ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

NGUYỄN ĐĂNG NGUYÊN

PHƯƠNG PHÁP XÂY DỰNG CÂY QUYẾT ĐỊNH DỰA TRÊN TẬP PHỤ THUỘC HÀM XẤP XỈ

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

THÁI NGUYÊN - 2017

ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

NGUYỄN ĐĂNG NGUYÊN

PHƯƠNG PHÁP XÂY DỰNG CÂY QUYẾT ĐỊNH DỰA TRÊN TẬP PHỤ THUỘC HÀM XẤP XỈ

Chuyên ngành: Khoa học máy tính
Mã số: 60 48 01 01

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

Người hướng dẫn khoa học: TS. LÊ VĂN PHÙNG

THÁI NGUYÊN - 2017

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn này do chính tôi thực hiện, dưới sự hướng dẫn khoa học của TS. Lê Văn Phùng, số liệu và kết quả nghiên cứu trong luận văn này hoàn toàn trung thực và chưa sử dụng để bảo vệ một công trình khoa học nào, các thông tin, tài liệu trích dẫn trong luận văn đã được chỉ rõ nguồn gốc. Mọi sự giúp đỡ cho việc hoàn thành luận văn đều đã được cảm ơn. Nếu sai tôi hoàn toàn chịu trách nhiệm.

Thái Nguyên, tháng 05 năm 2017

Học viên

Nguyễn Đăng Nguyên

LỜI CẢM ƠN

Trước hết em xin trân trọng cảm ơn các thầy giáo, cô giáo trường Đại học Công nghệ Thông tin và Truyền thông đã giảng dạy em trong quá trình học tập chương trình sau đại học. Dù rằng, trong quá trình học tập có nhiều khó khăn trong việc tiếp thu kiến thức cũng như sưu tầm tài liệu học tập, nhưng với sự nhiệt tình và tâm huyết của thầy cô cùng với những nỗ lực của bản thân đã giúp em vượt qua được những trở ngại đó.

Em xin bày tỏ lòng biết ơn sâu sắc tới thầy giáo TS.Lê Văn Phùng người hướng dẫn khoa học, đã tận tình hướng dẫn em trong suốt quá trình làm luận văn.

Xin chân thành cảm ơn các bạn bè, đồng nghiệp, các bạn học viên lớp cao học CK14A, những người thân trong gia đình đã động viên, chia sẻ, tạo điều kiện giúp đỡ trong suốt quá trình học tập và làm luận văn.

Một lần nữa em xin chân thành cảm ơn!

Thái Nguyên, tháng 05 năm 2017

Học viên

Nguyễn Đăng Nguyên

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT VÀ KÍ HIỆU SỬ DỤNG

Từ và Ký hiệu	Diễn giải
$R(U)$	Quan hệ trên tập thuộc U
$U = \{A_1, ..., A_m\}$	Tập m thuộc tính.
S = <U, F>	Lược đồ quan hệ với U là tập thuộc tính, F là tập các phụ thuộc hàm trên U
LĐQH	Lược đồ quan hệ
CSDL	Cơ sở dữ liệu
PTH	Phụ thuộc hàm
KPDL	Khai phá dữ liệu

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

THUẬT NGỮ TIẾNG ANH

Bảng quyết định	Decision Table
Cắt tỉa	Prune
Cây quyết định	Decision Tree
Độ tin cậy	Confidence
Giao tác	Transaction
Hệ thông tin	Information System
Khai phá phụ thuộc hàm xấp xỉ	Mining Approximate Functional Dependencies
Khóa	Key
Lớp tương đương	Equivalenc Classes
Luật quyết định	Decision Rule
Phân hoạch rút gọn	Stripped partitions
Phụ thuộc hàm	Functional Dependency
Phủ tối tiểu	Minimal Cover
Quan hệ	Relation
Rút gọn thuộc tính	Attribute Reduction
Siêu khóa	Super key
Sơ đồ quan hệ	Relation Schema
Tập ứng cử viên	Candidate_Set

MỞ ĐẦU

Công nghệ thông tin đã và đang trở thành lĩnh vực nghiên cứu, ứng dụng và phát triển hiệu quả trong đời sống kinh tế, xã hội. Việc ứng dụng công nghệ thông tin trong các ngành khoa học, kinh tế xã hội đã và đang mang lại những hiệu quả to lớn. Với những ngành khoa học, kinh tế - xã hội nơi có những kho dữ liệu khổng lồ thì việc tìm kiếm truy xuất và đưa ra những thông tin cần thiết phù hợp với thời gian và yêu cầu là không hề dễ dàng, chính vì điều này một thế hệ mới các phương pháp tiếp cận, phương pháp nghiên cứu và các kỹ thuật, công cụ cho phép phân tích tổng hợp, khai phá tri thức từ dữ liệu một cách thông minh và hiệu quả đã được các nhà khoa học quan tâm và nghiên cứu.

Một trong những lĩnh vực nghiên cứu các phương pháp ứng dụng khai phá dữ liệu, tìm kiếm chi thức, kết xuất tri thức… từ dữ liệu là cây quyết định (decision tree) cũng được nghiên cứu từ nhiều năm trước đây và đã có những kết quả khả quan và mang lại hướng ứng dụng có hiệu quả cao. Ngày nay, kỹ thuật khai phá dữ liệu dựa trên cây quyết định đã được áp dụng và mang lại hiệu quả cho nhiều ngành, nhiều lĩnh vực như: Kinh tế, tài chính, khoa học - kỹ thuật, ngân hang, thương mại, giáo dục, y tế… các kỹ thuật khai phá dự liệu bằng cây quyết định rất đa dạng và phong phú như các kỹ thuật dựa trên các thuật toán Hunt, ID3, C4.5,…và kỹ thuật xây dựng cây quyết định dựa trên các phụ thuộc hàm trong CSDL quan hệ.

Với mong muốn làm rõ hơn các kỹ thuật khai phá tri thức từ dữ liệu sử dụng cây quyết định nhằm phục vụ công tác nghiên cứu chuyên môn cũng như mong muốn đưa các kỹ thuật khai phá dữ liệu sử dụng cây quyết định vào thực tế nên tôi lựa chọn thực hiện luận văn tốt nghiệp là **"Phương pháp xây dựng cây quyết định dựa trên tập phụ thuộc hàm xấp xỉ"**. Mục đích khi thực hiện luận văn này là tổng hợp các kiến thức về kỹ thuật khai phá dữ liệu

bằng các kỹ thuật xây dựng cây quyết định dựa trên các tập phụ thuộc hàm của CSDL quan hệ.

Nội dung nghiên cứu luận văn gồm:

- Nghiên cứu tổng quan về khai phá dữ liệu và khai phá dữ liệu bằng cây quyết định, tập trung vào các phương pháp xây dựng cây quyết định.

- Nghiên cứu về phụ thuộc hàm, phụ thuộc hàm xấp xỉ trong CSDL quan hệ

- Nghiên cứu sâu về phương pháp xây dựng cây quyết định dựa vào phụ thuộc hàm xấp xỉ

- Xây dựng chương trình mô phỏng Phương pháp xây dựng cây quyết định dựa trên tập phụ thuộc hàm xấp xỉ

Cấu trúc luận văn gồm 3 chương bao gồm:

Chương 1: Tổng quan về cây quyết định và phụ thuộc hàm xấp xỉ.

Chương 2: Một số thuật toán xác định phụ thuộc hàm xấp xỉ và xây dựng cây quyết định.

Chương 3: Chương trình thử nghiệm xây dựng cây quyết định chẩn đoán bệnh tại Bệnh viện đa khoa Trung ương Thái Nguyên dựa trên việc khai phá tập phụ thuộc hàm xấp xỉ.

Chương 1
TỔNG QUAN VỀ CÂY QUYẾT ĐỊNH VÀ PHỤ THUỘC HÀM XẤP XỈ

1.1. Tổng quan về khai phá dữ liệu và cây quyết định

1.1.1. Khái niệm về khai phá dữ liệu, quá trình phát triển và ứng dụng trong việc phát hiện tri thức

"Khám phá tri thức là quá trình tìm ra những tri thức, đó là những mẫu tìm ẩn, trước đó chưa biết và là thông tin hữu ích đáng tin cậy". Còn khai phá dữ liệu (KPDL) là một bước quan trọng trong quá trình khám phá tri thức, sử dụng các thuật toán KPDL chuyên dùng với một số quy định về hiệu quả tính toán chấp nhận được để chiết xuất ra các mẫu hoặc các mô hình có ích trong dữ liệu. Nói một cách khác, mục đích của khám phá tri thức và KPDL chính là tìm ra các mẫu hoặc mô hình đang tồn tại trong các cơ sở dữ liệu (CSDL) nhưng vẫn còn bị che khuất bởi hàng núi dữ liệu [4].

Để có được những thông tin quý báu chúng ta phải tìm ra các mẫu có trong tập CSDL trước. Đầu ra của một chương trình là phát hiện những mẫu có ích được gọi là tri thức. Tri thức được phát hiện có các đặc điểm chính:

- Kiến thức cao cấp
- Độ chính xác cao
- Có tính hấp dẫn
- Có tính hiệu quả.

Nếu phát hiện tri thức là toàn bộ quá trình chiết xuất tri thức từ các CSDL thì KPDL là giai đoạn chủ yếu của quá trình đó. KPDL là một quá trình phát hiện các mẫu mới, thường bao gồm việc thử tìm mô hình phù hợp với tập dữ liệu và tìm kiếm các mẫu từ tập dữ liệu theo mô hình đó.

KPDL được sử dụng để tạo ra giả thuyết. Ví dụ như để xác định các yếu tố rủi ro khi cho vay tín dụng, kỹ thuật KPDL phải phát hiện được những người có thu nhập thấp và nợ nhiều là những người sẽ có mức rủi ro cao.

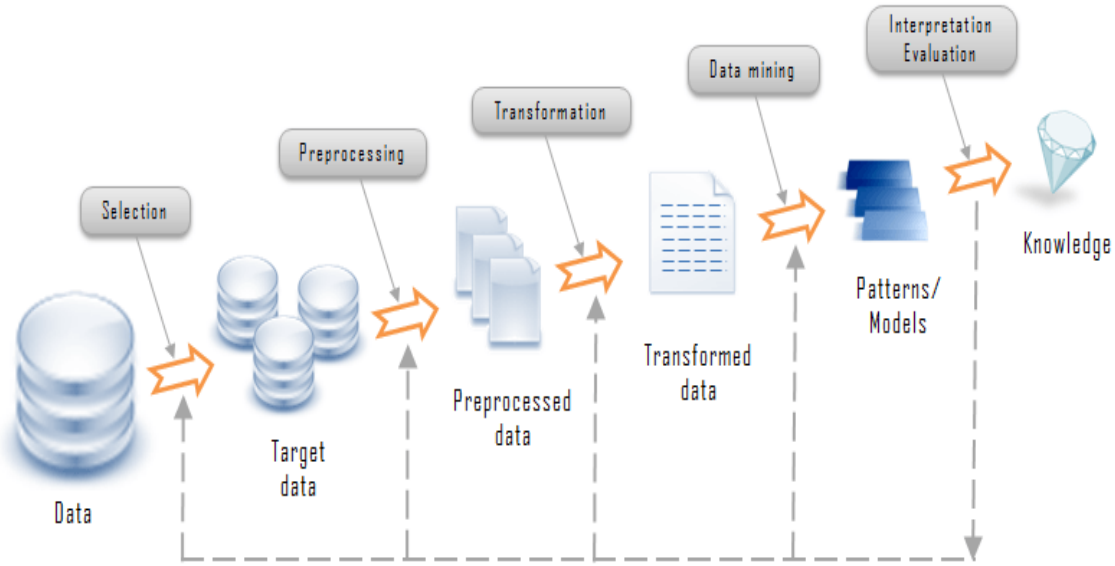
Ngoài ra kỹ thuật cũng có thể phát hiện ra những quy luật mà nhà phân tích có thể chưa tìm ra ví dụ như tỷ lệ giữa thu nhập trên nợ và tuổi cũng là các yếu tố xác định mức rủi ro. Để làm được điều này, KPDL sử dụng các thông tin trong quá khứ để học. Nó sẽ tìm kiếm các thông tin này trong các CSDL và sử dụng chúng để tìm ra các mẫu đáng quan tâm.

Nếu xét về mặt ý tưởng và mục đích ứng dụng, KPDL là một nhu cầu tất yếu, một sự nhạy cảm đáp lại sự mong mỏi của giới kinh doanh thì về mặt kỹ thuật, đó thực sự là một khó khăn và là cả sự thách thức đối với những nhà khoa học. KPDL được xây dựng dựa trên việc sử dụng các giải thuật mới, được định hướng theo như cầu kinh doanh để có thể giải quyết tự động các bài toán kinh doanh bằng các kỹ thuật dễ dùng và có thể hiểu được.

KPDL không thuộc một ngành công nghiệp nào. Nó sử dụng các kỹ thuật thông minh để khai phá các tri thức tiềm ẩn trong dữ liệu. Hiện nay trên thế giới đã có rất nhiều ngành công nghiệp sử dụng kỹ thuật KPDL để phục vụ cho hoạt động kinh doanh của mình và đã bước đầu thành công như ngành tài chính, y học, hóa học, bảo hiểm, sản xuất, giao thông, hàng không,… Các kết quả đạt được cho thấy mặc dù kỹ thuật KPDL hiện nay vẫn còn nhiều vấn đề nổi cộm, nhưng với những tri thức mà chuyên gia con người cũng chưa cung cấp được thì KPDL có một tiềm năng to lớn trong việc tạo ra những lợi nhuận đáng kể trong nền kinh tế.

Quá trình phát hiện tri thức từ CSDL là một quá trình có sử dụng nhiều phương pháp và công cụ tin học nhưng vẫn là một quá trình mà trong đó con người là trung tâm. Do đó, nó không phải là một hệ thống phân tích tự động mà là một hệ thống bao gồm nhiều hoạt động tương tác thường xuyên giữa con người và CSDL, tất nhiên là với sự hỗ trợ của các công cụ tin học. Người sử dụng hệ thống ở đây phải là người có kiến thức cơ bản về lĩnh vực cần phát hiện tri thức để có thể chọn được đúng các tập con dữ liệu, các lớp mẫu phù hợp và đạt tiêu chuẩn quan tâm so với mục đích. Tri thức

mà ta nói ở đây là các tri thức rút ra từ các CSDL, thường để phục vụ cho việc giải quyết một loạt nhiệm vụ nhất định trong một lĩnh vực nhất định. Do đó, quá trình phát hiện tri thức cũng mang tính chất hướng nhiệm vụ, không phải là phát hiện mọi tri thức bất kỳ mà là phát hiện tri thức nhằm giải quyết tốt nhiệm vụ đề ra.



Hình 1.1. Quá trình phát hiện tri thức

1.1.2. Khái quát về các phương pháp khai phá dữ liệu phổ biến

Quá trình khai phá dữ liệu là quá trình phát hiện mẫu, trong đó phương pháp khai phá dữ liệu để tìm kiếm các mẫu đáng quan tâm theo dạng xác định. Có thể kể ra đây một vài phương pháp như: sử dụng công cụ truy vấn, xây dựng cây quyết định, dựa theo khoảng cách (K-láng giềng gần), giá trị trung bình, phát hiện luật kết hợp, …

Vấn đề chính liên quan đến thuộc tính của bản ghi. Một bản ghi gồm hiều thuộc tính độc lập, nó bằng một điểm trong không gian tìm kiếm có số chiều lớn. Trong các không gian có số chiều lớn, giữa hai điểm bất kỳ hầu như có cùng khoảng cách. Vì thế mà kỹ thuật K-láng giềng không cho ta thêm một thông tin có ích nào, khi tất cả các cặp điểm đều là các láng giềng. Cuối

cùng, phương pháp K-láng giềng không đưa ra lý thuyết để hiểu cấu trúc dữ liệu. Hạn chế đó có thể được khắc phục bằng kỹ thuật cây quyết định [4].

1.1.2.1. Phương pháp sử dụng cây quyết định và luật

Với kỹ thuật phân lớp dựa trên cây quyết định, kết quả của quá trình xây dựng mô hình sẽ cho ra một cây quyết định. Cây này được sử dụng trong quá trình phân lớp các đối tượng dữ liệu chưa biết hoặc đánh giá độ chính xác của mô hình. Tương ứng với hai giai đoạn trong quá trình phân lớp là quá trình xây dựng và sử dụng cây quyết định.

Quá trình xây dựng cây quyết định bắt đầu từ một nút đơn biểu diễn tất cả các mẫu dữ liệu. Sau đó, các mẫu sẽ được phân chia một cách đệ quy dựa vào việc lựa chọn các thuộc tính. Nếu các mẫu có cùng một lớp thì nút sẽ trở thành lá, ngược lại ta sử dụng một độ đo thuộc tính để chọn ra thuộc tính tiếp theo làm cơ sở để phân chia các mẫu ra các lớp. Theo từng giá trị của thuộc tính vừa chọn, ta tạo ra các nhánh tương ứng và phân chia các mẫu vào các nhánh đã tạo. Lặp lại quá trình trên cho tới khi tạo ra được cây quyết định, tất cả các nút triển khai thành lá và được gán nhãn.

Quá trình đệ quy sẽ dừng lại khi một trong các điều kiện sau được thỏa mãn:

- Tất cả các mẫu thuộc cùng một nút.
- Không còn một thuộc tính nào để lựa chọn.
- Nhánh không chứa mẫu nào.

Phần lớn các giải thuật sinh cây quyết định đều có hạn chế chung là sử dụng nhiều bộ nhớ. Lượng bộ nhớ sử dụng tỷ lệ thuận với kích thước của mẫu dữ liệu huấn luyện. Một chương trình sinh cây quyết định có hỗ trợ sử dụng bộ nhớ ngoài song lại có nhược điểm về tốc độ thực thi. Do vậy, vấn đề tỉa bớt cây quyết định trở nên quan trọng. Các nút lá không ổn định trong cây quyết định sẽ được tỉa bớt.

Kỹ thuật tỉa trước là việc dừng sinh cây quyết định khi chia dữ liệu không có ý nghĩa.

1.1.2.2. Phương pháp phát hiện luật kết hợp

Phương pháp này nhằm phát hiện ra các luật kết hợp giữa các thành phần dữ liệu trong CSDL. Mẫu đầu ra của giải thuật khai phá dữ liệu là tập luật kết hợp tìm được. Ta có thể lấy một ví dụ đơn giản về luật kết hợp như sau: sự kết hợp giữa hai thành phần A và B có nghĩa là sự xuất hiện của A trong bản ghi kéo theo sự xuất hiện của B trong cùng bản ghi đó: A => B.

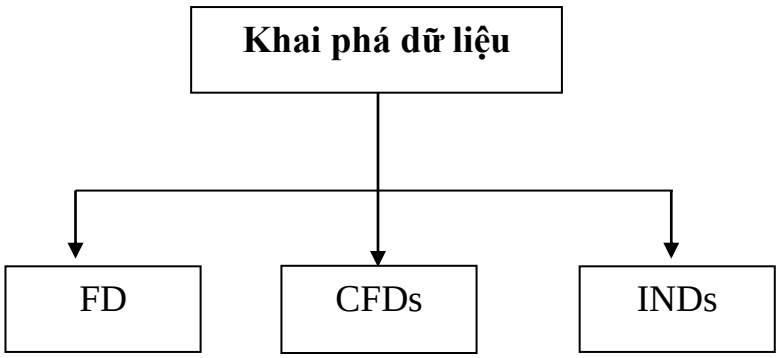
Các luật kết hợp có thể là một cách hình thức hóa đơn giản. Chúng rất thích hợp cho việc tạo ra các kết quả có dữ liệu dạng nhị phân. Giới hạn cơ bản của phương pháp này là ở chỗ các quan hệ cần phải thưa theo nghĩa không có tập thường xuyên nào chứa nhiều hơn 15 thuộc tính. Giải thuật tìm kiếm các luật kết hợp tạo ra số luật ít nhất phải bằng với số các tập phổ biến và nếu như một tập K phổ biến có kích thước K thì phải có ít nhất là 2 tập phổ biến. Thông tin về các tập phổ biến được sử dụng để ước lượng độ tin cậy của các tập luật kết hợp.

1.2. Phụ thuộc hàm xấp xỉ

1.2.1. Khái niệm về phụ thuộc hàm trong mô hình CSDL quan hệ

Phụ thuộc hàm biểu diễn mối quan hệ giữa các thuộc tính của một CSDL, một phụ thuộc hàm chỉ ra rằng giá trị của một thuộc tính được xác định duy nhất bởi giá trị của một số thuộc tính khác. Phụ thuộc hàm đóng vai trò quan trọng trong chuẩn hóa CSDL, phát hiện các phụ thuộc hàm cũng có thể giúp các nhà thiết kế CSDL tách một lược đồ quan hệ thành nhiều lược đồ quan hệ đạt dạng chuẩn cao hơn [5].

Phụ thuộc của các thuộc tính: có 3 loại phụ thuộc của các thuộc tính thường được khám phá là : phụ thuộc hàm (FD), phụ thuộc có điều kiện (CFDs) và phụ thuộc bao gồm (INDs). Hình 1.2 biểu diễn các loại phụ thuộc dữ liệu (theo [11]).



Hình 1.2. Các loại phụ thuộc dữ liệu

Cho tập hữu hạn khác rỗng các thuộc tính $U = \{A_1, ..., A_m\}$. Mỗi thuộc tính A_i có một miền giá trị tương ứng $Dom(A_i)$, $1 \le i \le m$. Một quan hệ trên *U*, ký hiệu *R* (*U*) hoặc *R* nếu không sợ nhầm lẫn, là một tập con của tích Descartes $Dom(A_1) \times Dom(A_2) \times ... \times Dom(A_m)$.

Một cách hình thức:

$$R(U) \subseteq Dom(A_1) \times Dom(A_2) \times ... \times Dom(A_m)$$

Các phần tử của quan hệ *R* được gọi là các *bộ*. Một quan hệ không chứa bộ nào được gọi là *quan hệ rỗng*.

Kí hiệu: t[*X*] là phép chiếu của bộ t trên tập thuộc tính *X*, $X \subseteq U$.

Định nghĩa 1.1: Một *phụ thuộc hàm* (*PTH*) trên quan hệ *R* (*U*) là một mệnh đề có dạng $X \to Y$ (trong đó *X*, $Y \subseteq U$). Ta nói PTH $X \to Y$ *đúng* trên quan hệ *R*, nếu: $(\forall t,s \in R): (t[X] = s[X] \Rightarrow t[Y] = s[Y])$

Khi PTH $X \to Y$ đúng trên quan hệ *R*. Người ta còn nói:

R thỏa PTH $X \to Y$ và ký hiệu *R* ($X \to Y$).

Ví dụ. Xét quan hệ *R* trên tập thuộc tính *U* = {*T*, *A*, *B*, *C*} cho trên bảng 1.1 như sau:

Bảng 1.1. Ví dụ về quan hệ

R =

T	*A*	*B*	*C*
1	*a1*	*b1*	*c1*
2	*a2*	*b2*	*c2*

	3	*a2*	*b2*	*c3*

Ta có: Chẳng hạn R thỏa các PTH $\{T\} \rightarrow \{A, B, C\}$, $\{C\} \rightarrow \{T, A, B\}$, $\{A\} \rightarrow \{B\}$, … và không thỏa PTH $\{B\} \rightarrow \{C\}$, …

Như vậy, có thể thấy PTH (trên quan hệ) là sự phụ thuộc của một số thuộc tính vào một số thuộc tính khác.

Một cặp $S = (U, F)$ với U là tập các thuộc tính và F là tập các PTH trên U được gọi là một *lược đồ quan hệ* (*LĐQH*).

Quan hệ R được gọi là *thỏa* tập PTH F, ký hiệu $R(F)$ nếu với mọi PTH $X \rightarrow Y \in F$ thì R $(X \rightarrow Y)$.

Cho tập các PTH F trên U và $X \rightarrow Y$ là một PTH bất kỳ trên U. Ta nói F *suy diễn logic* PTH $X \rightarrow Y$, ký hiệu $F \models X \rightarrow Y$. Nếu với mọi quan hệ R (U) sao cho R (F) thì R $(X \rightarrow Y)$.

Bao đóng của tập PTH F trên U, ký hiệu F^+ là tập nhỏ nhất các PTH trên U chứa F và thỏa các tính chất (A1) – (A3) như sau [5]:

Với $\forall X, Y, Z \subseteq U$:

(A1) *Tính phản xạ*

Nếu $Y \subseteq X$ thì $X \rightarrow Y \in F^+$

(A2) *Tính gia tăng*

Nếu $X \rightarrow Y \in F^+$ thì $X \cup Z \rightarrow Y \cup Z \in F^+$

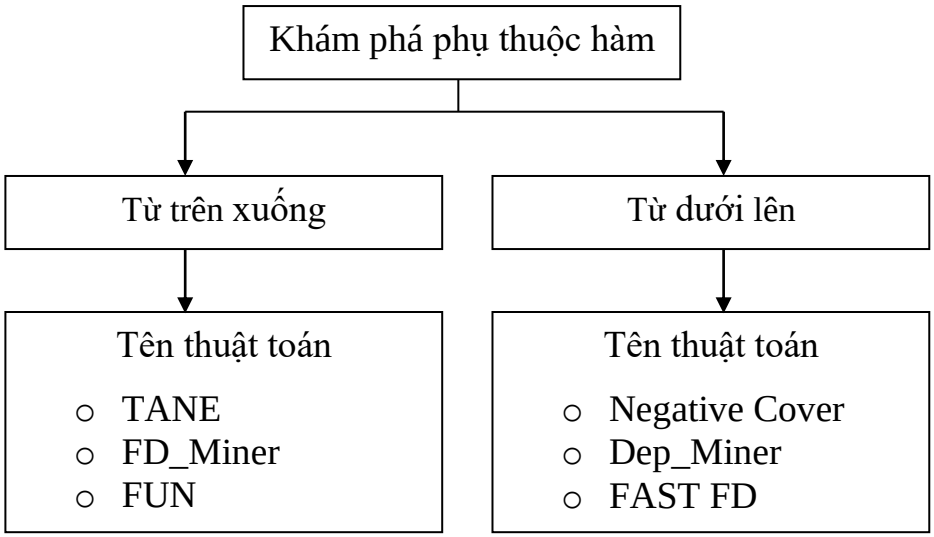
(A3) *Tính bắc cầu*

Nếu $X \rightarrow Y \in F^+$ và $Y \rightarrow Z \in F^+$ thì $X \rightarrow Z \in F^+$

Rõ ràng F^+ hữu hạn vì U hữu hạn.

Các tính chất từ (A1) – (A3) còn thường được gọi là hệ tiên đề *Armstrong* hay *tập quy tắc suy diễn Armstrong*.

Phát hiện phụ thuộc hàm từ một bảng quan hệ đã được nhiều nhà nghiên cứu quan tâm. Đã có nhiều thuật toán được đề xuất, hình 1.3 biểu diễn các phương pháp cùng một số thuật toán.



Hình 1.3. Kỹ thuật phát hiện phụ thuộc hàm

Bảng 1.2 trình bày một số thuật toán điển hình và các kỹ thuật sử dụng trong thuật toán đó (theo [11]).

Bảng 1.2. Các thuật toán khám phá phụ thuộc hàm

Năm	**Tên thuật toán**	**Kỹ thuật sử dụng**
1999	TANE	Phân hoạch (Partitions)
2000	Dep-Miner	Tập khác biệt (Difference Set)
2001	FUN	Phụ thuộc hàm nhúng (Embedded FD)
2002	FD_Mine	Phân hoạch và lớp tương đương (Partitions and Equivalences)
2008	FD_Discover	Các lớp tương đương và phủ tối thiểu (Equivalent Classes and Minimal Cover)
2010	FDs Using Rough Sets	Tập thô (Rough sets)
2011	FDs via Bayes Net Analyses	Sử dụng mạng Bayes Net và Heuristics (Bayes Net and Heuristics)

1.2.2. Khái niệm về phụ thuộc hàm xấp xỉ và các đặc trưng của chúng

Trong thực tế do có một số giá trị dữ liệu không chính xác hoặc một số ngoại lệ nào đó làm cho các phụ thuộc hàm không thỏa. Sự phụ thuộc tuyệt đối này dường như quá nghiêm ngặt khi ta hình dung tới một quan hệ có hàng nghìn bộ, trong khi đó chỉ có khoảng vài bộ vi phạm phụ thuộc hàm. Bỏ qua các phụ thuộc hàm này sẽ làm mất tính chất phụ thuộc vốn có giữa các thuộc tính. Vì vậy các nhà nghiên cứu đã mở rộng khái niệm phụ thuộc hàm thành phụ thuộc hàm xấp xỉ theo một cách thức, một nghĩa nào đó, các phụ thuộc hàm xấp xỉ (Approximate Functional Dependencies - AFDs) này cho phép có một số lượng lỗi nhất định của các bộ dữ liệu đối với phụ thuộc hàm [5].

1.2.2.1. Định nghĩa phụ thuộc hàm xấp xỉ

Định nghĩa 1.2. [11] Cho quan hệ R (U) và $X \rightarrow Y$ là một PTH trên U. Gọi $S \subseteq R$ là quan hệ sao cho có số bộ ít nhất cần loại bỏ khỏi R để $R \backslash S$ thỏa mãn PTH ($X \rightarrow Y$). Khi đó tỷ số của $|S|$ và $|R|$ được gọi là độ đo lỗi của PTH $X \rightarrow Y$ trên R, ký hiệu g_3 ($X \rightarrow Y$, R).

Như vậy

$$g_3(X \rightarrow Y, R) = \frac{\min\{|S| \, | \, S \subseteq R, (R \setminus S)\ (X \rightarrow Y)\}}{|R|}$$

Một cách tương đương

$$g_3(X \rightarrow Y, R) = 1 - \frac{\max\{|S| \, | \, S \subseteq R, S(X \rightarrow Y)\}}{|R|}$$

Độ đo lỗi trên còn được gọi là độ đo lỗi g_3.

Có thể thấy nếu $X \rightarrow Y$ là một PTH đúng trên R thì độ đo lỗi $g_3(X \rightarrow Y, R) = 0$.

Cho ngưỡng lỗi ε với $0 \leq \varepsilon \leq 1$. Người ta nói PTH $X \rightarrow Y$ là một PTH xấp xỉ trên R ứng với ngưỡng lỗi ε nếu $g_3(X \rightarrow Y, R) \leq \varepsilon$. Khi đó người ta còn nói PTH xấp xỉ $X \rightarrow Y$ đúng trên R ứng với ngưỡng lỗi ε.

Ví dụ. Xét quan hệ *R* trên tập thuộc tính *U* = {*A*, *B*, *C*, *D*} cho trên bảng 1.3 như sau:

Bảng 1.3. Bảng quan hệ ví dụ về PTH xấp xỉ

rr =	**Bộ**	**A**	**B**	**C**	**D**
	t_1	0	0	1	1
	t_2	0	1	1	1
	t_3	0	2	1	2
	t_4	1	2	0	0
	t_5	2	0	0	0
	t_6	2	0	2	0
	t_7	1	1	2	1

Xét PTH $AB \rightarrow C$ trên *R* (*U*).

Rõ ràng bỏ bộ 6 (hoặc bộ 5) thì PTH $AB \rightarrow C$ sẽ đúng trên quan hệ *R*. Tức số bộ ít nhất cần loại khỏi quan hệ *R* để PTH $AB \rightarrow C$ đúng trên các bộ còn lại là 1. Suy ra: $g_3(AB \rightarrow C,\ R) = \frac{1}{7}$

Với ngưỡng lỗi $\varepsilon = 0{,}5$, ta có: $g_3(AB \rightarrow C,\ R) = \frac{1}{7} \leq 0{,}5$.

Do đó PTH $AB \rightarrow C$ là một PTH xấp xỉ trên *R* ứng với $\varepsilon = 0{,}5$.

Lưu ý: Khi chúng ta nói xét một PTH trên quan hệ *R*(*U*) có nghĩa PTH đó xác định trên tập thuộc tính *U* và muốn tìm hiểu xem PTH đó có đúng trên quan hệ *R* hay không.

1.2.2.2. Một số độ đo cơ bản

Khi nghiên cứu phát hiện các PTH xấp xỉ thì vấn đề xác định độ đo cho các PTH xấp xỉ đóng vai trò cực kì quan trọng. Đã có nhiều tác giả đưa ra nhiều độ đo dựa vào nhiều cách khác nhau. Ở đây luận văn tìm hiểu một số độ đo cơ bản.

Trong định nghĩa về PTH xấp xỉ khi có một bộ làm cho PTH không đúng. Người ta gọi bộ này là một trường hợp ngoại lệ. Như vậy, PTH xấp xỉ chính là PTH với các trường hợp ngoại lệ. Các độ đo của PTH xấp xỉ dựa trên số lượng các trường hợp ngoại lệ. Nó là cơ sở để xác định, phát hiện các PTH xấp xỉ [15].

Dưới đây, ta xem xét một số cách mở rộng.

Cách 1: [4, 5, 6]

Cho quan hệ r và phụ thuộc hàm $X \to Y$. Ba độ đo lỗi của $X \to Y$ trong r được đề xuất như sau:

$$G_1(X \to Y, r) = |\{(t_1, t_2) \mid t_1, t_2 \in r, t_1[X] = t_2[X], t_1[Y] \neq t_2[Y]\}|$$

$$g_1(X \to Y, r) = \frac{G_1(X \to Y, r)}{|r|^2}$$

$$G_2(X \to Y, r) = |\{t_1 \mid t_1 \in r, \exists t_2 \in r : t_1[X] = t_2[X], t_1[Y] \neq t_2[Y]\}|$$

$$g_2(X \to Y, r) = \frac{G_2(X \to Y, r)}{|r|}$$

$$G_3(X \to Y, r) = |r| - \max\{|s| \mid s \subseteq r, s \models X \to Y\}$$

$$g_3(X \to Y, r) = \frac{G_3(X \to Y, r)}{|r|}$$

Đặt g là một trong ba độ đo g_1, g_2, g_3. Cho trước ε, $0 \le \varepsilon \le 1$. Ta nói $X \to Y$ là $\varepsilon - good$ đối với g trong r nếu $g(X \to Y, r) \le \varepsilon$. Ngược lại $X \to Y$ là $\varepsilon - bad$.

Cho quan hệ $r(R)$ khi đó độ đo lỗi g_3 của một phụ thuộc hàm $X \to A$ được xác định như sau:

$$g_3(X \to A) = 1 - \frac{\max\{|s| \mid s \subseteq r, s \models X \to Y\}}{|r|}$$

Từ đó $X \to A$ đứng trên r ứng với một ngưỡng lỗi $\varepsilon \in [0,1]$ khi và chỉ khi $g_3(X \to Y) \le \varepsilon$.

Cách 2:

Cho r là một quan hệ trên tập thuộc tính $U=\{A_1,A_2,...,A_n\}$ trong đó các thuộc tính $A_1,A_2...A_n$ có thể là thuộc tính định danh, rời rạc hoặc liên tục. Đối với những thuộc tính định danh, ta tiến hành thực hiện ánh xạ tất cả các giá trị có thể tới một tập các số nguyên dương liền kề.

Định nghĩa khoảng cách giữa hai bộ giá trị trên tập thuộc tính: Với hai bộ $t_1,t_2\in r$, ta kí hiệu $\rho(t_1(X),t_2(X))$ là khoảng cách giữa t_1 và t_2 trên tập thuộc tính $X\subseteq R$ được xác định như sau.

$$\rho(t_1(X),t_2(X))=\max(|t_1(A_i)-t_2(A_i)|)/\max(|t_1(A_i),t_2(A_i)|),A_i\in X)$$

- Hàm max(x,y) là hàm chọn số lớn nhất trong hai số x,y.

- Trường hợp $\max(|t_1(A_i),t_2(A_i)|)=0$, tức $t_1(A_i)=t_2(A_i)=0$ ta qui ước:

$$|t_1(A_i)-t_2(A_i)|)/\max(|t_1(A_i),t_2(A_i)|)=0$$

Khoảng cách giữa hai bộ giá trị trên tập thuộc tính có thể coi là hàm số của các đối số là các bộ giá trị của quan hệ và tập các thuộc tính.

Cách 3:

Cho $R(U)$ là một lược đồ quan hệ $X,Y\subseteq U$ và R là một quan hệ trên $R(U)$ với n bộ. Khi đó độ thỏa mà R thỏa $X\to Y$ ký hiệu là $TRUTH_R(X\to Y)$, được định nghĩa như sau:

$$TRUTH_R(X\to Y)=\frac{\sum_{t_i\neq t_j}^{\forall t_i,t_j\in r}TRUTH_{(t_i,t_j)}(X\to Y)}{NTP}$$

trong đó nếu $t_i(X)=t_j(X)$ và $t_i(Y)\neq t_j(Y)$ thì $TRUTH_{(t_i,t_j)}(X\to Y)=0$, ngược lại $TRUTH_{(t_i,t_j)}(X\to Y)=1$; NTP là số cặp bộ trong R và bằng $n(n-1)/2$.

Ví dụ: Với quan hệ R như bảng 1.5 dưới đây, ta có:

$TRUTH_R$(Trái cây$\to$ Đồ uống) = 77.78%.

Bảng 1.3: Bảng quan hệ ví dụ

ID	Trái cây	Đồ uống
1	Vải	Coca Cola
2	Vải	Coca Cola
3	Vải	Spirit
4	Vải	Spirit
5	Vải	N/A
6	Táo	Spirit
7	Táo	Spirit
8	Táo	Coca Cola

Cách 4:

Xây dựng quan hệ tương đương $E(X)$ trên r như sau:

$t_1,t_2 \in E(X) \Leftrightarrow t_1[X] = t_2[X], \forall t_1,t_2 \in r$

Quan hệ $E(X)$ phân hoạch r thành các lớp tương đương, mỗi lớp tương đương là một tập con của r chứa các bộ giống nhau trên X. Ký hiệu phân hoạch đó là $PART(X)$.

Cho $u \in PART(X)$. Nếu $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y] \forall t_1,t_2 \in u$ thì ta nói rằng u thỏa $X \to Y$. Ngược lại, u không thỏa $X \to Y$.

Cách 5: [7]

Phụ thuộc hàm điều kiện:

Định nghĩa 1.3: Một phụ thuộc hàm điều kiện có dạng $(X \to A, t_p)$, trong đó $X \to A$ là một phụ thuộc hàm (kinh điển) và t_p là một bộ mẫu với các thuộc tính trong X và A. Bộ mẫu t_p chứa các giá trị hằng và biến không tên "–". Biến không tên "–" có thể nhận một giá trị tùy ý trong miền thuộc tính tương ứng. Xét quan hệ s được cho dưới đây.

Bảng 1.4: Bảng quan hệ ví dụ về phụ thuộc hàm điều kiện

A	B	C	D
001	124	12	34
001	124	21	43
002	157	34	69
002	157	34	61
002	158	89	62
002	158	89	90
003	167	78	96

Ta thấy s không thỏa $B \to C$. Tuy nhiên s thỏa ràng buộc $[A = 002, B] \to [C]$.

Theo định nghĩa phụ thuộc hàm điều kiện ta thấy ràng buộc $[A = 002, B] \to [C]$ là một phụ thuộc hàm điều kiện có dạng $(AB \to C, (002, -\| -))$. Phụ thuộc hàm kinh điển là trường hợp riêng của phụ thuộc hàm điều kiện. Phụ thuộc hàm $X \to A$ chính là phụ thuộc hàm điều kiện dạng $(X \to A, t_p)$ với $t_p[B] = -, \forall B \in X \cup \{A\}$. Phụ thuộc hàm điều kiện được sử dụng trong bài toán làm sạch dữ liệu.

1.3. Kết luận chương 1

Nội dung chương 1 trình bày tổng quan về quá trình phát hiện tri thức và khai phá dữ liệu: khái niệm về khai phá dữ liệu, quá trình phát triển và ứng dụng trong việc khám phá tri thức, khái quát về các phương pháp khai phá dữ liệu phổ biến. Nội dung trọng tâm của chương 1 trình bày chi tiết về Phụ thuộc hàm xấp xỉ: Các khái niệm về phụ thuộc hàm trong mô hình CSDL quan hệ, khái niệm về phụ thuộc hàm xấp xỉ và các đặc trưng của chúng. Chương này cũng hệ thống hóa các kiến thức cơ bản của lý thuyết về cây quyết định cùng với những ví dụ minh họa cụ thể.

Chương 2
MỘT SỐ THUẬT TOÁN XÁC ĐỊNH PHỤ THUỘC HÀM XẤP XỈ VÀ XÂY DỰNG CÂY QUYẾT ĐỊNH

2.1. Thuật toán TANE xác định phụ thuộc hàm xấp xỉ từ quan hệ

2.1.1. Khái niệm lớp tương đương và phân hoạch

Cách tiếp cận của TANE để phát hiện PTH là dựa trên tập các bộ bằng nhau trên tập các thuộc tính nào đó. Việc kiểm tra các PTH có thỏa mãn hay không có thể thực hiện bằng cách kiểm tra vế phải có bằng nhau hay không khi thấy vế trái bằng nhau. Hơn thế nữa, hoàn toàn có thể đánh dấu các bộ mà có vế phải không bằng nhau. Cách tiếp cận này hoàn toàn tốt đối với PTH xấp xỉ trên cơ sở sử dụng khái niệm *lớp tương đương* và *phân hoạch [13]*.

Phương pháp phân hoạch chia các bộ dữ liệu thành các nhóm dựa trên những giá trị khác nhau của mỗi cột (thuộc tính). Với mỗi thuộc tính, số các nhóm bằng với số các giá trị khác nhau của mỗi thuộc tính đó. Mỗi nhóm được gọi là một *lớp tương đương*.

Hai bộ t và u là *tương đương* đối với một tập X các thuộc tính cho trước nếu $t[A] = u[A]$ với mọi A trong X. Mỗi tập thuộc tính bất kỳ X nào cũng có thể phân hoạch các bộ của quan hệ thành các lớp tương đương.

Chúng ta biểu thị lớp tương đương của một bộ $t \in r$ với một tập cho trước $X \subseteq R$ bởi $[t]_X$, tức là $[t]_X = \{u \in r \mid t[A] = u[A] \forall A \in X\}$.

Ta gọi tập $\pi_X = \{[t]_X \mid t \in r\}$ của các lớp tương đương là một *phân hoạch* của r theo X. Như vậy mỗi lớp tương đương ứng với một giá trị duy nhất cho tập thuộc tính X và hợp của các lớp tương đương sẽ đúng bằng quan hệ r. Lực lượng của phân hoạch π, ký hiệu là $|\pi|$, là số lớp tương đương trong π.

Bảng 2.1. Bảng quan hệ minh họa cho phân hoạch

TupleID	A	B	E	C	D
1	1	a	2	$	Hoa
2	1	A	2		Hoa Nhài
3	2	A	0	$	Hướng Dương
4	2	A	0	$	Hoa
5	2	b	0		Hoa Huệ
6	3	b	1	$	Hoa Lan
7	3	C	1		Hoa
8	3	C	1	#	Hoa Hồng

Thuộc tính A có giá trị "1" chỉ trong các bộ 1 và 2 vì vậy chúng tạo thành một lớp tương đương $[1]_{\{A\}} = [2]_{\{A\}} = \{1,2\}$. Tương tự, thuộc tính A có giá trị "2" trong các bộ 3, 4, 5 và có giá trị "3" trong các bộ 6, 7, 8.

Tương tự, $\pi_{\{A\}} = \{\{1,2\},\{3,4,5\},\{6,7,8\}\}$. Các lớp tương đương với tổ hợp các thuộc tính $\{B,C\}$ là $\pi_{(B,C)}$= {{1},{2},{3,4},{5},{6}, {7},{8}}.

2.1.2. Phân hoạch mịn hơn

Khái niệm phân hoạch mịn hơn liên quan trực tiếp với các phụ thuộc. Một phân hoạch π mịn hơn một phân hoạch π' khác nếu mỗi lớp tương đương trong π đều là tập con của một lớp tương đương nào đó của π'.

Bổ đề 1 [1]:

Một PTH $X \to A$ đúng nếu và chỉ nếu π_X mịn hơn $\pi_{\{A\}}$.

Để kiểm tra $X \to A$ đúng hay không chúng ta kiểm tra $|\pi_X|$ có bằng với $|\pi_{X\cup\{A\}}|$ hay không. Nếu π_X mịn hơn $\pi_{\{A\}}$ thì $\pi_{X\cup\{A\}}$ bằng với π_X (có cùng

số lớp tương đương với π_X). Mặt khác do $\pi_{X\cup\{A\}}$ luôn mịn hơn π_X nên $\pi_{X\cup\{A\}}$ và π_X chỉ có cùng số lớp tương đương khi $\pi_{X\cup\{A\}}$ bằng π_X

Bổ đề 2

Một PTH $X \to A$ *đúng nếu và chỉ nếu* $|\pi_X| = |\pi_{X\cup\{A\}}|$

Ví dụ: Xét quan hệ cho trên bảng 2.2 sau:

Bảng 2.2. Bảng quan hệ ví dụ cho phân hoạch mịn hơn

TupleID	A	B	E	C	D
1	1	a	2	$	Hoa
2	1	A	2		Hoa Nhài
3	2	A	0	$	Hướng Dương
4	2	A	0	$	Hoa
5	2	b	0		Hoa Huệ
6	3	b	1	$	Hoa Lan
7	3	C	1		Hoa
8	3	C	1	#	Hoa Hồng

Các lớp tương đương đối với thuộc tính $A:\{\{1,2\},\{3,4,5\},\{6,7,8\}\}$, các lớp tương đương đối với thuộc tính $E:\{\{1,2\},\{3,4,5\},\{6,7,8\}\}$. Vì các lớp tương đương của thuộc tính A mịn hơn các lớp tương đương của thuộc tính E nên qua đó có thể phát hiện PTH $A \to E$ được thỏa mãn.

Một định nghĩa chuẩn của một phụ thuộc xấp xỉ $X \to A$ là dựa trên số tối thiểu những hàng cần loại bỏ khỏi quan hệ r để phụ thuộc $X \to A$ đúng trong r. Sai số $g_3(X \to A) = 1 - (\max\{|s| \,|\, s \subseteq r$ và $X \to A$ đúng trong s$\}$)/|r|.

Độ đo g_3 có một cách hiểu tự nhiên như tỷ số của các hàng với những ngoại lệ hoặc sai số ảnh hưởng đến phụ thuộc. Cho một ngưỡng sai số ε, $0 \le \varepsilon \le 1$, chúng ta nói rằng $X \to A$ là một phụ thuộc xấp xỉ nếu và chỉ nếu $g_3(X \to A)$ lớn nhất là bằng ε.

Lỗi e(X→A) có thể được tính toán từ các phân hoạch π_X và $\pi_{X\cup A}$ theo cách: Mỗi lớp tương đương c của π_X là hợp của một hoặc một vài lớp tương đương $c'_1, c'_2 \ldots$ của $\pi_{X\cup A}$, và thực chất của việc loại đi tất cả các bộ để X→A trở thành đúng là việc loại đi một trong các lớp tương đương c'_i này. Số lượng nhỏ nhất các bộ cần loại bỏ là kích thước của lớp c trừ đi kích thước của lớp c'_i lớn nhất. Tính tổng cộng cho các lớp tương đương trong π_X sẽ được tổng số các bộ cần loại bỏ. Như vậy chúng ta có công thức tính lỗi g3 (X→A) từ các phân hoạch π_X và $\pi_{X\cup A}$:

$$g_3(X \to A) = 1 - \sum_{c\in\pi X} \max\{|c'| \, | \, c' \in \pi_{X\cup\{A\}} \text{ và } c' \subseteq c\}/|r| = e(X\to A)$$

cho biết tỷ lệ số bộ cần loại.

Ví dụ:

Bảng 2.3: Bảng quan hệ minh họa cho PTH xấp xỉ

TupleID	A	B	E	C	D
1	1	a	2	$	Hoa
2	1	A	2		Hoa Nhài
3	2	A	0	$	Hướng Dương
4	2	A	0	$	Hoa
5	2	b	0		Hoa Huệ
6	3	b	1	$	Hoa Lan
7	3	C	1		Hoa
8	3	C	1	#	Hoa Hồng

Để kiểm tra $A \to B$ thỏa hoặc không, chúng ta tìm các lớp tương đương: $\pi_A = \{\{1,2\},\{3,4,5\},\{6,7,8\}\}$

$\pi_B = \{\{1\},\{2,3,4\},\{5,6\},\{7,8\}\}$.

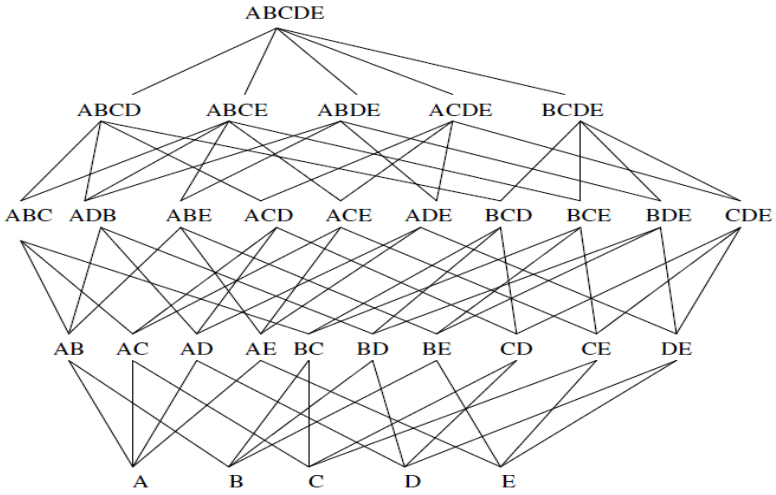
Vì lớp tương đương $\{1,2\}$ trong π_A không mịn hơn bất kỳ lớp nào trong π_B, và tương tự như vậy đối với các lớp khác của π_A. Do đó $A \to B$ không đúng.

Tuy nhiên trong ví dụ trên, $A \to B$ có thể đúng với một sai số g_3 nào đó nếu chúng ta loại bỏ một vài bộ khỏi quan hệ đã cho. Đầu tiên chúng ta tìm $\pi_{A\cup\{B\}} = \{\{1\},\{2\},\{3,4\},\{5\},\{6\},\{7,8\}\}$. Lớp tương đương $\{1,2\}$ trong π_A bằng $\{1\}\cup\{2\}$ lấy từ $\pi_{A\cup\{B\}}$, có kích thước lớn nhất của $\{1\}$ và $\{2\}$ là 1. Lớp tương đương $\{3,4,5\}$ trong π_A bằng $\{3,4\}\cup\{5\}$ lấy từ $\pi_{A\cup\{B\}}$, có kích thước lớn nhất của $\{3,4\}$ và $\{5\}$ là 2. Cuối cùng lớp tương đương $\{6,7,8\}$ của π_A bằng $\{6\}\cup\{7,8\}$ lấy từ $\pi_{A\cup\{B\}}$ có kích thước lớn nhất của $\{6\}$ và $\{7,8\}$ là 2.

Từ đó $g_3(A \to B) = 1-(1+2+2)/8 = 0.375$. Điều này nghĩa là ít nhất 3 bộ trong 8 bộ ở ví dụ trên cần loại bỏ để $A \to B$ được thỏa mãn. Như vậy có thể nói PTH: $A \to B$ thỏa mãn xấp xỉ với tỉ lệ sai số $\varepsilon = 0.375$.

Quá trình phát hiện các PTH xấp xỉ này được lặp lại cho tất cả thuộc tính và cho tất cả các tổ hợp của chúng. Lấy ví dụ, cho một quan hệ với 5 thuộc tính (A, B, C, D, E) tập các ứng viên là {Ø, A, B, C, D, E, AB, AC, AD, AE, BC, BD, BE, CD, CE, DE, ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE, CDE, ABCD, ABCE, ABDE, ACDE, BCDE, ABCDE} cho một tổng cộng 32 tổ hợp. Các thuộc tính ứng viên này của quan hệ được biểu diễn như một dàn được chỉ trong hình 2.1.

Mỗi nút trong hình 2.1. biểu diễn một tập thuộc tính ứng viên. Một đường nối giữa hai nút bất kỳ như E và ED chỉ ra phụ thuộc xấp xỉ: ED\E→E cần được kiểm tra.



Hình 2.1. Dàn cho các thuộc tính (A, B, C, D, E)

2.1.3. Thuật toán TANE cải tiến

Cách tiếp cận của TANE dựa trên số cực tiểu các bộ cần phải loại khỏi quan hệ r để X→A trở thành một PTH đúng trên r [13].

Công thức tính lỗi e(X→A)=min $\{|s|: s \subseteq r$ và X→A đúng trên r\s$\}$ /$|r|$, với $|s|, |r|$ là lực lượng của s và r. Một cách tự nhiên, độ đo e có thể hiểu là tỷ lệ các bộ ngoại lệ hay lỗi ảnh hưởng lên PTH.

Cho một ngưỡng ε nào đó nằm trong đoạn kín [0,1], khi đó X→A được gọi là PTH xấp xỉ nếu và chỉ nếu e(X→A) ≤ ε .

Thuật toán TANE được cải tiến để tìm tất cả các PTH xấp xỉ dạng $X \to A$ có độ đo lỗi e(X→A) ≤ ε, với X, A⊆R (R là lược đồ quan hệ tương ứng với quan hệ r), còn ε là ngưỡng lỗi cho trước.

2.1.4. Chiến lược tìm kiếm

2.1.4.1. Chiến lược chung

Để tìm tất cả các PTH tối thiểu có nghĩa, thuật toán được xây dựng theo chiến lược tìm kiếm trên dàn dữ liệu theo từng mức, từ nhỏ tới lớn. Bắt đầu từ tập thuộc tính đơn và lặp lại công việc đối với tập thuộc tính lớn hơn theo từng mức trên dàn chứa các thuộc tính. Một mức L_λ là một tập các thuộc tính có kích thước λ, trong đó các tập trong L_λ rất có khả năng tạo thành các PTH.

Ban đầu mức L_1= {{A}| A∈R} (kích thước 1: có 1 thuộc tính), sau đó mức L_2 sẽ được tính từ L_1, và L_3 được tính từ L_2,… theo các thông tin thu nhận được trong quá trình thực hiện [12,13].

Khi thuật toán xử lý tập X, nó kiểm tra tất cả các PTH có dạng X\{A}→A, với A∈X. Điều này bảo đảm rằng chỉ có các PTH có ý nghĩa được xem xét. Chiều tim kiếm từ nhỏ tới lớn của thuật toán bảo đảm rằng kết quả thu được chỉ gồm những PTH tối thiểu. Đồng thời nó cũng giúp làm giảm không gian tìm kiếm một cách hiệu quả.

Cũng như chiến lược tìm kiếm từ nhỏ tới lớn, thì chiến lược tìm kiếm theo mức cũng được áp dụng thành công trong rất nhiều các ứng dụng khai phá dữ liệu. Cùng với việc làm giảm không gian tìm kiếm hiệu quả, thuật toán tìm kiếm theo mức còn có hệu quả làm giảm việc tính toán ở mỗi mức bằng cách sử dụng kết quả từ những mức trước.

2.1.4.2. Giảm bớt không gian tìm kiếm

1- Thu nhỏ tập thuộc tính là ứng cử viên về phải Rhs (Rhs -Right_hand site) của PTH

TANE làm việc trên dàn cho đến khi các PTH tối thiểu đúng đắn được tìm thấy.

Để kiểm tra tính tối thiểu của một PTH có dạng X\{A}→A, chúng ta cần kiểm tra xem Y\{A}→A có phải là PTH hay không, trong đó Y là tập con thực sự của X. Chúng ta lưu thông tin này vào trong *tập C(Y)- tập các ứng cử viên về phải của Y*.

Tập C(X) với X là một tập thuộc tính, chỉ bao gồm các thuộc tính A sao cho A không PTH vào bất kỳ một tập con thực sự nào của X.

Chi tiết hơn, tập các thuộc tính ứng cử viên Rhs của tập thuộc tính X⊆R được tính như sau:

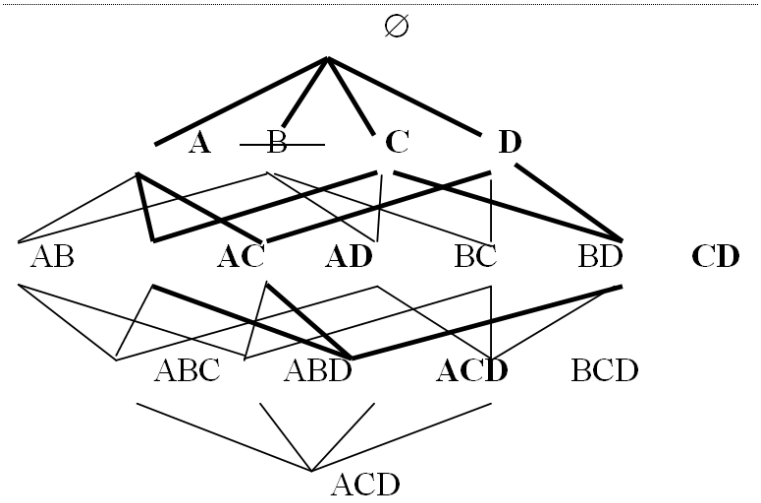
C(X)=R\$\overline{C(X)}$, trong đó $\overline{C(X)}$ ={A∈X: X\{A}→A là đúng}

Để tìm PTH tối thiểu, ta chỉ cần kiểm tra PTH có dạng X\{A}→A, trong đó A∈X và A∈ C(X)\{B} với ∀B∈X.

Ví dụ: giả sử TANE đang xét tập X={A,B,C} và giả sử C→A là một PTH. Do C→A đúng, nên ta có A∉C({A,C})= C(X)\{B}, do đó ta có {B,C}→A không tối thiểu.

Hạn chế không gian tìm kiếm trong TANE dựa trên yếu tố là nếu C(X)=∅ thì C(Y)=∅ với ∀Y là tập bao hàm X (supersets Y of X).

Do đó không có PTH nào ở dạng Y\{A}→A có thể là PTH tối thiểu, và không cần phải tính toán đối với tập Y. Việc tìm kiếm rộng thực hiện việc này rất hiệu quả, như trên hình sau:



Hình 2.2. Một tập đã được cắt tia chứa dàn cho {A,B,C,D}.

Do xóa B nên chỉ có các thành viên tô đậm mới được tham gia trong thuật toán ở mức cao hơn

2- Thu nhỏ tập mịn hơn chứa tập các thuộc tính là ứng cử viên về phải (Rhs$^+$)

Ứng cử viên về phải Rhs đủ để bảo đảm các PTH phát hiện được là tối thiểu. TANE còn có thể sử dụng tập mịn hơn các ứng cử viên Rhs+ là $C^+(X)$ nhằm giúp thu gọn hơn nữa không gian tìm kiếm PTH tối thiểu một cách hiệu quả.

$C^+(X)$= {A∈R | ∀B∈X: X\{A,B}→{B} không đúng}

Chú ý rằng A và B có thể bằng nhau. Bổ đề sau đây sẽ chỉ ra cách sử dụng Rhs+ để kiểm tra tính tối thiểu của PTH giống như đã làm với Rhs.

Bổ đề 3:

Cho A∈X và X\{A}→A là một PTH đúng,

PTH X\{A}→A là tối thiểu ⇔ với ∀B∈X ta có A∈C^+(X\{B})

Bổ đề 4:

Cho B∈X và X\{B}→B là một PTH đúng,

Nếu X→A đúng thì X\{B}→A đúng.

Bổ đề này cho phép chúng ta loại thêm được những thuộc tính trong tập những ứng cử viên Rhs : C+(X). Giả sử có X\{B}→B là một PTH, theo bổ đề ta có những PTH có X trong vế trái không phải là PTH tối thiểu, do chúng ta có thể loại B khỏi vế trái mà vẫn không làm thay đổi tính đúng đắn của PTH. Do đó, có thể loại khỏi C(X) những tập sau đây một cách an toàn

$\overline{C'(X)}$= { R\X nếu ∃ B∈X: X\{B}→B đúng
∅ nếu trái lại

Ví dụ [13]:

Giả sử TANE đang kiểm tra tập thuộc tính X={A,B,C} và {C}→B là một PTH đúng. Do A∈ $\overline{C'(\{B,C\})}$ = $\overline{C'(X\backslash\{A\})}$, nên X\{A}→A không phải là tối thiểu.

Hơn thế nữa, sỉa sử X có một tập con thực sự Y mà Y\B→B đúng với một thuộc tính B nào đó (B∈Y). Ta cũng có thể loại khỏi C(X) tất cả các thuộc tính A mà A∈X\Y. Tập các thuộc tính có thể loại đi được tính như sau:

$\overline{C''(X)}$= { A∈X : ∃ B∈X\{A} sao cho X\{A,B} →B đúng}

Ví dụ:

Giả sử TANE đang kiểm tra tập thuộc tính X={A,B,C,D} và có {C}→B là một PTH đúng. Do vậy, A∈ $\overline{C''(\{A,B,C\})}$ = $\overline{C''(X\backslash\{D\})}$, nên X\{A}→A không phải là tối thiểu.

Bổ đề 5:

$C^+(X)$ = {A∈R: ∀B∈X: X\{A,B}→B không đúng}

= ((R\ $\overline{C(X)}$)\ $\overline{C'(X)}$\ $\overline{C''(X)}$

3-Hạn chế không gian tìm kiếm theo khóa

Một tập thuộc tính là một ***siêu khóa*** nếu như không có 2 bộ nào bằng nhau trên tập thuộc tính đó, chẳng hạn như phân hoạch π_B chỉ bao gồm các lớp tương đương mà từng lớp này chỉ có duy nhất 1 bộ. Tập X là một ***khóa*** nếu như nó là một siêu khóa và không có một tập con thực sự nào của nó là siêu khóa. Khi 1 khóa được tìm ra trong quá trình tìm kiếm các PTH, thì có thể áp dụng cách để hạn chế không gian tìm kiếm nữa.

Bổ đề 6 [13]:

Cho B∈X và X\{B}→B là một PTH đúng. Nếu X là một siêu khóa, thì ta có X\{B} là một siêu khóa.

2.1.4.3. Tính toán với các phân hoạch

Để làm giảm thời gian và không gian chi việc tính toán trên các phân hoạch người ta áp dụng thêm hai phương pháp sau, thứ nhất là thay vì tính toán trên các phân hoạch, ta tính trên phân hoạch rút gọn, thứ hai, là phương pháp tính toán lỗi e một cách nhanh chóng.

Người ta đưa ra khái niệm siêu khóa xấp xỉ để có thể thực hiện các công việc trên một cách dễ dàng hơn.

Ta định nghĩa e(X) là số bộ nhỏ nhất cần loại khỏi r để X trở hành một siêu khóa. Nếu e(X) là đủ nhỏ, thì ta nói X là một siêu khóa xấp xỉ. Lỗi e(X) có thể dễ dàng tính được từ các phân hoạch theo công thức:

$$e(X) = 1 - |\pi_X| / |r|$$

1-Phương pháp Phân hoạch rút gọn [13]

Một phân hoạch rút gọn (Stripped partitions) là một phân hoạch trong đó đã loại bỏ các lớp tương đương có kích thước bằng 1. Một phân hoạch rút gọn của một phân hoạch π được kí hiệu là π^*. Chẳng hạn ta có, π^*_D= {{1,4,7}}. Việc loại bỏ các lớp tương đương có kích thước bằng 1 là rất dễ hiểu, bởi vì lớp tương đương có kích thước bằng 1 không làm ảnh hưởng đến tính đúng đắn của bất kỳ một PTH nào.

Một phân hoạch rút gọn cũng chứa đầy đủ thông tin như một phân hoạch bình thường. Ví dụ giá trị e(X) dễ dàng tìm được từ phân hoạch rút gọn theo công thức:

$$e(X)= (||\pi^*_X||-|\pi^*_X|) / |r| \qquad (1)$$

Trong đó, $||\pi^*_X||$ là tổng kích thước của các lớp tương đương trong π^*_X. Đồng thời, mối quan hệ làm mịn giữa các phân hoạch không vị thay đổi và bổ đề 1 vẫn đúng đối với phân hoạch rút gọn.

Bổ đề 2 không đúng đối với phân hoạch rút gọn, bởi vì $|\pi^*_X|$ không thể bằng $|\pi^*_{X\cup A}|$, hơn thế nữa, $\pi^* \neq \pi^*_{X\cup A}$. Tuy nhiên, do e(X)= e(Y) $\Leftrightarrow |\pi_X|=|\pi_Y|$, ta có bổ đề sau thay thế cho bổ đề 2:

Bổ đề 7:

Một PTH $X\rightarrow A$ đúng $\Leftrightarrow e(X) = e(X\cup\{A\})$

2-Phương pháp tính Lỗi e

Ta có công thức sau đây về ràng buuộc của e(X→A), đó là:

$$e(X) - e(X\cup\{A\}) \leq e(X\rightarrow A) \leq e(X) \qquad (2)$$

nếu $e(X) - e(X\cup\{A\}) > \varepsilon$ hoặc $e(X) < \varepsilon$ thì thuật toán không cần phải tính e(X→A) để kiểm tra PTH X→A thỏa mãn hay không.

3-Tính các phân hoạch

Trong mỗi bước lặp, TANE không tính các phân hoạch từ đầu bằng cách tính trực tiếp từ dữ liệu đầu vào, mà dựa trên các phân hoạch đã tính

được từ bước lặp trước bằng sản phẩm của hai phân hoạch π' và π'' đã được tính trước đây. Sản phẩm của 2 phân hoạch π' và π'' , được lí hiệu là π'. π'', chính là phân hoạch π mịn nhất, mịn hơn cả hai phân hoạch π' và π''. Ta có bổ đề sau:

Bổ đề 8

Với $\forall X,Y \subseteq R$ *ta có* $\pi'. \pi'' = \pi_{X \cup Y}$. ($\pi_X. \pi_Y = \pi_{X \cup Y}$)

Đầu tiên, các phân hoạch π_A sẽ được tính trực tiếp từ CSDL, với mọi thuộc tính $A \in R$. Sau đó, các phân hoạch π_X với kích thước của X là $|X| \geq 2$ sẽ được tính bằng một sản phẩm của các phân hoạch đối với 2 tập con của X mà có kích thước là |X|-1.

Sau khi có phân hoạch π_X, lỗi e(X) sẽ được tính, để kiểm tra tính thỏa mãn theo bổ đề 7. Phân hoạch đầy đủ chỉ cần thiết cho việc tính các phân hoạch cho mức tiếp theo.

Kể từ lần lặp tiếp theo, thuật toán chỉ thực hiện trên định danh của các bộ, do đó, thu được các lợi ích như, dễ thực hiện, vì không phải quan tâm đến các loại dữ liệu khác nhau, khi tính toán cho PTH xấp xỉ, các bộ bị thiếu giá trị vẫn có thể tính bình thường.

2.1.4.4. Thủ tục chính của thuật toán TANE cải tiến

Thuật toán TANE cải tiến để xác định các PTH xấp xỉ [13]:

Input: Quan hệ r trên lược đồ R

Output: Các PTH tối thiểu, không tầm thường đúng trên r

$C(X) = \{A \in X \mid X \setminus \{A\} \rightarrow A$ không đúng$\}$

$= R \setminus \{A \in X: X \setminus \{A\} \rightarrow A$ đúng$\} = R \setminus \overline{C(X)}$ (ký hiệu phủ định)

$C^+(X) = \{A \in R: \forall B \in X: X \setminus \{A,B\} \rightarrow B$ không đúng$\}$

1. $L_0 := \{\varnothing\}$
2. $C^+(\varnothing) := R$

3. $L_1 := \{\{A\} | A \in R\}$ /xét 1 thuộc tính A của R

4. $\lambda := 1$

5. while $L_\lambda \neq \varnothing$

6. COMPUTE_DEPENDENCIES(L_λ)

7. PRUNE (L_λ)

8. $L_{\lambda+1}$:= GENERATE_NEXT_LEVEL(L_λ)

9. $\lambda := \lambda + 1$

ở đây, việc tính GENERATE_NEXT_LEVEL(L_λ) là:

$L_{\lambda+1} = \{X: |X| = \lambda+1, \forall Y: Y \subset X, |Y| = \lambda$ chúng ta có $Y \in L_\lambda\}$

Thủ tục tính các PTH:

Chương trình con này sẽ tìm kiếm tất cả các PTH tối thiểu mà có vế trái nằm trong $L_{\lambda-1}$.

Procedure COMPUTE_DEPENDENCIES(L_λ)

1 for each $X \in L_\lambda$ do

2 $C^+(X) := \bigcap_{A \in X} C^+(X \setminus \{A\})$

3 for each $X \in L_\lambda$ do

4 for each $A \in X \bigcap C^+(X)$ do

5 **if** $e(X \setminus \{A\} \to A) \leq \varepsilon$ **then**

6 output $X \setminus \{A\} \to A$

7 remove A from $C^+(X)$

8 **If** $X \setminus \{A\} \to A$ thỏa mãn **then**

9 remove all B in $R \setminus X$ from $C^+(X)$

Theo bổ đề 3 *[Cho $A \in X$ và $X\setminus\{A\} \to A$ là một PTH, PTH $X\setminus\{A\} \to A$ là tối thiểu $\Leftrightarrow \forall B \in X$, chúng ta có: $A \in C+(X\setminus\{B\})$]:* các bước 2,4,5 bảo đảm chỉ có những PTH tối thiểu có dạng $X\setminus\{A\} \to A$ được đưa ra, trong đó $X \in L_\lambda$ và

$A \in X$. Việc kiểm tra tính đúng của PTH trong dòng 5 dựa theo bổ đề số 7: PTH $X \to A$ đúng $\Leftrightarrow e(X)=e(X \cup \{A\})$.

COMPUTE_DEPENDENCIES(L_λ) cũng tính các tập $C^+(X)$ với $\forall X \in L_\lambda$. Bổ đề sau chỉ ra rằng điều này là hoàn toàn đúng.

Bổ đề:

Cho $\forall Y \in L_{\lambda-1}$, cho $C^+(Y)$ đã được tính là đúng, sau khi thực hiện thủ tục COMPUTE_DEPENDENCIES(L_λ), $C^+(X)$ là được tính đúng đối với $\forall X \in L_\lambda$

Dòng 8 triển khai thực hiện sự khác biệt giữa $C^+(X)$ và $C(X)$. Nếu dòng này bị chuyển đi, thuật toán vẫn thực hiện chính xác nhưng việc hạn chế không gian tìm kiếm sẽ kém hiệu quả.

Thủ tục Tỉa các thuộc tính trên dàn:

Chương trình con này sẽ hạn chế không gian tìm kiếm bằng cách loại khỏi L_λ những tập như đã xem xét trong phần trước (section 3, từ chiến lược chung đến tính toán các phân hoạch).

Procedure PRUNE L_λ

1 for each $X \in L_\lambda$ do

2 if $C^+(X) = \varnothing$ do

3 delete X from L_λ

4 if X is a (super)key do

5 for each $A \in C^+(X) \setminus X$ do

6 If $A \in \cap_{B \in X} C^+(X \cup \{A\} \setminus \{B\})$ then

7 output $X \to A$

8 delete X from L_λ

Chương trình con này đã áp dụng 2 quy tắc hạn chế không gian tìm kiếm được mô tả trong phần trước:

1-Thứ nhất, X bị loại bỏ nếu $C^+(X)=\varnothing$,

2-Thứ hai, X bị loại bỏ nếu X là một khóa.

Trong trường hợp 1, nếu $C^+(X)=\varnothing$, thì các dòng 4-8 trong thủ tục *COMPUTE_DEPENDENCIES* , và các dòng 5-7 trong thủ tục PRUNE sẽ không làm gì cả. sau khi $C^+(Y)=\varnothing$, cung đối với $\forall Y \supset X$, việc xóa X sẽ không có hiệu quả trên output của thuật toán.

Còn về trường hợp 2, tỉa theo khóa, tính đúng của việc tỉa là dựa trên mệnh đề sau:

Cho X-siêu khóa và $A \in X$. PTH $X\{A\} \rightarrow A$ là đúng và cực tiểu $\Leftrightarrow$ $X\{A\}$là 1 khóa và, đối với $\forall B \in X$, $A \in C+(X\{B\})$

Một PTH $X \rightarrow A$ là output cua dòng 7 của thủ tục PRUNE $\Leftrightarrow$ x là 1 khóa,

$A \in C+(X\{X\})$ và $A \in C+(X \cup \{A\}\{B\})$, đối với $\forall B \in X$.

Mệnh đề trên đã chắc chắn chỉ ra rằng 1 PTH đúng và tối tiểu. Nó cũng cho biết nếu 1 PTH tối tiểu $X\{A\} \rightarrow A$ không phải là output trong thủ tục COMPUTE_DEPENDENCIES bởi vì việc tỉa đó, nó là output trong thủ tục PRUNE. Do vậy, việc tỉa là hoàn toàn chính xác.

Chương trình con sinh mức kế tiếp GENERATE _NEXT_LEVEL (L_λ) sẽ thực hiện sinh ra mức mới từ mức hiện tại.

Chương trình GENERATE _NEXT_LEVEL (L_λ)

Chương trình con sinh mức kế tiếp sẽ thực hiện sinh mức $L_{\lambda+1}$ từ mức L_λ. Mức $L_{\lambda+1}$ chỉ bao gồm các tập thuộc tính có kích thước $\lambda+1$ mà các tập con kích thước λ của chúng đều thuộc mức L_λ.Các phương pháp tỉa bảo đảm rằng không có PTH nog bị mất.

Procedure **GENERATE _NEXT_LEVEL (L_λ)**

1. $L_{\lambda+1}:=\varnothing$
2. For each $K \in$PREFIX_BLOCKS(L_λ) do
3. For each $\{Y,Z\} \subseteq K$, $Y \neq Z$ do

4. X:= Y∪Z

5. If for all A∈X, X\{A}∈ L_λ then

6. $L_{\lambda+1}$:= $L_{\lambda+1}$∪ {X}

7. RETURN $L_{\lambda+1}$

Trong đó thủ tục_PREFIX_BLOCKS(L_λ) thực hiện phân chia L_λ thành các tập con rời nhau như sau. Giả sử một tập X ∈ L_λ đã được sắp xếp các thuộc tính theo thứ tự. Hai tập X,Y ∈ L_λ thuộc về cùng một prefix block nếu chúng có cùng một tiền tố chiều dài λ-1, tức là chỉ khác nhau đúng vị trí cuối cùng khi đã sắp xếp. Mỗi prefix block tạo thành các khối liên tiếp nhau theo thứ tự từ điển trong L_λ. Các prefix block dễ dàng được tính từ thứ tự từ điển của L_λ.

2.1.4.5. Thuật toán TANE và việc phân hoạch

TANE dường như không liên quan gì đến việc phân hoạch, nhưng thực sự, trong dòng 5 của COMPUTE_DEPENDENCIES đòi hỏi phải biết e(X) và e(X\{A}), và việc kiểm tra siêu khóa trong dòng 4 của hàm PRUNE cũng dựa trên e(X). Trong TANE, giá trị e được tính từ các phân hoạch rút gọn theo công thức:

$$e(X)=(\| |\pi^*_x| \| - |\pi^*_x|) / |r|$$

trong đó $\| |\pi^*_x| \|$ là tổng kích thước của các lớp tương đương trong π^*_x , còn π^* là phân hoạch rút gọn của phân hoạch π.

Các phân hoạch được tính như sau [13]:

Ban đầu, các phân hoạch trên 1 thuộc tính được tính trực tiếp từ quan hệ r. Một phân hoạch π{A} được tính từ cột r[A] như sau. Đầu tiên, các giá trị của các cột được thay thế bằng các số nguyên 1,2,3,… nên quan hệ tương đương không bị thay đổi, nghĩa là các giá trị giống nhau được thay bằng cùng một số nguyên và các giá trị khác nhau với những số nguyên khác nhau. Điều này có thể thực hiện trong thời gian tuyến tính bằng cách sử dụng cấu trúc dữ liệu cây hoặc bảng băm để ánh xạ các giá trị gốc thành các số nguyên. Sau đó,

giá trị t[A] đó sẽ là định danh cho lớp tương đương $[t]_{\{A\}}$ của $\pi_{\{A\}}$, và do đó $\pi_{\{A\}}$ dễ dàng được xây dựng. Cuối cùng các lớp tương đương có kích thước bằng 1 sẽ được giản ước đi và tạo thành phân hoach rút gọn $\pi^*_{\{A\}}$.

Các phân hoạch trên tập thuộc tính X lớn hơn được tính toán khi X được đưa vào mức tương ứng, tại dòng 6 của GENERATE_NEXT_LEVEL. Tập X có dạng Y∪Z, và phân hoạch π_x được tính giống như sản phẩm $\pi_y.\pi_z$. Sản phẩm đó dược tính toán với thủ tục sau với độ phức tạp tuyến tính.

Procedure STRIPPED_PRODUCT

Input: phân hoach rút gọn

$\pi^{*'}= \{c'_1,c'_2,\ldots, c'_{|\pi^{*'}|}\}$ và $\pi^{*''}=\{c''_1,c''_2,\ldots, c''_{|\pi^{*''}|}\}$

Output: phân hoạch rút gọn $\pi^*=(\pi'. \pi'')^*$

1. $\pi^* := \varnothing$
2. For i:=1 to $|\pi^{*'}|$ do
3. For each $t \in c'_i$ do T[t]:=i
4. S[i] := ∅
5. For i:=1 to $|\pi^{*''}|$ do
6. For each $t \in c''_i$ do
7. if T[t]≠ NULL then S[T[t]]:=S[T[t]]∪{t}
8. for each $t \in c''_i$ do
9. if $|S[T[t]]| \geq 2$ then $\pi^* := \pi^* \cup \{S[T[t]]\}$
10. S[T[t]]:=∅
11. For i:=1 to $|\pi^{*'}|$ do
12. for each $t \in c'_i$ do T[t]:= NULL
13. Return π^*

Thủ tục này bảo đảm rằng T được khởi tạo là NULL, và trước khi ra khỏi chương trình lại gán NULL trở lại để dùng về sau.

Nhận xét:

Thuật toán TANE được gọi là thuật toán cải tiến (để tính tất cả các PTH xấp xỉ tối thiểu). Việc sửa đổi quan trọng là thay đổi việc kiểm tra tính đúng trên dòng 5 gốc (if $X\backslash\{A\}\rightarrow A$ đúng then) của thủ tục COMPUTE_DEPENDENCIES bởi:

5' **If** $e(X \setminus \{A\} \rightarrow A) \leq \varepsilon$ **then**

Cũng vậy, thay dòng 8 gốc (remove all B in $R \setminus X$ from $C^{+}(X)$) của thủ tục COMPUTE_DEPENDENCIES bởi:

8' **If** $X \setminus \{A\} \rightarrow A$ đúng **then**

9' remove all B in $R \setminus X$ from $C^{+}(X)$.

Thuật toán trên sẽ trả về những PTH xấp xỉ tối thiểu. Trong một vài ứng dụng, điều đó là có ích để nhận biết các PTH xấp xỉ mà không tối thiểu nhưng có lỗi nhỏ hơn.

Để thực hiện việc kiểm tra tại dòng 5', đầu tiên ta kiểm tra ràng buộc: **if** $X\backslash\{A\}\rightarrow A$ đúng **then**. Nếu không thỏa mãn ràng buộc này thì lúc đó ta mới cần tính $e(X\backslash\{A\}\rightarrow A)$ nhờ cách phân hoạch rút gọn [13]:

Procedure e

Input: các phân hoạch rút gọn π^*x và $\pi^*_{x\cup A}$,

Output: e($X\rightarrow A$).

1. e:=0
2. for each $c\in\pi^*_{x\cup A}$, do
3. chọn bất kỳ $t\in c$
4. T[t]:= $|c|$
5. For each $c\in\pi^*_x$ do
6. m:=1
7. For each $t\in$ c do m:=max{m,T[t]}
8. e:=e+[c]-m
9. For each $c\in\pi^*_{x\cup A}$ do
10. Chọn $t\in$ c (cùng t ở dòng 3)
11. T[t]:=0
12. Return e/ $|r|$

Chú ý thấy sự tương tự với thủ tục STRIPPED_PRODUCT. Ở đây cũng thấy rằng bảng T phải được khởi tạo về 0, nhưng không cần tạo lại sau đó.

Độ phức tạp của thuật toán TANE cải tiến

Với quan hệ R có |R| thuộc tính và |r| bộ. Độ phức tạp thời gian của thuật toán TANE cải tiến phụ thuộc vào số bộ trong CSDL |r|, phụ thuộc vào số tập trong tất cả các mức của dàn các thuộc tính ứng viên.

$$s = O(2^{|R|})$$

và số lượng khóa

$$K = O(2^{|R|} / \sqrt{|R|}).$$

Do đó độ phức tạp của thuật toán TANE sửa đổi là:

$$O(s(|r| + |R|^2) + K|R|^3)$$

2.2. Thuật toán xác định phụ thuộc hàm xấp xỉ dựa trên luật kết hợp

2.2.1. Luật kết hợp

2.2.1.1. CSDL giao tác

Cho tập I có *m* mục dữ liệu, ký hiệu là $I = \{I_1, I_2, ..., I_m\}$. Mỗi tập con $T \subseteq I$ được gọi là một *giao tác*. Một tập các mục dữ liệu được gọi là *tập mục*. Một tập mục chứa *k* mục được gọi là *k - tập mục*. Tập I trên chính là m-tập mục [3,14,15].

Một cơ sở dữ ***liệu giao tác*** *D* là một tập các giao tác.

Ví dụ :

Cho CSDL giao tác *D* trên tập gồm 5 mục *I* = {*A, H, L, E, F*}. CSDL giao tác *D* = {{*A, F, L, E*};{*F, H, E*}; *A, F, L, E*};{*A, F, H, E*};{*A, F, H, L, E*};{*F, H, L*}}

Bảng 2.4. Ví dụ về CSDL giao tác D

Định danh giao tác	Giao tác
1	{*A, F, L, E*}
2	{*F, H, E*}
3	{*A, F, L, E*}
4	{*A, F, H, E*}
5	{*A, F, H, L, E*}
6	{*F, H, L*}

Một giao tác *T* được gọi là *hỗ trợ* tập mục $X \subseteq I$ nếu nó chứa tất cả các mục của *X*, nghĩa là $X \subseteq$ T.

Độ hỗ trợ của tập mục X, ký hiệu *supp*(*X*) là tỷ số giữa số các giao tác có chứa *X* với số các giao tác của *D*. Tức là:

$$\text{supp}(X) = \frac{|\{T \in D \mid X \subseteq T\}|}{|D|}$$

Có thể thấy: $0 \leq supp\ (X) \leq 1$ với *mọi* tập mục *X*.

2.2.1.2. Tập mục phổ biến/ thường xuyên

Tập mục *X* được gọi là *tập mục phổ biến* (hay *tập mục thường xuyên*), nếu *supp* (*X*) ≥ *minsupp*. Trong đó *minsupp* ∈ [0, 1] là giá trị cho trước bởi NSD thường được gọi là *ngưỡng hỗ trợ tối thiểu*.

Mệnh đề [Bramer M. A. (2007)] (về một số tính chất cơ bản của Tập mục phổ biến)

Cho hai tập mục X, Y và X ⊆ Y. Khi đó

(1) (*Độ hỗ trợ của tập mục con*) *supp* (*X*) ≥ *supp* (*Y*).

(2) *Nếu Y là tập phổ biến thì X cũng là tập phổ biến.*

(3) *Nếu X là tập không phổ biến thì Y cũng là tập không phổ biến.*

Bảng 2.5. Ví dụ về các tập phổ biến với độ hỗ trợ tương ứng, minsupp = 50%

Các tập mục phổ biến	Độ hỗ trợ tương ứng
{*F*}	100% (6/6)
{*E*}, {*F, E*}	83% (5/6)
{*A*}{*H*}{*D*}{*A, F*}{*A, E*}{*F, H*}{*F, L*}{*A, F, E*}	67% (4/6)
{*A,L*}{*H,E*}{*L,E*}{*A,F,L*}{*A,L,E*}{*F,H,E*}{*F,L,E*}	50% (3/6)

2.2.1.3. Định nghĩa luật kết hợp

Một luật kết hợp (LKH) là một mệnh đề có dạng $X \Rightarrow Y$, trong đó $X, Y \subseteq I$ thỏa mãn điều kiện $X \cap Y = \varnothing$. Tập *X* gọi là tiền đề, tập *Y* gọi là kết luận của luật.

LKH có hai độ đo quan trọng là *độ hỗ trợ* và *độ tin cậy*:

Độ hỗ trợ (Support) của luật $X \Rightarrow Y$, ký hiệu là $s(X \Rightarrow Y)$, là tỷ lệ phần trăm các giao tác trong *D* có chứa $X \cup Y$.

Tức là: $s(X \Rightarrow Y) = \dfrac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|D|} = \text{supp}(X \cup Y)$

Do đó theo quan điểm xác suất, độ hỗ trợ của luật đặc trưng cho tần suất xuất hiện của các mẫu trong luật.

$$s(X \Rightarrow Y) = P(X \cup Y)$$

Độ tin cậy (Confidence) của luật $X \Rightarrow Y$, ký hiệu $c(X \Rightarrow Y)$. Là tỷ lệ phần trăm giữa các giao tác trong D có chứa $X \cup Y$ với các giao tác trong D có chứa X. Tức là

$$c(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$$

Như vậy, có thể hiểu độ tin cậy $c(X \Rightarrow Y)$ là tỷ lệ phần trăm giao tác trong D có chứa X thì cũng có chứa Y. Theo quan điểm xác suất $c(X \Rightarrow Y)$ chính là xác suất có điều kiện mà giao tác cho trước T hỗ trợ X thì T cũng hỗ trợ Y:

$$c(X \Rightarrow Y) = P(Y / X) = \frac{P(Y \cup X)}{P(X)}$$

Như vậy, độ tin cậy của LKH $X \Longrightarrow Y$ thể hiện sự tương quan giữa X và Y. Độ tin cậy đo sức mạnh của luật và người ta chỉ quan tâm đến các LKH có độ tin cậy cao.

2.2.1.4. Định nghĩa luật kết hợp mạnh

LKH $X \Longrightarrow Y$ được gọi là *LKH mạnh,* nếu $s(X \Rightarrow Y) \geq$ minsup và $c(X \Rightarrow Y) \geq$ minconf . Trong đó, *minsup* và *minconf* là các giá trị cho trước, được xác định bởi người dùng, có miền giá trị thuộc đoạn [0, 1].

Ý nghĩa của LKH được thể hiện ở những điểm sau đây [14].

Giả sử có LKH là những luật có dạng:

+70% khách hàng mua đường thì mua thêm sữa, 30% giao tác có mua cả đường lẫn sữa.

+75% bệnh nhân có hút thuốc lá và sống ở ven vùng ô nhiễm thì bị ung thư phổi. Trong đó có 25% số bệnh nhân vừa hút thuốc lá, vừa sống ven vùng ô nhiễm vừa bị ung thư phổi.

Ở đây: Vế trái của luật: "mua đường", "hút thuốc lá và sống ven vùng ô nhiễm". Vế phải của luật: "mua sữa", "ung thư phổi". Còn những con số như 30%, 25%: độ hỗ trợ của luật; 70%, 75%: độ tin cậy của luật.

Ta thấy tri thức đem lại bởi LKH ở trên có một sự khác biệt cơ bản so với thông tin thu được từ các câu lệnh truy vấn dữ liệu thông thường. Đó thường là những tri thức, những mối liên hệ chưa được biết trước và *mang tính chất dự báo, đang tiềm ẩn trong dữ liệu*. Những tri thức này không đơn giản chỉ là kết quả của các phép nhóm, tính tổng, sắp xếp mà là kết quả của một quá trình tính toán khai phá phức tạp và tốn nhiều thời gian [14].

Tuy nhiên LKH là dạng luật khá đơn giản nhưng lại mang rất nhiều ý nghĩa. Thông tin mà dạng luật này đem lại là rất đáng kể và hỗ trợ không nhỏ trong quá trình ra quyết định.

LKH có một số tính chất cơ bản sau [Bramer M. A. (2007)]:

1. *Nếu* $X \Rightarrow Y$ *là LKH mạnh thì* $X \Rightarrow Z$ *với* $Z \subseteq Y$ *cũng là LKH mạnh.*

2. *Nếu* $X \Rightarrow Z$ *và* $Y \Rightarrow Z$ *là các LKH mạnh thì không nhất thiết* $X \cup Y \Rightarrow Z$ *cũng là LKH mạnh*.

3.*Nếu* $X \cup Y \Rightarrow Z$ *là LKH mạnh thì* $X \Rightarrow Z$ *và* $Y \Rightarrow Z$ *chưa chắc LKH mạnh.*

4. *Nếu* $X \Rightarrow Y$ *và* $Y \Rightarrow Z$ *là LKH mạnh thì chưa chắc* $X \Rightarrow Z$ *là LKH mạnh.*

2.2.2.Biểu diễn PTH xấp xỉ qua LKH

Chúng ta trình bày lại một số khái niệm của LKH theo quan điểm của PTH xấp xỉ [14].

Xét tập R gồm m thuộc tính R={A_1, A_2,…A_m}, quan hệ *r* (*R*),

PTH xấp xỉ: $X \rightarrow Y$ trên *r*, $X,Y \subseteq R$.

Giả sử $Dom(X) = \{x_1, ..., x_k\}$, $Dom(Y) = \{y_1, ..., y_m\}$,

n = |r| (số các bộ trong r)

n_{xi}= |{t∈r: t[X]= x_i}|, n_{yi}= |{t∈r: t[Y]= y_i}|

và n_{xiyj}= |{t∈r: t[X]= x_i và t[Y]= y_j}|

Ta hiểu mỗi mục là một đối tượng (bộ) gắn với một thuộc tính của *R*. Như vậy, với mỗi thuộc tính $A_i \in R$ cho ta một mục i_{A_i}. Do đó, mục i_{A_i} là một đối tượng độc lập với các thể hiện của *R* và miền giá trị của A_i. Khi đó tập mục I_X được xác định như sau: $I_X = \{i_{A_i} \mid A_i \in X\}$

Và tập tất cả các mục I_R được xác định là: $I_R = \{i_{A1},...,i_{Am}\}$

Tập các giao tác, ký hiệu *TD*, được định nghĩa là: Với mỗi cặp các bộ :

$(t, s) \in r \times r$ có một giao tác $ts \in TD$: $i_{A_i} \in ts \Leftrightarrow t[A_i] = s[A_i]$

Như vậy, mỗi giao tác trong *TD* tương ứng một cặp các bộ trong quan hệ *r*. Sự có mặt của một mục i_{A_i} trong một giao tác *ts* có nghĩa là bộ *t* và bộ *s* có cùng giá trị trong A_i. Trên cơ sở này, người ta quan niệm một PTH xấp xỉ như là một LKH như sau:

Định nghĩa [14]

Cho X, Y ⊂ R sao cho X ∩ Y = ϕ. Khi đó một PTH xấp xỉ $X \rightarrow Y$ *trên quan hệ R là một LKH* $I_X \Rightarrow I_Y$ *trên cơ sở giao tác TD.*

Theo cách tiếp cận này, một LKH $i_{A_i} \Rightarrow i_{A_j}$. Trong đó *TD* có ý nghĩa tương tự như PTH $A_i \rightarrow A_j$ trong quan hệ *R*. Sử dụng phương pháp chuyển đổi này, chúng ta có thể tìm kiếm các PTH xấp xỉ trong quan hệ *R* bằng cách tìm kiếm các LKH tương ứng trong *TD*.

Như chúng ta đã biết, độ hỗ trợ của tập mục $I_X(\text{supp}(I_X))$ là tỷ lệ phần trăm các giao tác chứa tập mục. Độ hỗ trợ của một LKH $I_X \Rightarrow I_Y(s(I_X \Rightarrow I_Y))$

là tỷ lệ phần trăm các giao tác chứa tập mục $I_X \cup I_Y$. Độ tin cậy là một độ đo độ chính xác thông thường (cổ điển) đối với các LKH:

$$c(I_X \Rightarrow I_Y) = \frac{s(I_X \Rightarrow I_Y)}{s(I_X)} = \frac{s(I_{XY})}{s(I_X)}$$

Do đó để kiểm tra các PTH xấp xỉ có thỏa mãn, chúng ta có thể sử dụng độ tin cậy và độ hỗ trợ của các LKH tương ứng theo cách như dưới đây.

Định nghĩa [14]

Độ hỗ trợ và độ tin cậy của một PTH xấp xỉ $X \rightarrow Y$ là độ hỗ trợ và độ tin cậy của LKH $I_X \Rightarrow I_Y$ tương ứng. Tức là:

$$s(X \rightarrow Y) = s(I_X \Rightarrow I_Y)$$

$$c(X \rightarrow Y) = c(I_X \Rightarrow I_Y)$$

Theo cách biểu diễn này thì độ hỗ trợ của tập thuộc tính X là:

Supp (X) = *supp* (I_X).

Khi đó tính chất quan trọng của đặc trưng này là mệnh đề sau.

Mệnh đề [14]

$$PTH\ X \rightarrow Y\ là\ hàm \Leftrightarrow c(I_X \Rightarrow I_Y) = 1.$$

Ví dụ

Bảng 2.6. Một quan hệ R

Ma *Xe*	**Diachi *Xe***	**SL*Xe***	**Ten *SP***
1	Ha Noi	4	Man
2	Ha Noi	2	Man
3	Ha Noi	4	Sau rieng

Khi chuyển qua tập các giao tác ta có:

$$I_U = \{i_{MaXe}, i_{DiaChiXe}, i_{SoluongXe}, i_{TenSP}\}.$$

Trong tập giao tác *TD*, mỗi hàng mô tả một giao tác và mỗi cột là một mục. Giá trị 1 (tương ứng 0) trong một ô nghĩa là tập mục đó có (tương ứng

không) trong giao tác. Ta được bảng như sau:

Bảng 2.7.Tập các giao tác TD của R

Ts	i_{MaXe}	$i_{DiaChiXe}$	i_{SLXe}	i_{TenSP}
(1,1)	1	1	1	1
(1,2)	0	1	0	1
(1,3)	0	1	0	1
(2,1)	0	1	0	1
(2,2)	1	1	1	1
(2,3)	0	1	1	1
(3,1)	0	1	0	1
(3,2)	0	1	1	1
(3,3)	1	1	1	1

Như vậy, ta có xác định được độ hỗ trợ và độ tin cậy của LKH trong *TD* và các PTH xấp xỉ tương ứng trong *R*.

Bảng 2.8. Một số LKH trong TD tương ứng với PTH xấp xỉ trong R

LKH	**Độ tin cậy**	**Độ hỗ trợ**	**PTH xấp xỉ**
$\{T_{DiachiXe}\} \Rightarrow \{T_{SLXe}\}$	5/9	5/9	$\{DiachiXe\} \rightarrow \{SLXe\}$
$\{T_{DiachiXe}, T_{SLXe}\} \Rightarrow \{T_{TenSP}\}$	3/5	1/3	$\{DiachiXe, SLXe\} \rightarrow \{TenSP\}$

2.2.3. Độ hỗ trợ của PTH xấp xỉ và tính không tầm thường

Một PTH xấp xỉ là tầm thường trong một quan hệ r khi không có một trường hợp ngoại lệ nào dữ liệu xảy ra [14]. Đặc biệt không có cặp bộ nào trong quan hệ có giá trị giống nhau trên tập thuộc tính X trong PTH $X \rightarrow Y$. Khi đó số các giá trị khác nhau của X trong R tiệm cận với n = |r| và do đó số các giá trị khác nhau của $X \cup Y$ trong r cũng tiệm cận với n.

Vấn đề này có thể được giải quyết bằng cách xem xét độ hỗ trợ của X và $X \rightarrow Y$. Độ hỗ trợ của X trong một quan hệ R có thể được tính như sau:

$$\text{supp}(X) = \frac{1}{n^2} \sum_{i=1}^{K} n_{x_i}^2$$

Tương tự, độ hỗ trợ của một PTH xấp xỉ $X \rightarrow Y$ được xác định như sau:

$$s(X \rightarrow Y) = \frac{1}{n^2} \sum_{i=1}^{K} \sum_{j=1}^{M} n_{x_i y_j}^2$$

Rõ ràng miền giá trị của độ hỗ trợ thuộc tính và phụ thuộc nằm trong dãy từ 0 (khi n = 0) đến 1 và giá trị nhỏ nhất cho quan hệ khác rỗng là 1/n. Đây cũng chính là giá trị cho một phụ thuộc hoàn toàn tầm thường.

Như vậy, độ hỗ trợ là một độ đo hợp lý cho tính tầm thường của các PTH xấp xỉ. Do vậy, thủ tục thông thường cho quá trình khai phá LKH với độ hỗ trợ lớn hơn ngưỡng tối thiểu cho phép chúng ta loại bỏ các LKH cũng như các PTH xấp xỉ tầm thường tương ứng với độ hỗ trợ nhỏ hơn ngưỡng. Điều này dẫn đến, chúng ta có thể sử dụng các thuật toán khai phá LKH đã biết. Từ đó có thể cắt tỉa bớt quá trình tìm kiếm, tránh duyệt một số lượng lớn phụ thuộc không đáng quan tâm [15].

Độ tin cậy không phải là một độ đo độ chính xác của một LKH và do đó là PTH xấp xỉ. Từ hạn chế này của độ tin cậy, nhiều độ đo khác được đề xuất. Một trong những độ đo thay thế độ tin cậy với những tính chất tốt, đáp ứng được độ chính xác của các LKH là *hệ số chắc chắn*.

Định nghĩa [15]

Hệ số chắc chắn của LKH $I_X \Rightarrow I_Y$, kí hiệu $cf(I_X \Rightarrow I_Y)$. Là độ đo được xác định như sau:

$$cf(I_X \Rightarrow I_Y) = \begin{cases} \dfrac{c(I_X \Rightarrow I_Y) - \text{supp}(I_Y)}{1 - \text{supp}(I_Y)}, & \text{nếu } c(I_X \Rightarrow I_Y) > \text{supp}(I_Y) \\ \dfrac{c(I_X \Rightarrow I_Y) - \text{supp}(I_Y)}{\text{supp}(I_Y)}, & \text{ngược lại} \end{cases}$$

Trường hợp nếu $\text{supp}(I_Y) = 1$ thì ta quy ước $\text{cf}(I_X \Rightarrow I_Y) = 1$ còn nếu $\text{supp}(I_Y) = 0$ thì ta quy ước $\text{cf}(I_X \Rightarrow I_Y) = -1$.

Khi xét PTH xấp xỉ $X \to Y$ thì hệ số chắc chắn của nó được xác định là $cf(X \to Y) = cf(I_X \Rightarrow I_Y)$.

Hệ số chắc chắn được hiểu như một biến thể độ đo của xác suất mà hệ quả trong một giao tác khi chúng ta xem xét chi tiết những giao tác là tiền đề. Đặc biệt hơn, hệ số chắc chắn đo sự giảm xác xuất mà hệ quả không ở trong một giao tác. Từ quan điểm xác xuất và phạm vi của các PTH xấp xỉ, giá trị tuyệt đối của hệ số chắc chắn dương (tương ứng âm) đo phần trăm giảm xác xuất mà hai bộ không bằng nhau (tương ứng bằng) trong *Y*, khi chúng ta biết chúng bằng nhau trên *X*.

Người ta chứng minh được độ đo $cf(I_X \Rightarrow I_Y)$ là thỏa tính chất của một độ đo chính xác.

Mệnh đề [15]

Hệ số chắc chắn có các tính chất sau:

Tính chất 1: Nếu X và Y là độc lập thống kê thì $cf(X \to Y) = 0$

Tính chất 2:

(1) $cf(I_X \Rightarrow I_Y) \le c(I_X \Rightarrow I_Y)$

(2) $cf(I_X \Rightarrow I_Y) = c(I_X \Rightarrow I_Y)$ nếu và chỉ nếu $cf(I_X \Rightarrow I_Y) = 1$ và $\text{supp}(I_Y) < 1$.

Tính chất 3: PTH xấp xỉ $X \to Y$ *là hàm* $\Leftrightarrow cf(I_X \Rightarrow I_Y) = 1$

2.2.4. Định nghĩa PTH xấp xỉ mạnh [14]

PTH xấp xỉ mạnh là phụ thuộc hàm với *hệ số chắc chắn* và *độ hỗ trợ* tương ứng lớn hơn hai ngưỡng tối thiểu *minCF* và *minsupp*. Ở đây *minCF* và *minsupp* được cho bởi NSD.

Lưu ý rằng, ở đây chúng ta chỉ quan tâm đến việc tìm kiếm các luật với

hệ số chắc chắn dương. Bởi vì các PTH xấp xỉ là sự kết hợp dương.

2.2.5. Biểu diễn độ đo, độ hỗ trợ, độ chính xác qua lý thuyết PTH xấp xỉ

Các độ đo của PTH xấp xỉ cần được xác định rõ ràng. Bởi vì trước quá trình khai phá chúng ta phải chọn các ngưỡng *minCF* và *minsupp* và khi các PTH xấp xỉ thu được chúng ta cần hiểu chúng tốt như thế nào. Trường hợp nếu chúng ta xem PTH xấp xỉ $X \to Y$ như là một LKH $I_X \Rightarrow I_Y$ trong *TD* thì các biểu diễn của độ hỗ trợ và độ chính xác là rõ ràng. Tuy nhiên, chúng ta xem các PTH xấp xỉ trong quan hệ *R* và muốn biểu diễn các độ đo này trong *R*.

Chúng ta sẽ biểu diễn độ hỗ trợ và độ chính xác của một PTH xấp xỉ qua độ hỗ trợ và độ chính xác của các LKH. Các LKH trong lý thuyết của một PTH xấp xỉ là được định nghĩa trên quan hệ và liên quan đến sự hiện diện của các giá trị của các thuộc tính trong các bộ. Nghĩa là các mục là các cặp (thuộc tính, giá trị), chúng ta kí hiệu là [thuộc tính = giá trị] và các giao tác là các bộ.

Định nghĩa [14]

Kí hiệu $Th_{[X \to Y]}$ là một tập các LKH được xác định như sau:

$$Th_{[X \to Y]} = \left\{ Ru_{ij} \mid \exists t \in R,\ \mathrm{t}[X] = x_i \text{ và } \mathrm{t}[Y] = y_j \right\}$$

Trong đó Ru_{ij} là LKH có dạng $[X = x_i] \Rightarrow [Y = y_i]$. Với s_{ij}, c_{ij} và cf_{ij} tương ứng là độ hỗ trợ, độ tin cậy và hệ số chắc chắn của Ru_{ij}.

Các LKH được định nghĩa trên quan hệ *r* theo cách cổ điển. Đó là chúng liên quan đến các sự hiện diện của các mục có dạng (thuộc tính, giá trị) trong các giao tác tương ứng với các bộ. Do đó, chúng khác với các LKH trên tập các giao tác *TD* mà định nghĩa PTH xấp xỉ ở trên.

Ví dụ

Xét quan hệ *r* ở trên. Khi đó chẳng hạn xét PTH xấp xỉ $X \to C$ thì ta có $Th_{[A \to C]}$ sẽ gồm 5 LKH sau:

[A = 0] ⇒ [c = 1] [A = 1] ⇒ [c = 0]

[A = 2] ⇒ [c = 0] [A = 2] ⇒ [c = 2]

[A = 1] ⇒ [c = 2]

Chúng ta sẽ biểu diễn các độ đo PTH xấp xỉ qua các độ đo của LKH.

2.2.5.1. Độ hỗ trợ

Đỗ hỗ trợ của một PTH xấp xỉ "$X \to Y$" có thể thu được từ độ hỗ trợ.

Mệnh đề [14]

$$s(X \to Y) = \frac{1}{n^2} \sum_{Ru_{ij} \in Th_{[X \to Y]}} n_{x_i y_j}^2 = \sum_{Ru_{ij} \in Th_{[X \to Y]}} s_{ij}^2$$

Ví dụ

Xét quan hệ *R* ở ví dụ trên và xét PTH xấp xỉ "$A \to C$". Khi đó ta có độ hỗ trợ của các LKH trong $Th_{[X \to Y]}$ là:

$$s([A=0] \Rightarrow [c=1]) = \frac{3}{7}, s([A=2] \Rightarrow [c=0]) = \frac{1}{7},$$

$$s([A=1] \Rightarrow [c=2]) = \frac{1}{7}, s([A=1] \Rightarrow [c=0]) = \frac{1}{7},$$

$$s([A=2] \Rightarrow [c=2]) = \frac{1}{7}.$$

Theo mệnh đề trên, suy ra:

$$s(A \to C) = \left(\frac{3}{7}\right)^2 + \left(\frac{1}{7}\right)^2 + \left(\frac{1}{7}\right)^2 + \left(\frac{1}{7}\right)^2 + \left(\frac{1}{7}\right)^2 = \frac{13}{49}$$

Mệnh đề [14]

Nếu mỗi LKH trong $Th_{[X \to Y]}$ *có cùng độ hỗ trợ* s_0 *thì*:

$$|Th_{X \to Y}| = \frac{1}{s_0} \text{ và } s(X \to Y) = s_0.$$

2.2.5.2. Độ tin cậy

Mặc dù chỉ có độ hỗ trợ và hệ số chắc chắn là được sử dụng trong việc tìm kiếm các PTH xấp xỉ. Nhưng trong mục này chúng ta vẫn khảo sát độ tin cậy, vì các hệ số chắc chắn là được xây dựng trên nó [14].

Bổ đề

Xét ánh xạ $f : R \to R$. *Khi đó:*

$$\sum_{i=1}^{K} f\left(n_{x_i}\right) = \sum_{Ru_{ij} \in Th_{[X \to Y]}} c_{ij} f\left(n_{x_i}\right)$$

Khi đó độ tin cậy của một PTH xấp xỉ $"X \to Y"$ có thể thu được từ độ hỗ trợ và độ tin cậy của các LKH trong $Th_{[X \to Y]}$.

Mệnh đề

$$\frac{1}{c(X \to Y)} = \sum_{Ru_{ij} \in Th_{[X \to Y]}} \left(\left(\frac{s_{ij}^2}{\sum_{Ru_{ij} \in Th_{[X \to Y]}} s_{pq}^2} \right) \frac{1}{c_{ij}} \right)$$

Ví dụ: Xét quan hệ *r* ở ví dụ trên và xét PTH xấp xỉ $"A \to C"$. Khi đó ta có độ hỗ trợ và độ tin cậy của các LKH trong $Th_{[X \to Y]}$ là:

$$s([A=0] \Rightarrow [C=1]) = \frac{3}{7}, \quad c([A=0] \Rightarrow [C=1]) = 1,$$

$$s([A=1] \Rightarrow [C=0]) = \frac{1}{7}, \quad c([A=1] \Rightarrow [C=0]) = \frac{1}{2},$$

$$s([A=2] \Rightarrow [C=0]) = \frac{1}{7}, \quad c([A=2] \Rightarrow [C=0]) = \frac{1}{2},$$

$$s([A=2] \Rightarrow [C=2]) = \frac{1}{7}, \quad c([A=2] \Rightarrow [C=2]) = \frac{1}{2},$$

$$s([A=1] \Rightarrow [C=2]) = \frac{1}{7}, \quad c([A=1] \Rightarrow [C=2]) = \frac{1}{2}.$$

Suy ra: $\sum_{Ru_{pq} \in Th_{[A \to C]}} s_{pq}^2 = s(A \to C) = \frac{13}{49}$

Do đó, ta có: $c(A \to C) = \frac{13}{17}$

Hệ quả

$$\frac{1}{c(X \to Y)} = \sum_{Ru_{ij} \in Th_{[X \to Y]}} \left(\left(\frac{s_{ij}^2}{s(X \to Y)} \right) \frac{1}{c_{ij}} \right)$$

Mệnh đề

Nếu mỗi LKH trong $Th_{[X \to Y]}$ *có độ tin cậy* c_0 *thì độ tin cậy của PTH xấp xỉ* $X \to Y$ *là* c_0.

2.2.5.3. Hệ số chắc chắn [14]

Mệnh đề (cung cấp một biểu diễn tương tự khác cho hệ số chắc chắn của một PTH xấp xỉ)

Nếu mỗi LKH trong $Th_{[X \to Y]}$ *có độ tin cậy* c_0 *và hệ số chắc chắn* cf_0 *thì: + Hoặc* $c_0 = 1$, *hoặc mỗi mục* $[Y = y_i]$ *có độ hỗ trợ* $s_1 = \frac{1}{M}$ *trong R. +* $cf(X \to Y) = cf_0$.

Mệnh đề

Nếu mỗi LKH trong $Th_{[X \to Y]}$ *có độ hỗ trợ* s_0, *có độ tin cậy* c_0 *và hệ số chắc chắn* cf_0 *thì khi đó: Mỗi mục* $[X = x_i]$ *có độ hỗ trợ* $s_2 = \frac{1}{K}$ *trong R. và-*$cf_0 \geq 0$.

Các biểu diễn trên giúp chúng ta hiểu ý nghĩa các giá trị của độ hỗ trợ và hệ số chắc chắn của một PTH xấp xỉ theo quan điểm chất lượng của lý thuyết PTH xấp xỉ, và dễ dàng lựa chọn các ngưỡng *minsupp* và *minCF*.

2.2.6. Thuật toán xác định PTH xấp xỉ dựa trên LKH

Chúng ta có thể cải tiến các thuật toán khai phá LKH hiệu quả để thu được các thuật toán khai phá PTH xấp xỉ. Tuy nhiên, áp dụng trực tiếp các thuật toán khám phá LKH thì có một hạn chế: Số các giao tác trong bảng giao tác *TD* là bình phương của số bộ trong quan hệ cho trước *R*. Để giải quyết vấn đề này, cần suy diễn PTH xấp xỉ trên tập n bộ trong *R* thay vì trên tập n^2 giao tác. Cách làm này được phát biểu và chứng minh qua kết quả dưới đây [3]:

Mệnh đề

$$\text{supp}(I_X) = \frac{1}{n}\sum_{i=1}^{K}\sum_{p=1}^{n_{x_i}}(2p-1)$$

Thuật toán

Input: Quan hệ $R(U)=\{t_1, ..., t_n\}$ và $X \subseteq U$.

Output: Độ hỗ trợ $\text{supp}(I_X)$.

1. $\text{supp}(I_X) := 0;$
2. *For* each $(i \in \{1, ..., K\})$
3. $N(X, x_i) := 0;$
4. *For* each $(t \in R)$
5. $N(N, x_i) := N(X, x_i) + 1;$
6. $\text{supp}(I_X) := \text{supp}(I_X) + 2N(X, x_i) - 1;$
7. $\text{supp}(I_X) := \text{supp}(I_X)/n^2.$

Trong thuật toán trên, $N(X, t_i[X])$ là ký hiệu cho số bộ $t_j (j \le i)$

Từ đây, chúng ta có thể áp dụng bất kỳ thuật toán khai phá LKH hiệu quả nào để phát hiện ra LKH mạnh và tương ứng là PTH xấp xỉ mạnh. Khi đó, độ phức tạp của thuật toán suy diễn ra PTH xấp xỉ mạnh chính là độ phức tạp của thuật toán khai phá LKH mạnh hiệu quả đã biết đó. Chẳng hạn, với thuật toán khai phá LKH Aprirori thì thuật toán suy diễn PTH xấp xỉ mạnh được đề xuất như dưới đây.

Thuật toán (ISAD) (*Thuật toán suy diễn PTH xấp xỉ mạnh*)

Input: Quan hệ $R(U) = \{t_1, ..., t_n\}$ với $|U| = m$ và hai ngưỡng tối thiểu *minsupp* và *minCF*.

Output: Tập tất cả các PTH xấp xỉ mạnh *AD* trên quan hệ *R*.

1) $AD := \emptyset;$

2) Sử dung thuật toán Apriori để tìm tất cả các tập mục phổ biến, bắt đầu từ các tập 1 - mục. Tuy nhiên, để tính độ hỗ trợ của các tập mục $I_X\left(X \subseteq U\right)$ chúng ta sử dụng thuật toán *tính độ hỗ trợ của tập mục* I_X .

3) Với mỗi tập phổ biến k – mục ($1 \le k \le$ m). Vận dụng thuật toán *Sinh các LKH mạnh* (GAR) để sinh ra các LKH mạnh tương ứng. Giả sử $I_X \Rightarrow I_Y$, trong đó chúng ta thay ngưỡng tin cậy tối thiểu *minconf* bằng ngưỡng tối thiểu *minCF*.

4) Với mỗi LKH mạnh $I_X \Rightarrow I_Y$ tìm được ở bước (3). Ta suy ra PTH xấp xỉ mạnh tương ứng $X \to Y$: $AD := \text{AD} \cup \{X \to Y\}$

5) Kết luận: *Tập các giao tác AD* là tập tất cả các PTH xấp xỉ mạnh cần tìm.

Nhận xét:

- Thuật toán ISAD cho ra đúng tất cả các PTH mạnh cần tìm.

- Thuật toán đề xuất ISAD này dựa trên ý tưởng trong [14]. Từ đây, có thể thấy nếu thay thuật toán Apriori bằng các thuật toán khai phá LKH hiệu quả khác thì sẽ thu được thuật toán suy diễn PTH xấp xỉ mạnh hiệu quả tương ứng.

2.3. Thuật toán xác định phụ thuộc hàm xấp xỉ dựa trên phủ tối thiểu và lớp tương đương

Khái niệm về lớp tương đương đã được trình bày trong phần 2.1. Ở đây, chúng ta chỉ cần làm rõ khái niệm Phủ tối thiểu.

2.3.1. Khái niệm về Phủ tối thiểu và các mệnh đề liên quan

Cho F và G là những tập PTH trên tập thuộc tính R.

Ta nói F phủ G nếu có $G^+ \subseteq F^+$, và nói F và G tương đương nếu $G^+ = F^+$, và kí hiệu là F ~ G [1].

Dễ dàng kiểm tra xem F và G có tương đương không. Cần chú ý rằng: nếu mọi PTH trong F đều thuộc G^+ thì mọi PTH trong F^+ đều suy diễn được

từ G^+ (nghĩa là $F^+ \subseteq G^+$). Với PTH $X \to Y \in F$, để kiểm tra $X \to Y \in G^+$, ta sẽ tính X_G^+ và kiểm chứng $Y \subseteq X_G^+$.

Định nghĩa Phủ tối thiểu [16]***:***

Tập PTH F được gọi là Phủ *tối thiểu* (có tài liệu gọi là tập PTH tối tiểu) nếu :

(i) Vế phải của mỗi PTH trong F gồm đúng một thuộc tính. (Không dư thừa thuộc tính nào ở vế phải).

(ii) Không tồn tại $X \to A$ trong F sao cho $(F \setminus \{X \to A\})$ tương đương với F. (Không thể bỏ đi một PTH nào trong F mà vẫn thu được một tập PTH tương đương với nó).

(iii) Không tồn tại $X \to A$ trong F và tập con thực sự Z của X sao cho $(F \setminus \{X \to A\} \cup \{Z \to A\})$ tương đương với F. (Không thể bỏ đi bất kì một thuộc tính nào ở vế trái của một PTH bất kì trong F mà vẫn thu được một tập PTH tương đương với nó).

Điều kiện thứ nhất bảo đảm rằng mọi PTH trong F không dư thừa thuộc tính nào ở vế phải. Điều kiện thứ hai đảm bảo rằng F không có PTH nào dư thừa. Điều kiện cuối cùng bảo đảm rằng mọi PTH trong F không dư thừa thuộc tính nào ở vế trái.

Định lí:

Với mỗi tập PTH F, đều tồn tại một tập PTH tối tiểu F' tương đương với nó.

Ví dụ:

Cho $F = \{A \to B, B \to A, A \to C, C \to A, B \to C\}$. Ta có thể tìm được hai tập phụ thuộc tối tiểu tương đương với F là:

$F_1 = \{A \to B, B \to C, C \to A\}$ loại bỏ từ trái sang phải ở điều kiện (ii)

$F_2 = \{A \to B, B \to A, A \to C, C \to A\}$ loại bỏ từ phải sang trái.

2.3.2. Thuật toán tìm Phủ tối thiểu

Thuật toán: Tìm phủ tối tiểu của tập PTH F trên tập thuộc tính R.

INPUT: s = <F, R>

OUTPUT: G {G là phủ tối thiểu của F}

Bước 1: G= φ. Tách tất cả các PTH f của F thành PTH mà vế phải chỉ có một thuộc tính:

FOR $\forall f \in F, f = X \rightarrow Y$ DO /Y={A_1,A_2,..A_k} với $A_i \in R$

G= (G\{f}) $\cup$ {X$\rightarrow A_i$, $A_i \in Y$}

Bước 2: Loại bỏ những phụ thuộc hàm không đầy đủ:

WHILE $\exists f \in G$ mà f:X$\rightarrow$A, $\exists Z \subset X$:Z$\rightarrow$A

G:=(G\{f})$\cup${Z$\rightarrow$A}

Bước 3: Loại bỏ những PTH dư thừa:

FOR $\forall f \in G$ DO

IF (G\{f})~G THEN G:=G\ {f}

Bước 4: RETURN (G).

Ta có thể diễn giải lại thuật giải này như sau:

Bước 1: Tách tất cả các PTH của F thành PTH mà vế phải chỉ có một thuộc tính.

Ví dụ:

AB$\rightarrow$CD được tách thành AB$\rightarrow$C, AB$\rightarrow$D (luật tách)

Bước 2: Loại bỏ những PTH không đầy đủ. Khi loại bỏ, ta phân biệt hai loại PTH không đầy đủ sau:

Loại 1: PTH mà vế phải là tập con của vế trái (PTH tầm thường)

(loại AB$\rightarrow$B)

Loại 2: Hai PTH có vế phải giống nhau, nếu vế trái của PTH này chứa vế trái của PTH kia thì ta loại ra khỏi F.

Ví dụ, Nếu có ABC$\rightarrow$D và BC$\rightarrow$D thì ta loại ABC$\rightarrow$D khỏi F.

Bước 3: Loại bỏ những PTH dư thừa:

Giả sử hai PTH có vế phải giống nhau: f1 = X $\rightarrow$ A và f2 = Y $\rightarrow$ A, lúc đó nếu có $A \in X_F^+ \setminus \{f1\}$ thì loại f1 ra khỏi F:

Ví dụ:

Cho $F = \{A \rightarrow B, B \rightarrow A, A \rightarrow C, C \rightarrow A, B \rightarrow C\}$. Ta có thể tìm được hai PTH tối tiểu tương đương với F là:

$F_1 = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$; $F_2 = \{A \rightarrow B, B \rightarrow A, A \rightarrow C, C \rightarrow A\}$

2.3.3. Thuật toán khai phá PTH xấp xỉ nhờ phủ tối thiểu và lớp tương đương

Thuật toán tìm PTH xấp xỉ dựa trên một số khái niệm của lý thuyết thiết kế CSDL quan hệ là phủ tối thiểu và lớp tương đương do tác giả Jalal Atoum (Khoa Khoa học Máy tính, Đại học Công nghệ (PSUT), Jordan) đề xuất và công bố năm 2009 [16].

2.3.3.1. Mô tả thuật toán

Thuật toán khai phá PTH xấp xỉ từ những CSDL lớn được trình bày dưới đây dựa trên độ đo xấp xỉ g_3. Trong đó có sử dụng một số khái niệm của lý thuyết thiết kế CSDL quan hệ, cụ thể là các khái niệm phủ tối tiểu và những lớp tương đương (Mining Approximate Functional Dependencies from Databases based on Minimal Cover and Equivalenc **C**lasses viết tắt là "AFDMCEC"). Thuật toán này làm giảm bớt số các thuộc tính và các PTH xấp xỉ cần kiểm tra bởi có kết hợp với một vài khái niệm từ lý thuyết thiết kế CSDL quan hệ.

Những khái niệm đầu tiên bao gồm việc tăng trưởng tính phủ tối tiểu của các PTH xấp xỉ trong mỗi giai đoạn phát hiện những PTH này. Mục đính của cách làm này là để giảm tối thiểu số các PTH xấp xỉ cần kiểm tra. Trong khi đó khái niệm thứ hai bao gồm việc tính toán về sự tương đương của các thuộc tính dựa trên bao đóng không tầm thường của chúng. Đối với mỗi cặp thuộc tính có cùng bao đóng, thuật toán loại đi một trong chúng khỏi tập hợp

các thuộc tính ứng viên. Như vậy thuật toán cũng xem hai thuộc tính này là tương đương xấp xỉ. *Hai thuộc tính tương đương xấp xỉ được ký hiệu bằng "$\leftrightarrow$".* Điều này sẽ làm giảm bớt số thuộc tính cần phải kiểm tra trong mỗi giai đoạn của thuật toán AFDMCEC [8].

2.3.3.2. Thủ tục chính của thuật toán AFDMCEC

Thuật toán khai phá PTH xấp xỉ sử dụng phủ tối thiểu và lớp tương đương

Input: Bảng dữ liệu D (quan hệ r) và những thuộc tính $X_1, X_2, ..., X_n$ của nó

ε: Ngưỡng sai số, $0 \leq \varepsilon \leq 1$

Output: Tập PTH xấp xỉ tối tiểu **MinimalApproximate_FDSet**, tập các ứng viên cho mức tiếp theo, **EQ_Set**

1. Bước khởi tạo

Set $R=(X_1, X_2, ..., X_n)$

Nrows = Số hàng trong r

Set FD_Set = $\varnothing$

Approximate_FDSet = $\varnothing$

Set EQ_Set = $\varnothing$

Set Candidate_Set = $\{X_1, X_2, ..., X_n\}$

2. While Candidate_Set $\neq \varnothing$ Do

For all $X_i \in$ Candidate_Set Do

Approximate_FDSet = ComputeMinimaiApproximate_FD(X_i) GenerateNextLevelCandidates(Candidate_Set)

3. Display ApproximateFD_Set

Thủ tục chính của thuật toán AFDMCEC gọi thủ tục tính toán PTH xấp xỉ tối tiểu ComputeMinimalApproximate_FD(X_i) cho mỗi X_i trong tập ứng viên. Với mỗi thuộc tính $Y, Y \subset R - X_i$, nếu $g_3(X_i \rightarrow Y) \leq \varepsilon$ thì bổ sung

$X_i \rightarrow Y$ vào tập PTH xấp xỉ Approximate_FDSet, và nếu bao đóng xấp xỉ của X_i bằng với bao đóng xấp xỉ của Y thì bổ sung $Y \rightarrow X_i$ vào tập PTH xấp xỉ Approximate_FDSet, bổ sung $X_i \leftrightarrow Y$ vào tập EQ_Set và loại bỏ Y khỏi tập ứng viên Candidate_Set.

Sau đây là các thủ tục con được sử dụng trong chương trình chính:

Thủ tục ComputeMinimalApproximate_FD(X_i)

// Thủ tục tính PTH xấp xỉ tối tiểu

Procedure ComputeMinimalApproximate_FD(X_i)

Max = 0 /Xi→Y

TempList = ∅

For each $Y \subset R - X_i$ do

M= $|\pi_{Xi}|$ // Tập các lớp tương đương của thuộc tính X_i

N= $|\pi_{Xi \cup Y}|$ // Tập các lớp tương đương của thuộc tính $X_i \cup Y$

For all $S \subset N$ do

For all $T \subset M$ do

if $T \subset S$ then

$W = W \cup T$

if Max<Len(T) then Max = Len(W)

Add Max to Templist

For I = 1 to Len(Templist)

J = J + Templist(I)

Result = 1 – J/Nrows

If Result $\leq \varepsilon$ Then

Add $X_i \rightarrow Y$ to ApproximateFD_Set

If (approximate closure(X_i) = approximate closure (Y)) then

Add $Y \rightarrow Xi$ to ApproximateFD_Set

Add Xi to closure [Y]

Add $Xi \leftrightarrow Y$ to EQ_Set

Remove Y from candidate_set

Thủ tục GenerateNextLevelCandidates procedure:

//Thủ tục sinh các ứng viên mức tiếp

Procedure GenerateNextLevelCandidates(CANDIDATE_SET)

For each $Xi \in$ Candidate_Set do

For each $Xj \in$ Candidate_Set do

If $(Xi[1] = Xj[1],...,Xi[k-2] = Xj[k-2]$ and $Xi[k-1]<Xj[k-1])$ then

Set $Xij = Xi$ join Xj

If $\exists$ $Xij \in$ TempList then

Delete Xij

else

Compute the partition π_{Xij} of Xij

2.3.4. Độ phức tạp của thuật toán khai phá PTH xấp xỉ sử dụng phủ tối thiểu và lớp tương đương

Thuật toán AFDMCEC đề xuất sẽ kiểm tra toàn bộ bảng gồm $|r|$ bộ để tìm tất cả những lớp tương đương với độ phức tập thời gian tỷ lệ với $|r|$. Sau đó phần chính của thuật toán AFDMCEC có một vòng lặp lặp lại $|R|$ lần. Vì vậy, phần chính này có độ phức tạp thời gian tỷ lệ với $|R|$. Trong mỗi lần lặp của vòng lặp, có một lần gọi cho mỗi thủ tục sau [8]:

1. ComputeMinimaiApproximate_FD(), mỗi lần gọi thủ tục này thực hiện $|R|$ lần lặp. Trong mỗi lần lặp có một vòng lặp kiểm tra toàn bộ những ứng viên trong mức đó có kích thước $s = 2^{|R|}$. Do đó tổng thời gian của bước này là $s*|R|$

2. GenerateNextLevelCandidates(Candidate_set) thủ tục này thực hiện hai vòng lặp lồng nhau, mỗi vòng lặp với $|R|$ lần lặp, cần một tổng thời gian bằng $|R|^2$. Vì vậy, tổng độ phức tạp thời gian đòi hỏi giải thuật AFDMCEC là:

$$O(|r|+|R|(s|R|+|R|^2)) = O(|r|+s|R|^2+|R|^3)$$

2.4. Thuật toán xây dựng cây quyết định dựa trên phụ thuộc hàm xấp xỉ

2.4.1. Giải thuật chung xây dựng cây quyết định

Có rất nhiều giải thuật xây dựng cây quyết định, nhưng nhìn chung các giải thuật đều dựa trên chiến lược chia để trị và từ trên xuống. Xuất phát từ một tập các bộ để huấn luyện, với các nhãn lớp. Tập huấn luyện được phân chia đệ quy thành các tập con khi cây được xây dựng. Giải thuật chung nhất để xây dựng cây quyết định được đặc tả như sau [2]:

Generate_Decision_Tree (TraningSet D, attribute_list)

1. Tạo một nút N
2. **If** các bộ trong D thuộc cùng một lớp C **then**
3. **Return** N là nút lá với nhãn là C
4. **If** attribute_list là rỗng **then**
5. **Return** N là nút lá với nhãn là lớp đa số trong D
6. Áp dụng hàm Attribute_selection_method (TraningSet D, attribute_list) để tìm thuộc tính phân nhánh tốt nhất
7. Gán nhãn cho nút N là thuộc tính phân nhánh vừa chọn được
8. **If** thuộc tính phân nhánh là rời rạc và được phép chia nhiều nhánh **then**
9. attribute_list <- attribute_list- thuộc tính phân nhánh
10. **for each** j thuộc kết quả của hàm lựa chọn thuộc tính

tính Dj là tập các bộ thỏa mãn j

if Dj là rỗng **then**

Thêm 1 nút mới với nhãn là lớp đa số trong D vào nút N

Else

Thêm 1 nút trả về bởi

Generate_Decision_Tree (Dj, attribute_list) vào nút N

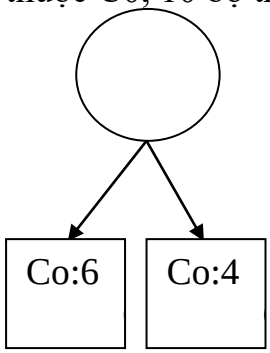
End for

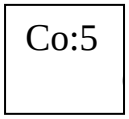
11. Return N

Giải thuật có 2 nhiệm vụ quan trọng là lựa chọ thuộc tính để tiếp tục tiến hành phân nhánh cây quyết định và tiếp tục phát triển thêm nhánh con.

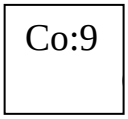
Vấn đề chọn thuộc tính phân nhánh [9]:

1. Việc chọn thuộc tính phân nhánh theo cách ngẫu nhiên (Hunt) hay chọn theo độ đo (ID3, C4.5) đều phải chú ý đến kiểu thuộc tính như giá trị rời rạc (số giá trị lớn phải gộp lại, xác định ngưỡng), liên tục (phải rời rạc hóa), thiếu, số lượng thuộc tính lớn). Phân nhánh theo thuộc tính nào tốt nhất? Cần thước đo độ hỗn tạp. Ví dụ, giả sử trước khi phân nhánh thì đang có 10 bộ thuộc C0, 10 bộ thuộc lớp C1:





Không đồng nhất, Độ hỗn tạp



Gần như đồng nhất, Độ hỗn tạp thấp

Về phương pháp luận, có thể tự đưa ra một độ đo khác nhau, vì đó chỉ là mẹo (heuristic), người ta thường dùng Information gain, Gain ratio, Gini index.

2. Vấn đề thứ hai đặt ra khi chọn thuộc tính phân nhánh là cài đặt vòng lặp với điều kiện j như thế nào, vì với mỗi cách xác định điều kiện j khác nhau sẽ cho một dáng cây khác nhau.

Vấn đề cắt tỉa cây

Khi xây dựng cây, chú ý đến sự xuất hiện của các phần tử ngoại lai và các nhiễu (hiện tượng Overfitting) nên cần cắt tỉa cây (tỉa trước và tỉa sau) [2]:

Cắt tỉa trước nghĩa là trước khi quyết định có tiếp tục phân nhánh không người ta dựa vào một độ đo nào đó. Cách này khá phức tạp, nhưng nếu chọn được một ngưỡng thich hợp thì ta hoàn toàn có thể là cho cây trở nên đơn giản hơn.

Cắt tia sau nghĩa là sau khi hoàn thành việc xây dựng cây đầy đủ thì mới tiến hành cắt tỉa. Một cây con sẽ bị tỉa đi một nhánh của nó và thay bằng một nút lá, với nhãn là lớp đa số trong cây con đã bị loại bỏ. Độ đo đưa ra khi tiến hành cắt tỉa cũng là một đại lượng heuristic. Có nhiều phương pháp khác nhau như *hàm lượng giá* của giải thuật CART (Classification And Regression Tree) hoặc cắt tỉa dựa trên độ dài chuỗi bit dùng để mã hóa cây. Phương pháp cắt tỉa sử dụng hàm lượng giá, ước lượng tỷ lệ lỗi của các nút đưa ra trong giải thuật CART cũng có thể đưa ra áp dụng để cắt tỉa cây sinh ra bởi giải thuật ID3. Tại mỗi nút, thuộc tính dùng để kiểm tra được chọn khi lượng Information Gain lớn nhất.

ID3 cũng có nhược điểm:

1-Chưa giải quyết được trường hợp thuộc tính có giá trị liên tục

2-Khi một thuộc tính có rất nhiều giá trị thì có thể sinh ra 1 cây quyết định không có nhiều ý nghĩa vì các thuộc tính phân lớp khi bị chia nhỏ sẽ khó có độ hỗn tạp thấp.

Chiến thuật cắt tỉa cây quyết định [4]:

Chúng ta thấy rằng việc xây dựng cây bằng cách phát triển nhánh đầy đủ theo chiều sâu để phân lớp hoàn toàn dữ liệu huấn luyện đôi khi gặp khó khăn trong các trường hợp dữ liệu bị nhiễu hoặc bị thiếu, số lượng mẫu quá nhỏ không đủ để đại diện cho một quy luật; tức là tạo ra các nút có số mẫu rất nhỏ. Trong những trường hợp này, nếu thuật toán vẫn cứ phát triển cây thì ta

sẽ dẫn đến một tình huống mà ta gọi là tình trạng "*Over fitting*" trong cây quyết định.

Để giải quyết tình trạng *Over fitting* này người ta sử dụng phương pháp cắt tỉa cây quyết định. Cắt tỉa cây chính là việc làm: tại một nút của cây, nếu sự chính xác của khi không chia tách cao hơn sự chính xác khi được chia tách, khi đó hãy thay thế cây con này bằng một nút lá tương ứng, nhãn của nút lá này được gán là nhãn của lớp đa số (phổ biến) trong tập các mẫu tại nút đó. Có hai chiến lược cắt tỉa cây quyết định:

1. Tiền cắt tỉa

Tiền cắt tỉa nghĩa là sẽ dừng sớm việc phát triển cây trước khi nó vươn đến điểm mà việc phân lớp các mẫu huấn luyện được hoàn thành. Nghĩa là trong quá trình xây dựng cây, một nút có thể sẽ không được tách thêm bước nữa nếu như kết quả của phép tách đó rơi vào một ngưỡng gần như chắc chắn. Nút đó trở thành nút lá và được gán nhãn là nhãn của lớp phổ biến nhất của tập các mẫu tại nút đó.

2. Hậu cắt tỉa

Chiến thuật này ngược với chiến thuật tiền cắt tỉa. Tức là, cây được phát triển đầy đủ sau đó mới thực hiện cắt tỉa (Trong quá trình xây dựng cây cho phép tình trạng *Over fitting* xẩy ra). Nó cho phép phát triển cây đầy đủ sau đó mới cắt tỉa bỏ các nhánh không hợp lý. Nếu một nút mà các cây con của nó bị cắt thì nó sẽ trở thành nút lá và nhãn của lá được gán là nhãn của lớp phổ biến nhất của các con trước đó của nó.

Trong thực tế, hậu cắt tỉa là một phương pháp khá thành công cho việc tìm ra các giả thuyết chính xác cao. Chiến thuật hậu cắt tỉa được tiến hành thông qua việc tính toán lỗi như sau:

Giả sử ta gọi: E(S) là lỗi tĩnh của một nút S; BackUpError(S) là lỗi từ các nút con của S (*Back Up Error*); Error(S) là lỗi của nút S. Các giá trị này được tính như sau:

Error(S) = Min(E(S), BackUpError(S))

E(S) = (N - n + 1) / (N + 2)

Trong đó: N là tổng số mẫu ở nút S, n là số mẫu của lớp phổ biến nhất trong S.

Trong trường hợp tổng quát, nếu thuộc tính lớp có K giá trị (K lớp) thì:

E(S) = (N-n+K-1) / (N+K)

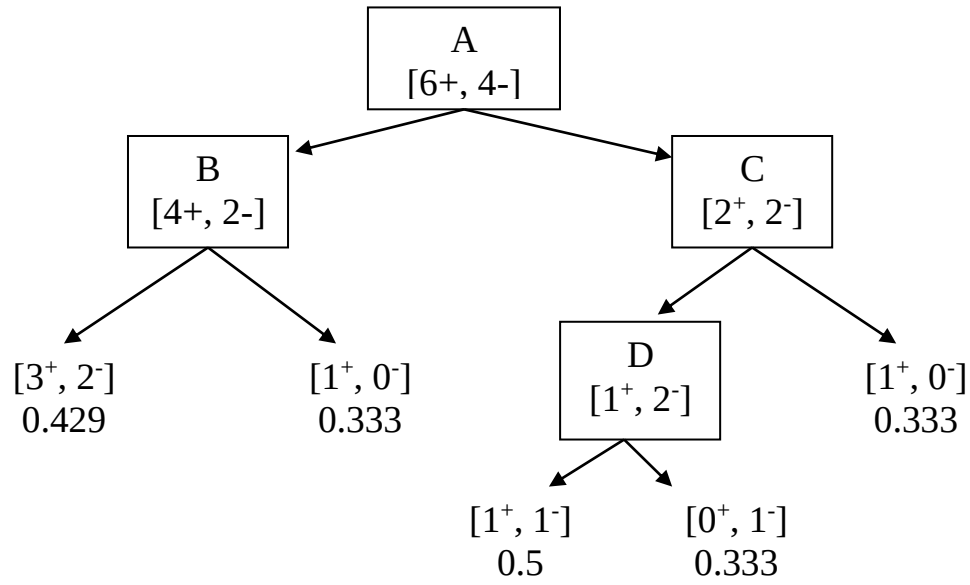
$$\text{BackUpError}(S) = \sum_i P_i \ \text{Error}(S_i)$$

Trong đó: S_i là nút con của S, P_i là tỷ lệ số mẫu trong S_i trên số mẫu trong S.

Như vậy các nút lá sẽ có lỗi $\text{Error}(S_i) = E(S_i)$ do nút lá không có nút con dẫn đến không có lỗi BackUpError. Nếu BackUpError(S) >= E(S) thì chiến thuật hậu cắt tỉa cây quyết định sẽ cắt tại nút S (nghĩa là cắt bỏ các cây con của S).

Ví dụ:

Ta có cây trước khi cắt tỉa như hình dưới đây:



Hình 2.3. Cây trước khi cắt tỉa

Tiến hành tính lỗi cho các nút trong ta có:

Xét nút B ta có: N= 6, n= 4

E(B) = (6-4+1)/(6+2) = 0.375

BackUpError(B) = (5/6)* 0.429 + (1/6)* 0.333 = 0.413

(Trong đó:Eb1=(N-n+k-1)/N+k=(5-3+2-1)/5+2= 0,429 và tương tự Eb2=0,333 như hình trên)

Error(B) = Min(E(B), BackUpError(B)) = Min (0.375, 0.413) = 0.375

Do BackUpError(B) > E(B) nên cắt bỏ các cây con của nút B. Thay B bằng nút lá với lỗi của nút là 0,375.

Xét tại nút D ta có: N= 3, n= 2

E(D) = (3-2+1) /(3+2)= 0.4

BackUpError(D) = (2/3)* 0.5 + (1/3)* 0.333 = 0.444

Error(D) = Min(E(D), BackUpError(D)) = Min (0.4, 0.444) = 0.4

Do BackUpError(D) > E(D) nên cắt bỏ cây con của nút D. Thay D bằng nút lá với lỗi của nút là 0,4.

Xét tại nút C ta có: N=4, n=4

E(C) = (4-2+1) /(4+2)=0.5

BackUpError(D) = (3/4)* 0.4 + (1/4)* 0.333 = 0.383

Error(C) = Min(E(C), BackUpError(C)) = Min (0.5, 0.383) = 0.383

Do BackUpError(C) < E(C) nên không cắt bỏ các cây con của nút C.

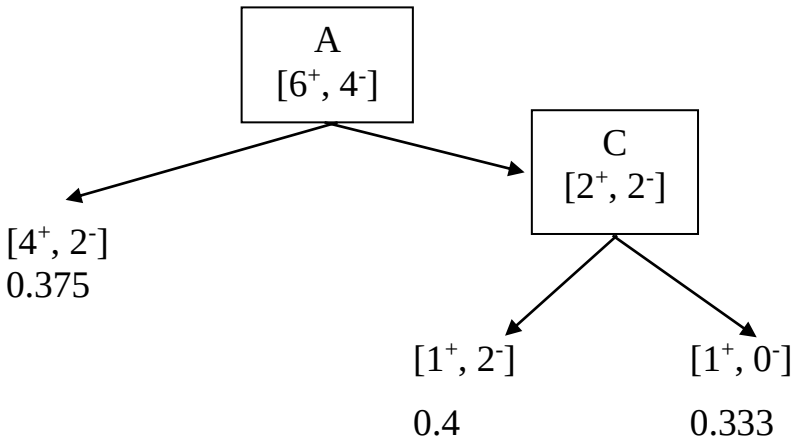
Xét tại nút A ta có: N=10, n=6

E(A) = (10-6+1) /(10+2)=0.417

BackUpError(A) = (6/10)* 0.375 + (4/10)* 0.383 = 0.378

Error(A) = Min(E(A), BackUpError(A)) = Min (0.417, 0.378) = 0.378

Do BackUpError(A) < E(A) nên không cắt bỏ các cây con của nút A. Sau khi thực hiện cắt bỏ các cây con tại nút B và D ta thu được cây như sau:



Hình 2.4. Cây sau khi cắt tỉa

Tóm lại, cắt tỉa cây nhằm tối ưu hoá cây kết quả. Tối ưu về kích cỡ cây và về độ chính xác của việc phân lớp bằng cách cắt bỏ các nhánh không phù hợp.

2.4.2. Giải thuật xây dựng cây quyết định dựa trên tập PTH xấp xỉ phân lớp

(PTH xấp xỉ phân lớp: approximate Class Functional Dependency-CFD)

Cây quyết định hoàn toàn có thể xây dựng được dựa trên các PTH xấp xỉ phát hiện được từ tập huấn luyện, thay vì tìm kiếm trên 1 thuộc tính khi tạo thêm nút mới, thì ta có thể tìm kiếm trên các thuộc tính có liên quan đến nhau, và như vậy có thể tạo ra cây quyết định có nhỏ hơn mà không cần phải tính độ đo "Gain".

Thông thường tập huấn luyện khá lớn, nên chúng ta phải khai phá tập các PTH xấp xỉ một cách tự động và TANE là một trong những thuật toán như thế.

Giải thuật xây dựng cây quyết định dựa trên tập PTH xấp xỉ cũng được phát triển từ giải thuật chung nhất, việc lựa chọn thuộc tính để phân nhánh, thông thường theo cách tiếp cận truyền thống, là dựa trên độ đo thông tin của thuộc tính ấy.

Kwok-Wa Lam, Victor C.S. Lee (2004) (Kwok-Wa Lam, Victor C.S. Lee (2004), *Building Decision Trees Using Functional Dependencies*, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)) đã đưa ra một cách tiếp cận mới. Đó là, mỗi khi tiến hành phân nhánh phát triển cây, sẽ tìm ra một tập thuộc tính có liên quan tới nhau. Chính xác hơn, tại mỗi bước thực hiện, ta sẽ chọn ra các thuộc tính nằm trong vế trái của *PTH xấp xỉ phân lớp (approximate Class Functional Dependency- CFD)* có số lỗi nhỏ nhất để phân nhánh. CFD là các PTH mà có vế phải là thuộc tính phân lớp.

Giải thuật xây dựng cây quyết định dựa trên tập PTH xấp xỉ:

Input: tập các *PTH xấp xỉ phân lớp (CFD)*

Output: cây quyết định.

Function BuildDecisionTree(examples, attributes, default)

IF examples is empty **then**

return default

else if all examples have the same classification **then**

return the classification

else *if* *attributes is empty* ***then***

return *MajorityClass(Examples)*

else

minCDF <- ChooseMinApproxCFD(attributes, examples)

tree <- a new decision tree with root test minCFD

For each value v_i of minCFD **do**

Example$_i$ <- {elements of Examples with minCFD=v_i}

Subtree <- BuildDecisionTree(example$_i$, attributes - minCFD)

MajorityClass(Examples))

Add *a branch to tree with label v_i and subtree*

End do

Return tree

Trong thuật toán trên, hàm MajorityClass dùng để xác định lỗi xấp xỉ của PTH (tức là lớp đa số trong tập mẫu hay nhiễu của dữ liệu mẫu có cùng đặc tính nhưng khác phân lớp) Còn hàm ChooseMinApproxCFD dùng để chọn ra một PTH xấp xỉ phân lớp có số lỗi nhỏ nhất.

2.4.3. Sinh luật từ cây quyết định

Vì việc sử dụng kết quả đã được khai phá là mục đích cuối cùng. Người ta sẽ không nghiên cứu nếu nó không được sử dụng vào đâu cả.

Quy ước, một luật phân lớp r có dạng r = <a,c>, trong đó a là tập điều kiện, là dãy các phép kiểm tra (được định giá là đúng hay sai), còn c là kết quả, là lớp thỏa mãn những trường hợp cụ thể của luật.

Giải thuật sinh luật từ cây quyết định:

Input: cây quyết định T

Output: tập các luật R

1. R:=∅
2. **For each** đường đi từ gốc tới lá trong T **do**
3. a:=true
4. **for each** nút không phải là lá **do**
5. a:=a AND (nhãn của nút kết hợp với nhãn của cạnh đâm ra từ nút này)
6. c:=nhãn của nút lá
7. R:= R UNION r=<a,c>,

2.5. Kết luận chương 2

Trong chương này trình bày một số thuật toán tìm tập tất cả các thuộc tính rút gọn, họ tất cả các tập rút gọn của cây quyết định, xác định các phụ thuộc hàm từ cây quyết định nhất quán, thuật toán xây dựng cây quyết định từ tập phụ thuộc hàm và một số thuật toán về cơ sở dữ liệu quan hệ

Chương 3
CHƯƠNG TRÌNH THỬ NGHIỆM XÂY DỰNG CÂY QUYẾT ĐỊNH CHẨN ĐOÁN BỆNH TẠI BỆNH VIỆN ĐA KHOA TRUNG ƯƠNG THÁI NGUYÊN DỰA TRÊN VIỆC KHAI PHÁ TẬP PTH XẤP XỈ

3.1. Mô tả Bài toán chẩn đoán bệnh cúm tại bệnh viện đa khoa Trung ương Thái Nguyên và yêu cầu chương trình

3.1.1. Giới thiệu về bệnh Cúm

Cúm là một bệnh truyền nhiễm cấp tính lây theo đường hô hấp, do các vi rút cúm A,B,C gây nên. Bệnh khởi phát đột ngột bằng sốt cao, nhức đầu, đau mỏi toàn thân và những dấu hiệu hô hấp, dễ dẫn đến viêm phổi, tỷ lệ tử vong cao.

Trong thế kỷ XX nhiều đại dịch cúm đã xảy ra với số mắc và tỷ lệ tử vong cao. Tuy nhiên lâm sàng bệnh đã được mô tả nhiều thế kỷ trước (A Hirsd, 1881-1886). Năm 1933 W.Smith, C.Andrews, P.Laidpow xác định được vi rút cúm A. Năm 1940 T.Francis và T.Magill phát hiện vi rút cúm B, năm 1949 R.Taylor phát hiện vi rút cúm C. Bằng các kỹ thuật sinh học phân tử, các nhà khoa học đã xác định thủ phạm gây ra vụ đại dịch cúm đầu tiên năm 1918-1919 (cúm TâyBan Nha) là vi rút cúm A chủng H1N1 gây tử vong 20 triệu người, đại dịch cúm châu á năm 1957- 1958 là do cúm A chủng H2N2làm khoảng 1 triệu người tử vong. Cúm Hồng Kông năm 1968- 1969 do cúm A H3N2, cúm Nga năm 1977 do chủng H1N1…Virút cúm A có khả năng thay đổi cấu trúc kháng nguyên. Quá trình lai ghép, tái tổ hợp giữa virut cúm A ở người Virut cúm A ở động vật sẽ tạo thành chủng virut cúm mới. Vì vậy virut cúm A là thủ phạm gây ra các đại dịch, virut cúm B thường gây các vụ dịch. Khu vực, virut cúm C thường gây các dịch tản phát. Cứ khoảng 10-

14 năm lại có một đại dịch cúm xảy ra. Dịch cúm A(H5 N1) lây từ gia cầm sang người xảy ra ở Hồng Kông đang có nguy cơ lan rộng thành đại dịch.

3.1.2. Quy trình chẩn đoán xác định bệnh cúm

Bước 1: Khám lâm sàng

Khởi phát đột ngột, sốt cao thời gian sốt 4-7 ngày.

Khởi bệnh:

- *Thời kỳ khởi phát:*

Thường đột ngột bằng sốt cao 39 - 40^0C, kèm theo rét run, nhức đầu choáng váng, buồn nôn và đau mỏi toàn thân, mệt mỏi không muốn làm việc.

- *Thời kỳ toàn phát:*

+ Hội chứng nhiễm khuẩn nhiễm độc sốt cao liên tục 39 - 40^0C, thời gian sốt 4 - 7 ngày, khi hết sốt nhiệt độ giảm nhanh. Một số bệnh nhân sốt kiểu "V cúm " đang sốt cao nhiệt độ tụt xuống ngay sau đó lại tăng lên rồi mới hạ xuống lần thứ 2.

+ Bệnh nhân mệt mỏi nhiều ăn ngủ kém, môi khô lưỡi bẩn, mạch nhanh huyết áp dao động, nước tiểu vàng.

+ Bạch cầu máu ngoại vi số lượng không tăng, tỷ lệ bạch cầu Lymphocyte tăng, tốc độ lấy máu không tăng.

+ Hội chứng hô hấp: Các triệu chứng thường gặp là:

- Viêm long đường hô hấp trên: Sổ mũi, hắt hơi, rát họng, ho khan mắt xung huyết, chảy nước mắt, sợ ánh sáng.
- Viêm phế quản cấp viêm phổi: Đau tức ngực, khó thở, ho khạc đờm trắng dính. Khám phổi thấy ran ngáy ran rít, hoặc một số ran ẩm nhỏ hạt.
- Viêm thanh hầu và khí quản: Bệnh nhân khàn tiếng, ho khan.
- X quang phổi: Thường không phản ánh được dấu hiệu lâm sàng ở phổi.

+ Triệu chứng khác:

- Đau đầu liên tục, đau nhiều ở vùng thái dương, vùng trán, đôi khi dội lên từng cơn kèm theo hoa mắt chóng mặt ù tai.
- Đau mỏi toàn thân đau cơ bắp và khớp, đau dọc sống lưng, đau ngang thắt lưng, xoa bóp cơ khớp thì đỡ đau.

Dựa vào các triệu chứng bệnh nêu trên. Nếu hết giai đoạn khám lâm sàng này, bác sĩ không có nghi ngờ gì về bệnh cúm thì sẽ đưa ra câu trả lời phủ định bệnh cúm, có thể gợi ý khả năng bệnh nhân mắc một bệnh khác. Bệnh nhân sẽ được khuyên là nên quay lại nếu bệnh nặng hơn mà không rõ căn nguyên.

Bước 2: Làm xét nghiệm

Số lượng bạch cầu máu ngoại vi bình thường hoặc giảm, Lymphocyte tăng.

Để chẩn đoán xác định mầm bệnh phải dựa vào các xét nghiệm đặc hiệu.

Phản ứng Hirst: Là phản ứng huyết thanh dựa trên nguyên lý kỹ thuật ức chế ngưng kết hồng cầu (HI). Lấy máu 2 lần cách nhau 7-10 ngày lần đầu lấy càng sớm càng tốt. Kết quả dương tính khi hiệu giá kháng thể đạt 1/1280 hoặc hiệu giá kháng thể lần 2 tăng gấp 4 lần trở lên.

Phản ứng kết hợp bổ thể.

Phản ứng miễn dịch huỳnh quang: Cho phép chẩn đoán sớm, kết quả chính xác tỷ lệ (+) 60- 70% sau 3-4 giờ.

Phân lập vi rút: Có giá trị chẩn đoán quyết định. Lấy dịch mũi họng, lấy máu, cấy trên tổ chức phôi gà.

Các kỹ thuật xét nghiệm: Elisa, Mac- Elisa, PCR, RT- PCR, kính hiển vi điện tử…được áp dụng để xác định các chủng virut cúm đặc biệt khi có các typ mới xuất hiện.

Bước 3: Điều trị

Nguyên tắc điều trị

Bệnh nhân cúm thể thông thường

Cách ly nghỉ ngơi tại giường cho tới khi hết sốt đề phòng các biến chứng. Ăn lỏng đủ dinh dưỡng, uống đủ nước, tăng cường các loại sinh tố.

Cho bệnh nhân thuốc an thần: Seduxen, rotunda… thuốc giảm ho long đờm, sirocodein, tecpincodein.

Kháng sinh chỉ dùng trong trường hợp bội nhiễm vi khuẩn.

Bệnh nhân cúm thể nặng (ác tính), nhiều virut cúm H5 N1

Bệnh nhân nghi ngờ phải cách ly.

Dùng thuốc kháng virut càng sớm càng tốt, ngay từ những ngày đầu của bệnh.

Hồi sức chống suy hô hấp là cơ bản.

Điều trị bội nhiễm, biến chứng suy đa phủ tạng.

Điều trị nguyên nhân

Thuốc kháng virut: Chỉ định cho những trường hợp nặng.

Tamiflu (Oseltamivir)

Trẻ em từ 1- 13 tuổi: dùng dung dịch uống tuỳ theo trọng lượng cơ thể.

< 15kg : 30mg x 2 lần/ ngày x 7 ngày.

16- 23kg : 45mg x 2 lần/ ngày x 7 ngày.

24- 40kg : 60mg x 2 lần/ ngày x7 ngày.

Người lớn và trẻ em trên 13 tuổi: 75mg x 2 lần/ ngày x 7 ngày.

Cần theo dõi chức năng gan, thận để điều chỉnh cho phù hợp.

Amatadine

1-9 tuổi : 50mg x 2lần/ ngày x 7 ngày.

> 9 tuổi : 100mg x 2 lần/ ngày x 7 ngày.

Ribavirin viên 400mg

1- 9 tuổi : 1 viên x 3 lần/ ngày x 7 ngày.

> 9 tuổi : 2- 3 viên x 3 lần/ ngày x 7 ngày.

Gammaglobulin chống cúm lấy từ huyết thanh người cho máu

Người lớn: 1- 6ml tiêm bắp thịt một lần.

Trẻ em: 1- 3ml tiêm bắp thịt 1-2 lần.

Huyết thanh khô chống cúm của Nga dạng bột phun vào mũi 1- 2 lần

InTerferon: Để bảo vệ những tế bào chưa bị virut phá huỷ.

Điều trị theo cơ chế bệnh sinh

Điều trị suy hô hấp cấp

Thở ôxy 1- 5 lít/phút để SPO2 > 90%.

Thở ôxy cao áp: Khi thở ôxy qua mũi không cải thiện được tình trạng giảm ôxy máu bắt đầu cho thở với CPAP = 5 cm H2O, sau đó điều chỉnh mức CPAP theo tình trạng bệnh nhân với mức thay đổi 1 cm H2O để duy trì SPO2 > 90%. Mức CPAP tối đa có thể đạt tới 10m H2O.

Thông khí nhân tạo khi 2 biện pháp trên không cải thiện được tình trạng hô hấp.

Truyền dịch bù nước điện giải: Trung bình 1200 - 1500ml/ ngày cho bệnh nhân là người lớn, chú ý tránh phù phổi.

Trợ tim mạch, chống sốc.

Cocticoid: Có thể dùng các thuốc.

Methylprenisolon 0,5 - 1,0 mg/kg/ ngày x 7 ngày, tiêm tĩnh mạch chậm.

Hydrocortisone 100mg x 2 lần/ ngày x 7 ngày.

Depersolon 30mg x 2 lần/ ngày x 7 ngày.

Prednisolon 0,5 - 1,0 mg/kg/ ngày x 7 ngày uống.

Kháng sinh: Liều cao phối hợp để phòng và điều trị bội nhiễm vi khuẩn như các thuốc nhóm Cephalosporin, Quinolon…

Bảo đảm chế độ dinh dưỡng và chăm sóc: Cho ăn sữa bột dinh dưỡng qua ống thông dạ dày. Nuôi dưỡng bằng đường tĩnh mạch nếu không ăn được.

Chống loét: cho bệnh nhân nằm đệm nước, xoa bóp thay đổi tư thế.

Chăm sóc hô hấp: Giúp bệnh nhân ho, khạc vỗ rung vùng ngực, hút đờm.

3.2. Tập dữ liệu huấn luyện (input)

CSDL về các bệnh nhân cúm được cung cấp bởi bác sĩ CKII. Hoàng Thị Thư - Trưởng khoa Bệnh Nhiệt Đới - Bệnh viện Đa khoa Trung ương Thái Nguyên. Xây dựng CSDL ban đầu gồm có 50 bệnh nhân. Mỗi bệnh nhân gồm có 12 thuộc tính điều kiện và thuộc tính quyết định (Cúm ={Có, Không}).

Triệu chứng	Giá trị thể hiện
1. Daudau (Đau đầu)	Có/ không
2. Dauco (Đau cơ)	Có/ không
3. Thannhiet (Thân nhiệt)-sốt	cao(có)/ Bìnhthường(không)
4. Onlanh (Ớn lạnh)	Có/ không
5. Chongmat (Chóng mặt)	Có/ không
6. Metmoi (Mệt mỏi)	Có/ không
7. Ho (Ho)	Có/ không
8. Dauhong (Đau họng)	Có/ không
9. Chaynuocmui (Chảy nước mũi)	Có/ không
10. Nghetmui (Nghẹt mũi)	Có/ không
11. Non (Nôn)	Có/ không
12. Tieuchay (Tiêu chảy)	Có/ không
13. **Cum** (Cúm)	Có/ không

3.3. Ứng dụng hai thuật toán 2.3 và 2.4 để xác định tập phụ thuộc hàm xấp xỉ và xây dựng cây quyết định chẩn đoán bệnh

Mục đích của bài toán chẩn đoán bệnh lâm sàng là dựa vào các triệu chứng bệnh nhân mắc phải mà đưa ra được kết luận bệnh nhân có mắc bệnh hay không và bệnh nhân có những triệu chứng nào thì kết luận được bệnh nhân mắc bệnh nên đối với bài toán chẩn đoán bệnh lâm sàng thì ta chủ yếu sử dụng mối quan hệ giữa cây quyết định và phụ thuộc hàm để xây dựng các suy diễn để chẩn đoán trong quá trình chẩn đoán bệnh.

Trong thực tế ta thường dựa vào các triệu chứng để chẩn đoán bệnh và mỗi một bệnh nhân lại có các triệu chứng có thể là rất khác nhau vì vậy việc xây dựng cây quyết định từ tập các phụ thuộc hàm không thật sự cần thiết hay mang nhiều ý nghĩa trong thực tế đối với công tác chẩn đoán bệnh vì nó không tham gia vào quá trình chẩn đoán bệnh. Nếu dựa vào cây quyết định mới xây dựng được để sử dụng trong công tác chẩn đoán đôi khi không đem lại hiệu quả như mong muốn.

Bài toán chẩn đoán bệnh lâm sàng được thiết kế xử lý theo chiều xây dựng các phụ thuộc hàm xấp xỉ từ cây quyết định. Từ CSDL với 12 thuộc tính điều kiện cho một bệnh nhân, suy ra được luật, chẳng hạn như:

If

(Daudau = có) and (Dauco = có) and (Thannhiet = cao) and (Onlanh = có) and (Metmoi = có) and (Ho = có) and (Dauhong = có)

then

Cum = có.

Từ cây quyết định ta dễ dàng xác định được những thuộc tính nào là cần thiết tham gia vào việc quyết định bệnh nhân có mắc bệnh cúm hay không. Dựa trên phủ tối thiểu ta rút ra được tập phụ thuộc hàm tối tiểu thể hiện tri thức về chẩn đoán bệnh.

3.4. Thiết kế chương trình

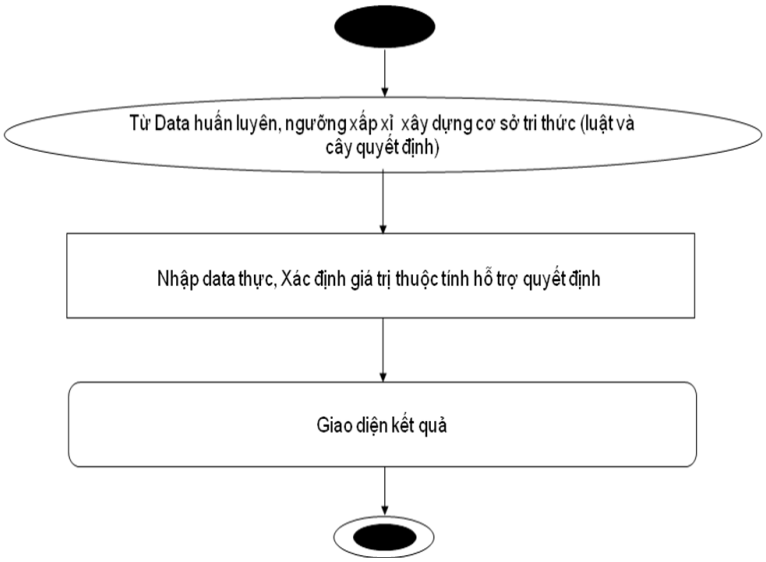
- Bài toán thử nghiệm chẩn đoán lâm sàng bằng phương pháp Phủ tối thiểu để xác định tập PTH tối thiểu gồm các chức năng sau:

Chức năng: Giới thiệu về chương trình và liên kết các mô đun.

Chức năng: Quản lý danh sách các thuộc tính quyết định.

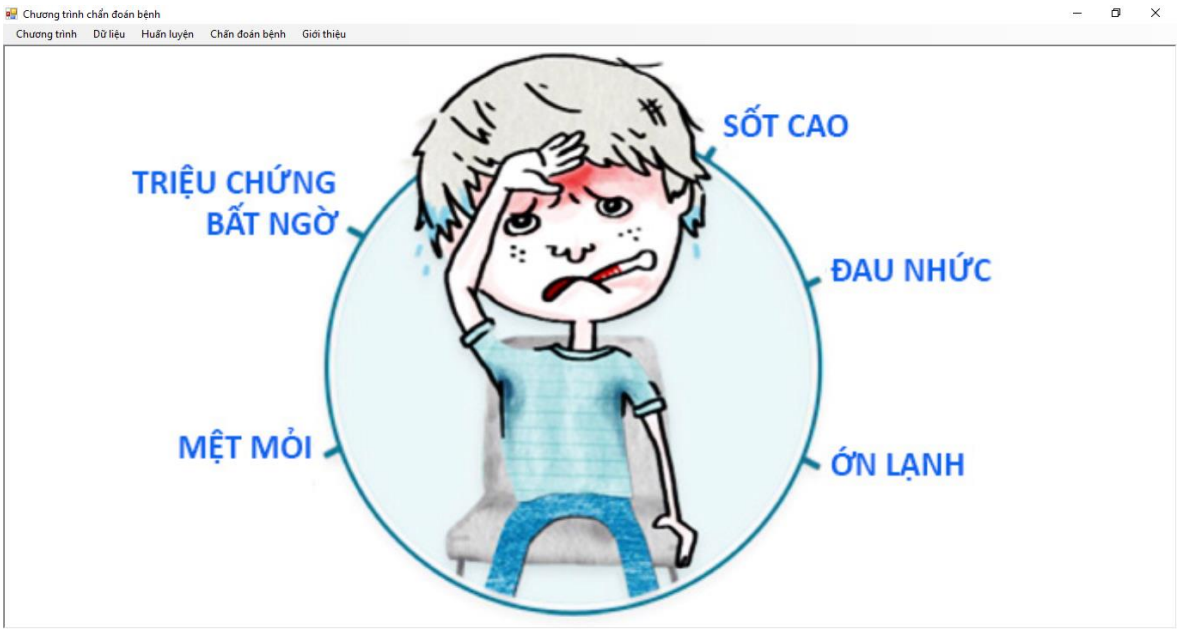
Chức năng: Xác định Phủ tối thiểu và cây quyết định.

Chức năng: Thực hành chẩn đoán bệnh nhân bị cúm.



3.5. Các giao diện chính của chương trình

❖ Giao diện màn hình chính



Chương trình gồm 3 menu chính "Dữ liệu", "Huấn luyện", "Chẩn đoán bênh" và 2 menu khác là "Chương trình" và "Giới thiệu".

- Menu "Dữ liệu": mở giao diện cho nhập và xem dữ liệu huấn luyện để xây dựng cây quyết định chẩn đoán bệnh cúm.

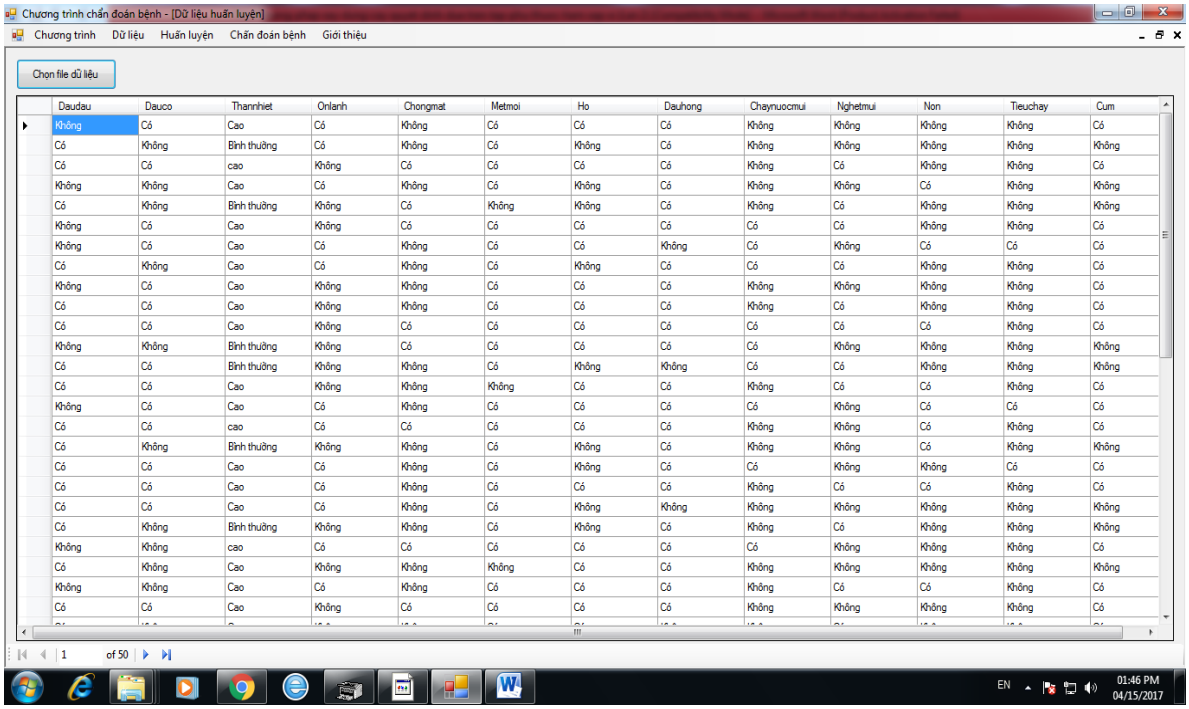
- Menu "Huấn luyện" mở giao diện thực hiện tìm tập phủ tối tiểu các phụ thuộc hàm xấp xỉ và thực hiện xây dựng cây quyết định.

- Menu “Chẩn đoán bệnh cúm” mở giao diện cho phép người dùng chẩn đoán khả năng mắc bệnh cúm của bệnh nhân với các triệu chứng thu được.

- Menu “Chương trình”: chứa nút lệnh tắt chương trình.

- Menu “Giới thiệu”: mở form giới thiệu về chương trình, tác giả.

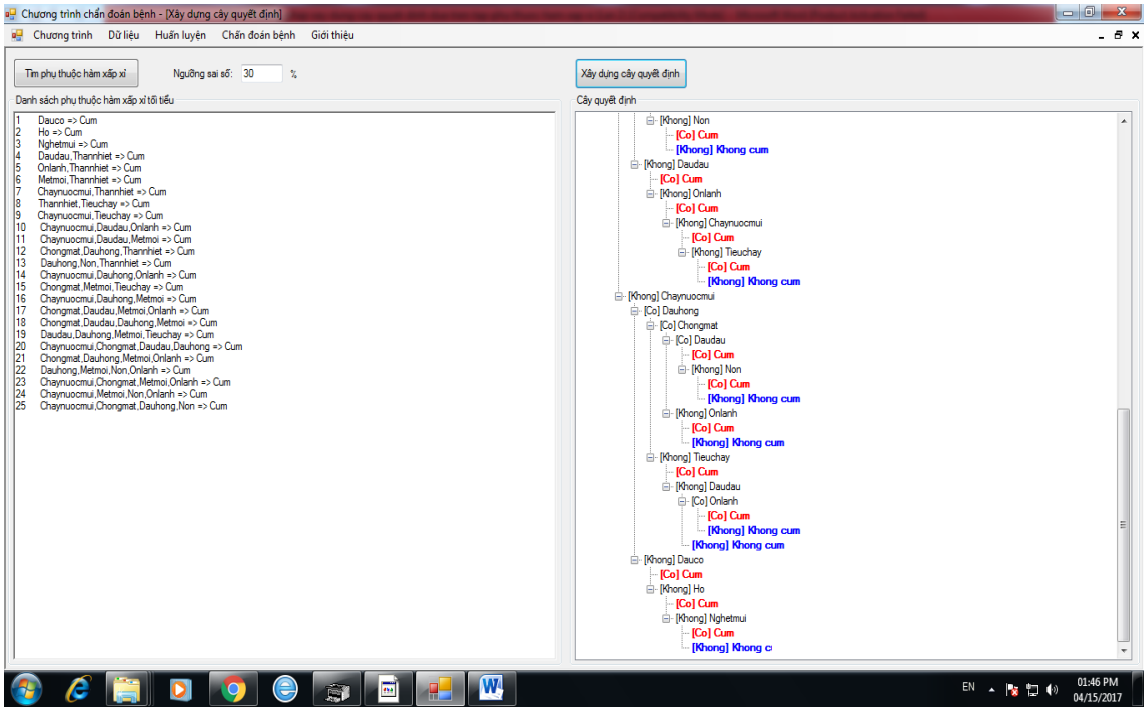
❖ *Giao diện nhập dữ liệu huấn luyện*



- Nút chọn dữ liệu để chọn file excel chứa bảng dữ liệu huấn luyện về bệnh cúm. Bảng dữ liệu gồm 13 cột (12 thuộc tính triệu chứng và cột "Cúm" là kết quả có mắc bệnh cúm hay không của người bệnh). Chương trình cho phép đọc dữ liệu từ file excel 2003 hoặc 2007 trở lên (*.xls và *.xlsx).

- Khi chọn file dữ liệu hợp lệ, dữ liệu được cập nhật vào hệ thống và hiển thị trong giao diện.

❖ *Giao diện huấn luyện*



- Giao diện huấn luyện gồm 2 vùng thể hiện cho 2 thuật toán: tìm tập phủ tối tiểu của phụ thuộc hàm xấp xỉ theo ngưỡng sai số và xây dựng cây quyết định.

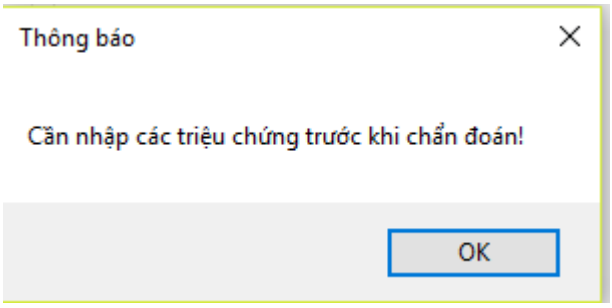
- Tìm tập phủ tối tiểu của phụ thuộc hàm xấp xỉ: người dùng nhập ngưỡng sai số tính theo % ($0<\varepsilon<100$) và nhấn nút "Tìm phụ thuộc hàm xấp xỉ". Chương trình thực hiện tính toán trên tập dữ liệu huấn luyện đã nhập ở bước trước, tìm tất cả các phụ thuộc hàm xấp xỉ thỏa mãn ngưỡng sai số. Sau đó tìm phủ tối tiểu của tập phụ thuộc hàm này và hiển thị lên giao diện bên tay trái.

- Xây dựng cây quyết định: khi nhấn nút "Xây dựng cây quyết định", chương trình thực hiện xây dựng cây quyết định từ dữ liệu huấn luyện và tập phụ thuộc hàm xấp xỉ vừa tìm được. Kết quả được hiển thị bên giao diện bên phải. Các trường thuộc tính được in đậm, các giá trị được in nghiêng.

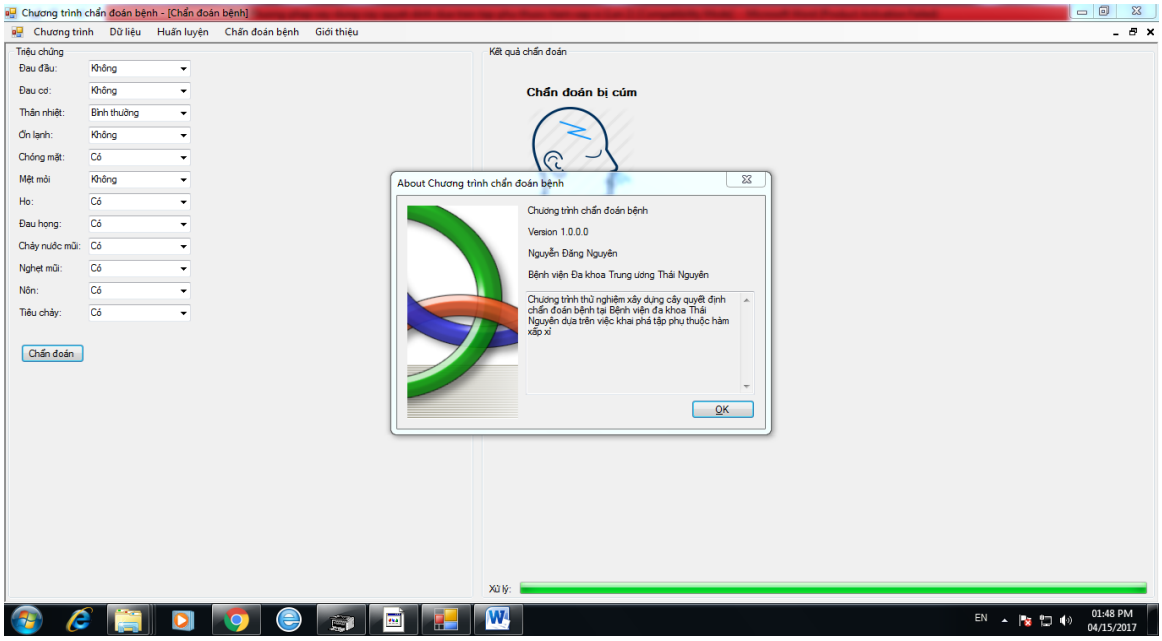
❖ *Giao diện chẩn đoán bệnh*

- Để chẩn đoán bệnh, người dùng nhập các thông tin về triệu chứng của người bệnh. Dựa trên cây quyết định đã xây dựng, chương trình nhanh chóng hiện thị kết quả chẩn đoán người bệnh có bị bệnh cúm hay không. Kết quả được hiển thị dạng text và hình ảnh minh họa.

- Khi nhập thiếu thông tin triệu chứng hoặc chưa xây dựng cây quyết định, chương trình đưa ra cảnh báo.



❖ *Giao diện giới thiệu chương trình*



3.6. Đánh giá kết quả thử nghiệm

Cấu hình máy kiểm thử:

- Proccessor: Intel Core™ i7 – 2670QM 2.20GHz
- Memory: 6G RAM
- Hệ điều hành: Window 10 Pro 64bit

Kết quả kiểm thử: *Đơn vị thời gian: giây*

Lần thử	Phụ thuộc hàm xấp xỉ			Cây quyết định	
	Ngưỡng sai số (%)	Thời gian tính (s)	Số PTH xấp xỉ	Thời gian sinh cây (s)	Cấy quyết định (số node)
1	0	16	47	5	167
2	5	14	119	18	435
3	10	7	166	47	655
4	15	3	145	27	533
5	20	0,5	87	11	345
6	25	0,3	40	4	187
7	30	0,1	25	2	81
8	35	0,07	31	2,5	97
9	40	0,001	15	1	41
10	45	0,001	15	0,7	41

11	50	0,001	15	0,7	41

Nhận xét

Qua thực nghiệm cho thấy, thời tìm tất cả phụ thuộc hàm xấp xỉ phụ thuộc rất nhiều vào ngưỡng sai số và dữ liệu. Tập thực nghiệm có 13 trường, nên tập ứng viên (tổ hợp đa mức ~12,5 tỷ ứng viên cần kiểm tra) là vô cùng lớn. Ngưỡng sai số ảnh hưởng đến tính chất phụ thuộc hàm và khả năng giảm bớt ứng viên. Thời gian tính phụ thuộc hàm giảm dần khi ngưỡng sai số tăng. Trên tập dữ liệu thử nghiệm, với ngưỡng sai số 5-15% có số phụ thuộc hàm xấp xỉ tìm được lớn nhất.

Thời gian sinh cây quyết định phụ thuộc vào số lượng phụ thuộc hàm xấp xỉ bởi phép duyệt đệ quy khi sinh cây quyết định. Số phụ thuộc hàm xấp xỉ càng nhiều, thời gian sinh cây càng lớn.

Phương pháp xây dựng cây quyết định dựa trên phụ thuộc hàm xấp xỉ có thể đạt hiệu quả tốt hơn trong một số trường hợp ngưỡng sai số phù hợp. Một số trường hợp khác (như $\varepsilon < 32$ trong thực nghiệm), thời gian tìm tập phụ thuộc hàm xấp xỉ ảnh hướng lớn đến tổng thời gian thực hiện thuật toán.

3.7. Kết luận chương 3

Nội dung chương 3 trình bày các vấn đề thiết kế và thực hiện chương trình thử nghiệm tìm phủ tối thiểu trên cơ sở tập huấn luyện về cúm ở Bệnh viện đa khoa Thái Nguyên

Kết quả thử nghiệm khai phá dữ liệu trên đã khẳng định khả năng ứng dụng có hiệu quả những vấn đề lý thuyết trong việc xác định phụ thuộc hàm xấp xỉ và xây dựng cây quyết định đã trình bày ở chương 2.

Kết quả ban đầu xác định các phụ thuộc hàm xấp xỉ và cây quyết định do chương trình thực nghiệm tìm được sẽ giúp cho việc hỗ trợ nghiệp vụ và quản lý triển khai chẩn đoán bệnh ở bệnh viện được tốt hơn.

KẾT LUẬN CHUNG

1. Kết quả đạt được trong luận văn

Về mặt lý thuyết:

- Tổng quan được các kiến thức cơ bản về quá trình phát hiện tri thức, khai phá dữ liệu.
- Sưu tập và tổng hợp các thuật toán xác định phụ thuộc hàm xấp xỉ và xây dựng cây quyết định tương ứng.

Về mặt thực nghiệm:

- Thử nghiệm thành công chương trình chẩn đoán bệnh cúm dựa vào các triệu chứng lâm sàng bằng ngôn ngữ lập trình Visual Studio trên cơ sở thuật toán xác định PTH xấp xỉ bằng phương pháp Phủ tối thiểu và lớp tương đương.

2. Hướng phát triển của đề tài

- Trên cơ sở những nghiên cứu đã được trình bày trong luận văn, tiếp tục nghiên cứu rộng hơn một số thuật toán liên quan đến việc xác định phụ thuộc hàm xấp xỉ và xây dựng cây quyết định.
- Hoàn thiện chương trình để có thể ứng dụng tốt tại Bệnh viện Đa khoa Trung ương Thái Nguyên, nơi mà tác giả đang công tác.
- Mở rộng cho nhiều lĩnh vực khác ở Bệnh viện đa khoa Trung ương Thái Nguyên để hỗ trợ chẩn đoán nhiều bệnh khác trên cơ sở sưu tập đầy đủ và chính xác các tập huấn luyện chuyên sâu trong từng lĩnh vực khám và điều trị bệnh.

TÀI LIỆU THAM KHẢO

A. Tiếng Việt:

[1]. Trần Khánh (2015), *Khai phá phụ thuộc hàm xấp xỉ sử dụng phủ tối thiểu và lớp tương đương*, Luận văn Thạc sỹ, ĐH CNTT&TT-Thái Nguyên.

[2]. Lê Thị Hoàng Liên (2007), *Khai phá dữ liệu với cây quyết định*, Luận văn thạc sỹ, Đại học Công nghệ, Đại học quốc gia Hà Nội.

[3]. Phạm Thị Thanh Nga (2015), Khai phá phụ thuộc hàm xấp xỉ sử dụng luật kết hợp và ứng dụng, Luận văn Thạc sỹ, ĐH CNTT&TT-Thái Nguyên.

[4]. Lê Văn Phùng, Quách Xuân Trưởng (2012), *Khai phá dữ liệu*, Nhà xuất bản Thông tin và truyền thông.

[5]. Phạm Hạ Thủy (2006), *Một số vấn đề liên quan đến file dữ liệu trong hệ thống CSDL*, Luận án Tiến sỹ, Viện CNTT, Viện Hàn lâm khoa học và công nghệ Việt Nam.

B. Tiếng Anh:

[6]. Flach, Petter and Savnik, Iztok (2000), *Database Dependency Discovery: a Machine Learning Approach*, Al Comm. Vol.12,no.3, pg 139-160 (Method FDEP).

[7]. Han J. and Kamber M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufman, Academic Press.

[8]. Jalal Atoum (2009), Mining Approximate Function Dependencies from Databases based on minimal cover and equivalent classes, European Journal of Scientific Research ISSN 1450-216X vol.33 No.2, pp.338-346.

[9]. Kwok-Wa Lam, Victor C.S. Lee (2004), *Building Decision Trees Using Functional Dependencies*, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04).

[10]. Lopes Stepane; Pettit, Jean-Marc and Lakhal, Lotfi (2000). *Efficient Discovery of Functional Dependencies and Armstrong Relations.*

Proceeding of ECDT. Lecture Notes in Computer Sciense, vol 1777. (Method DEP-MINER)

[11]. Novelli N., Ciccbetti R (2001), *FUN: An efficient algorithm for mining functional and embedded dependencies*. Proceedings of the 8th International Confẻnce on Databese Theory (ICDT), pp.189-203.

[12]. Ullas Nambiar, Subbarao Kambhampati (2004), *Mining Approximate Functional Dependencies and Concept Similarities to Answer Imprecise Queries*, Seventh International Workshop on the Web and Database, June 17-18, , Paris, France.

[13]. Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka and Hannu Toivonen (1999). TANE: *An efficient algorithm for discovering functional and approximate dependencies*,The Computer Journal, vol. 42, No2.

[14] Sánchez D., Serrano J.M., Blanco I (2008), "Using association ruler to mine for strong approximate dependencies"

[15] Berzal F, Blanco I, Sánchez D, Vila M (2002), "Measuring the accuracy and interest of association rules: A new framework" Intell Data Anal

[16] Jalal Atoum (2009), Mining Approximate Functional Dependencies from Databases Base on Minimal Cover and Equivalent Classes, European Journal of Scientific Research, pp338-346, EuroJournals Publlishing

C. Internet:

[17]. http://www.jaist.ac.jp/~bao/, Ho Tu Bao (2000), Knowledge Discovery and Data Mining Techniques and Pratice.