NGUYỄN THU TRÀ

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

LUẬN VĂN THẠC SỸ KHOA HỌC

NGÀNH: CÔNG NGHỆ THÔNG TIN

NGHIÊN CỨU VÀ ÁP DỤNG MỘT SỐ KỸ THUẬT KHAI PHÁ DỮ LIỆU VỚI CƠ SỞ DỮ LIỆU NGÀNH THUẾ VIỆT NAM

CÔNG NGHỆ THÔNG TIN

2004-2006

NGUYỄN THU TRÀ

Hà Nội
2006

Hà Nội 2006

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT....4
DANH MỤC CÁC BẢNG....5
DANH MỤC CÁC HÌNH VẼ....6
MỞ ĐẦU....8
CHƯƠNG 1. KHAI PHÁ DỮ LIỆU....12
1.1. Tổng quan khai phá dữ liệu....12
1.1.1 Dữ liệu....14
1.1.2 Tiền xử lý dữ liệu....16
1.1.3 Mô hình khai phá dữ liệu....18
1.2. Các chức năng cơ bản khai phá dữ liệu....19
1.2.1 Phân lớp (Classification)....19
1.2.2 Hồi qui....31
1.2.3 Phân nhóm....34
1.2.4 Khai phá luật kết hợp....38
CHƯƠNG 2. MỘT SỐ THUẬT TOÁN KHAI PHÁ DỮ LIỆU....46
2.1. Thuật toán khai phá luật kết hợp....46
2.1.1 Thuật toán Apriori....46
2.1.2 Thuật toán AprioriTid....49
2.1.3 Thuật toán AprioriHybrid....51
2.2. Cải tiến hiệu quả thuật toán Apriori....54
2.2.2 Phương pháp FP-tree....56
2.2.3 Thuật toán PHP....59
2.2.4 Thuật toán PCY....63
2.2.5 Thuật toán PCY nhiều chặng....65
2.3. Thuật toán phân lớp bằng học cây quyết định....67
2.3.1 Các định nghĩa....68
2.3.2 Thuật toán ID3....69
2.3.3 Các mở rộng của C4.5....70
CHƯƠNG 3. ÁP DỤNG KHAI PHÁ TRÊN CSDL NGÀNH THUẾ..72
3.1. CSDL ngành Thuế....72
3.2. Lựa chọn công cụ khai phá....73
3.2.1 Lựa chọn công cụ....73
3.2.2 Oracle Data Mining (ODM)....76
3.2.3 DBMS_DATA_MINING....78
3.3. Mục tiêu khai thác thông tin của ngành Thuế....79

3.4. Thử nghiệm khai phá luật kết hợp 81
3.5. Phân lớp bằng học cây quyết định 91
3.5.1 Phân lớp ĐTNT dựa vào so sánh tỷ suất các năm 93
3.5.2 Phân lớp ĐTNT theo số liệu của một năm 96
CHƯƠNG 4. KẾT LUẬN 102
HƯỚNG NGHIÊN CỨU TIẾP THEO 103
TÀI LIỆU THAM KHẢO 104
PHỤ LỤC 106

DANH MỤC CÁC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

Ký hiệu, chữ viết tắt	Ý nghĩa
Association Rules	Các luật kết hợp
Candidate itemset	Một itemset trong tập C_k được sử dụng để sinh ra các large itemset
C_k	Tập các candidate k-itemset ở giai đoạn thứ k
Confidence	Độ chắc chắn của luật kết hợp = support(X∪Y)/support(X) phản ánh khả năng giao dịch hỗ trợ X thì cũng hỗ trợ Y
CSDL	Cơ sở dữ liệu
DM	Data mining – Khai phá dữ liệu
DW	Data warehouse – Kho dữ liệu
ĐTNT	Đối tượng nộp thuế, chỉ tới các cá nhân hoặc tổ chức nộp thuế
Frequent/large itemset	Một itemset có độ hỗ trợ (support) >= ngưỡng độ hỗ trợ tối thiểu
ID	Identifier
Item	Một phần tử của itemset
Itemset	Tập của các item
k-itemset	Một itemset có độ dài k
L_k	Tập các Large itemset ở giai đoạn thứ k
ODM	Oracle Data Mining – 1 công cụ khai phá dữ liệu
TID	Unique Transaction Identifier
Transaction	Giao dịch

DANH MỤC CÁC BẢNG

Bảng 1.1: CSDL đơn giản gồm các ví dụ huấn luyện 25
Bảng 1.2 Mô hình CSDL giao dịch đơn giản 39
Bảng 2.1 Cơ sở dữ liệu giao dịch T 56
Bảng 2.2 Bảng các sản phẩm khai phá dữ liệu 74

DANH MỤC CÁC HÌNH VẼ

Hình 1.1 Quá trình khám phá tri thức 14
Hình 1.2 Khuôn dạng đơn bản ghi và đa bản ghi 16
Hình 1.3: Cây quyết định đơn giản với các tests trên các thuộc tính X và Y . 22
Hình 1.4: Sự phân lớp một mẫu mới dựa trên mô hình cây quyết định 23
Hình 1.5 Cây quyết định cuối cùng cho CSDL T đã nêu trong bảng 1.1 29
Hình 1.6 Cây quyết định ở dạng giả code cho CSDL T (bảng 1.1)..... 29
Hình 1.7 Hồi qui tuyến tính 32
Hình 1.8 Gộp nhóm theo phương pháp k-means (Điểm đánh dấu + là tâm) 36
Hình 1.9 Phân hoạch vun đống hoặc tách dần 37
Hình 1.10 Bước lặp đầu tiên của thuật toán *Apriori* cho CSDL DB 41
Hình 1.11 Lần lặp thứ 2 của thuật toán *Apriori* cho CSDL DB 42
Hình 1.12 Lần lặp thứ 3 của thuật toán *Apriori* cho CSDL DB 42
Hình 2.1 Thuật toán Apriori..... 46
Hình 2.2 Thuật toán AprioriTid 50
Hình 2.3 Ví dụ..... 51
Hình 2.4: Thời gian thực hiện cho mỗi lần duyệt của Apriori và AprioriTid 52
Hình 2.5: Một ví dụ của cây phân cấp khái niệm cho khai phá các frequent itemsets nhiều mức..... 55
Hình 2.6: FP-tree cho CSDL T trong bảng 2.1 57
Hình 2.7 Thuật toán PHP 62
Hình 2.8 Bộ nhớ với 2 lần duyệt của thuật toán PCY 63
Hình 2.9 Sử dụng bộ nhớ cho các bảng băm nhiều chặng..... 66
Hình 3.1 Công sức cần cho mỗi giai đoạn khai phá dữ liệu 82
Hình 3.2 Các bước khai phá luật kết hợp trên CSDL ngành Thuế..... 83
Hình 3.3 Nhánh cây phân cấp ngành nghề 85
Hình 3.4 Các luật khai phá từ ODM (độ dài luật = 2) 87

Hình 3.5 Các luật khai phá từ ODM (độ dài luật = 3) 89
Hình 3.6 Cây quyết định dùng ODM – Bài toán phân tích tỷ suất..... 95
Hình 3.7 Cây quyết định dùng See5 – Bài toán phân tích tỷ suất 96
Hình 3.8 Cây quyết định dùng ODM – Bài toán xét số liệu một năm..... 99
Hình 3.9 Cây quyết định dùng See5 – Bài toán phân tích trong năm..... 100

MỞ ĐẦU

Thời đại phát triển mạnh của Internet, Intranet, Data warehouse, cùng với sự phát triển nhanh về công nghệ lưu trữ đã tạo điều kiện cho các doanh nghiệp, các tổ chức thu thập và sở hữu được khối lượng thông tin khổng lồ. Hàng triệu CSDL đã được dùng trong quản trị kinh doanh, quản lý chính phủ, quản lý dữ liệu khoa học và nhiều ứng dụng khác. Với khả năng hỗ trợ mạnh của các Hệ quản trị CSDL, các CSDL này càng lớn lên nhanh chóng. Câu "Sự lớn mạnh của các CSDL dẫn đến sự cần thiết phải có các kỹ thuật và các công cụ mới để thực hiện chuyển đổi tự động dữ liệu một cách thông minh thành thông tin và tri thức hữu ích" [10] đã trở thành đặt vấn đề của nhiều bài viết về khai phá thông tin và tri thức từ các CSDL lớn.

Công tác trong ngành Thuế, nơi Công nghệ thông tin được áp dụng vào quản lý Thuế từ những năm 1986, CSDL thông tin liên quan đến các lĩnh vực quản lý Thuế là một CSDL lớn và chắc chắn tiềm ẩn nhiều thông tin quý báu. Với mong muốn bước đầu áp dụng kỹ thuật khai phá dữ liệu trên CSDL ngành Thuế, luận văn đã tập trung nghiên cứu về các kỹ thuật khai phá dữ liệu và tiến hành khai phá thử nghiệm trên CSDL ngành Thuế.

Khả năng mở rộng tri thức có ích ẩn trong dữ liệu để đưa ra những hành động cần thiết dựa trên tri thức đó đang trở nên ngày càng quan trọng trong thế giới cạnh tranh hiện nay. Toàn bộ quá trình dùng các phương pháp luận dựa trên tính toán, bao gồm các kỹ thuật mới để phát hiện ra tri thức từ dữ liệu được gọi là khai phá dữ liệu (data mining). [9]

Khai phá dữ liệu là sự tìm kiếm thông tin mới, có giá trị và không tầm thường trong một khối lượng dữ liệu lớn. Nó là sự phối hợp nỗ lực của con người và máy tính. Các kết quả tốt nhất nhận được bằng việc cân bằng giữa

tri thức của các chuyên gia con người trong việc mô tả các vấn đề và mục đích với khả năng tìm kiếm của máy tính.

Hai mục đích chính của khai phá dữ liệu là để dự đoán (prediction) và mô tả (description). Dự đoán bao gồm việc dùng một vài biến hoặc trường trong tập dữ liệu để dự đoán các giá trị tương lai hoặc chưa biết của các biến cần quan tâm. Còn mô tả tập trung vào việc tìm ra các mẫu mô tả dữ liệu mà con người có thể hiểu được/ biên dịch được. Có thể đưa các hoạt động khai phá dữ liệu vào một trong hai loại sau:

— *Khai phá dữ liệu dự báo*, tạo ra mô hình của hệ thống được mô tả bởi tập dữ liệu cho trước, hoặc

— *Khai phá dữ liệu mô tả*, với việc tạo ra thông tin mới, không tầm thường dựa trên tập dữ liệu có sẵn.

Một số chức năng khai phá dữ liệu chính như:

— Mô tả khái niệm: Mô tả đặc điểm và phân biệt. Tìm ra các đặc điểm khái quát hoá, tổng kết, các đặc điểm khác nhau trong dữ liệu.

— Kết hợp: xem xét về tương quan và quan hệ nhân quả.

— Phân lớp và dự báo (Classification and Prediction): Xác định mô hình mô tả các lớp riêng biệt và dùng cho dự đoán tương lai.

— Phân tích nhóm (Cluster analysis): Chưa biết nhãn lớp, thực hiện nhóm dữ liệu thành các lớp mới dựa trên nguyên tắc cực đại hoá sự tương tự trong cùng lớp và cực tiểu hoá sự khác tương tự giữa các lớp khác nhau.

— Phân tích nhiễu (Outlier analysis): Hữu ích trong việc phát hiện lỗi, phân tích các sự kiện hiếm.

— Phân tích xu hướng và sự phát triển

Khai phá dữ liệu là một trong những lĩnh vực phát triển nhanh nhất trong công nghiệp máy tính. Từ chỗ là một miền quan tâm nhỏ trong khoa học

máy tính và thống kê, nó đã nhanh chóng mở rộng thành một lĩnh vực/ngành của riêng nó. Một trong những lớn mạnh nhất của khai phá dữ liệu là sự ảnh hưởng trong phạm vi rộng của các phương pháp luận và các kỹ thuật được ứng dụng đối với một loạt các bài toán, các lĩnh vực.

Trong kinh doanh, khai phá dữ liệu có thể được dùng để khám phá ra những xu hướng mua sắm mới, kế hoạch cho các chiến lược đầu tư, và phát hiện những sự tiêu dùng không chính đáng từ hệ thống kế toán. Nó có thể giúp cải tiến các chiến dịch marketing để mang lại nhiều hỗ trợ và quan tâm hơn tới khách hàng. Các kỹ thuật khai phá dữ liệu có thể được áp dụng đối với các bài toán thiết kế lại quy trình kinh doanh, trong đó mục đích là để hiểu được các tương tác và quan hệ trong thông lệ kinh doanh và các tổ chức kinh doanh.

Nhiều đơn vị thi hành luật, các đơn vị điều tra đặc biệt, có nhiệm vụ tìm ra các hành động không trung thực và phát hiện ra các xu hướng phạm tội, cũng đã sử dụng khai phá dữ liệu một cách thành công. Các kỹ thuật khai phá dữ liệu cũng có thể được dùng trong các tổ chức tình báo nơi lưu giữ nhiều nguồn dữ liệu lớn liên quan đến các hoạt động, các vấn đề về an ninh quốc gia.

Với mục đích nghiên cứu một số phương pháp khai phá dữ liệu và thử nghiệm khai phá trên CSDL ngành Thuế, luận văn được trình bày với các phần sau:

Chương 1 – Khai phá dữ liệu: Tìm hiểu các chức năng khai phá dữ liệu.

Chương 2 – Một số thuật toán khai phá dữ liệu. Nghiên cứu trên hai kiểu khai phá: Khai phá luật kết hợp - một kỹ thuật thông dụng trong học không giám sát. Phân lớp bằng học cây quyết định - kỹ thuật học có giám sát.

Chương 3 – Áp dụng khai phá trên CSDL ngành Thuế: Thử nghiệm khai phá luật kết hợp và phân lớp trên CSDL ngành Thuế

Chương 4 – Kết luận và những kết quả đạt được

Cuối cùng là một số hướng nghiên cứu tiếp theo.

Em xin chân thành cảm ơn PGS. TS Nguyễn Ngọc Bình đã hướng dẫn và cho em những ý kiến quý báu, chân thành cảm ơn các thầy cô giáo của trường Đại học Bách khoa Hà Nội đã trang bị kiến thức giúp em hoàn thành luận văn này.

CHƯƠNG 1. KHAI PHÁ DỮ LIỆU

1.1. Tổng quan khai phá dữ liệu

Khai phá dữ liệu có nguồn gốc từ các phương pháp riêng biệt, 2 dạng quan trọng nhất là thống kê và học máy. Thống kê có nguồn gốc từ toán học và do đó nhấn mạnh đến độ chính xác toán học, mong muốn thiết lập cái mà có thể nhận ra trên nền toán học trước khi kiểm thử nó trong thực tế. Ngược lại, học máy có nguồn gốc rất nhiều trong thực tiễn tính toán. Điều này dẫn đến sự hướng thực tiễn, sẵn sàng kiểm thử để biết nó thực hiện tốt thế nào mà không cần chờ một chứng minh chính thức. [9]

Có thể có định nghĩa về Khai phá dữ liệu như sau: Khai phá dữ liệu là quá trình phát hiện các mô hình, các tổng kết khác nhau và các giá trị được lấy từ tập dữ liệu cho trước. [9]

Hay, Khai phá dữ liệu là sự thăm dò và phân tích lượng dữ liệu lớn để khám phá từ dữ liệu ra các mẫu hợp lệ, mới lạ, có ích và có thể hiểu được [14]. Hợp lệ là các mẫu đảm bảo tính tổng quát, mới lạ là mẫu chưa được biết trước đó, có ích là có thể dựa vào mẫu đó đưa ra các hành động phù hợp, hiểu được là có thể biên dịch và hiểu thấu đáo các mẫu.

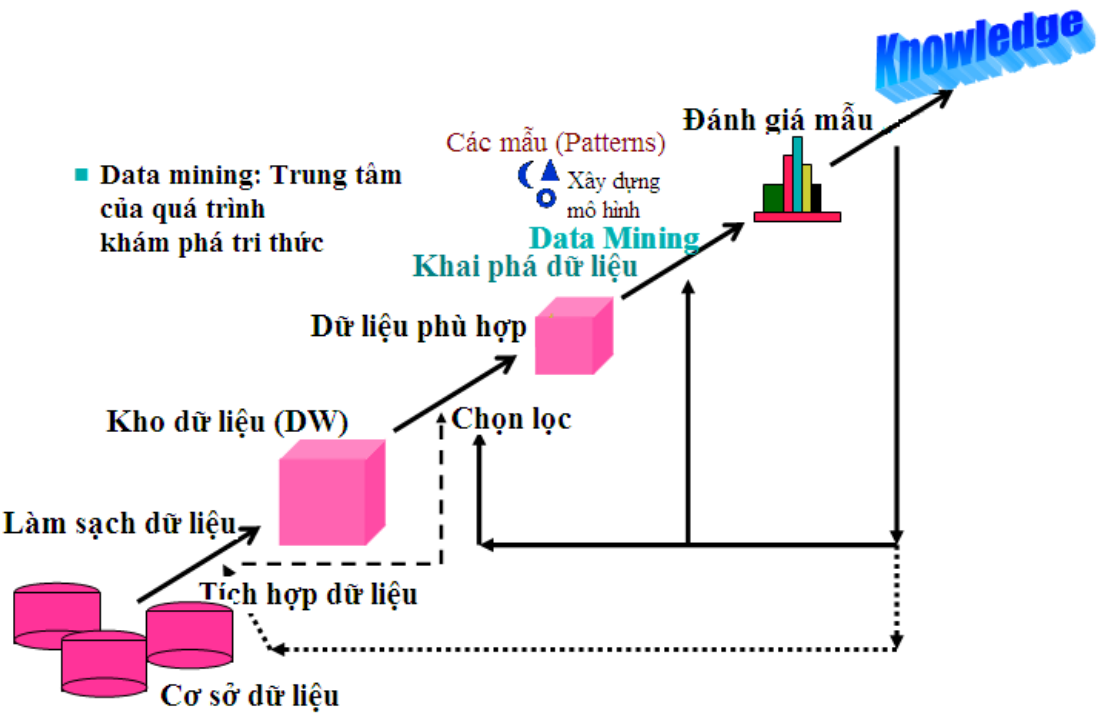
Các kỹ năng phân tích của con người là không đầy đủ do: Kích thước và chiều của dữ liệu; tốc độ tăng trưởng của dữ liệu là rất lớn. Thêm vào đó là những đáp ứng mạnh mẽ của kỹ thuật về khả năng: thu thập dữ liệu, lưu trữ, năng lực tính toán, phần mềm, sự thành thạo về chuyên môn. Ngoài ra còn có môi trường cạnh tranh về dịch vụ, chứ không chỉ cạnh tranh về giá (đối với Ngân hàng, công ty điện thoại, khách sạn, công ty cho thuê …) với câu "Bí quyết của sự thành công là biết những gì mà không ai khác biết" (Aristotle Onassis [14]). Tất cả những điều đó chính là những nguyên nhân thúc đẩy Khai phá dữ liệu phát triển.

Quá trình khám phá tri thức:

Trước tiên, phân biệt giữa các thuật ngữ "mô hình (model)" và "mẫu (pattern)" dùng trong khai phá dữ liệu. Mô hình là một cấu trúc "quy mô lớn", có thể là tổng kết các quan hệ qua nhiều trường hợp (case) (đôi khi là tất cả các trường hợp), trong khi mẫu là một cấu trúc cục bộ, thoả mãn bởi một số ít trường hợp hoặc trong một miền nhỏ của không gian dữ liệu. Trong khai phá dữ liệu, một mẫu đơn giản là một mô hình cục bộ.

Quá trình khám phá tri thức tiến hành theo các bước sau:

1. Xác định bài toán nghiệp vụ: Trước tiên phải tìm hiểu lĩnh vực của ứng dụng nghiệp vụ; Tìm hiểu các tri thức liên quan và các mục đích của ứng dụng.
2. Khai phá dữ liệu
    - Lựa chọn dữ liệu: Xác định các tập dữ liệu đích và các trường liên quan
    - Làm sạch dữ liệu: Xoá bỏ nhiễu, tiền xử lý. Phần việc này có thể chiếm tới 60% công sức.
    - Giảm bớt dữ liệu và chuyển đổi dữ liệu: Tìm ra những đặc trưng hữu dụng, giảm bớt các chiều hoặc các biến, biểu diễn lại các đại lượng bất biến
    - Lựa chọn chức năng khai phá dữ liệu: Tổng kết, phân lớp, Hồi qui, kết hợp, phân nhóm.
    - Lựa chọn thuật toán khai phá.
    - Thực hiện khai phá dữ liệu (***Data Mining***): Tìm kiếm các mẫu quan tâm
    - Đánh giá các mẫu và biểu diễn tri thức



Hình 1.1 Quá trình khám phá tri thức

3. Áp dụng khám phá tri thức
4. Đánh giá và đo đạc
5. Triển khai và tích hợp vào các qui trình nghiệp vụ

1.1.1 Dữ liệu

Do có nhiều kiểu dữ liệu, các CSDL sử dụng trong các ứng dụng cũng khác nhau, nên người dùng luôn mong đợi một hệ thống khai phá dữ liệu có thể điều khiển được tất cả các loại dữ liệu. Thực tế CSDL có sẵn thường là CSDL quan hệ và hệ thống khai phá dữ liệu cũng thực hiện hiệu quả việc khai phá tri thức trên dữ liệu quan hệ. Với những CSDL của ứng dụng chứa các kiểu dữ liệu phức tạp, như dữ liệu hypertext và multimedia, dữ liệu tạm và không gian (spatial), dữ liệu kế thừa (legacy)… thường phải có các hệ thống khai phá dữ liệu riêng biệt xây dựng để khai phá cho các kiểu dữ liệu cụ thể.

Dữ liệu được khai phá có thể là dữ liệu có cấu trúc, hoặc không có cấu trúc. Mỗi bản ghi dữ liệu được coi như một trường hợp hoặc một ví dụ (case/example).

Phân biệt hai kiểu thuộc tính: *phân loại (categorical)* và *số (numerical)*. Các thuộc tính kiểu phân loại là những thuộc tính có các giá trị thuộc vào một số lượng nhỏ các phân loại hoặc các lớp riêng rẽ và giữa chúng không có thứ tự ẩn nào. Nếu chỉ có 2 giá trị, ví dụ là yes và no, hoặc male và female, thuộc tính được coi là binary. Nếu có hơn 2 giá trị, ví dụ, nhỏ, vừa, lớn, rất lớn, thuộc tính được coi là đa lớp (multiclass).

Các thuộc tính số là những thuộc tính lấy các giá trị liên tục, ví dụ, thu nhập hàng năm, hoặc tuổi. Thu nhập hàng năm hoặc tuổi có thể về lý thuyết là bất kỳ một giá trị nào từ 0 tới vô hạn, mặc dù mỗi giá trị thường xuất hiện phù hợp với thực tế. Các thuộc tính số có thể được biến đổi thành categorical: Ví dụ, thu nhập hàng năm có thể được chia thành các loại: thấp, trung bình, cao.

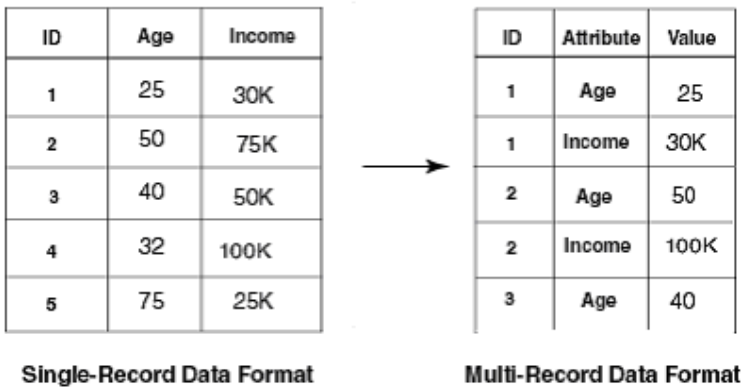
Dữ liệu không có cấu trúc có thể áp dụng các thuật toán khai phá dữ liệu thường là dữ liệu kiểu Text.

Khuôn dạng bảng của dữ liệu có thể thuộc hai loại:

- Dữ liệu dạng đơn bản ghi (còn gọi là kiểu không giao dịch), đây là các bảng dữ liệu quan hệ thông thường.
- Dữ liệu dạng đa bản ghi (còn gọi là kiểu giao dịch), được dùng cho dữ liệu với nhiều thuộc tính.

Ở dạng đơn bản ghi (kiểu không giao dịch), mỗi bản ghi được lưu trữ như 1 dòng trong bảng. Dữ liệu đơn bản ghi không đòi hỏi cung cấp khoá để xác định duy nhất mỗi bản ghi. Nhưng, khoá là cần cho các trường hợp kết hợp (associate) để có kết quả cho học có giám sát.

Trong dạng đa bản ghi (kiểu giao dịch), mỗi trường hợp (case) được lưu trong nhiều bản ghi trong một bảng với các cột: dãy số định danh, tên thuộc tính, giá trị.



Hình 1.2 Khuôn dạng đơn bản ghi và đa bản ghi

1.1.2 Tiền xử lý dữ liệu

Dữ liệu được chọn lọc sẽ phải qua bước tiền xử lý trước khi tiến hành khai phá phát hiện tri thức. Bước thu thập và tiền xử lý dữ liệu là bước rất phức tạp. Để một giải thuật DM thực hiện trên toàn bộ CSDL sẽ rất cồng kềnh, kém hiệu quả. Trong quá trình khai phá dữ liệu, nhiều khi phải thực hiện liên kết/tích hợp dữ liệu từ rất nhiều nguồn khác nhau. Các hệ thống sẵn có được thiết kế với những mục đích và đối tượng phục vụ khác nhau, khi tập hợp dữ liệu từ những hệ thống này để phục vụ khai phá dữ liệu, hiện tượng dư thừa là rất phổ biến, ngoài ra còn có thể xảy ra xung đột gây mấy dữ liệu, dữ liệu không đồng nhất, không chính xác. Rõ ràng yêu cầu chọn lọc và làm sạch dữ liệu là rất cần thiết.

Nếu đầu vào của quá trình khai phá là dữ liệu trong DW thì sẽ rất thuận tiện, vì dữ liệu này đã được làm sạch, nhất quán và có tính chất hướng chủ đề.

Tuy nhiên nhiều khi vẫn phải có thêm một số bước tiền xử lý để đưa dữ liệu về đúng dạng cần thiết.

Ngoài một số xử lý thông thường như: biến đổi, tập hợp dữ liệu từ nhiều nguồn về một kho chung, xử lý để đảm bảo nhất quán dữ liệu (khử các trường hợp lặp, thống nhất cách ký hiệu, chuyển đổi về khuôn dạng thống nhất (đơn vị tiền tệ, ngày tháng..)). Một số xử lý đặc biệt cần chú ý trong bước tiền xử lý dữ liệu:

Xử lý với dữ liệu thiếu (missing data): Thường thì khi khai phá dữ liệu không đòi hỏi NSD phải xử lý các giá trị thiếu bằng cách thức đặc biệt nào. Khi khai phá, thuật toán khai phá sẽ bỏ qua các giá trị thiếu. Tuy nhiên trong một vài trường hợp cần chú ý để đảm bảo thuật toán phân biệt được giữa giá trị có nghĩa ("0") với giá trị trống. (tham khảo trong [11]).

Các giá trị gây nhiễu (Outliers): Một outlier là một giá trị ở xa bên ngoài của miền thông thường trong tập hợp dữ liệu, là giá trị chênh lệch với chuẩn về ý nghĩa. Sự có mặt của outliers có thể có ảnh hưởng đáng kể trong các mô hình khai phá dữ liệu.

Outliers ảnh hưởng đến khai phá dữ liệu trong bước tiền xử lý dữ liệu hoặc là khi nó được thực hiện bởi NSD hoặc tự động trong khi xây dựng mô hình.

Binning: Một vài thuật toán khai phá dữ liệu có thể có lợi nhờ việc binning với cả hai loại dữ liệu number và categorical. Các thuật toán Naive Bayes, Adaptive Bayes Network, Clustering, Attribute Importance, và Association Rules có thể có lợi từ việc binning.

Binning nghĩa là nhóm các giá trị liên quan với nhau, như vậy giảm số lượng các giá trị riêng biệt của một thuộc tính. Có ít hơn các giá trị riêng biệt dẫn đến mô hình gọn nhẹ và xây dựng được nhanh hơn, nhưng nó cũng có thể

dẫn đến việc mất đi độ chính xác [11] (Các phương pháp tính toán ranh giới bin [11]).

1.1.3 Mô hình khai phá dữ liệu

Mô hình khai phá dữ liệu là một mô tả về một khía cạnh cụ thể của một tập dữ liệu. Nó tạo ra các giá trị đầu ra cho tập các giá trị đầu vào.

Ví dụ: Mô hình Hồi qui tuyến tính, mô hình phân lớp, mô hình phân nhóm.

Một mô hình khai phá dữ liệu có thể được mô tả ở 2 mức:

— Mức chức năng (Function level): Mô tả mô hình bằng những thuật ngữ về dự định sử dụng. Ví dụ: Phân lớp, phân nhóm.

— Mức biểu diễn (representation level): Biểu diễn cụ thể một mô hình. Ví dụ: Mô hình log-linear, cây phân lớp, phương pháp láng giềng gần nhất.

Các mô hình khai phá dữ liệu dựa trên 2 kiểu học: có giám sát và không giám sát (đôi khi được nói đến như là học trực tiếp và không trực tiếp – directed and undirected learning) [11].

Các hàm học có giám sát (Supervised learning functions) được sử dụng để dự đoán giá trị. Các hàm học không giám sát được dùng để tìm ra cấu trúc bên trong, các quan hệ hoặc tính giống nhau trong nội dung dữ liệu nhưng không có lớp hay nhãn nào được gán ưu tiên. Ví dụ của các thuật toán học không giám sát gồm phân nhóm k-mean (k-mean clustering) và các luật kết hợp Apriori. Một ví dụ của thuật toán học có giám sát bao gồm Naive Bayes cho phân lớp (classification).

Tương ứng có 2 loại mô hình khai phá dữ liệu:

— Các mô hình dự báo (học có giám sát):

- Phân lớp: nhóm các items thành các lớp riêng biệt và dự đoán một item sẽ thuộc vào lớp nào.
- Hồi qui (Regression): xấp xỉ hàm và dự báo các giá trị liên tục
- Độ quan trọng của thuộc tính: xác định các thuộc tính là quan trọng nhất trong các kết quả dự báo

— Các mô hình mô tả (học không giám sát):

- Phân nhóm (Clustering): Tìm các nhóm tự nhiên trong dữ liệu
- Các mô hình kết hợp (Association models): Phân tích "giỏ hàng"
- Trích chọn đặc trưng (Feature extraction): Tạo các thuộc tính (đặc trưng) mới như là kết hợp của các thuộc tính ban đầu

1.2. Các chức năng cơ bản khai phá dữ liệu

1.2.1 Phân lớp (Classification)

Trong bài toán phân lớp, ta có dữ liệu lịch sử (các ví dụ được gán nhãn - thuộc lớp nào) và các dữ liệu mới chưa được gán nhãn. Mỗi ví dụ được gán nhãn bao gồm nhiều thuộc tính dự báo và một thuộc tính đích (biến phụ thuộc). Giá trị của thuộc tính đích chính là nhãn của lớp. Các ví dụ không được gán nhãn chỉ bao gồm các thuộc tính dự báo. Mục đích của việc phân lớp là xây dựng mô hình dựa vào dữ liệu lịch sử để dự báo chính xác nhãn (lớp) của các ví dụ không gán nhãn. [11]

Nhiệm vụ phân lớp bắt đầu với việc xây dựng dữ liệu (dữ liệu huấn luyện) có các giá trị đích (nhãn lớp) đã biết. Các thuật toán phân lớp khác nhau dùng các kỹ thuật khác nhau cho việc tìm các quan hệ giữa các giá trị của thuộc tính dự báo và các giá trị của thuộc tính đích trong dữ liệu huấn luyện. Những quan hệ này được tổng kết trong mô hình, sau đó được dùng

cho các trường hợp mới với các giá trị đích chưa biết để dự đoán các giá trị đích.

Mô hình phân lớp có thể được dùng trên bộ dữ liệu kiểm thử/dữ liệu đánh giá với mục đích so sánh các giá trị dự báo với các câu trả lời đã biết. Kỹ thuật này được gọi là kiểm tra mô hình, nó đo độ chính xác dự báo của mô hình.

Áp dụng mô hình phân lớp đối với dữ liệu mới được gọi là sử dụng mô hình, và dữ liệu được gọi là dữ liệu sử dụng hay dữ liệu trung tâm (apply data or scoring data). Việc sử dụng dữ liệu thường được gọi là '*scoring the data*'.

Sự phân lớp được dùng trong phân đoạn khách hàng, phân tích tín dụng, và nhiều ứng dụng khác. Ví dụ, công ty thẻ tín dụng muốn dự báo những khách hàng nào sẽ không trả đúng hạn trên các chi trả của họ. Mỗi khách hàng tương ứng với một trường hợp; dữ liệu cho mỗi trường hợp có thể bao gồm một số thuộc tính mô tả thói quen tiêu dùng của khách hàng, thu nhập, các thuộc tính nhân khẩu học,… Đây là những thuộc tính dự báo. Thuộc tính đích chỉ ra có hay không người khách hàng đã vỡ nợ/không trả đúng hạn; như vậy, có hai lớp có khả năng, tương ứng với vỡ nợ hoặc không. Dữ liệu huấn luyện sẽ được dùng để xây dựng mô hình dùng cho dự báo các trường hợp mới sau này (dự báo khách hàng mới có khả năng chi trả nợ không).

Chi phí (Costs):

Trong bài toán phân lớp, có thể cần xác định chi phí bao hàm trong việc tạo ra một quyết định sai lầm. Việc này là quan trọng và cần thiết khi có chênh lệch chi phí lớn giữa các phân lớp sai (misclassification). Ví dụ, bài toán dự báo có hay không một người sẽ trả lời với thư quảng cáo. Đích có 2 phân loại: YES (khách hàng trả lời) và NO (khách hàng không trả lời). Giả sử trả lời tích cực đối với quảng cáo sinh ra $500 và nó trị giá $5 để gửi thư. Nếu

mô hình dự báo YES và giá trị thực tế là YES, giá trị của phân lớp sai là $0. Nếu mô hình dự báo YES và giá trị thực tế là NO, giá trị của phân lớp sai là $5. Nếu mô hình dự báo NO và giá trị thực tế là YES, giá trị của phân lớp sai là $500. Nếu mô hình dự báo NO và giá trị thực là NO, chi phí là $0.

Ma trận chi phí, có chỉ số hàng ương ứng với các giá trị thực; chỉ số cột tương ứng với các giá trị dự báo. Với mỗi cặp chỉ số thực-dự báo, giá trị của ma trận chỉ ra chi phí của sự phân lớp sai.

Một vài thuật toán, như Adaptive Bayes Network, tối ưu ma trận chi phí một cách trực tiếp, sửa đổi mô hình mục đích tạo ra các giải pháp chi phí cực tiểu. Các thuật toán khác, như Naive Bayes (dự báo xác suất), dùng ma trận chi phí trong khi tìm kết quả trên dữ liệu thật để đưa ra giải pháp chi phí ít nhất.

1.2.1.1 Phân lớp - một quá trình hai bước

Bước 1. Xây dựng mô hình (Học)

Xây dựng mô hình bằng cách phân tích tập dữ liệu huấn luyện, sử dụng các thuật toán phân lớp và thể hiện mô hình theo luật phân lớp, cây quyết định hoặc các công thức toán học, mạng nơron…

Bước này còn được coi là bước tạo ra bộ phân lớp (classifier).

Bước 2. Sử dụng mô hình (Phân lớp)

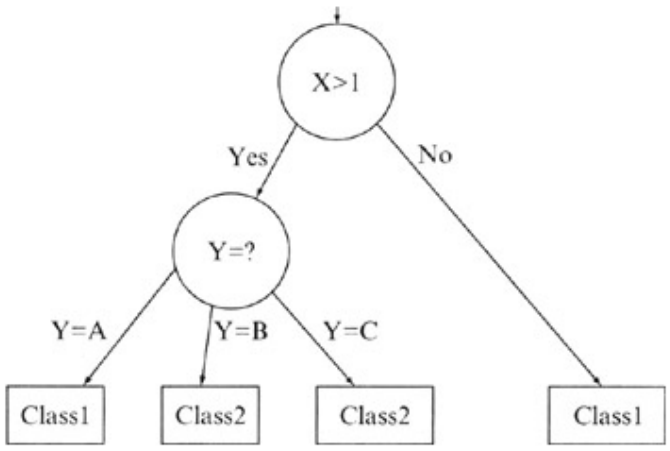
Áp dụng mô hình cho tập dữ liệu kiểm thử với các lớp đã xác định để kiểm tra và đánh giá độ chính xác của mô hình. Nếu độ chính xác là chấp nhận được, mô hình sẽ được sử dụng để phân lớp cho các dữ liệu mới.

Như vậy có 3 tập dữ liệu có cấu trúc và các thuộc tính dự đoán giống nhau: Tập huấn luyện và tập kiểm thử đã biết lớp; Tập mới chưa xác định lớp.

1.2.1.2 Phân lớp bằng học cây quyết định

Cây quyết định

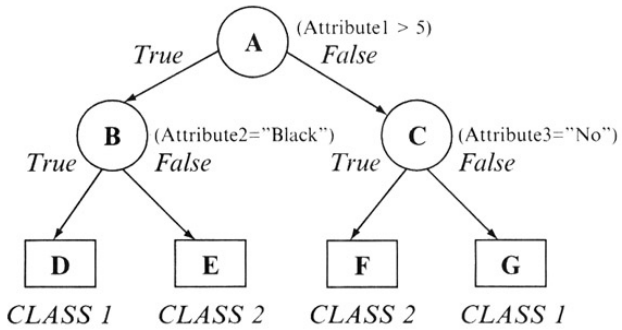
Phương pháp hiệu quả đặc biệt cho việc tạo ra các bộ phân lớp từ dữ liệu là sinh ra cây quyết định. Biểu diễn của cây quyết định là phương pháp logic được sử dụng rộng rãi nhất [9]. Một cây quyết định bao gồm các nodes mà ở đó các thuộc tính được kiểm tra (tested). Các nhánh ra của một node tương ứng với tất cả các kết quả có thể của việc kiểm tra tại node. Ví dụ, cây quyết định đơn giản cho việc phân lớp các mẫu với 2 thuộc tính đầu vào X và Y được cho trong hình 1.3. Tất cả các mẫu với các giá trị đặc trưng X>1 và Y=B thuộc vào Class2, trong khi các mẫu với giá trị X<1 đều thuộc vào Class1, dù Y lấy bất kỳ giá trị nào.



Hình 1.3: Cây quyết định đơn giản với các tests trên các thuộc tính X và Y

Phần quan trọng nhất của thuật toán là quá trình sinh ra một cây quyết định khởi đầu từ tập các mẫu huấn luyện. Kết quả, thuật toán sinh ra một bộ phân lớp ở dạng của một cây quyết định; Một cấu trúc với 2 kiểu nodes: Node lá, để chỉ 1 lớp, hoặc một node quyết định chỉ ra kiểm tra được thực hiện trên một giá trị thuộc tính đơn, với một nhánh và cây con cho mỗi khả năng đầu ra của kiểm tra.

Một cây quyết định có thể được dùng để phân lớp một mẫu mới bằng cách khởi đầu tại gốc của cây và di chuyển qua nó đến khi gặp một lá. Tại mỗi node quyết định không là lá, đầu ra với kiểm tra tại node được xác định và lựa chọn di chuyển tới gốc của cây con. Ví dụ, nếu mô hình phân lớp của bài toán được cho với cây quyết định trong hình 1.4.1 và mẫu cho việc phân lớp trong hình 1.4.2, thì thuật toán sẽ tạo đường đi qua các nodes A, C, và F (node lá) đến khi nó tạo quyết định phân lớp cuối cùng: CLASS2.



1) Decision tree

Attribute	Value
Attribute1	5
Attribute2	Black
Attribute3	*No*

2) An example for classification

Hình 1.4: Sự phân lớp một mẫu mới dựa trên mô hình cây quyết định

Thuật toán phát triển cây (tree-growing) cho việc sinh ra cây quyết định dựa trên các phân tách đơn biến là ID3 với phiên bản mở rộng là C4.5.

Giả sử có nhiệm vụ lựa chọn một kiểm tra với n đầu ra (n giá trị cho một đặc trưng đã cho) mà chia tập các mẫu học T thành các tập con T_1, T_2, …, T_n. Thông tin dùng cho việc hướng dẫn là sự phân tán của các lớp trong T và các tập con T_i của nó. Nếu S là tập bất kỳ các mẫu, gọi freq (C_i, S) biểu thị số lượng các mẫu trong S mà thuộc vào lớp C_i, và |S| biểu diễn số lượng các mẫu trong tập S.

Thuật toán ID3 gốc dùng một tiêu chuẩn được gọi là lợi ích (gain) để lựa chọn thuộc tính được kiểm tra, dựa trên khái niệm lý thuyết thông tin: entropy. Quan hệ sau đây đưa ra tính toán của entropy của tập S:

$$Info(S) = -\sum_{i=1}^{k} p_i \log_2 p_i = -\sum_{i=1}^{k} ((freq(C_i, S) / |S|) * \log_2 (freq(C_i, S) / |S|)$$

Xem xét tập T sau khi được phân chia tương ứng với n đầu ra của một thuộc tính kiểm tra X. Yêu cầu về thông tin mong đợi có thể được tìm ra như là tổng trọng số của các entropies trên các tập con:

$$Info_x(T) = -\sum_{i=1}^{n} ((|T_i| / |T|) * Info(T_i))$$

Độ đo lợi ích thông tin Gain: Một thuộc tính có lợi ích thông tin cao, nghĩa là nếu biết được các giá trị của thuộc tính đó thì việc phân lớp sẽ tiến gần tới đích. Như ví dụ trên hình 1.3, nếu biết X>1 thì biết được ngay thuộc lớp Class1. Gain của thuộc tính X được đo bằng độ giảm entropy trung bình của tập T sau khi đã biết giá trị của X:

$$Gain(X) = Info(T) - Info_x(T)$$

Ví dụ minh hoạ việc áp dụng các phép đo khi tạo cây quyết định:

Giả sử CSDL T với 14 trường hợp (ví dụ) được mô tả với 3 thuộc tính đầu vào và thuộc vào 2 nhóm cho trước: CLASS1 hoặc CLASS2. CSDL cho trước trong bảng 1.1

9 mẫu thuộc vào CLASS1 và 5 mẫu thuộc CLASS2, vậy entropy trước khi phân tách là:

$$Info(T) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.940 \text{ bits}$$

Sau khi dùng Attribute1 để chia tập ban đầu của các mẫu T thành 3 tập con (kiểm tra x_1 biểu diễn lựa chọn một trong 3 giá trị A, B hoặc C), thông tin kết quả được cho bởi:

$$\begin{aligned} Info_{x1}(T) = {} & 5/14 \, (- 2/5 \log_2 (2/5) - 3/5 \log_2 (3/5)) \\ & + 4/14 \, (- 4/4 \log_2 (4/4) - 0/4 \log_2 (0/4)) \\ & + 5/14 \, (- 3/5 \log_2 (3/5) - 2/5 \log_2 (2/5)) \\ = {} & 0.694 \text{ bits} \end{aligned}$$

Bảng 1.1: CSDL đơn giản gồm các ví dụ huấn luyện

CSDL T:			
Attribute1	Attribute2	Attribute3	Attribute4
A	70	True	CLASS1
A	90	True	CLASS2
A	85	False	CLASS2
A	95	False	CLASS2
A	70	False	CLASS1
B	90	True	CLASS1
B	78	False	CLASS1
B	65	True	CLASS1
B	75	False	CLASS1
C	80	True	CLASS2
C	70	True	CLASS2
C	80	False	CLASS1
C	80	False	CLASS1
C	96	False	CLASS1

Thông tin thu được bằng kiểm tra x1 này là:

Gain $(x_1) = 0.940 – 0.694 = 0.246$ bits

Nếu kiểm tra và phân tách dựa trên Attribute3 (kiểm tra x_2 biển diễn lựa chọn một trong 2 giá trị True hoặc False), một tính toán tương tự sẽ cho các kết quả mới:

$$\text{Info}_{x2}(T) = 6/14 (– 3/6 \log_2 (3/6) – 3/6 \log_2 (3/6)) + 8/14 (– 6/8 \log_2 (6/8) – 2/8 \log_2 (2/8))$$

$$= 0.892 \text{ bits}$$

Và gain tương ứng là

Gain(x_2) = 0.940 – 0.892 = 0.048 bits

Dựa trên điều kiện lợi ích (gain criterion), thuật toán cây quyết định sẽ lựa chọn kiểm tra x_1 như một kiểm tra khởi đầu cho việc phân tách CSDL T bởi vì giá trị lợi ích cao hơn. Để tìm ra kiểm tra tối ưu, cần phải phân tích kiểm tra trên Attribute2, là một đặc trưng số với các giá trị liên tục.

Ở trên đã giải thích kiểm tra chuẩn cho các thuộc tính phân loại. Dưới đây sẽ nêu thêm về thủ tục cho việt thiết lập các kiểm tra trên các thuộc tính với các giá trị số. Các kiểm tra trên các thuộc tính liên tục sẽ khó công thức hoá, vì nó chứa một ngưỡng bất kỳ cho việc phân tách tất cả các giá trị vào 2 khoảng.

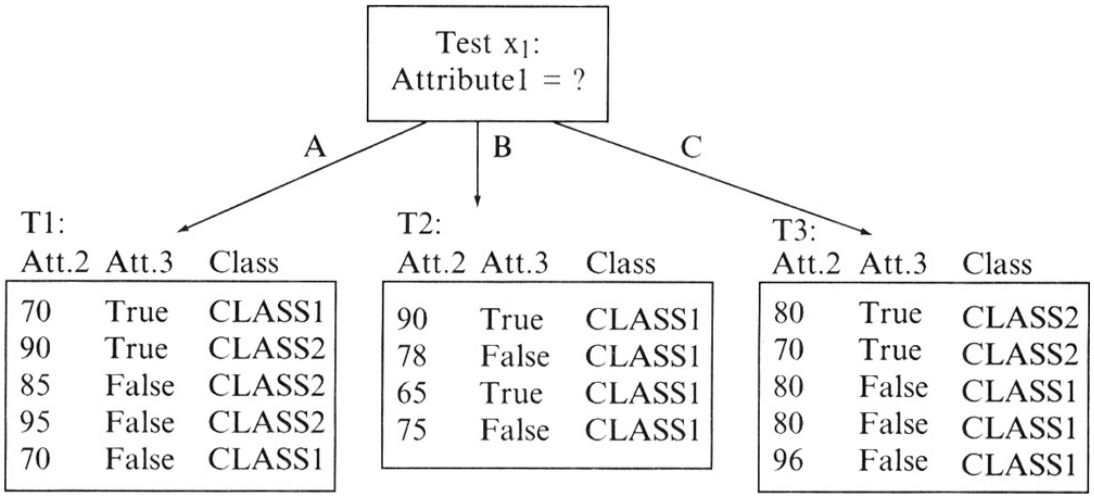
Có một thuật toán cho việc tính toán giá trị ngưỡng tối ưu Z. Các mẫu học đầu tiên được sắp xếp trên các giá trị của thuộc tính Y đang được xem xét. Chỉ có một số có hạn của các giá trị này, vì vậy ký hiệu chúng trong thứ tự đã được sắp xếp là $\{v_1, v_2 \ldots, v_m\}$. Bất kỳ giá trị ngưỡng nào nằm giữa v_i và v_{i+1} sẽ có cùng hiệu quả nếu ngưỡng đó chia các trường hợp thành những phần mà giá trị của thuộc tính Y của chúng nằm trong $\{v_1, v_2 \ldots, v_i\}$ và trong $\{v_{i+1}, v_{i+2}, \ldots, v_m\}$. Chỉ có m-1 khả năng trên Y, tất cả chúng cần được kiểm tra một cách có hệ thống để thu được một phân tách tối ưu. Thường chọn ngưỡng là điểm giữa của mỗi khoảng $(v_i + v_{i+1})/2$.

Ví dụ minh hoạ quá trình tìm ngưỡng này: Với CSDL T, phân tích các khả năng phân tách Attribute2. Sau khi sắp xếp, tập các giá trị cho Attribute2 là {65, 70, 75, 78, 80, 85, 90, 95, 96} và tập các giá trị ngưỡng tiềm năng Z là {65, 70, 75, 78, 80, 85, 90, 95}. Z tối ưu (với thông tin lợi ích cao nhất) cần được lựa chọn. Trong ví dụ này, giá trị Z tối ưu là Z = 80 và quá trình tính toán thông tin lợi ích tương ứng cho kiểm tra x_3 (Attribute2 $\leq$ 80 or Attribute2 > 80) như sau:

$$Info_{x3}(T) = 9/14\ (-7/9 \log_2(7/9) - 2/9 \log_2(2/9)) + 5/14\ (-2/5 \log_2(2/5) - 3/5 \log_2(3/5)) = 0.837 \text{ bits}$$

$$Gain(x_3) = 0.940 - 0.837 = 0.103 \text{ bits}$$

So sánh thông tin lợi ích cho 3 thuộc tính trong ví dụ, ta có thể thấy Attribute1 vẫn cho lợi ích cao nhất 0.246 bits và do đó thuộc tính này sẽ được lựa chọn cho việc phân tách đầu tiên trong việc xây dựng cây quyết định. Nút gốc sẽ có kiểm tra cho các giá trị của Attribute1, và 3 nhánh sẽ được tạo, mỗi nhánh cho một giá trị thuộc tính. Cây ban đầu này với các tập con tương ứng của các mẫu trong các nodes con được biểu diễn trong hình 1.5.



Hình 1.5 Cây quyết định ban đầu
và tập con các trường hợp cho một CSDL trong bảng 1.1

Sau việc phân tách ban đầu, mỗi node con có một vài mẫu từ CSDL, và toàn bộ quá trình lựa chọn và tối ưu kiểm tra sẽ được lặp lại cho mọi node con. Bởi vì node con cho kiểm tra x_1: Attribute1 = B có 4 trường hợp và tất cả chúng là trong CLASS1, node này sẽ là node lá, và không có các kiểm tra bổ sung nào cần cho nhánh này của cây.

Cho node con còn lại, có 5 trường hợp trong tập con T1, các kiểm tra trên các thuộc tính còn lại có thể được thực hiện; một kiểm tra tối ưu (với thông tin có ích cực đại) sẽ là kiểm tra x_4 với 2 lựa chọn: Attribute2 $\leq$ 70 or Attribute2 > 70.

Info $(T_1) = -2/15 \log_2 (2/5) - 3/15 \log_2 (3/5) = 0.940$ bits

Dùng Attribute2 để chia T_1 thành 2 tập con (kiểm tra x_4 biểu diễn lựa chọn của một trong 2 khoảng), thông tin kết quả được cho bởi:

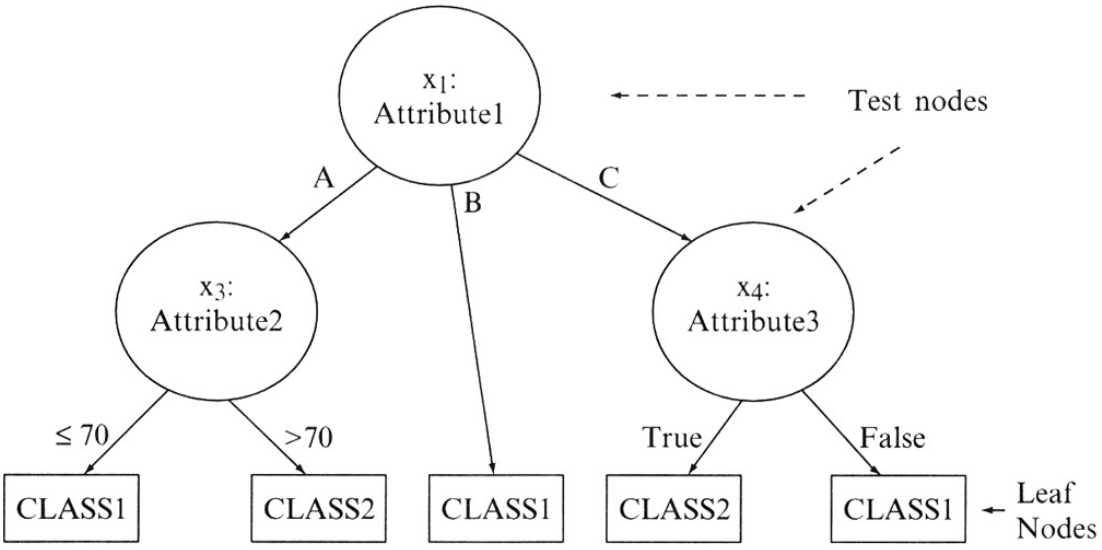
$$\text{Info}_{x4}(T_1) = 2/5 (- 2/2 \log_2 (2/2) - 0/2 \log_2 (0/2)) + 3/5 (- 0/3 \log_2 (0/3) - 3/3 \log_2 (3/3)) = 0 \text{ bits}$$

Gain thu được bởi test này là cực đại:

$\text{Gain}(x_4) = 0.940 - 0 = 0.940$ bits

Và 2 nhánh sẽ tạo các node lá cuối cùng vì các tập con của các trường hợp trong mỗi nhánh thuộc vào cùng một class.

Tính toán tương tự sẽ được tiến hành/tiếp tục cho con thứ 3 của node gốc. Cho tập con T_3 của CSDL T, kiểm tra x_5 tối ưu được chọn là việc kiểm tra trên các giá trị của Attribute3. Các nhánh của cây, Attribute3 = True và Attribute3 = False, sẽ tạo các tập con đồng nhất của các trường hợp mà thuộc vào cùng một lớp. Cây quyết định cuối cùng cho CSDL T được biểu diễn trong hình 1.5.



Hình 1.5 Cây quyết định cuối cùng cho CSDL T đã nêu trong bảng 1.1

Tuỳ chọn, một cây quyết định cũng có thể được biểu diễn ở dạng một mã thực hiện (hoặc giả mã) với các cấu trúc if-then cho việc tách nhánh thành một cấu trúc cây. Cây quyết định cuối cùng trong ví dụ trên được đưa trong giả code như hình 1.6.

```
If      Attribute1 = A
        Then
                If      Attribute2 <= 70
                        Then
                                Classification = CLASS1;
                        Else
                                Classification = CLASS2;
Elseif  Attribute1 = B
        Then
                                Classification = CLASS1;
Elseif  Attribute1 = C
        Then
                If      Attribute3 = True
                        Then
                                Classification = CLASS2;
                        Else
                                Classification = CLASS1.
```

Hình 1.6 Cây quyết định ở dạng giả code cho CSDL T (bảng 1.1)

1.2.1.3 Phân lớp Bayees

Phân lớp Bayees là phương pháp phân lớp thống kê dự đoán xác suất các thành viên thuộc lớp. Phân lớp Bayees cho tính chính xác và tốc độ cao khi áp dụng vào các CSDL lớn. Phương pháp Naive Bayees là một phương pháp phân lớp Bayees đơn giản. Phương pháp này giả thiết ảnh hưởng của một giá trị thuộc tính tới lớp là độc lập với các giá trị thuộc tính khác - gọi là độc lập điều kiện lớp.

Lý thuyết Bayees

Cho X là dữ liệu ví dụ của một lớp chưa biết. H là giả thiết X thuộc lớp C. Bài toán phân lớp sẽ xác định P(H|X) – là xác suất giả thuyết H chứa ví dụ X. Đó là xác suất hậu nghiệm của H với điều kiện X.

Công thức Bayees là:

$P(H|X) = P(X|H) * P(H) / P(X)$ (**1.1**)

Với P(X|H) là xác suất hậu nghiệm của X với điều kiện H.

P(X) là xác suất tiên nghiệm của X.

Phân lớp Naive Bayees

1. Mỗi dữ liệu ví dụ được biểu diễn bằng một vecto X=(x1, .. xn) mô tả n độ đo của n thuộc tính A1,.., An.
2. Giả sử có m lớp $C_1,\ldots, C_m$. Cho một trường hợp X chưa biết lớp, phân lớp sẽ dự đoán X thuộc về lớp C_i có xác suất điệu kiện X cao nhất, nghĩa là

   $X \subset C_i \Leftrightarrow P(C_i|X)>P(C_j | X) \ \forall \ \ 1<=j<=m \ j \# i$

   Theo công thức Bayees có: $P(C_i|X) = P(X | C_i)P(C_i)/ P(X)$

   Trong đó C_i được gọi là giả thuyết hậu nghiệm lớn nhất.
3. Nếu P(X) là hằng chỉ cần tìm max $P(X|C_i)P(C_i)$. Nếu xác suất tiên nghiệm chưa biết và giả sử $P(C_1)=P(C_2)$... thì tìm C_i có max $P(X|C_i)P(C_i)$. (**1.2**)

4. Nếu dữ liệu có nhiều thuộc tính, chi phí tính toán $P(X|C_i)$ có thể rất lớn, vì vậy với giả thiết các thuộc tính độc lập điều kiện lớp thì có thể tính $P(X|C_i)= \prod P(\mathbf{X}_k|C_i)$ (k=1..n)

   Trong đó $P(\mathbf{X}_k|C_i)$ được tính như sau :

   Với giả thiết A_k là thuộc tính giá trị tên thì $P(X_k|C_i)= S_{ik}/S_i$, trong đó S_{ik} số ví dụ huấn luyện của lớp C_i có giá trị X_k với A_k, S_i là số ví dụ thuộc lớp C_i.
5. Để phân lớp cho đối tượng X chưa biết lớp: Tính các giá trị $P(X|C_i)$ cho mọi lớp C_i và X thuộc lớp C_i khi và chỉ khi: $P(X|C_i)=Max(P(X|C_i)P(C_i)$.

1.2.2 Hồi qui

Hồi qui: Một kỹ thuật phân tích dữ liệu dùng thống kê để xây dựng các mô hình dự báo cho các trường dự báo có giá trị liên tục. Kỹ thuật tự động xác định một công thức toán học mà cực tiểu hoá một vài phép đo lỗi giữa cái dự báo từ mô hình Hồi qui với dữ liệu thực.

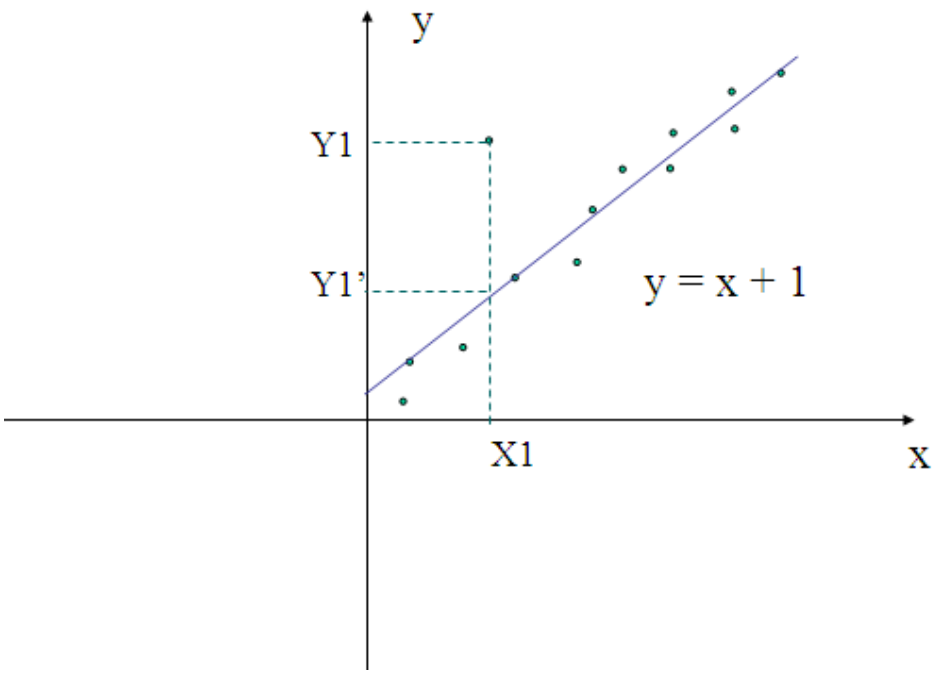
Dạng đơn giản nhất của một mô hình hồi qui chứa một biến phụ thuộc (còn gọi là "biến đầu ra", "biến nội sinh" hay "biến Y") và một biến độc lập đơn (còn gọi là "hệ số", "biến ngoại sinh", "hay biến X").

Ví dụ như: sự phụ thuộc của huyết áp Y theo tuổi tác X, hay sự phụ thuộc của trọng lượng Y theo khẩu phần ăn hàng ngày. Sự phụ thuộc này được gọi là hồi qui của Y lên X.

Hồi qui tạo các mô hình dự báo. Sự khác nhau giữa Hồi qui và phân lớp là hồi qui xử lý với các thuộc tính đích là kiểu số hoặc liên tục, trong khi phân lớp xử lý với các thuộc tính đích riêng lẻ hoặc phân loại (discrete/categorical). Nói cách khác, nếu thuộc tính đích chứa các giá trị liên tục, sẽ đòi hỏi dùng kỹ thuật hồi qui. Nếu thuộc tính đích chứa các giá trị phân loại (xâu ký tự hoặc số nguyên rời rạc), kỹ thuật phân lớp sẽ được sử dụng.

Dạng thông dụng nhất của hồi qui là Hồi qui tuyến tính (linear regression), trong đó một đường thẳng vừa nhất với dữ liệu được tính toán, đó là, đường thẳng cực tiểu hoá khoảng cách trung bình của tất cả các điểm từ đường đó.

Đường này trở thành mô hình dự báo khi giá trị của biến phụ thuộc là chưa biết; giá trị của nó được dự báo bởi điểm nằm trên đường mà tương ứng với giá trị của các biến phụ thuộc cho bản ghi đó.



Hình 1.7 Hồi qui tuyến tính

Một số khái niệm:

- Các biến ngẫu nhiên X_1, …, X_k (các biến dự báo) và Y (biến phụ thuộc)
- X_i có miền (domain) là dom(X_i), Y có miền là dom(Y)
- P là một phân bố xác suất trên dom(X_1) x … x dom(X_k) x dom(Y)
- CSDL huấn luyện D là một mẫu ngẫu nhiên từ P

— Bộ dự báo (predictor) là một hàm:

d: dom(X_1) … dom(X_k) → dom(Y)

Nếu Y là số, bài toán là bài toán Hồi qui. Y được gọi là biến phụ thuộc, d được gọi là hàm Hồi qui.

Gọi r là một bản ghi ngẫu nhiên lấy từ P. Định nghĩa tỷ suất lỗi trung bình bình phương của d là:

$RT(d,P) = E(r.Y - d(r.X_1, \ldots, r.X_k))^2$

Định nghĩa bài toán: Tập dữ liệu D cho trước là một mẫu ngẫu nhiên từ phân tán xác suất P, tìm hàm Hồi qui d mà RT(d, P) là cực tiểu.

Thuật toán SVM cho Hồi qui

Support Vector Machine (SVM) xây dựng cả hai mô hình phân lớp và Hồi qui. SVM là một công cụ dự báo phân lớp và Hồi qui dùng lý thuyết học máy để cực đại độ chính xác dự báo trong khi tự động tránh vượt ngưỡng (over-fit) đối với dữ liệu.

Các mạng neural và các hàm radial basis (RBFs), hai kỹ thuật khai phá thông dụng, có thể được xem là trường hợp đặc biệt của SVMs.

SVMs thực hiện tốt với các ứng dụng thế giới thực như phân lớp dữ liệu văn bản (text), nhận dạng các chữ viết tay, phân lớp các hình ảnh,... Việc giới thiệu nó trong những năm đầu 1990s dẫn đến sự bùng nổ các ứng dụng và các phân tích lý thuyết chuyên sâu thiết lập SVM cùng với mạng neural như các công cụ chuẩn cho học máy và khai phá dữ liệu. [11]

Không có giới hạn trên nào trên số lượng các thuộc tính và ứng viên đích cho SVMs.

Chi tiết về chuẩn bị dữ liệu và các thiết đặt lựa chọn cho SVM – tham khảo trong [13].

1.2.3 Phân nhóm

Phân nhóm là kỹ thuật hữu ích cho việc khám phá dữ liệu. Nó hữu ích khi có nhiều trường hợp và không có các nhóm tự nhiên rành mạch. Ở đây, các thuật toán khai phá dữ liệu phân nhóm có thể được dùng để tìm ra bất kỳ nhóm tự nhiên nào có thể tồn tại.

Phân tích phân nhóm xác định các nhóm trong dữ liệu. Một nhóm là tập hợp/thu thập các đối tượng dữ liệu tương tự nhau trong một vài nghĩa nào đó so với đối tượng khác. Phương pháp phân nhóm tốt tạo ra các nhóm chất lượng cao để đảm bảo rằng sự tương tự giữa các nhóm khác nhau là thấp và sự tương tự bên trong nhóm là cao; nói cách khác, các thành viên của một nhóm giống nhau hơn so với các thành viên trong nhóm khác.

Phân nhóm cũng có thể phục vụ như một bước tiền xử lý dữ liệu hiệu quả để xác định các nhóm không đồng nhất trong đó để xây dựng các mô hình dự báo. Các mô hình phân nhóm khác với các mô hình dự báo trong đó đầu ra của quá trình không được hướng dẫn bằng kết quả đã biết, đó là, không có thuộc tính đích. Các mô hình dự báo dự đoán các giá trị cho một thuộc tính đích và một tỷ suất lỗi giữa các giá trị đích và giá trị dự báo có thể được tính toán để hướng dẫn việc xây dựng mô hình. Các mô hình phân nhóm khám phá các nhóm tự nhiên trong dữ liệu. Mô hình có thể sau đó được dùng để gán các nhãn của các nhóm (cluster IDs) tới các điểm dữ liệu.

Ví dụ, cho tập dữ liệu với 2 thuộc tính: AGE và HEIGHT, luật sau đây biểu diễn phần lớn dữ liệu được gán cho cluster 10:

If AGE >= 25 and AGE <= 40

and HEIGHT >= 5.0ft and HEIGHT <= 5.5ft

then CLUSTER = 10

Một đầu vào của một phân tích cluster có thể được mô tả như một cặp có thứ tự (X, s), hoặc (X, d), trong đó X là tập các mô tả của các mẫu và s và

d theo thứ tự là các tiêu chuẩn cho độ tương tự hoặc không tương tự (khoảng cách) giữa các mẫu. Đầu ra từ hệ thống clustering là một partition $\Lambda = \{G_1, G_2, \ldots, G_N\}$ trong đó G_k, k = 1, …, N là tập con thực sự (crisp subset) của X:

$G_1 \cup G_2 \cup \ldots \cup G_1 = X$, và

$G_i \cap G_j = \varnothing \qquad i \neq j$

Các thành viên $G_1, G_2, \ldots, G_N$ của Λ được gọi là các clusters. Mọi cluster có thể được mô tả với một vài đặc trưng. Trong việc phân nhóm (clustering) trên cơ sở khám phá, cả cluster (tập riêng biệt các điểm trong X) và các mô tả của chúng hoặc các đặc trưng được sinh ra như là một kết quả của một thủ tục clustering.

Phần lớn các thuật toán clustering dựa trên 2 cách tiếp cận sau:

1. Phân nhóm theo partitional theo thứ tự lỗi lặp (Iterative square-error partitional clustering) – Phân hoạch
2. Phân nhóm phân cấp (Hierarchical clustering)

1.2.3.1 Các phương pháp phân hoạch

Cho một CSDL với N đối tượng, phương pháp phân hoạch xây dựng K phân hoạch (K<=N), trong đó mỗi phân hoạch biểu diễn một nhóm. Nghĩa là phân chia dữ liệu thành K nhóm thoả mãn:

— Mỗi nhóm phải chứa ít nhất 1 đối tượng

— Mỗi đối tượng chỉ thuộc 1 nhóm

Một kỹ thuật đơn giản nhất là chia N thành K tập con có thể, thử phân hoạch lần đầu, sau đó lặp các bước phân hoạch bằng cách chuyển các đối tượng từ nhóm này sang nhóm khác và đánh giá theo tiêu chuẩn gần nhất của các đối tượng trong nhóm. Quá trình phân hoạch này có thể đòi hỏi sử dụng mọi khả năng có thể của K tập con trong N nhóm. Các ứng dụng có thể sử dụng một trong hai thuật toán ***K-mean***, mỗi nhóm được đại diện bằng một giá

trị trung bình của các đối tượng trong nhóm hoặc ***K-medoid*** trong đó mỗi nhóm được đại diện bằng 1 trong các đối tượng gần tâm nhóm.

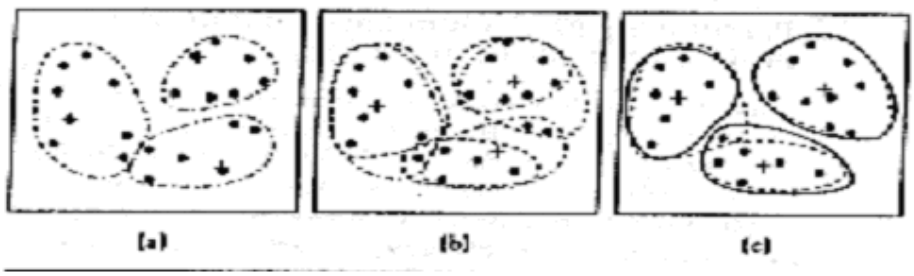
Kỹ thuật dựa tâm - Thuật toán k-mean

Thuật toán K-mean sẽ phân hoạch dữ liệu thành K (tham số) nhóm xử lý như sau:

- Lựa chọn ngẫu nhiên K đối tượng, mỗi đối tượng được coi là khởi tạo giá trị trung bình hoặc tâm của nhóm.
- Các đối tượng còn lại sẽ được gán vào các nhóm được coi là gần đối tượng đó nhất dựa trên khoảng cách giữa đối tượng với tâm.
- Tính toán lại giá trị trung bình của mỗi nhóm
- Các bước trên được lặp cho đến khi đạt hội tụ. Tiêu chuẩn sai số toàn phương:

  $E = \sum \sum | p - m_i |^2$

Trong đó E là tổng các sai số bình phương của mọi đối tượng trong CSDL, p là điểm trong không gian biểu diễn các đối tượng, m_i là trung bình của nhóm C_i. Tiêu chuẩn này tiến đến K nhóm là kín và tách rời nhau. Thuật toán sẽ tiến đến K phân hoạch sao cho hàm sai số bình phương là tối thiểu (LMS). Thuật toán này chỉ áp dụng được nếu xác định được giá trị trung bình. Do vậy thuật toán sẽ không áp dụng được cho dữ liệu tên (ký tự).



Hình 1.8 Gộp nhóm theo phương pháp k-means (Điểm đánh dấu + là tâm)

Các biến thể mở rộng của K-mean:

K-medoid mở rộng phân hoạch cho các thuộc tính có giá trị tên bằng kỹ thuật dựa trên tần suất xuất hiện.

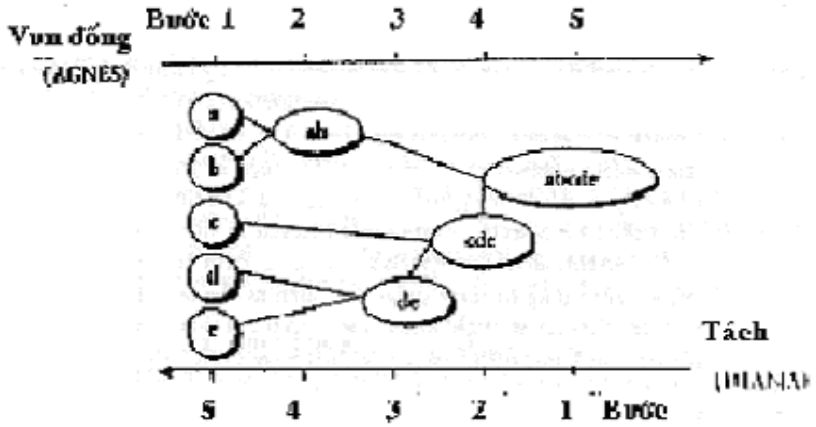
1.2.3.2 Các phương pháp phân cấp

Phương pháp phân cấp làm việc bằng cách nhóm các đối tượng dữ liệu thành cây. Các phương pháp phân cấp có thể phân lớp theo kiểu vun đống hoặc tách dần:

— Gộp nhóm phân cấp kiểu vun đống: Chiến lược từ dưới lên bắt đầu bằng cách đặt mỗi đối tượng đơn vào một nhóm sau đó trộn vài nhóm thành nhóm lớn hơn và lớn hơn nữa, cho đến khi thành một nhóm đơn hoặc thoả mãn điều kiện kết thúc nào đó.

— Gộp nhóm phân cấp kiểu tách dần: Chiến lược này ngược với chiến lược từ trên xuống. Bắt đầu bằng cách đặt tất cả các đối tượng thành một nhóm đơn. Sau đó chia thành các nhóm nhỏ dần, cho đến khi thành các nhóm với một đối tượng hoặc thoả mãn điều kiện kết thúc nào đó.

Trong các thuật toán này cần xác định điều kiện để kết thúc.



Hình 1.9 Phân hoạch vun đống hoặc tách dần

1.2.4 Khai phá luật kết hợp

Các luật kết hợp là một trong những kỹ thuật chính của khai phá dữ liệu và nó có thể là thông dụng nhất từ khám phá mẫu địa phương trong hệ thống học không giám sát.

1.2.4.1 Phân tích giỏ hàng

Giỏ hàng là tập thu thập các mục hàng mua sắm của khách hàng trong một giao dịch đơn lẻ. Những người bán hàng thu thập các giao dịch bằng việc ghi lại các hoạt động kinh doanh (bán hàng) qua thời gian dài. Một phân tích thông thường thực hiện trên CSDL các giao dịch là tìm ra các tập hàng hoá xuất hiện cùng nhau trong nhiều giao dịch. Tri thức của những mẫu này có thể được dùng để cải tiến chỗ để của những mặt hàng trong kho hoặc sắp đặt lại thứ tự ở catalog trong trang mail và các trang Web. Một itemset chứa i items được gọi là i-itemset. Số phần trăm các giao dịch chứa một itemset được gọi là độ hỗ trợ của itemset. Itemset được quan tâm, độ hỗ trợ của nó cần phải cao hơn giá trị cực tiểu mà người sử dụng đưa ra. Những itemsets như vậy được gọi là thường xuyên (frequent).

Việc tìm ra các frequent itemsets là không đơn giản. Lý do trước tiên, số giao dịch của khách hàng có thể rất lớn và thường không vừa với bộ nhớ của máy tính. Thứ hai, số lượng các frequent itemsets tiềm năng là theo hàm mũ đối với số lượng các items khác nhau, mặc dù số lượng thực của các frequent itemsets có thể nhỏ hơn nhiều. Do vậy, yêu cầu đối với các thuật toán là có thể mở rộng (độ phức tạp của nó phải tăng tuyến tính, không theo hàm mũ với số lượng các giao dịch) và nó kiểm tra càng ít các infrequent itemsets càng tốt.

Mô tả bài toán:

Từ một CSDL của các giao dịch bán hàng, cần tìm ra các mối liên hệ quan trọng trong các items: Sự có mặt của một vài items trong giao dịch sẽ dẫn đến sự có mặt của các items khác trong cùng giao dịch.

Có các items I = $\{i_1, i_2, \ldots, i_m\}$. DB là tập các giao dịch, trong đó mỗi giao dịch T là tập của các items vậy là T□ I. Ở đây không quan tâm đến số lượng hàng (items) được mua trong giao dịch, nghĩa là mỗi item là một biến nhị phân chỉ ra item đó có được mua hay không. Mỗi giao dịch tương ứng với một định danh được gọi là transaction identifier hoặc TID. Một ví dụ về CSDL giao dịch như vậy được đưa ra trong bảng 1.2

Bảng 1.2 Mô hình CSDL giao dịch đơn giản

Database DB:	
TID	Items
001	A C D
002	B C E
003	A B C E
004	B E

Gọi X là tập các items. Giao dịch T được gọi là chứa X nếu và chỉ nếu $X \subseteq T$. Một luật kết hợp ngụ ý dạng $X \Rightarrow Y$, trong đó $X \subseteq I$, $Y \subseteq I$, và $X \cap Y = \varnothing$. Luật $X \Rightarrow Y$ có trong tập giao dịch DB với độ chắc chắn c (confidence) nếu c% giao dịch trong D có chứa X thì cũng chứa Y. Luật $X \Rightarrow Y$ có độ hỗ trợ s trong tập giao dịch D nếu s% các giao dịch trong DB chứa $X \cup Y$. Độ chắc chắn biểu thị độ lớn của ý nghĩa của luật và độ hỗ trợ biểu thị tần số của các mẫu xuất hiện trong luật. Thường được quan tâm là những luật có độ hỗ trợ lớn hợp lý. Những luật với độ chắc chắn cao và độ hỗ trợ mạnh được xem như là những luật mạnh. Nhiệm vụ của khai phá các luật kết hợp là phát hiện

các luật kết hợp mạnh trong các CSDL lớn. Bài toán khai phá các luật kết hợp có thể được chia thành 2 pha:

1. Khám phá các itemsets lớn, ví dụ các tập items có độ hỗ trợ của giao dịch ở trên ngưỡng tối thiểu cho trước.
2. Dùng các itemsets lớn để sinh ra các luật kết hợp cho CSDL mà có độ chắc chắn c trên ngưỡng tối thiểu cho trước

Hiệu năng chung của việc khai phá các luật kết hợp được xác định chủ yếu bởi bước đầu tiên. Sau khi các itemsets lớn được xác định, các luật kết hợp tương ứng có thể được lấy theo cách đơn giản. Tính toán hiệu quả của các itemsets lớn là trọng tâm của phần lớn các thuật toán khai phá.

1.2.4.2 Thuật toán Apriori

Thuật toán Apriori tính toán các frequent itemsets trong CSDL qua một vài lần lặp. Bước lặp thứ i tính toán tất cả các frequent i-itemsets (các itemsets với i thành phần). Mỗi lần lặp có 2 bước: b1) Sinh ra các candidate. b2) Tính toán và chọn candidate.

Trong pha đầu tiên của bước lặp đầu tiên, tập các candidate itemsets được sinh ra chứa tất cả các 1-itemsets (nghĩa là, tất cả các items trong CSDL). Trong pha tính toán, thuật toán tính độ hỗ trợ của chúng tìm trên toàn bộ CSDL. Cuối cùng, chỉ những 1-itemsets với s ở trên ngưỡng yêu cầu sẽ được chọn là frequent. Như vậy sau lần lặp đầu tiên, tất cả các frequent 1-itemsets sẽ được xác định.

Tiếp tục với lần lặp thứ 2. Sinh ra các candidate của 2-itemset như thế nào? Tất cả các cặp của items đều là các candidates. Dựa trên tri thức về các infrequent itemsets thu được từ các lần lặp trước, thuật toán Apriori giảm tập các candidate itemsets bằng cách cắt tỉa các candidate itemsets không là frequent. Việc cắt tỉa dựa trên sự quan sát là nếu một itemset là frequent thì

tất cả các tập con của nó cũng là frequent. Do vậy, trước khi vào bước tính toán candidate, thuật toán sẽ thực hiện loại bỏ mọi candidate itemset mà có các tập con là infrequent.

Xem xét CSDL trong bảng 1.2. Giả sử rằng độ hỗ trợ tối thiểu s=50% như vậy một itemset là frequent nếu nó được chứa trong ít nhất là 50% các giao dịch. Trong mỗi bước lặp, thuật toán Apriori xây dựng một tập candidate của các large itemsets, đếm số lần xuất hiện của mỗi candidate, và sau đó xác định large itemsets dựa trên độ hỗ trợ cực tiểu cho trước s=50%.

Trong bước đầu tiên của lần lặp đầu tiên, tất cả các items đơn lẻ đều là candidates. Apriori đơn giản duyệt tất cả các giao dịch trong CSDL DB và sinh ra danh sách các candidates. Trong bước tiếp theo, thuật toán đếm số lần xuất hiện của mỗi candidate và dựa trên ngưỡng s chọn ra các frequent itemsets. Tất cả những bước này được đưa trong hình 1.10. Năm 1-itemsets được sinh ra trong C_1 và chỉ có bốn được chọn là lớn trong L_1 (có s ≥50%).

1-itemset C_1
{A}
{C}
{D}
{B}
{E}

1. Generate phase

1-itemsets	Count	S{%}
{A}	2	50
{C}	3	75
{D}	1	25
{B}	3	75
{E}	3	75

2. Count phase

Large 1-itemsets L_1	Count	S{%}
{A}	2	50
{C}	3	75
{B}	3	75
{E}	3	75

3. Select phase

Hình 1.10 Bước lặp đầu tiên của thuật toán *Apriori* cho CSDL DB

Để khám phá ra tập các large 2-itemsets, bởi vì bất kỳ tập con nào của large itemset cũng có độ hỗ trợ cực tiểu, thuật toán Apriori dùng $L_1 * L_1$ để sinh ra các candidates. Thao tác * được định nghĩa tổng quát là

$$L_k * L_k = \{X \cup Y \text{ where } X, Y \in L_k, |X \cap Y| = k - 1\}$$

Với k=1 toán hạng biểu diễn sự ghép chuỗi lại đơn giản. Do đó, C_2 chứa 2-itemsets được sinh ra bởi toán hạng $|L_1| \cdot (|L_1| - 1)/2$ như là các candidates trong lần lặp 2. Trong ví dụ của chúng ta, số này là $4.3/2 = 6$. Duyệt CSDL DB với danh sách này, thuật toán tính độ hỗ trợ cho mọi candidate và cuối cùng lựa chọn large 2-itemsets L_2 có $s \geq 50\%$. Tất cả những bước này và các kết quả tương ứng của lần lặp 2 được cho trong hình 1.11.

2-itemset C_2
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

1. Generate phase

2-itemsets	Count	S{%}
{A, B}	1	25
{A, C}	2	50
{A, E}	1	25
{B, C}	2	50
{B, E}	3	75
{C, E}	2	50

2. Count phase

Large 2-itemsets L_2	Count	S{%}
{A, C}	2	50
{B, C}	2	50
{B, E}	3	75
{C, E}	2	50

3. Select phase

Hình 1.11 Lần lặp thứ 2 của thuật toán *Apriori* cho CSDL DB

Tập các candidate itemsets C_3 được sinh ra từ L_2 dùng toán hạng đã định nghĩa trước đây L_2*L_2. Thực tế, từ L_2, hai large 2-itemsets với item đầu tiên giống nhau như {B,C} và {B,E}, được xác định trước. Như vậy, Apriori kiểm tra có 2-itemset {C,E}, cái mà chứa items thứ 2 trong tập {B,C} và {B,E}, tạo thành một larger 2-itemset hay không. Bởi vì {C,E} là large itemset, như vậy tất cả các tập con của {B,C,E} đều là large và do đó {B,C,E} trở thành candidate 3-itemset. Không có candidate 3-itemset nào khác từ L_2 trong CSDL DB. Apriori sau đó duyệt tất cả các giao dịch và khám phá ra large 3-itemsets L_3, như trong hình 1.12

3-itemset C_3
{B, C, E}

1.Generate phase

3-itemsets	Count	S{%}
{B, C, E}	2	50

2. Count phase

Large 3-itemsets L_3	Count	S{%}
{B, C, E}	2	50

3. Select phase

Hình 1.12 Lần lặp thứ 3 của thuật toán *Apriori* cho CSDL DB

Trong ví dụ, vì không có candidate 4-itemset để tiếp tục từ L_3, Apriori kết thúc quá trình lặp.

Apriori không chỉ tính toán độ hỗ trợ của tất cả các frequent itemsets, mà cả độ hỗ trợ của các infrequent candidate itemsets không thể loại ra trong pha cắt tỉa. Tập tất cả các candidate itemsets mà là infrequent nhưng độ hỗ trợ của nó được tính toán bởi Apriori được gọi là negative border. Như vậy, một itemset là trong negative border nếu nó là infrequent nhưng tất cả các tập con của nó là frequent. Trong ví dụ, phân tích hình 1.10 và 1.12 chúng ta có thể thấy rằng negative border bao gồm các itemsets {D}, {A, B}, và {A, E}. Negative border đặc biệt quan trọng cho một vài cải tiến thuật toán Apriori, như là tăng hiệu quả trong việc sinh ra các large itemsets hoặc thu được các luật kết hợp negative.

1.2.4.3 Từ các frequent itemsets tới các luật kết hợp

Pha thứ hai trong khai phá các luật kết hợp dựa trên tất cả các frequent i-itemsets, cái đã được tìm thấy trong pha đầu tiên dùng Apriori hoặc một vài thuật toán tương tự, là tương đối đơn giản và dễ hiểu. Với luật ngầm ý $\{x_1, x_2, x_3,\} \rightarrow x_4$, thì cần phải có itemset $\{x_1, x_2, x_3\ x_4\}$ và $\{x_1, x_2, x_3\}$ là frequent. Vậy, độ chắc chắn của luật $c= s(x_1, x_2, x_3, x_4\}/s(x_1\ x_2, x_3)$. Các luật kết hợp mạnh là các luật với giá trị độ chắc chắn c lớn hơn ngưỡng a cho trước.

Với CSDL DB trong bảng 1.2, để kiểm tra luật kết hợp $\{B, C\} \rightarrow E$ có là luật mạnh không, đầu tiên lấy các độ hỗ trợ tương ứng từ L_2 và L_3:

s(B,C) = 2, s(B, C, E) = 2

Và dùng những độ hỗ trợ này tính toán độ chắc chắn của luật:

c({B, C} → E) = s(B, C, E) / s(B, C) = 2/2 = 1 (hoặc 100%)

Với bất kỳ ngưỡng đã chọn cho các luật kết hợp mạnh (ví dụ, c_T= 0.8 hoặc 80%), luật này sẽ đạt vì độ chắc chắn của nó đạt cực đại, nghĩa là, nếu một giao dịch chứa items B và C, nó cũng sẽ chứa item E. Các luật khác cũng có thể có cho CSDL DB, như là A →C vì c(A →C) = s (A, C)/s(A) =1, và cả hai itemsets {A} và {A, C} là frequent dựa trên thuật toán Apriori. Do vậy trong pha này, cần thiết để phân tích một cách có hệ thống tất cả các luật kết hợp có thể sinh ra từ các frequent itemsets, và lựa chọn những luật kết hợp mạnh có độ chắc chắn trên ngưỡng cho trước.

Tuy nhiên không phải tất cả các luật kết hợp mạnh được phát hiện (nghĩa là, qua được các yêu cầu về độ hỗ trợ s và độ chắc chắn c) đều có ý nghĩa sử dụng. Ví dụ, xem xét trường hợp sau đây khai phá các kết quả điều tra ở trường học có 5000 sinh viên. Người bán lẻ bữa sáng ngũ cốc điều tra các hoạt động mà các sinh viên tham gia trong mọi buổi sáng. Dữ liệu cho thấy 60% sinh viên (là 3000 sinh viên) chơi basketball, 75% sinh viên (3759 sinh viên) ăn ngũ cốc, và 40% (là 2000 sinh viên) chơi basketball và cũng ăn ngũ cốc. Giả sử rằng chương trình khai phá dữ liệu cho khai phá các luật kết hợp thực hiện trên các thiết lập sau: Độ hỗ trợ cực tiểu là 2000 (s=0.4) và độ chắc chắn cực tiểu là 60% (c=0.6). Luật kết hợp sau đây sẽ được tạo ra: “Chơi basketball) → (ăn ngũ cốc)", vì luật này chứa độ hỗ trợ sinh viên cực tiểu và tương ứng với độ chắc chắn c = 2000/3000 = 0.66 là lớn hơn giá trị ngưỡng. Nhưng, luật kết hợp trên là sai lạc vì tỷ lệ sinh viên ăn ngũ cốc là 75%, lớn hơn 66%. Đó là, chơi basketball và ăn ngũ cốc trong thực tế là không có quan hệ. Không hiểu đầy đủ khía cạnh này, người ta có thể tạo những kinh doanh sai hoặc những quyết định mang tính khoa học/kỹ thuật cao sai từ các luật kết hợp thu được.

Để lọc ra các kết hợp sai lệch, người ta có thể định nghĩa một luật kết hợp A → B là đáng quan tâm nếu độ chắc chắn của nó vượt quá một độ đo

chắc chắn. Tham số đơn giản ta dùng trong ví dụ trên đưa ra kinh nghiệm đo sự kết hợp cần phải là:

$s(A, B) / s(A) – s(B) > d$

hoặc có thể chọn là:

$s(A, B) – s(A) \cdot s(B) > k$

Trong đó d hoặc k là các hằng số phù hợp. Các biểu thức ở trên về cơ bản biểu diễn các kiểm tra của tính độc lập thống kê. Rõ ràng, yếu tố phụ thuộc thống kê trong số các itemsets được phân tích đã được đưa vào xem xét để quyết định tính hữu dụng của các luật kết hợp. Trong ví dụ đơn giản của chúng ta về các sinh viên cuộc kiểm tra này là không thành công do khám phá ra luật kết hợp:

$s (A, B) – s(A) \cdot s(B) = 0.4 – 0.6 \cdot 0.75 = – 0.05 < 0$

Và do đó, mặc dù các giá trị cao cho các tham số s và c, luật vẫn không được quan tâm. Trong trường hợp này, là sự kiện sai lạc.

CHƯƠNG 2. MỘT SỐ THUẬT TOÁN KHAI PHÁ DỮ LIỆU

2.1. Thuật toán khai phá luật kết hợp

2.1.1 Thuật toán Apriori

Thuật toán này thực hiện theo từng mức:

1. Cho ngưỡng hỗ trợ s. Lần duyệt đầu tiên tìm các items xuất hiện ít nhất trong s phần của giỏ hàng. Có tập L_1, các frequent items.
2. Các candidate C_2 cho lần duyệt thứ hai là các cặp items trong L_1. Các cặp trong C_2 có số đếm >= s là các cặp frequent pairs L_2.
3. Bộ ba candidate C_3 là những tập {A, B, C} mà tất cả {A, B}, {A. C} và {B, C} đều trong L_2. Lần duyệt thứ 3, bộ ba candidate trong C_3 có số lần xuất hiện >= s là các bộ 3 frequent, L_3.
4. Có thể tiếp tục đến khi các tập trở thành rỗng. L_i là frequent itemsets có kích thước i; C_{i+1} là itemsets kích thước i+1 mà mỗi tập con kích thước i đều nằm trong L_i.

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)      C_k = apriori-gen(L_{k-1}); // New candidates
4)      forall transactions t ∈ D do begin
5)          C_t = subset(C_k, t); // Candidates contained in t
6)          forall candidates c ∈ C_t do
7)              c.count++;
8)      end
9)      L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ⋃_k L_k;
```

Hình 2.1 Thuật toán Apriori

2.1.1.1 Sinh ra Candidate của Apriori

Hàm apriori-gen lấy tham số L_{k-1}, tập của tất cả các large (k-1)-itemsets. Nó trả về một superset của tập tất cả các large k-itemsets. Hàm làm việc như sau. Đầu tiên, trong bước join, kết nối L_{k-1} với L_{k-1}

insert into C_k

select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

from L_{k-1} p, L_{k-1} q

where $p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$;

Tiếp theo, trong bước cắt tỉa, ta xoá tất cả các itemsets c thuộc C_k mà các tập con (k-1) của c không nằm trong L_{k-1}:

forall itemsets $c \in C_k$ **do**

forall *(k-1)*-**subsets** s **of** c **do**

if $(s \notin L_{k-1})$ **then**

delete c **from** C_k;

Ví dụ, Cho L_3 là { {1 2 3}, {1 2 4}, {1 3 4}, {1 3 5}, {2 3 4} }. Sau bước kết nối, C_4 sẽ là { {1 2 3 4}, {1 3 4 5}}. Bước cắt tỉa sẽ xoá itemsets {1 3 4 5} vì itemset {1 4 5} là không trong L_3. Sau đó chỉ còn {1 2 3 4} trong C_4.

Tính đúng đắn – Có $C_k \supseteq L_k$. Rõ ràng bất kỳ tập con nào của large itemset cũng cần phải có độ hỗ trợ cực tiểu. Do vậy, nếu mở rộng mỗi itemset trong L_{k-1} với tất cả các items có thể và sau đó xoá đi tất cả những cái mà có (k-1)-subsets của nó không có trong L_{k-1}, ta sẽ còn lại một superset của các itemsets trong L_k.

Kết nối tương đương với mở rộng L_{k-1} với mỗi item trong CSDL và sau đó xoá những itemsets mà với nó *(k-1)* itemset được tạo bằng cách xoá item thứ *(k-1)* là không có trong L_{k-1}. Điều kiện $p.item_{k-1} < q.item_{k-1}$ đơn giản đảm bảo rằng không sinh ra trùng nhau. Như vậy sau bước join, $C_k \supseteq L_k$. Với lý do tương tự, bước cắt tỉa, ở đó xoá khỏi C_k tất cả các itemsets mà *(k-1)*-subsets

của nó không có trong L_{k-1}, sẽ không xoá mất bất kỳ itemset nào có thể có trong L_k.

2.1.1.2 Hàm Subset

Các candidate itemsets C_k được sắp xếp trong một hash-tree. Một node của hash-tree hoặc chứa một danh sách các itemsets (một node lá) hoặc là một hash table (một node trong). Một node trong, mỗi bucket của hash table (lớp có cùng giá trị hàm băm) chỉ tới một node khác. Gốc của hash-tree được định nghĩa là có độ sâu 1. Một node trong ở độ sâu d chỉ tới các node tại độ sâu d+1. Các itemsets được lưu trong các lá. Khi thêm một itemset c, ta bắt đầu từ gốc và đi xuống đến khi gặp một lá. Tại một node trong ở độ sâu d, chúng ta quyết định đi theo nhánh nào bằng cách áp dụng hàm băm (hash function) cho item thứ d của itemset. Tất cả các nodes được tạo ban đầu như các node lá. Khi số lượng các itemsets trong một node lá vượt quá một ngưỡng nào đó, node lá được chuyển thành node trong.

Bắt đầu từ node gốc, hàm subset tìm tất cả các candidates chứa trong một giao dịch t như sau. Nếu ở tại lá, ta tìm itemset nào trong lá được chứa trong t và thêm các tham chiếu tới chúng vào tập trả lời. Nếu ở node trong và ta đã đến tới nó bằng cách dùng hàm băm cho item i, ta sẽ băm trên mỗi item đến sau i trong t và áp dụng đệ quy thủ tục này tới node trong bucket tương ứng. Với node gốc, chúng ta băm trên mọi item trong t.

Để biết vì sao hàm subset trả về tập các tham chiếu mong muốn, xem xét tại node gốc. Với bất kỳ itemset c nào chứa trong giao dịch t, item đầu tiên của c cần phải có trong t. Tại gốc, bằng cách băm trên mọi item trong t, đảm bảo rằng ta chỉ bỏ qua các itemsets bắt đầu với một item không có trong t. Tương tự các tham số áp dụng ở các độ sâu thấp hơn. Vì các items trong bất

kỳ itemset nào cũng được sắp thứ tự, nên nếu ta đạt đến node hiện thời bằng cách băm item i, thì chỉ cần xem xét các items xuất hiện sau i trong t.

2.1.2 Thuật toán AprioriTid

Thuật toán AprioriTid cho thấy trong hình 2.2, cũng dùng hàm apriori-gen để xác định các candidate itemsets trước khi lần duyệt bắt đầu. Đặc điểm đáng quan tâm của thuật toán này là CSDL D không được dùng cho việc đếm độ hỗ trợ sau lần duyệt đầu tiên. Tập $\overline{C_k}$ được dùng cho mục đích này. Mỗi thành viên của tập $\overline{C_k}$ là ở dạng <TID, $\{X_k\}$>, trong đó mỗi X_k là một large k-itemset tiềm năng biểu diễn trong giao dịch với TID. Với k=1, $\overline{C_1}$ tương ứng với CSDL_D, mặc dù về khái niệm mỗi item i được thay thế bởi itemset {i}. Với k>1, $\overline{C_k}$ được sinh ra bằng thuật toán (bước 10). Thành viên của $\overline{C_k}$ tương ứng với giao dịch t là <t.TID, $\{ c \in C_k \mid c$ được chứa trong t$\}$. Nếu một giao dịch không chứa bất kỳ candidate k-itemset nào, thì $\overline{C_k}$ sẽ không có một entry cho giao dịch này. Như vậy số lượng các entries trong $\overline{C_k}$ có thể là nhỏ hơn số lượng các giao dịch trong CSDL, đặc biệt cho các giá trị lớn của k. Hơn nữa, với các k giá trị lớn, mỗi entry có thể là nhỏ hơn giao dịch tương ứng vì giao dịch có thể chứa rất ít candidates. Nhưng, với các giá trị k nhỏ, mỗi entry có thể lớn hơn giao dịch tương ứng vì một entry trong C_k bao gồm tất cả các candidate k-itemset có trong giao dịch.

1) L_1 = {large 1-itemsets};
2) $\overline{C}_1$ = database $\mathcal{D}$;
3) **for** ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**
4) C_k = apriori-gen(L_{k-1}); // New candidates
5) $\overline{C}_k = \emptyset$;
6) **forall** entries $t \in \overline{C}_{k-1}$ **do begin**
7) // determine candidate itemsets in C_k contained
// in the transaction with identifier t.TID
$C_t = \{c \in C_k \mid (c - c[k]) \in t.\text{set-of-itemsets} \wedge$
$(c - c[k-1]) \in t.\text{set-of-itemsets}\}$;
8) **forall** candidates $c \in C_t$ **do**
9) c.count++;
10) **if** $(C_t \neq \emptyset)$ **then** $\overline{C}_k$ += $< t.\text{TID}, C_t >$;
11) **end**
12) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
13) **end**
14) Answer = $\bigcup_k L_k$;

Hình 2.2 Thuật toán AprioriTid

Ví dụ: Xem xét CSDL trong hình 3 và cho rằng độ hỗ trợ cực tiểu là 2 giao dịch. Gọi apriori-gen với L_1 tại bước 4 cho các candidate itemsets C_2. Trong các bước 6 đến 10, chúng ta đếm độ hỗ trợ của các candidates trong C_2 bằng cách lặp qua các entries trong $\overline{C_1}$ và sinh ra $\overline{C_2}$. Entry đầu tiên trong $\overline{C_1}$ là { {1} {3} {4} }, tương ứng với giao dịch 100. C_t tại bước 7 tương ứng với entry t này là { {1 3} }, vì {1 3} là một thành viên của C_2 và cả hai ({1 3} - {1}) và ({1 3} - {3}) là các thành viên của t.set-of-itemsets.

Gọi hàm apriori-gen với L_2, có được C_3. Duyệt một lần qua $\overline{C_2}$ và C_3 tạo ra $\overline{C_3}$. Chú ý rằng không có entry nào trong $\overline{C_3}$ cho các giao dịch với TIDs 100 và 400, vì chúng không chứa bất kỳ tập itemsets trong C_3. Candidate {2 3 5} trong C_3 trở thành large và là thành viên duy nhất của L_3. Khi sinh C_4 dùng L_3, kết quả rỗng và kết thúc.

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

$\overline{C}_1$

TID	Set-of-Itemsets
100	{ {1}, {3}, {4} }
200	{ {2}, {3}, {5} }
300	{ {1}, {2}, {3}, {5} }
400	{ {2}, {5} }

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

$\overline{C}_2$

TID	Set-of-Itemsets
100	{ {1 3} }
200	{ {2 3}, {2 5}, {3 5} }
300	{ {1 2}, {1 3}, {1 5}, {2 3}, {2 5}, {3 5} }
400	{ {2 5} }

L_2

Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

$\overline{C}_3$

TID	Set-of-Itemsets
200	{ {2 3 5} }
300	{ {2 3 5} }

C_3

Itemset	Support
{2 3 5}	2

L_3

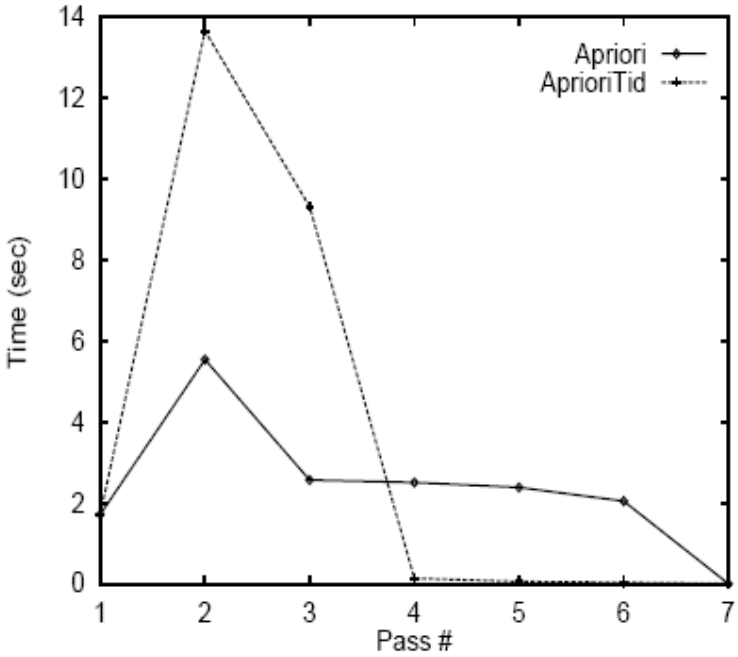
Itemset	Support
{2 3 5}	2

Hình 2.3 Ví dụ

2.1.3 Thuật toán AprioriHybrid

Không cần thiết phải dùng cùng một thuật toán trong tất cả các lần duyệt qua dữ liệu. Hình 2.4 cho thấy thời gian thực hiện cho Apriori và AprioriTid cho các lần duyệt khác nhau qua tập dữ liệu T10.I4.D100K kích thước 4.4MB (T10.I4.D100K có T=10, I=4, D=100, T là kích thước trung bình của giao dịch, I là kích thước trung bình của các large itemsets, D là số lượng các giao dịch). Trong các lần duyệt đầu, Apriori làm tốt hơn AprioriTid. Nhưng AprioriTid lại hơn Apriori trong các lần duyệt sau. Nguyên nhân: Apriori và AprioriTid dùng cùng một thủ tục sinh ra candidate và do đó đếm cùng các itemsets. Trong các lần duyệt sau, số lượng các candidate itemsets giảm. Nhưng, Apriori vẫn kiểm tra mọi giao dịch trong CSDL. Trong khi đó, không duyệt toàn bộ CSDL, AprioriTid duyệt $\overline{C_k}$ để thu

được số đếm độ hỗ trợ, và kích thước của $\overline{C_k}$ trở thành nhỏ hơn kích thước của CSDL. Khi tập $\overline{C_k}$ có thể vừa vặn trong bộ nhớ, thì thậm chí còn không phải chịu chi phí của việc ghi nó lên đĩa.



Hình 2.4: Thời gian thực hiện cho mỗi lần duyệt của Apriori và AprioriTid (T10.I14.D100K, độ hỗ trợ cực tiểu = 0.75%)

Dựa trên những quan sát này, người ta đã thiết kế một thuật toán lai, gọi là AprioriHybrid: dùng Apriori trong trong các lần duyệt đầu và chuyển tới AprioriTid khi nó cho rằng tập $\overline{C_k}$ tại cuối lần duyệt sẽ vừa vặn trong bộ nhớ. Dùng kinh nghiệm sau đây để đánh giá xem $\overline{C_k}$ có vừa vặn trong bộ nhớ không trong lần duyệt tiếp theo. Tại cuối mỗi lần duyệt hiện thời, ta có số đếm các candidates trong C_k. Từ đây, ta ước lượng kích thước của $\overline{C_k}$ sẽ là gì nếu nó được sinh ra. Kích thước này, tính bằng words, là

$$\sum_{\text{Candidates } c \in C_k} \text{support}(c) + \text{số các giao dịch}$$

Nếu $\overline{C_k}$ trong lần duyệt này là nhỏ đủ vừa trong bộ nhớ, và có các large candidates trong lần duyệt hiện tại ít hơn lần duyệt trước, ta sẽ chuyển tới AprioriTid. Điều kiện thứ hai được thêm vào để tránh chuyển khi $\overline{C_k}$ trong lần duyệt hiện tại vừa với bộ nhớ nhưng $\overline{C_k}$ trong lần duyệt sau lại không vừa.

Chuyển từ Apriori tới AprioriTid không bao gồm chi phí. Giả sử rằng quyết định chuyển từ Apriori tới AprioriTid tại cuối của lần duyệt thứ k. Trong lần duyệt thứ (k+1), sau khi tìm ra các candidate itemsets được chứa trong giao dịch, ta cũng sẽ phải thêm IDs của chúng vào $\overline{C_{k+1}}$. Như vậy có một chi phí bổ sung phải chịu trong lần duyệt này chỉ liên quan tới việc chạy Apriori. Chỉ trong lần duyệt thứ (k+2) khi thực sự bắt đầu chạy AprioriTid. Như vậy, nếu không có large (k+1)-itemsets, hoặc không có (k+2)-candidates, ta sẽ phải chịu chi phí của việc chuyển mà không thu được bất kỳ cái gì từ việc dùng AprioriTid.

So sánh hiệu năng của AprioriHybrid với Apriori và AprioriTid với cùng các tập dữ liệu: AprioriHybrid thực hiện tốt hơn Apriori trong phần lớn các trường hợp. Với trường hợp AprioriHybrid làm tốt hơn Apriori từ lần duyệt có sự chuyển nhưng việc chuyển lại xuất hiện ở lần duyệt cuối cùng; như vậy AprioriHybrid chịu chi phí của việc chuyển mà không thực sự có lợi ích. Nói chung, thuận lợi của AprioriHybrid hơn Apriori phụ thuộc vào kích thước của tập $\overline{C_k}$ suy biến như thế nào trong các lần duyệt sau. Nếu $\overline{C_k}$ giữ nguyên large đến gần cuối và sau đó đột ngột rớt xuống, thì sẽ không thu được nhiều từ việc dùng AprioriHybrid vì ta chỉ có thể dùng AprioriTid cho một chu kỳ thời gian ngắn sau khi chuyển (như với tập dữ liệu T20.I16.D100K). Ngược lại, nếu có một sự suy giảm dần trong kích thước của $\overline{C_k}$, AprioriTid có thể được dùng một lúc sau khi chuyển, và ý nghĩa cải tiến có thể thu được trong thời gian thực hiện.

2.2. Cải tiến hiệu quả thuật toán Apriori

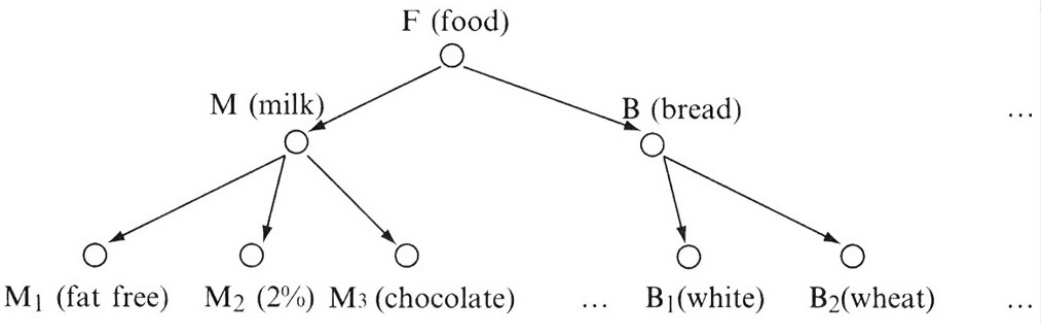
Vì lượng của dữ liệu xử lý trong khai phá các frequent itemsets có xu hướng rất lớn, nên việc phát minh ra những thuật toán hiệu quả để khai phá dữ liệu như vậy là rất quan trọng. Thuật toán Apriori cơ bản duyệt CSDL một vài lần, phụ thuộc vào kích thước của frequent itemset lớn nhất. Một vài tinh chỉnh đã được đưa ra tập trung vào việc giảm số lượng lần duyệt CSDL, số lượng các candidate itemsets được tính toán trong mỗi lần duyệt, hoặc cả hai.

Partition-based Apriori (Apriori dựa trên Partition) là thuật toán đòi hỏi chỉ 2 lần duyệt CSDL giao dịch. CSDL được chia thành các phần (partitions) rời nhau, mỗi phần đủ nhỏ để vừa với bộ nhớ sẵn có. Trong lần duyệt đầu tiên, thuật toán đọc mỗi partition và tính toán các frequent itemsets địa phương trong mỗi partition. Trong lần duyệt thứ hai, thuật toán tính toán độ hỗ trợ của tất cả các frequent itemsets địa phương đối với toàn bộ CSDL. Nếu itemset là frequent với toàn bộ CSDL, nó chắc chắn là frequent trong ít nhất một partition. Đó là kinh nghiệm dùng trong thuật toán. Do đó, lần duyệt thứ 2 qua CSDL đếm superset của tất cả các frequent itemsets tiềm năng.

Lấy mẫu: Khi kích thước của CSDL rất lớn, việc lấy mẫu trở thành cách tiếp cận hấp dẫn với việc khai phá dữ liệu. Thuật toán dựa trên mẫu điển hình yêu cầu 2 lần duyệt CSDL. Thuật toán đầu tiên lấy mẫu từ CSDL và sinh ra tập các candidate itemsets sẽ là frequent trong toàn bộ CSDL với khả năng chắc chắn cao. Trong lần duyệt chuỗi con trên CSDL, thuật toán tính toán chính xác độ hỗ trợ của các itemsets này và độ hỗ trợ của ranh giới negative. Nếu không có itemset nào trong negative border là frequent, thì thuật toán đã khám phá tất cả các frequent itemsets. Mặt khác, một vài superset của một itemset trong negative border có thể là frequent, nhưng độ hỗ trợ của nó chưa được tính toán. Thuật toán sinh ra và tính toán tất cả các frequent itemsets tiềm năng trong các lần duyệt chuỗi con trên CSDL.

Cập nhật dần (Incremental updating): Việc tìm ra các frequent itemsets trong các CSDL lớn là rất có giá trị, các kỹ thuật *incremental updating* cần được phát triển để bảo trì các frequent itemsets đã phát hiện được (và các luật kết hợp tương ứng) với mục đích tránh khai phá lại toàn bộ CSDL. Các cập nhật trên CSDL có thể không chỉ làm sai một vài frequent itemsets đang tồn tại mà con chuyển một vài itemsets mới thành frequent. Vấn đề bảo trì các frequent itemsets đã phát hiện từ trước trong các CSDL lớn và biến động không đơn giản. Ý tưởng là dùng lại thông tin của các frequent itemsets cũ và tích hợp thông tin về độ hỗ trợ của các frequent itemsets mới để giảm về căn bản lượng các candidates được kiểm tra lại.

Khai phá luật kết hợp tổng quát: Trong nhiều ứng dụng, các kết hợp của các items dữ liệu thường xuất hiện tại *mức khái niệm tương đối cao*. Ví dụ, một phân cấp của các thành phần thức ăn được biểu diễn trong hình 2.5, trong đó M (sữa), B (bánh mì), là khái niệm phân cấp, có thể có một vài nội dung thành phần con. Các thành phần mức thấp nhất trong phân cấp là các loại sữa và bánh mì. Có thể khó tìm ra quy tắc với mức khái niệm nguyên thủy, như là sữa socola và bánh mì bằng lúa mì. Nhưng dễ dàng tìm ra quy tắc ở mức khái niệm cao, như: hơn 80% khách hàng mua sữa cũng mua bánh mì.



Hình 2.5: Một ví dụ của cây phân cấp khái niệm cho khai phá các frequent itemsets nhiều mức

Do đó, khai phá ra các frequent itemsets tại mức trừu tượng tổng quát hoặc tại các mức đa khái niệm (multiple-concept levels) là rất quan trọng.

Vì số lượng dữ liệu được xử lý trong khai phá các luật kết hợp có xu hướng rất lớn, nên đưa ra những thuật toán hiệu quả để xây dựng việc khai phá trên những dữ liệu như vậy là rất quan trọng. Trong phần này, trình bày một vài thuật toán cải tiến cho Apriori.

2.2.2 Phương pháp FP-tree

Phương pháp Frequent pattern growth (FP-growth) là một phương pháp hiệu quả để khai phá các frequent itemsets trong các CSDL lớn. Thuật toán khai phá các frequent itemsets mà không có quá trình sinh candidate tốn thời gian mà là yếu tố cần thiết cho Apriori. Khi CSDL lớn, FP-growth đầu tiên thực hiện một phép chiếu CSDL của các frequent items; sau đó nó chuyển tới khai phá bộ nhớ chính bằng cách xây dựng cấu trúc dữ liệu gọn nhẹ được gọi là FP-tree. [9]

Để giải thích thuật toán, ta dùng CSDL giao dịch trong bảng 2.1 và chọn ngưỡng độ hỗ trợ cực tiểu là 3.

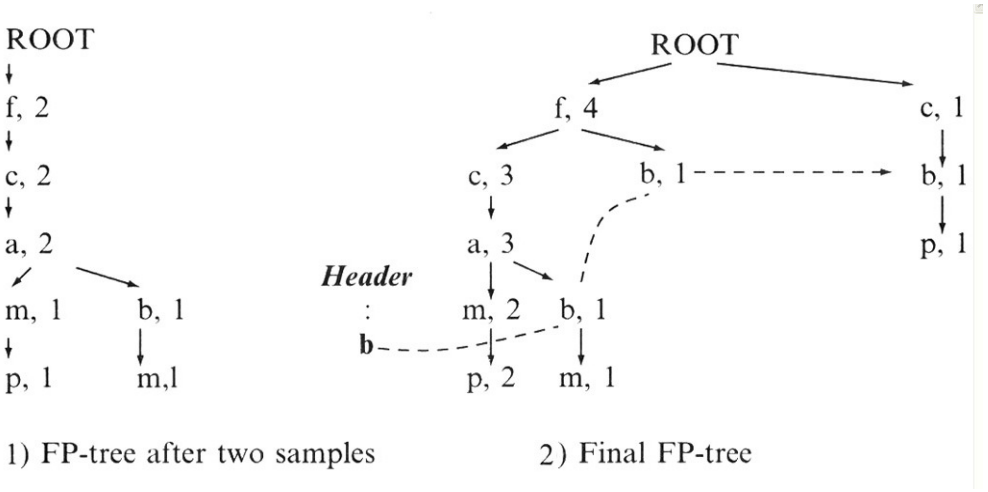
Bảng 2.1 Cơ sở dữ liệu giao dịch T

Cơ sở dữ liệu giao dịch T	
TID	**Itemset**
01	f, a, c, d, g, i, m, p
02	a, b, c, f, l, m, o
03	b, f, h, j, o
04	b, c, k, s, p
05	a, f, c, e, l, p, m n

Thứ nhất, việc duyệt CSDL T lấy danh sách L các frequent items xuất hiện lớn hơn hoặc bằng 3 lần trong CSDL. Những items này là (với độ hỗ trợ của nó): L = {(f, 4), (c, 4), (a, 3), (b, 3), (m, 3), (p, 3)}

Các items được hiển thị trong thứ tự giảm dần của tần số xuất hiện. Thứ tự này là quan trong vì mỗi đường đi của FP-tree sẽ theo thứ tự này.

Thứ 2, gốc của cây, đánh nhãn ROOT, được tạo ra. CSDL T được duyệt lần thứ 2. Duyệt giao dịch đầu tiên dẫn đến xây dựng nhánh đầu tiên của FP-tree: {(f, 1), (c, 1), (a, 1), (m, 1), (p, 1)}. Chỉ những items này trong danh sách các frequent items L được lựa chọn. Các chỉ số cho các node trong nhánh (tất cả là 1) biểu diễn số tích luỹ của các mẫu tại node này trong cây, và tất nhiên sau mẫu đầu tiên, tất cả là 1. Thứ tự của các nodes không phải trong mẫu mà là trong danh sách các frequent items L. Với giao dịch thứ 2, vì nó dùng chung các items f, c và a, nó dùng chung tiền tố {f, c, a} với nhánh trước và mở rộng tới nhánh mới {(f, 2), (c, 2), (m, 1), (p, 1),} tăng thêm 1 cho các chỉ số của tiền tố chung. Phiên bản trung gian mới của FP-tree, sau 2 mẫu từ CSDL được đưa ra trong hình 2.6.1. Các giao dịch còn lại có thể được chèn vào tương tự và cây FP cuối cùng cho thấy trong hình 2.6.2.



Hình 2.6: FP-tree cho CSDL T trong bảng 2.1

Để thuận tiện cho việc duyệt trên cây, một item header table được xây dựng, trong đó mỗi item trong danh sách L kết nối các nodes trong FP-tree với các giá trị của nó qua các đường nối node (node-links). Tất cả các nodes f được nối trong 1 danh sách, tất cả các nodes c trong danh sách khác,… Để đơn giản, chỉ biển diễn danh sách cho các node b trong hình 2.6.2. Dùng cấu trúc cây thu gọn (compact-tree), Thuật toán FP-growth khai phá tập đầy đủ các frequent itemsets.

Tương ứng với danh sách L của frequent items, tập đầy đủ các frequent itemsets có thể được chia thành các tập con (6 với ví dụ này) không chồng nhau: 1) frequent itemsets có item p (cuối của danh sách L); 2) itemsets có item m nhưng không có p; 3) frequent itemsets với b mà không có cả m và p… 6) large itemsets chỉ có f. Sự phân lớp này là hợp lệ cho ví dụ ở đây, nhưng các luật giống nhau có thể được sử dụng cho các CSDL khác và các danh sách L khác.

Dựa trên kết nối liên kết node, ta thu thập tất cả các giao dịch có p tham gia vào bằng cách bắt đầu từ header table của p và tiếp theo các node-links của p. Trong ví dụ này, 2 đường (paths) sẽ được chọn trong FP-tree: {(f, 4), (c, 3), (a, 3), (m, 2), ((p, 2)} và {(c, 1), (b, 1), (p, 1)}, trong đó các mẫu với frequent item p là {(f, 2), (c, 2), (a, 2), (m, 2), ((p, 2) và {(c, 1), (b, 1), (p, 1)}. Giá trị ngưỡng cho trước (3) thoả mãn chỉ frequent itemsets {(c, 3), (p, 3)}, hoặc đơn giản {c, p}. Tất cả các itemsets khác với p đều dưới giá trị ngưỡng.

Tập con tiếp theo của frequent itemsets là những cái có m và không có p. FP-tree nhận ra các paths {(f, 4), (c, 3), (a, 3), (m, 2)} và {(f, 4) (c, 3), (a, 3), (b, 1), (m, 1)}, hoặc các samples tích luỹ tương ứng {(f, 2), (c, 2), (a, 2), (m, 2)} và (f, 1), (c, 1),(a, 1), (b, 1), (m, 1)}. Việc phân tích các mẫu phát hiện frequent itemset {(f, 3), (c, 3), (a, 3), (m, 3)} hoặc, đơn giản, {f, c, a, m}.

Việc lặp lại cùng quá trình cho các tập con 3 đến 6 trong ví dụ này, thêm các frequent itemsets nữa được khai phá. Đây là những itemsets {f, c, a} và {f, c}, nhưng chúng là tập con của frequent itemset {f, c, a, m}. Do đó, đáp án cuối cùng trong phương pháp FP-growth là tập các frequent itemsets {{c, p}, {f, c, a, m}}.

Thử nghiệm đã cho thấy rằng thuật toán FP-growth nhanh hơn thuật toán Apriori. Một vài kỹ thuật tối ưu được thêm vào thuật toán FP-growth, và do đó tồn tại một số các phiên bản khác cho việc khai phá các dãy và các mẫu với các ràng buộc.

2.2.3 Thuật toán PHP

(Thuật toán băm và cắt tỉa hoàn hảo – perfect hashing and pruning)

Trong thuật toán DHP, nếu có thể định nghĩa một bảng băm lớn để mỗi itemsets khác nhau được ánh xạ tới các vị trí khác nhau trong bảng băm, thì các mục của bảng băm cho số đếm thực sự của mỗi itemset trong CSDL. Trong trường hợp đó, ta không có bất kỳ false positives nào và kết quả là xử lý quá mức cho việc đếm số lần xuất hiện của mỗi itemset được loại bỏ. Ta đã biết lượng dữ liệu phải duyệt trong quá trình phát hiện large itemset là một vấn đề liên quan đến hiệu năng. Giảm số lượng các giao dịch phải duyệt và cắt bớt số các items trong mỗi giao dịch sẽ cải tiến được hiệu quả khai phá dữ liệu trong các chặng sau.

Thuật toán này dùng băm hiệu quả cho bảng băm được sinh ra ở mỗi lần duyệt và giảm kích thước của CSDL bằng cách cắt tỉa các giao dịch không chứa bất kỳ frequent item nào. Do vậy thuật toán được gọi là *Perfect Hashing and Pruning (PHP)*. Thuật toán như sau:

Trong lần duyệt đầu tiên, bảng băm với kích thước bằng với các items riêng biệt trong CSDL được tạo ra. Mỗi item riêng biệt trong CSDL được ánh

xạ tới vị trí khác nhau trong bảng băm, và phương pháp này được gọi là *perfect hashing*. Phương thức *add* của bảng băm thêm một mục mới nếu một mục cho item x chưa tồn tại trong bảng băm và khởi tạo đếm của nó bằng 1, nếu không nó tăng số đếm của x trong bảng lên 1. Sau lần duyệt đầu tiên, bảng băm chứa số chính xác lần xuất hiện của mỗi item trong CSDL. Chỉ duyệt một lần qua bảng băm, nằm trong bộ nhớ, thuật toán dễ dàng sinh ra các frequent 1-itemsets. Sau thao tác đó, phương thức cắt tỉa *prune* của bảng băm cắt tỉa tất cả các mục mà độ hỗ trợ của chúng nhỏ hơn độ hỗ trợ cực tiểu.

Trong các lần duyệt tiếp theo, thuật toán cắt tỉa CSDL bằng cách loại bỏ các giao dịch không có items nào trong frequent itemsets, và cũng cắt các items mà không là frequent khỏi các giao dịch. Cùng lúc, nó sinh ra các candidate k-itemsets. Quá trình này tiếp tục đến khi không có F_k mới nào được tìm ra. Thuật toán cho thấy trong hình 2.7. Thuật toán này tốt hơn thuật toán DHP vì sau khi hình thành bảng băm, nó không cần đến số lần xuất hiện của các candidate k-itemsets như trong thuật toán DHP. Thuật toán cũng tốt hơn thuật toán Apriori vì tại mỗi lần lặp, kích thước của CSDL được giảm đi, và nó đem lại hiệu năng cao cho thuật toán khi kích thước của CSDL rất lớn mà số lượng các frequent itemsets lại tương đối nhỏ.

Input: Database
Output: Frequent k-itemset

/* *Database = set of transactions;*
Items = set of items;
transaction = <TID, {x ∈ Items}>;
F_1 is a set of frequent l-itemsets */

$F_1 = \phi$;

/* *H_1 is the hash table for 1-itemsets*
Read the transactions, and count the occurrences of each item, and generate H_1 */

for each transaction $t \in$ Database do begin
    for each item x in t do
        $H_1.add(x)$;
end;

// *Form the set of frequent 1-itemset*

for each itemset y in H_1 do
    if $H_1.hassupport(y)$
    then $F_1 = F_1 \cup y$
end

/* *Remove the hash values without the minimum support* */

$H_1.prune(\min \sup)$;
$D_1 = Database$;

// *D_k is the pruned database*

/* *Find F_k, the set of frequent k-itemsets, where $k \geq 2$ and prune the database* */

$k = 2$;
repeat
    $D_k = \phi$;
    $F_k = \phi$;
    for each transaction $t \in D_{k-1}$ do begin
        // *w is k-1 subset of items in t*
        if $\forall w \mid w \notin F_{k-1}$
        then skip *t*;
        else
            $items = \phi$;
            for each *k* -itemset *y* in *t* do
                if $\neg\exists z \mid z = k-1$ subset of y
                    $\wedge \neg H_{k-1}.hassupport(z)$
                then $H_k.add(y)$;
                items=*items* $\cup y$;
            end
            $D_k = D_k \cup t$ // *such that t contains*
                                // *items only in the set items*
    end

    for each itemset *y* in H_k do
        if $H_k.hassupport(y)$
        then $F_k = F_k \cup y$
    end

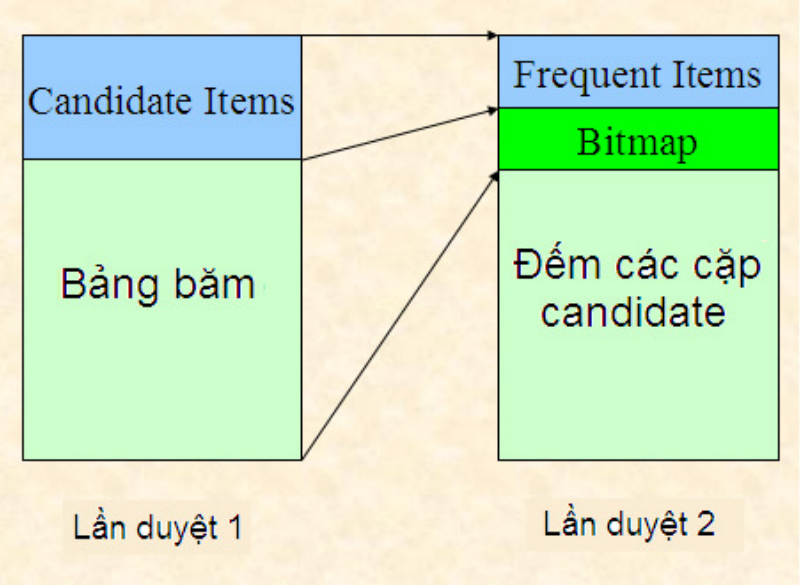
    /* *Remove the hash values without the minimum support from* H_k */
    $H_k.prune(\min sup)$;
    k++;
until $F_{k-1} = \phi$;
Answer = $\cup_k F_k$;

Hình 2.7 Thuật toán PHP

Thêm nữa, sau mỗi lần lặp, CSDL D_k chứa các giao dịch chỉ với các frequent items. Thuật toán hình thành tất cả các tập con k items trong mỗi giao dịch và chèn tất cả các tập con k-1 lớn (large) vào bảng băm. Vì lý do này thuật toán không để thiếu bất kỳ frequent itemset nào. Vì thuật toán thực hiện cắt tỉa trong khi chèn các candidate k-itemsets vào H_k, kích thước của bảng băm sẽ không lớn và vừa với bộ nhớ.

2.2.4 Thuật toán PCY

Park, Chen, và Yu đưa ra việc dùng hash table (bảng băm) để xác định trong lần duyệt đầu tiên (khi L_1 đang được xác định) nhiều cặp không thể là frequent. Thực tế có thuận lợi là bộ nhớ chính thường lớn hơn nhiều số lượng các items. Trong 2 lần duyệt để tìm L_2, bộ nhớ chính được cho thấy trong hình 2.8.



Hình 2.8 Bộ nhớ với 2 lần duyệt của thuật toán PCY

Giả sử rằng dữ liệu được lưu như một flat file, với các bản ghi bao gồm basket ID và một danh sách các items của nó.

Lần duyệt 1:

— (a) Đến số lần xuất hiện của tất cả các items

— (b) Với mỗi bucket, bao gồm các items $\{i_1, \ldots i_k\}$, băm tất cả các cặp tới một bucket của bảng băm, và tăng số đếm của bucket lên 1

— (c) Cuối lần duyệt, xác định L_1, các items với số đếm ít nhất = s

— (d) Cũng tại cuối lần duyệt, xác định những buckets với độ hỗ trợ ít nhất là s

* Điểm mấu chốt: một cặp (i, j) không thể là frequent trừ khi nó băm tới một frequent bucket, vì vậy các cặp mà băm tới các buckets khác không cần là candidate trong C_2.

Thay bảng băm bởi một *bitmap*, với một bit cho một bucket: 1 nếu bucket là frequent, 0 nếu không là frequent.

Lần duyệt 2:

— (a) Bộ nhớ chính giữ một danh sách của tất cả các frequent items, nghĩa là L_1.

— (b) Bộ nhớ chính cũng giữ bitmap (bản đồ bit) tổng kết các kết quả của việc băm từ lần duyệt 1.

* Điểm mấu chốt: Các buckets cần dùng 16 hoặc 32 bits cho một số đếm (count), nhưng nó được nén vào 1 bit. Như vậy, thậm chí bảng băm chiếm gần như toàn bộ bộ nhớ chính trong lần duyệt 1, bitmap của nó chiếm không lớn hơn 1/16 bộ nhớ chính trong lần duyệt thứ 2

— (c) Cuối cùng, bộ nhớ chính cũng giữ một bảng với tất cả các cặp candidate và các đếm của chúng. Cặp (i, j) có thể là một candidate trong C_2 chỉ khi tất cả các điều sau là đúng:

i) i là trong L_1.

ii) j là trong L_1.

iii) (i, j) băm tới một frequent bucket.

Điều kiện cuối cùng là phân biệt PCY với a-priori đơn giản và giảm các yêu cầu về bộ nhớ trong lần duyệt 2

— (d) Trong lần duyệt 2, ta xem xét mỗi basket, và mỗi cặp các items của nó, thực hiện các kiểm tra theo nguyên tắc nêu trên. Nếu 1 cặp đảm bảo tất cả các điều kiện, cộng thêm vào số đếm của nó trong bộ nhớ, hoặc tạo một mục cho nó nếu nó chưa tồn tại.

Khi nào PCY hơn Apriori? Khi có quá nhiều cặp các items từ L_1, không thể để vừa một bảng các cặp candidate và các đếm của chúng trong bộ nhớ chính, thì số các frequent buckets trong thuật toán PCY là đủ nhỏ để giảm kích thước của C_2 xuống vừa với bộ nhớ (thậm chí cả với khi 1/16 bộ nhớ dùng cho bitmap).

Khi nào phần lớn các buckets sẽ là infrequent trong PCY? Khi có ít cặp frequent, phần lớn các cặp là infrequent mà thậm chí khi các đếm của tất cả các cặp băm tới một bucket đã có được cộng thêm, chúng vẫn không chắc chắn cộng được tới lớn hơn hoặc bằng s.

2.2.5 Thuật toán PCY nhiều chặng

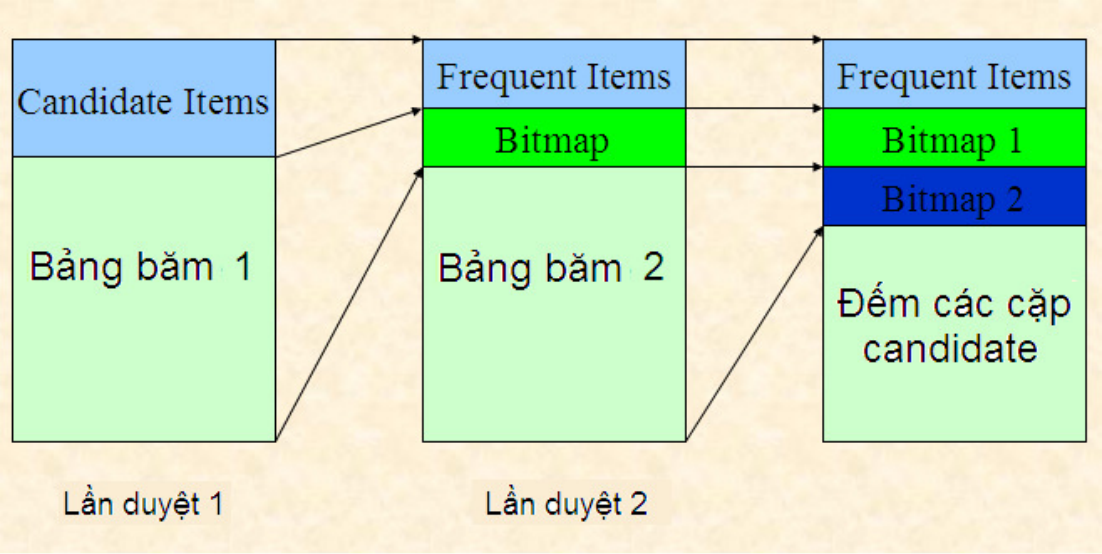
Thay cho việc kiểm tra các candidates trong lần duyệt 2, ta chạy một bảng băm khác (hàm băm khác!) trong lần duyệt 2, nhưng ta chỉ băm những cặp mà thoả mãn điều kiện kiểm tra của PCY; nghĩa là, cả hai đều trong L_1 và được băm tới một frequent bucket trong lần duyệt 1. Ở lần duyệt thứ 3, giữ các bitmaps từ cả hai bảng băm, và coi 1 cặp là một candidate trong C_2 chỉ khi:

— (a) Cả hai items đều trong L_1

— (b) Cặp được băm tới frequent buckets trong lần duyệt 1

— (c) Cặp cũng được băm tới frequent bucket trong lần duyệt 2

Hình 2.9 giới thiệu việc dùng bộ nhớ. Lược đồ này có thể được mở rộng tới nhiều lần duyệt hơn, nhưng có một giới hạn, vì thực tế bộ nhớ trở nên đầy với bitmaps, và ta không thể đếm được candidate nào.



Hình 2.9 Sử dụng bộ nhớ cho các bảng băm nhiều chặng

Khi nào nhiều bảng băm có ích? Khi phần lớn các buckets trên lần duyệt đầu tiên của PCY có các đếm cách xa dưới ngưỡng *s*. Khi đó, có thể gấp đôi các đếm trong buckets và vẫn có phần lớn các buckets dưới ngưỡng.

Khi nào nhiều chặng có ích? Khi số lượng các frequent buckets trên lần duyệt đầu tiên là cao (ví dụ 50%), nhưng không phải tất cả các buckets. Khi đó, lần băm thứ 2 với một vài cặp bị bỏ qua có thể giảm số lượng các frequent buckets một cách đáng kể.

2.3. Thuật toán phân lớp bằng học cây quyết định

ID3 và C4.5 là các thuật toán được Quilan giới thiệu để thu được các mô hình phân lớp từ dữ liệu (cũng được gọi là các cây quyết định).

Cây quyết định quan trọng không phải vì nó tổng kết từ tập huấn luyện mà ta mong đợi nó sẽ **phân lớp chính xác** các trường hợp mới. Như vậy khi xây dựng các mô hình phân lớp ta cần có cả dữ liệu huấn luyện để xây dựng mô hình, cả dữ liệu kiểm thử để đánh giá cây quyết định tốt mức nào.

Cho một tập các bản ghi. Mỗi bản ghi có cùng một cấu trúc, gồm một số cặp thuộc tính/giá trị. Một trong các thuộc tính này biểu diễn phân loại của bản ghi. Bài toán là xác định cây quyết định dựa trên các câu trả lời với các câu hỏi về các thuộc tính không phân loại (non-category) dự báo chính xác giá trị của thuộc tính phân loại. Thông thường thuộc tính phân loại chỉ lấy các giá trị {*true, false*}, hoặc {*success, failure*} hoặc tương tự. Trong bất kỳ trường hợp nào, một trong các giá trị của nó cũng có nghĩa là sai.

Các thuộc tính không phân loại có thể là rời rạc hoặc liên tục. ID3 không trực tiếp xử lý với những trường hợp thuộc tính liên tục.

Ý tưởng cơ sở của ID3 là:

— Trong cây quyết định mỗi node trong tương ứng với một thuộc tính không phân loại và mỗi cành tới một giá trị có thể của thuộc tính đó. Lá của cây chỉ giá trị mong đợi của thuộc tính phân lớp cho các bản ghi được mô tả bởi đường đi từ gốc tới lá. [Đây là định nghĩa Cây quyết định là gì]

— Trong cây quyết định tại mỗi node cần tương ứng với thuộc tính không phân loại chứa nhiều thông tin nhất trong số các thuộc tính chưa được xem xét trong đường đi từ gốc . [Việc này thiết lập cây quyết định "tốt"]

— *Entropy* được dùng để đo thông tin thế nào là node. [Điều này định nghĩa "tốt" là gì]

C4.5 là một mở rộng của ID3 với tính toán các giá trị thiếu, các miền giá trị liên tục, cắt tỉa cây quyết định, suy diễn luật…

2.3.1 Các định nghĩa

Nếu có n thông điệp có khả năng xảy ra như nhau, thì xác xuất p của mỗi thông điệp là 1/n và thông tin chuyển tới bởi thông điệp là $-\log_2(p) = \log_2(n)$. Đó là, nếu có 16 thông điệp, thì $\log_2(16) = 4$ và ta cần 4 bits để định danh mỗi thông điệp.

Nói chung, nếu ta có phân tán xác xuất (probability distribution) P = (p1, p2, .., pn) thì thông tin chuyển tải bởi sự phân tán này -*Entropy* của P- là:

$I(P) = -(p1*\log_2(p1) + p2*\log_2(p2) + .. + pn*\log_2(pn))$

Nếu tập T của các bản ghi được phân chia thành các lớp riêng biệt C1, C2, …, Ck trên cơ sở giá trị của thuộc tính phân loại, thì thông tin cần để xác định lớp của phần tử của T là **Info(T)** = I(P), trong đó P là phân tán xác suất của các phần (C1, C2,..Ck):

P = (|C1|/|T|, |C2|/|T|, ..., |Ck|/|T|)

Nếu đầu tiên ta chia phần T trên cơ sở giá trị của các thuộc tính không phân loại X thành các tập T1, T2, .. Tn thì thông tin cần để xác định lớp của một phần tử của T trở thành trọng số trung bình của thông tin cần để xác định lớp của một phần tử của T, nghĩa là trọng số trung bình của Info(Ti):

$$Info(X, T) = \sum_{i=1}^{n} Info_x(T) = - \sum ((|T_i| / |T|) * Info(T_i))$$

Giá trị lợi ích Gain(X,T) được định nghĩa là

Gain(X,T) = Info(T) - Info(X,T)

Điều này biểu diễn sự khác nhau giữa *thông tin cần để xác định một phần tử của T và thông tin cần để xác định một phần tử của T sau khi giá trị thuộc tính X đã được biết*, đó là, *lợi ích thông tin (gain) trong thuộc tính X*.

Ta có thể dùng khái niệm **gain** để giới hạn (rank) các thuộc tính và để xây dựng các cây quyết định với mỗi node được định vị thuộc tính với gain lớn nhất trong số các thuộc tính chưa được xem xét trong đường đi từ gốc.

2.3.2 Thuật toán ID3

Thuật toán ID3 được dùng để xây dựng cây quyết định, cho một tập các thuộc tính không phân loại C1, C2, …, Cn, thuộc tính phân loại C và tập các bản ghi để huấn luyện T.

function ID3 (R: tập các thuộc tính không phân loại,
C: thuộc tính phân loại,
S: tập huấn luyện) trả lại một cây quyết định;
begin
Nếu S rỗng, trả về một node đơn lẻ với giá trị Failure;
Nếu S chứa các bản ghi tất cả có cùng giá trị cho thuộc tính
phân loại, trả về một node đơn lẻ với giá trị đó.
Nếu R là trống, thì trả về một node đơn lẻ với giá trị có
tần suất lớn nhất trong các giá trị của thuộc tính
phân loại mà tìm thấy trong các bản ghi của S; [chú ý rằng
sau đó sẽ có các lỗi, đó là, các bản ghi mà sẽ bị phân lớp
không rõ ràng];
Cho D là thuộc tính với Gain lớn nhất Gain(D,S)
trong số các thuộc tính trong R;
Cho {dj | j=1,2, .., m} là các giá trị của thuộc tính D;
Cho {Sj | j=1,2, .., m} là các tập con của S gồm thứ tự

các bản ghi với giá trị dj cho thuộc tính D;

Trả về cây với gốc gán nhãn D và các cung được gán nhãn

d1, d2, .., dm lần lượt tới các cây

ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm);

end ID3;

Dùng tỷ suất lợi ích (Gain Ratios)

Khái niệm lợi ích (Gain) có xu hướng ưu tiên các thuộc tính có số lượng lớn các giá trị. Ví dụ, nếu một thuộc tính D có giá trị riêng biệt cho mỗi bản ghi, thì Info(D,T) là 0, như vậy Gain(D,T) là cực đại. Để khắc phục, dùng tỷ lệ sau thay cho Gain:

GainRatio(D,T) = Gain(D,T) / SplitInfo(D,T)

Trong đó SplitInfo(D,T) là thông tin do phân tách của T trên cơ sở giá trị của thuộc tính phân loại D. SplitInfo(D,T) là

I(|T1|/|T|, |T2|/|T|, .., |Tm|/|T|)

Trong đó {T1, T2, .. Tm} là sự phân hoạch T do giá trị của D.

2.3.3 Các mở rộng của C4.5

C4.5 mở rộng một số xử lý từ thuật toán gốc ID3:

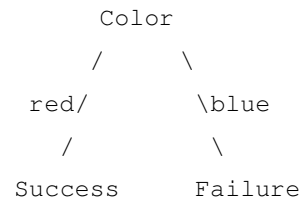
Trong việc xây dựng cây quyết định: Xử lý các tập huấn luyện có các bản ghi chứa giá trị thuộc tính thiếu bằng cách đánh giá lợi ích, hoặc tỷ lệ lợi ích cho một thuộc tính chỉ qua xem xét các bản ghi có giá trị của thuộc tính đó.

Trong việc dùng một cây quyết định, ta có thể phân lớp các bản ghi có các giá trị thuộc tính thiếu bằng cách đưa ra kết quả là dự đoán xác suất của mỗi kết quả khác nhau.

Xử lý với trường hợp các thuộc tính với phạm vi liên tục (**continuous ranges**) như sau. Có thuộc tính Ci liên tục. Kiểm tra các giá trị của thuộc tính này trong tập huấn luyện. Nói chúng là theo thứ tự tăng, A1, A2, ..,Am. Vậy cho mỗi giá trị Aj, j=1,2,..m, ta phân hoạch (partition) các bản ghi thành những phần mà có các giá trị Ci từ nhỏ tới Aj, và những phần có giá trị lớn hơn Aj. Với mỗi phần phân hoạch này ta tính toán gain, hoặc gain ratio, và chọn partition mà cực đại lợi ích (gain).

Cắt tỉa cây quyết định: Cây quyết định xây dựng dùng tập huấn luyện, với cách xây dựng cây là xử lý chính xác với phần lớn các bản ghi của tập huấn luyện. Thực tế, để làm như vậy, cây có thể trở thành quá phức tạp, với các đường đi thậm chí rất dài.

Việc cắt tỉa cây quyết định được làm bằng cách thay thế toàn bộ cây con bằng một node lá. Sự thay thế thực hiện nếu một luật quyết định xây dựng mà tỷ suất lỗi trong cây con là lớn hơn trong lá đơn lẻ. Ví dụ, nếu cây quyết định đơn giản



Được xây dựng với một bản ghi thành công màu đỏ và 2 bản ghi lỗi màu xanh, và như vậy trong tập kiểm thử ta tìm thấy 3 lỗi đỏ và 1 thành công xanh, ta có thể xem xét thay thế cây con này bằng một node lỗi (Failure) đơn lẻ. Sau khi thay thế ta sẽ chỉ có 2 lỗi thay vì 5 lỗi.

CHƯƠNG 3. ÁP DỤNG KHAI PHÁ TRÊN CSDL NGÀNH THUẾ

3.1. CSDL ngành Thuế

Áp dụng công nghệ tin học vào công tác quản lý Thuế từ những năm 1986, đến nay ngành Thuế đã xây dựng được hệ thống Công nghệ thông tin đồ sộ, đáp ứng được nhiệm vụ quản lý Thuế trong giai đoạn mới. Từ những ứng dụng phát triển trên máy đơn lẻ, đến nay toàn ngành đã có một CSDL phân tán tại 64 Cục Thuế trên cả nước. Hệ thống kết nối mạng máy tính, trao đổi thông tin, dữ liệu toàn ngành, từ Tổng cục đến 64 Cục Thuế và gần 700 Chi cục Thuế quận, huyện. Hệ thống các ứng dụng phục vụ các công tác đăng ký và cấp mã số thuế, hệ thống quản lý thu thuế tự động hoá các khâu xử lý quan trọng trong qui trình quản lý như quản lý số phải thu, quản lý số thu, quản lý nợ tính thuế, tính nợ, tổng hợp các báo cáo kế toán, thống kê thuế…

Sở hữu một kho thông tin liên quan đến lĩnh vực Thuế, CSDL ngành Thuế đóng một vai trò quan trọng không chỉ trong ngành mà còn có giá trị với cả nước. Một phần thông tin trong CSDL ngành Thuế - đó là thông tin liên quan đến các tổ chức, cá nhân nộp thuế - sẽ góp phần đóng góp cho CSDL quốc gia ngành Tài chính.

Trước đây, CSDL ngành Thuế mới được sử dụng phục vụ các tác nghiệp hàng ngày, các báo cáo, thống kê. Những năm gần đây, những năm đầu của thời kỳ Cải cách Thuế, CSDL ngành Thuế mới đáp ứng một phần cho công tác phân tích thông tin.

Trong giai đoạn Cải cách hành chính về Thuế, ngành Thuế đã đưa dần thực hiện cơ chế tự khai tự tính, tự nộp thuế. Với nhiệm vụ trọng tâm là xây dựng lại toàn bộ quy trình quản lý nộp thuế trên cơ sở chức năng mới, cá thể hoá trách nhiệm của cơ quan quản lý thuế và đối tượng nộp thuế, đơn giản và làm rõ hơn về quy trình và thủ tục giấy tờ trong việc kê khai, nộp thuế. Giao

cho đối tượng nộp thuế quyền tự chủ, tự chịu trách nhiệm xác định số thuế và nộp thuế, cơ quan Thuế sẽ tập trung đẩy mạnh hai khâu công tác lớn là tuyên truyền, hướng dẫn, cung cấp dịch vụ hỗ trợ đối tượng nộp thuế và thanh tra, kiểm tra. Như vậy trong giai đoạn mới này, có thể thấy thông tin có một giá trị rất quan trọng, tổ chức khai thác thông tin tốt sẽ góp phần lớn hỗ trợ công tác thanh tra, kiểm tra, đảm bảo ngăn chặn các hành vi trốn thuế, đảm bảo giữ công bằng cho các đối tượng nộp thuế trong nghĩa vụ đóng góp ngân sách cho Nhà nước. Phân tích, dự báo thông tin đúng cũng góp phần giúp công tác thanh tra, kiểm tra Thuế xác định được đúng đối tượng cần thanh kiểm tra, giúp hạn chế những tiêu cực trong công tác thanh tra, kiểm tra thuế.

Nghiên cứu lý thuyết khai phá dữ liệu, áp dụng khai phá dữ liệu trên cơ sở dữ liệu ngành Thuế với mong muốn bước đầu tìm hiểu những kết quả khai phá thú vị từ kho thông tin Thuế. Những kết quả khai phá trong phạm vi luận văn có thể chưa có ý nghĩa thiết thực, nhưng hy vọng sẽ là bước đầu cho dự án Xây dựng hệ thống phân tích thông tin hỗ trợ các công tác quản lý và thanh tra thuế.

3.2. Lựa chọn công cụ khai phá

3.2.1 Lựa chọn công cụ

Có rất nhiều sản phẩm hỗ trợ việc khai phá tri thức từ CSDL.

Bảng dưới đây liệt kê một số sản phẩm khai phá dữ liệu của các hãng khác nhau và những tính năng của mỗi sản phẩm (http://www.xore.com/prodtable.html).

Bảng 2.2 Bảng các sản phẩm khai phá dữ liệu

Company	Product	NN	Tree	Naïve Bayes	k-Mns	k-NN	Stats	Pred	Time Series	Clust	Assoc	Win 32	UNIX	Par	API SDK	SQL Ext
Angoss International Ltd.	KnowledgeSEEKER		Y					Y				Y	Y	Y		
	KnowledgeSTUDIO	Y	Y		Y			Y		Y		Y	Y	Y	Y	Y
Business Objects	BusinessMiner		Y									Y				
Cognos Incorporated	4Thought	Y						Y	Y			Y				
	Scenario		Y									Y				
Fair, Isaac/HNC Software	DataBase Mining Marksman	Y						Y		Y		Y		Y		
Informix/RedBrick Software Inc.	Red Brick Data Mine			Y				Y				Y	Y			Y
International Business Machines	Intelligent Miner	Y	Y			Y		Y	Y	Y	Y	Y	Y	Y	Y	
Accrue Software	Decision Series	Y	Y	Y				Y		Y	Y		Y	Y	Y	
NeuralWare	NeuralSIM	Y						Y				Y				
Oracle Corp.	Darwin	Y	Y			Y		Y					Y	Y		
Salford Systems	CART		Y					Y				Y	Y			
SAS Institute	Enterprise Miner	Y	Y				Y	Y	Y	Y	Y	Y	Y			
SPSS, Inc.	Answer Tree		Y				Y	Y				Y	Y			
	Clementine	Y	Y					Y	Y	Y	Y	Y	Y			
	Neural Connection	Y					Y	Y				Y	Y			
Unica Technology	Pattern Recognition Workbench	Y			Y	Y	Y	Y	Y	Y		Y			Y	
	Model 1	Y	Y	Y	Y		Y	Y				Y	Y	Y		

CSDL ngành Thuế sử dụng là CSDL Oracle. Do vậy việc chọn công cụ khai phá dữ liệu của hãng Oracle cũng là một lựa chọn tất yếu.

Khai phá dữ liệu bằng sản phẩm của hãng Oracle, có thể lựa chọn:

1. Darwin: Là một ứng dụng khai phá dữ liệu đặc biệt để xử lý với nhiều gigabytes dữ liệu và cung cấp những câu trả lời cho các bài toán phức tạp như phân lớp dữ liệu, dự đoán và dự báo.

Phần mềm Darwin giúp ta chuyển đổi một khối lượng dữ liệu lớn thành những tri thức kinh doanh (tri thức nghiệp vụ - Business intelligence). Darwin giúp tìm ra những mẫu và các liên kết có ý nghĩa trong toàn bộ dữ liệu – Các mẫu cho phép ta hiểu tốt hơn và dự đoán được hành vi của khách hàng.

2. Oracle Data Mining (ODM) được thiết kế cho người lập trình, những nhà phân tích hệ thống, các quản trị dự án và cho tất cả những ai quan tâm đến việc phát triển các ứng dụng CSDL dùng khai phá dữ liệu để phát hiện ra các mẫu ẩn và dùng tri thức đó để tạo các dự đoán.

ODM là công cụ khai phá dữ liệu được nhúng trong CSDL Oracle. Dữ liệu không tách rời CSDL - dữ liệu, và tất cả những hoạt động chuẩn bị dữ liệu, xây dựng mô hình và áp dụng mô hình đều được giữ trong CSDL. Việc này cho phép Oracle xây dựng nền tảng cho những nhà phân tích dữ liệu và những ngươờiphát triển ứng dụng có thể tích hợp khai phá dữ liệu một cách liền mạch với các ứng dụng CSDL.

Darwin là sản phẩm khai phá dữ liệu chỉ chạy trên nền Unix. Hiện tại trong ngành Thuế vẫn đang sử dụng hệ điều hành Windows, và cũng chưa mua bản quyền sử dụng Darwin.

Các thành phần liên quan đến CSDL Oracle sử dụng tại ngành Thuế đều có mua bản quyền của hãng. ODM là có sẵn trong CSDL Oracle. Do vậy ODM là công cụ khai phá dữ liệu được lựa chọn trong luận văn này.

3.2.2 Oracle Data Mining (ODM)

Oracle Data Mining (ODM) cung cấp cả hai giao diện lập trình ứng dụng PL/SQL và Java API cho việc tạo ra các mô hình khai phá dữ liệu có giám sát và không giám sát. Hai APIs là tương tác hoàn toàn với nhau, vì vậy mô hình có thể được tạo ra với một API và sau đó sửa đổi hoặc sử dụng dùng API khác.

Java API là một thực hiện của Oracle theo chuẩn JDM 1.0, theo đúng framework mở rộng của chuẩn JSR-73.

PL/SQL API: Có thể sử dụng các package để xây dựng mô hình khai phá, kiểm thử mô hình, và áp dụng mô hình với dữ liệu để thu được các thông tin dự đoán và mô tả.

Các API của Oracle Data Mining hỗ trợ cả 2 chức năng khai phá dự đoán và mô tả. Các chức năng dự đoán được biết như học có giám sát, dùng dữ liệu huấn luyện để dự đoán giá trị đích. Các chức năng mô tả, được biết như học không giám sát, xác định các quan hệ bản chất bên trong dữ liệu. Mỗi chức năng khai phá xác định một lớp các bài toán được giải quyết và mỗi chức năng có thể được thực hiện với một hoặc nhiều thuật toán. Các API cũng cung cấp các phương tiện chuyển đổi dữ liệu cơ sở cho việc chuẩn bị dữ liệu khai phá.

Oracle Data Mining cung cấp:

1. Các chức năng dự đoán sau:

Chức năng	Mô tả	Các thuật toán
Phân lớp Classification	Mô hình phân lớp dùng dữ liệu lịch sử để dự đoán dữ liệu rời rạc hoặc phân loại mới	Naive Bayes, Adaptive Bayes Network, Support Vector Machine, Decision Tree
Phát hiện bất thường Anomaly Detection	Mô hình phát hiện bất thường dự đoán có hay không một điểm dữ liệu là điển hình cho sự phân tán cho trước. PL/SQL và Java APIs hỗ trợ phát hiện bất thường qua chức năng phân lớp	One-Class Support Vector Machine (SVM). PL//SQL và Java APIs hỗ trợ One-Class SVM dùng chức năng khai phá phân lớp và thuật toán SVM không có đích.
Hồi qui Regression	Mô hình Hồi qui dùng dữ liệu lịch sử để dự đoán dữ liệu số, liên tiếp mới	Support Vector Machine
Độ quan trọng của thuộc tính Attribute Importance	Mô hình độ quan trọng của thuộc tính xác định tầm quan trọng liên quan của một thuộc tính trong việc dự đoán một đầu ra cho trước.	Minimal Descriptor Length

2. Các chức năng mô tả sau:

Chức năng	Mô tả	Các thuật toán
Phân nhóm Clustering	Mô hình phân nhóm xác định các nhóm tự nhiên trong tập dữ liệu	Enhanced k-means, Orthogonal Clustering (O-Cluster - Thuật toán bản quyền của Oracle)
Các luật kết hợp Association Rules	Mô hình kết hợp xác định các quan hệ và khả năng xuất hiện của chúng trong tập dữ liệu	Apriori
Trích chọn đặc trưng Feature Extraction	Mô hình trích chọn đặc trưng tạo tập dữ liệu tối ưu làm cơ sở cho mô hình trên đó.	Non-Negative Matric Factorization

3.2.3 DBMS_DATA_MINING

Phương pháp phát triển cho khai phá dữ liệu dùng giao diện DBMS_DATA_MINING được chia thành hai pha.

Pha đầu tiên bao gồm việc phân tích và thiết kế dữ liệu của ứng dụng, trong đó thực hiện hai bước sau:

1. Phân tích bài toán, lựa chọn hàm khai phá và thuật toán khai phá

2. Phân tích dữ liệu được dùng cho xây dựng các mô hình khai phá (build data), kiểm thử các mô hình dự đoán (test data), và sử dụng dữ liệu mới trên mô hình (scoring data).

Pha thứ hai bao gồm việc phát triển ứng dụng khai phá dùng các packages DBMS_DATA_MINING và DBMS_DATA_MINING_TRANSFORM.

3. Chuẩn bị dữ liệu xây dựng, kiểm thử, áp dụng (build, test, scoring data) dùng package DBMS_DATA_MINING_TRANSFORM hoặc công cụ third-party hoặc dùng trực tiếp các scripts SQL hoặc PL/SQL trong mẫu phù hợp với hàm và thuật toán lựa chọn. Việc quan trọng là ba tập dữ liệu đã nêu ở trên phải được chuẩn bị theo cách giống nhau để việc khai phá ra các kết quả có ý nghĩa.

4. Chuẩn bị các bảng thiết lập tham số thay thế cho các thiết đặt ngầm định của thuật toán, của chức năng khai phá. Bước này là tuỳ chọn.

5. Xây dựng mô hình khai phá cho tập dữ liệu huấn luyện đã cho

6. Với các mô hình dự đoán (phân lớp và hồi qui), kiểm thử mô hình cho tính chính xác và đo hiệu năng. Việc này là áp dụng mô hình trên dữ liệu kiểm thử.

7. Lấy dấu hiệu của mô hình để xác định các thuộc tính khai phá sẽ được dùng với mô hình khi áp dụng. Thông tin này sẽ giúp biết chắc chắn dữ liệu khai phá là phù hợp với mô hình đã cho. Đây là bước tuỳ chọn.

8. Áp dụng mô hình phân lớp, hồi qui, phân nhóm, hoặc mô hình trích chọn đặc trưng với dữ liệu mới để sinh ra các dự đoán và/hoặc các tổng kết mô tả và các mẫu về dữ liệu

9. Lấy các chi tiết của mô hình để hiểu được vì sao mô hình mô hình cho ra dữ liệu trong mỗi mẫu cụ thể. Đây là bước tuỳ chọn

10. Lặp lại bước 3 đến bước 9, đến khi ta thu được các kết quả vừa ý.

3.3. Mục tiêu khai thác thông tin của ngành Thuế

Tại hầu hết các đơn vị, tổ chức có áp dụng công nghệ thông tin vào quản lý hiện nay, ứng dụng mới dừng lại ở mức độ là ứng dụng tác nghiệp thông thường với chức năng hỗ trợ đưa thông tin vào và kết xuất ra các báo cáo đầu ra. Những ứng dụng hỗ trợ cao cho phân tích, hỗ trợ ra quyết định

chưa nhiều. Tuy nhiên với xu hướng phát triển hiện tại, chắc chắn sẽ rất cần đến những ứng dụng khai phá tri thức tiềm ẩn trong CSDL.

Hiện nay, ngành Thuế đang trong những năm đầu thực hiện cải cách hành chính Thuế. Theo chiến lược này hướng quản lý của ngành Thuế sẽ thay đổi lớn, tập trung vào hai công tác chính:

— Công tác tuyên truyền, hỗ trợ và cung cấp các dịch vụ phục vụ cho Đối tượng nộp thuế.

— Công tác thanh tra kiểm tra Thuế.

Khai phá dữ liệu tốt có tác dụng hỗ trợ công tác tuyên truyền hỗ trợ ĐTNT: Phân tích trên dữ liệu, có thể tìm ra được những kết quả giúp định hướng việc hỗ trợ, tuyên truyền, giúp xác định những ĐTNT nào nên áp dụng cách thức tuyên truyền nào cho hiệu quả.

Với công tác thanh tra kiểm tra Thuế: Khai phá dữ liệu còn mang lại ý nghĩa to lớn hơn. Trước đây công tác thanh tra chủ yếu dựa vào kinh nghiệm của các cán bộ thanh tra, xem xét số liệu trên các báo cáo tài chính của ĐTNT, so sánh số liệu các năm của doanh nghiệp đó, so sánh số liệu trong năm của doanh nghiệp với tình hình phát triển chung của ngành để phát hiện ra những điểm nghi ngờ cần xác minh. Ngày nay, số lượng doanh nghiệp tăng trưởng ngày càng nhiều, sẽ đến lúc mỗi cán bộ thanh tra không thể xem xét từng trường hợp, từng số liệu cụ thể của mỗi ĐTNT được. Như vậy rất cần công cụ hỗ trợ.

Một vấn đề nữa không chỉ có ngành Thuế quan tâm, đó là hạn chế những phiền toán cho Doanh nghiệp khi phải thanh tra Thuế. Muốn vậy, cần xác định được ĐTNT nghi ngờ, phải thanh tra thuế với độ chắc chắn cao.

Mặc dù chưa có ứng dụng khai phá dữ liệu nào, nhưng qua một số thông tin học hỏi từ Thuế các nước, Thuế Việt Nam cũng bắt đầu đi theo hướng cải tiến này. Ngành Thuế bắt đầu xem xét việc yêu cầu Doanh nghiệp

cung cấp các báo cáo tài chính liên quan, để làm cơ sở xem xét, phân tích ĐTNT, như Bảng cân đối kế toán, Báo cáo kết quả hoạt động kinh doanh, Báo cáo lưu chuyển tiền tệ trực tiếp/gián tiếp… Từ những báo cáo này, kết hợp với số liệu quản lý thuế (số thuế mỗi ĐTNT phải nộp, số đã nộp, còn nợ…) để xác định các chỉ tiêu phân tích. Ứng dụng hiện tại mới dừng ở mức đưa ra báo cáo liệt kê các chỉ tiêu đã phân tích (phân tích các chỉ tiêu một cách riêng lẻ), dựa vào đó để cán bộ thanh tra xem xét ra quyết định. Mong muốn của cán bộ thanh tra là có được ứng dụng tự động phân tích dựa trên nhiều chỉ tiêu và khi đưa số liệu của một ĐTNT vào sẽ có câu trả lời là điểm đánh giá mức độ vi phạm của ĐTNT này.

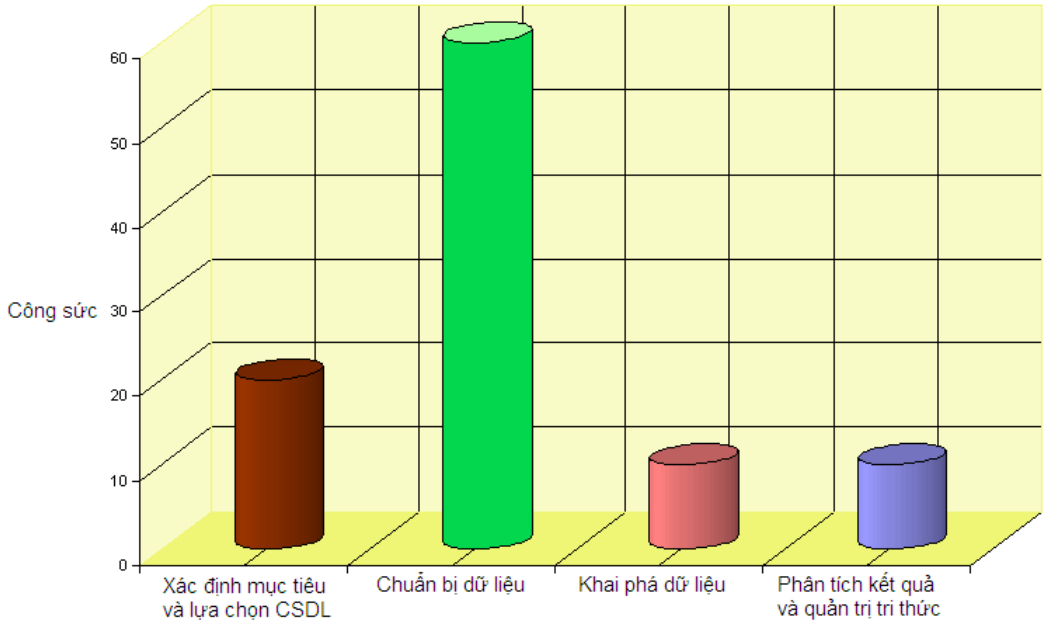
Với những tìm hiểu trên, có thể thấy nhiều kiểu khai phá dữ liệu có thể áp dụng được để đáp ứng yêu cầu và giúp nâng cao hiệu quả của công tác quản lý Thuế. Tuy nhiên trong khuôn khổ của Luận văn, hai chức năng khai phá được chọn để khai phá thử nghiệm trên CSDL ngành Thuế, đó là:

- Khai phá luật kết hợp: Với mong muốn tri thức phát hiện ra có thể giúp ích cho công tác tuyên truyền và hỗ trợ ĐTNT
- Phân lớp: Dựa vào một số chỉ tiêu phân tích để phân lớp các ĐTNT và dự báo về khả năng vi phạm của ĐTNT. Hỗ trợ thanh tra Thuế.

3.4. Thử nghiệm khai phá luật kết hợp

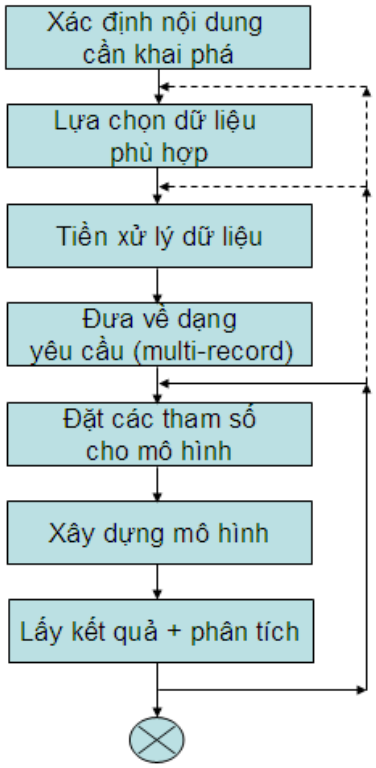
Dữ liệu quản lý Thuế được tổ chức phân tán tại 64 Cục Thuế. Tại Tổng cục Thuế có tập trung dữ liệu ở một mức độ nhất định tuỳ theo loại thông tin. Ví dụ với dữ liệu thông tin các Đối tượng nộp thuế được tập trung khá đầy đủ tại Tổng cục thuế (trừ phần dữ liệu lịch sử, tại Tổng cục chỉ lưu thông tin đầy đủ đến thời điểm hiện tại), còn dữ liệu về quản lý thuế thì chỉ có số liệu tổng hợp tại Tổng cục, dữ liệu chi tiết được quản lý tại các Cục Thuế.

Công việc khai phá dữ liệu nói chung có thể tổng kết theo 4 nhiệm vụ chính: Xác định mục tiêu và lựa chọn dữ liệu, Chuẩn bị dữ liệu, Khai phá dữ liệu, Phân tích kết quả và quản trị tri thức. Trong 4 nhiệm vụ trên thì việc chuẩn bị dữ liệu sẽ mất nhiều công sức nhất. Có thể thấy minh hoạ ở hình 3.1. Công sức dành cho viêc chuẩn bị dữ liệu để khai phá đối với CSDL tác nghiệp thực sự sẽ khó khăn hơn nhiều so với thực hiện trên dữ liệu giả định.



Hình 3.1 Công sức cần cho mỗi giai đoạn khai phá dữ liệu

Sử dụng ODM để khai phá luật kết hợp gồm những bước chính: Chuẩn bị dữ liệu, xây dựng mô hình – chính là bước xác định các frequent itemsets, lấy ra các luật khai phá được. Các bước tiến hành thử nghiệm khai phá luật kết hợp trên CSDL ngành Thuế thực hiện trong luận văn này đều được tiến hành theo quy trình sau:



Hình 3.2 Các bước khai phá luật kết hợp trên CSDL ngành Thuế

Khi đặt các tham số cho mô hình khai phá luật kết hợp có thể là cao quá với dữ liệu, kết quả sẽ không thu được luật. Khi đó thực hiện điều chỉnh tham số của mô hình. Trường hợp thay đổi các tham số vẫn không hiệu quả, có thể phải xem xét lại từ bước tiền xử lý dữ liệu. Trường hợp không loại bỏ các items phổ biến trong tập dữ liệu cũng có thể dẫn đến kết quả khai phá không như mong muốn. Hoặc xem xét lại cách xử lý với dữ liệu thiếu. Cũng có thể phải xem xét lại dữ liệu lựa chọn cho khai phá đã đúng chưa.

Thử nghiệm khai phá luật kết hợp được thực hiện theo các bước nêu trên và dưới đây là kết quả cuối cùng. Các mã lệnh tương ứng được trình bày trong phần phụ lục.

Như đã nêu trong mục 3.3, bài toán khai phá luật kết hợp khá phù hợp cho việc phát hiện tri thức phục vụ cho công tác tuyên truyền, hỗ trợ ĐTNT. Những luật phát hiện được có thể giúp cán bộ tuyên truyền, hỗ trợ xác định được phạm vi ĐTNT để đưa các hình thức tuyên truyền phù hợp.

Dưới đây là một khai phá thử nghiệm phát hiện mối liên hệ giữa ngành nghề, quy mô doanh nghiệp (theo doanh thu), số thuế phải nộp và tình trạng nộp chậm thuế.

Xác định nội dung khai phá:

Nhằm xác định phạm vi ĐTNT nào cần tập trung tuyên truyền nâng cao ý thức nghiêm chỉnh chấp hành nghĩa vụ Thuế. Bài toán sẽ dựa vào những thông tin có khả năng liên quan đến tình trạng nộp chậm Thuế, bao gồm: ngành nghề kinh doanh, quy mô doanh nghiệp (tính theo doanh thu), số thuế phải nộp.

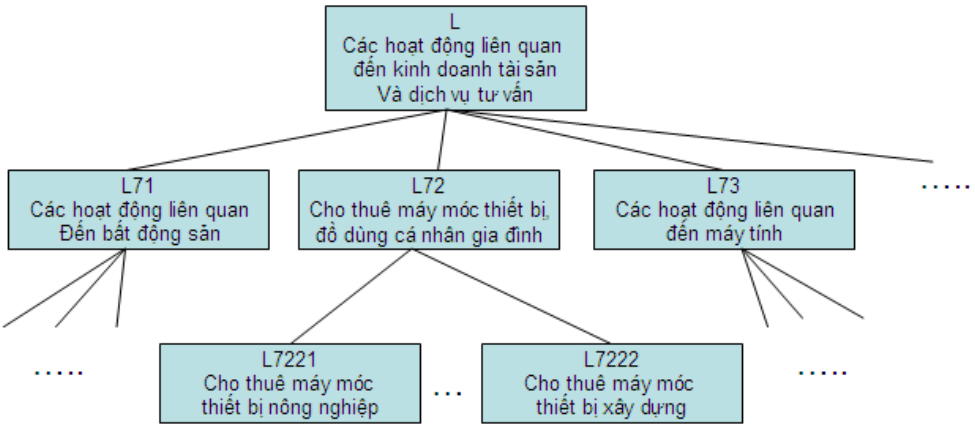
Lựa chọn dữ liệu:

Thông tin từ Báo cáo kết quả sản xuất kinh doanh của ĐTNT: Có được thông tin về doanh thu, số thuế phải nộp.

Dữ liệu về ngành nghề của các ĐTNT:

- ID
- Mã số thuế
- Mã ngành nghề
- Trường xác định dữ liệu lịch sử hay hiện tại

Mã ngành nghề biểu diễn bởi 5 ký tự (ví dụ: L7221 – Cho thuê máy móc thiết bị nông nghiệp). Sự phân cấp ngành nghề được tổ chức ngay trong mã. Ví dụ một nhánh cây phân cấp trong hình 3.3.



Hình 3.3 Nhánh cây phân cấp ngành nghề

Tình trạng nộp chậm thuế: Được lấy từ thông tin tính phạt nộp chậm trong hệ thống thông tin Quản lý thuế. Ở đây chỉ lấy thông tin ĐTNT có nộp chậm thuế (1) hay không (0).

Tiền xử lý dữ liệu:

Với ngành nghề nếu để mức thấp sẽ khó phát hiện luật. Sẽ thực hiện khai phá ở mức khái niệm cao hơn. Như vậy khi lấy giá trị ngành nghề sẽ có biến đổi: lấy ngành nghề kinh doanh của mỗi đối tượng theo 3 ký tự đầu của ngành nghề.

Quy mô doanh nghiệp được phân loại dựa theo doanh thu trung bình tháng của mỗi đối tượng (tính trung bình trong 1 năm), và chia thành các mức: Rất nhỏ (từ 0 đến 100.000.000), nhỏ (từ 100.000.000 đến 500.000.000), trung bình (từ 500.000.000 đến 1.000.000.000), lớn (từ 1.000.000.000 đến 5.000.000.000), rất lớn (trên 5.000.000.000).

Số thuế phải nộp trung bình tháng cũng được phân nhóm thành các khoảng 5 triệu, 10 triệu, 20 triệu, 30 triệu, 50 triệu, 100 triệu, 500 triệu, 1 tỷ, 5 tỷ.

Đưa dữ liệu về dạng phù hợp với yêu cầu khai phá:

Dữ liệu được đưa về dạng:

(Mã số thuế, ngành sx, 1

Union

Mã số thuế, doanh thu, 1

Union

Mã số thuế, thuế phải nộp, 1

Union

Mã số thuế, nộp chậm, 1)

Và chuyển về dạng nested table:

```
CREATE VIEW TR_dondoc_AR AS
SELECT TIN,
       CAST(COLLECT(DM_Nested_Numerical(
         SUBSTRB(nganhsx, 1, 10), has_it))
       AS DM_Nested_Numericals) tinnganhsx
  FROM tr_dondoc
GROUP BY TIN;
```

Đặt tham số cho mô hình:

Ngưỡng độ hỗ trợ cực tiểu: 0.1

Ngưỡng độ chắc chắn cực tiểu: 0.1

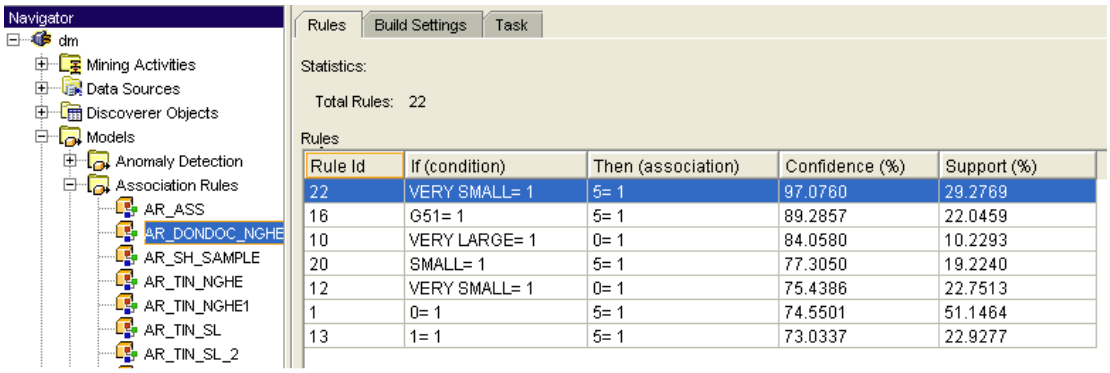
Độ dài luật khai phá: 2

Tạo mô hình và đưa ra kết quả:

Item	Độ hỗ trợ (support)	Số items
G51	.246913580246913580246913580246913580025	1
SMALL	.248677248677248677248677248677248677248677725	1
VERY SMALL	.301587301587301587301587301587301587301587	1
1-1	.313932980599647266313932980599647266631	1
0-1	.686067019400352733686067019400352733369	1
5	.740740740740740740740740740740740740740740	1
0	.227513227513227513227513227513227513227513227513	2

```
VERY SMALL .22751322751322751322751322751322751323     2
1          .22927689594356261022927689594356261023     2
5          .22927689594356261022927689594356261023     2
5          .29276895943562610229276895943562610229     2
VERY SMALL .29276895943562610229276895943562610229     2
0          .51146384479717813051146384479717813051     2
5          .51146384479717813051146384479717813051     2
```

Các luật khai phá được:



Hình 3.4 Các luật khai phá từ ODM (độ dài luật = 2)

LUẬT	CONFIDENCE	SUPPORT
VERY SMALL => 5	97.07603	29.276896
G51 => 5	89.28571	22.045855
VERY LARGE => 0	84.05797	10.229277
SMALL => 5	77.30496	19.223986
VERY SMALL => 0	75.4386	22.751324
0 => 5	74.550125	51.146385
1 => 5	73.03371	22.92769

Nhận xét:

Khai phá được các luật trên đều có độ chắc chắn lớn.

1. `VERY SMALL => 5`: Quy mô rất nhỏ thì 97% có số thuế phải nộp dưới 5 triệu/tháng
2. `G51 => 5`: Ngành nghề 'Bán buôn và đại lý (trừ xe có động cơ và môtô, xe máy)' thì 89% có số thuế phải nộp dưới 5 triệu/tháng

3. VERY LARGE => 0: ĐTNT có quy mô rất lớn thì có 84% không nộp chậm thuế
4. SMALL => 5: ĐTNT có quy mô nhỏ, có 77% nộp thuế dưới 5 triệu/tháng
5. VERY SMALL => 0: ĐTNT có quy mô rất nhỏ thì 75% thực hiện tốt nghĩa vụ Thuế, không nộp chậm thuế.
6. 0 => 5: Trong số các ĐTNT không nộp chậm thuế thì có 74% là ĐTNT phải nộp dưới 5 triệu/tháng
7. 1 => 5: Trong số các ĐTNT nộp chậm thuế thì có 73% là ĐTNT phải nộp dưới 5 triệu/tháng

Một số ý nghĩa rút ra được từ các luật trên:

Những ĐTNT thuộc diện nộp thuế dưới 5 triệu/tháng có hiện tượng chậm nộp thuế. Tuy nhiên về số lượng thì số ĐTNT chấp hành tốt nghĩa vụ đóng thuế thuộc diện nộp thuế dưới 5 triệu/tháng lớn hơn nhiều so với số lượng chậm nộp thuế (theo luật 6 và 7). Thêm vào đó số thuế thường nhỏ nên tổng thu từ những ĐTNT này không lớn. Cần tổ chức các hình thức tuyên truyền công cộng, đỡ tốn phí tuyên truyền cho các ĐTNT này.

Những đối tượng có quy mô rất lớn nghiêm chỉnh chấp hành nghĩa vụ Thuế sẽ rất có lợi cho nhà nước (luật 3). Bởi vậy cần có chế độ, chính sách khen thưởng kịp thời những ĐTNT này.

Khai phá thêm các luật với độ dài luật khai phá = 3

Đặt tham số cho mô hình:

Ngưỡng độ hỗ trợ cực tiểu: 0.1

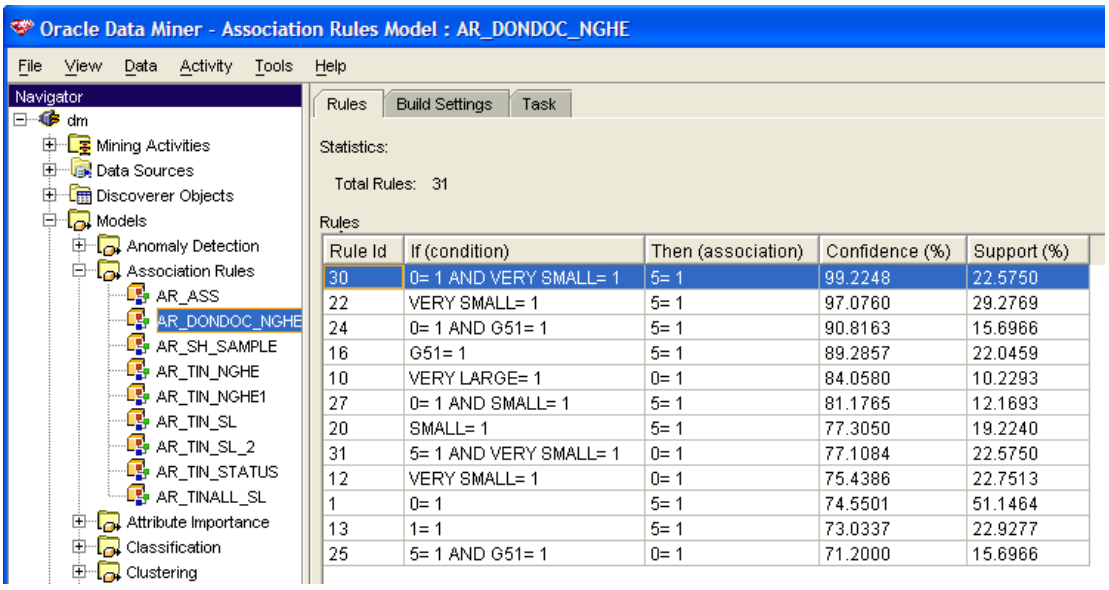
Ngưỡng độ chắc chắn cực tiểu: 0.1

Độ dài luật khai phá: 3

Tạo mô hình và đưa ra kết quả:

Item	Độ hỗ trợ (support)	Số items
G51	.246913580246913580246913580246913580 25	1
SMALL	.248677248677248677248677248677248677 25	1
VERY SMALL	.301587301587301587301587301587301587 3	1
1	.313932980599647266313932980599647266 31	1
0	.686067019400352733686067019400352733 69	1
5	.740740740740740740740740740740740740 74	1
0	.227513227513227513227513227513227513 23	2
VERY SMALL	.227513227513227513227513227513227513 23	2
1	.229276895943562610229276895943562610 23	2
5	.229276895943562610229276895943562610 23	2
5	.292768959435626102292768959435626102 29	2
VERY SMALL	.292768959435626102292768959435626102 29	2
0	.511463844797178130511463844797178130 51	2
5	.511463844797178130511463844797178130 51	2

Các luật khai phá được:



Hình 3.5 Các luật khai phá từ ODM (độ dài luật = 3)

LUẬT	CONFIDENCE	SUPPORT
0 AND VERY SMALL => 5	99.22481	22.574955
VERY SMALL => 5	97.07603	29.276896
0 AND G51 => 5	90.81633	15.696649
G51 => 5	89.28571	22.045855
VERY LARGE => 0	84.05797	10.229277
0 AND SMALL => 5	81.17647	12.1693125
SMALL => 5	77.30496	19.223986
5 AND VERY SMALL => 0	77.10844	22.574955
VERY SMALL => 0	75.4386	22.751324
0 => 5	74.550125	51.146385
1 => 5	73.03371	22.92769
5 AND G51 => 0	71.2	15.696649

Nhận xét:

Khai phá được các luật trên đều có độ chắc chắn lớn. Các luật độ dài bằng 2 đã được khai phá từ bước trước và có diễn giải. Dưới đây chỉ nêu luật độ dài hơn 2.

1. `0 AND VERY SMALL => 5`: Trong số ĐTNT không nộp chậm thuế và thuộc loại ĐTNT quy mô rất nhỏ thì 99% trong số đó có số thuế phải nộp dưới 5 triệu/tháng.
2. `0 AND G51 => 5`: ĐTNT chấp hành tốt nghĩa vụ Thuế và thuộc ngành nghề 'Bán buôn và đại lý (trừ xe có động cơ và môtô, xe máy)' thì 90% số đó có số thuế phải nộp hàng tháng dưới 5 triệu
3. `0 AND SMALL => 5`: Trong số ĐTNT không nộp chậm thuế và thuộc loại ĐTNT quy mô nhỏ thì 81% trong số đó có số thuế phải nộp dưới 5 triệu/tháng.
4. `5 AND VERY SMALL => 0`: ĐTNT phải nộp thuế dưới 5 triệu/tháng và có quy mô rất nhỏ thì 77% là nộp thuế đúng hạn

5. `5 AND G51 => 0`: 71% ĐTNT có số thuế phải nộp dưới 5 triệu/tháng và kinh doanh ngành nghề 'Bán buôn và đại lý (trừ xe có động cơ và môtô, xe máy)' thực hiện tốt nghĩa vụ nộp thuế.

Một số ý nghĩa từ các luật trên:

ĐTNT có quy mô nhỏ, rất nhỏ và có số thuế phải nộp dưới 5 triệu/tháng, đặc biệt ĐTNT thuộc ngành nghề 'Bán buôn và đại lý (trừ xe có động cơ và môtô, xe máy)' sẽ không phải quan tâm nhiều đến việc đốc thúc thu thuế, vì ĐTNT thuộc phạm vi này thường nghiêm chỉnh chấp hành việc nộp thuế.

3.5. Phân lớp bằng học cây quyết định

Trong phân lớp bằng học cây quyết định, sau khi xác định bài toán và lựa chọn dữ liệu thì cần thực hiện bước tạo ra bộ dữ liệu huấn luyện dùng để xây dựng mô hình, bộ để kiểm thử và đánh giá độ chính xác của mô hình. Mô hình đạt được độ chính xác chấp nhận được sẽ được sử dụng với bộ dữ liệu mới.

Sử dụng ODM để phân lớp sẽ qua các bước chính sau:

- Chuẩn bị 3 bộ dữ liệu (xác định thuộc tính phân loại, tổ chức của 3 bộ dữ liệu phải tương tự nhau)
- Thiết lập các tham số: Lựa chọn thuật toán nào, xác định ma trận chi phí.
- Xây dựng mô hình dựa vào các tham số đã thiết lập. Ngoài ra, chỉ rõ: Sử dụng ma trận chi phí nào, thuộc tính khoá xác định duy nhất một bản ghi, chỉ ra thuộc tính đích (là thuộc tính phân lớp), chỉ ra bộ dữ liệu huấn luyện

— Kiểm thử trên bộ dữ liệu kiểm thử: Áp dụng mô hình để phân loại trên dữ liệu kiểm thử và so sánh với thuộc tính đích để đánh giá độ chính xác. Ở đây có thể lựa chọn phân loại có dùng hoặc không dùng ma trận chi phí.

— Cuối cùng là sử dụng mô hình nếu mô hình có độ chính xác chấp nhận được: Áp dụng mô hình trên dữ liệu chưa phân loại, đưa ra các dự báo.

Áp dụng phân lớp trên CSDL ngành Thuế có thể:

— Dùng để dự báo ĐTNT nợ thuế, phục vụ cho công tác đôn đốc thu.

— Dùng để dự báo ĐTNT nghi ngờ vi phạm, gian lận… phục vụ cho công tác thanh tra Thuế.

Những chỉ tiêu thường được lấy làm căn cứ phân tích phục vụ công tác thanh tra Thuế gồm những thông tin sau:

— Các tỷ suất thể hiện khả năng thanh toán, tỷ suất sinh lời, tỷ suất hiệu quả, cơ cấu tài sản và cơ cấu nguồn vốn, tỷ suất liên quan đến kê khai thuế

— Quy mô doanh nghiệp: Quy mô theo doanh thu, nguồn vốn, theo Tài sản cố định

— Xác định rủi ro theo: Quy mô của doanh nghiệp, loại hình doanh nghiệp, theo mức độ tuân thủ về nộp thuế, hiệu quả sản xuất kinh doanh, tình hình kê khai thuế của doanh nghiệp

Có nhiều cách phân tích dựa trên các chỉ tiêu trên. Có thể tính toán các tỷ suất của một doanh nghiệp và so sánh với chính doanh nghiệp đó qua các thời kỳ khác nhau hoặc cùng so sánh với tỷ suất chuẩn của ngành. Có thể xem xét tỷ suất theo nhiều năm của các doanh nghiệp trong cùng ngành kinh tế và tỷ suất trung bình ngành theo từng năm. So sánh doanh thu, chi phí của mỗi doanh nghiệp qua các năm và so với doanh thu, chi phí trung bình của ngành.

Thực tế phối hợp được nhiều chỉ tiêu trong phân tích và số liệu thu thập được càng chính xác sẽ có được những nhận định có độ chắc chắn cao. Sự phối hợp thông tin giữa các ngành khác nhau cũng rất quan trọng, ví dụ lấy số liệu thống kê ngành nghề từ Cục Thống Kê.

Với mục đích khai phá thử nghiệm, những bài toán khai phá trong luận văn có thể coi là những minh hoạ cho khả năng khai phá dữ liệu, để từ đó phát triển sau này với sự phân tích đầy đủ các chỉ tiêu.

3.5.1 Phân lớp ĐTNT dựa vào so sánh tỷ suất các năm

Xác định nội dung khai phá

Dựa vào cách phân tích tỷ suất của một ĐTNT qua các năm và so sánh với tỷ suất chung của Ngành, đưa ra bài toán: Căn cứ vào tỷ suất Sinh lợi của mỗi ĐTNT qua hai năm và tỷ suất Sinh lợi của ngành để đưa ra nhận định ĐTNT có thuộc diện cần phải xem xét không.

Tỷ suất Sinh lợi = (Lợi nhuận thuần + Chi phí lãi vay)/Doanh thu thuần

Lựa chọn dữ liệu

Số liệu được lấy từ Báo cáo Kết quả hoạt động kinh doanh của ĐTNT.

Báo cáo kết quả hoạt động kinh doanh:

- Mã số thuế
- Loại báo cáo
- Năm
- Chỉ tiêu báo cáo
- Số tiền

Mã ngành nghề của ĐTNT được lấy theo dữ liệu ngành nghề.

Tiền xử lý dữ liệu

Lấy các chỉ tiêu cần thiết để tính Tỷ suất Sinh lợi, lấy dữ liệu của 2 năm 2004 và 2005 để so sánh.

Tính toán Tỷ suất Sinh lợi trung bình của ngành trong năm 2004 và 2005.

Để thử nghiệm trên cả công cụ khai phá của Oracle và See5, sẽ lọc lấy một phần nhỏ dữ liệu. Và lấy một số ngành nghề như: K70 - Hoạt động khoa học và công nghệ, D26 - Sản xuất các sản phẩm từ khoáng chất, I60 - Vận tải đường bộ, D22 - Xuất bản, in và sảo bản ghi các loại, C14 – Khai thác than đã và khai thác mỏ đá, C10 – Khai thác than cứng, than non, than bùn, J65 – Trung gian tài chính (Trừ bảo hiểm và trợ cấp hưu trí).

Dữ liệu cho xây dựng cây quyết định như sau:

- Mã số thuế (TIN)
- Ngành sản xuất (chỉ lấy mức 3 ký tự) (NGANHSX)
- Chênh lệch tỷ suất sinh lợi giữa 2 năm (SoTSSinhLoi)
- Chênh lệch tỷ suất sinh lợi của ngành nghề (SoTS)
- Trường phân loại xác định ĐTNT có thuộc diện phải xem xét hay không (XEMXET)

Thiết đặt các tham số và xác định ma trận chi phí:

Ma trận chi phí:

Chi phí		Dự báo cần xem xét 1	Dự báo không xem xét 0
Xem xét (thực tế)	1	0	5
Không xem xét (thực tế)	0	1	0

Chọn sử dụng thuật toán cây quyết định

Tạo mô hình:

Đây chính là bước xây dựng cây quyết định

Kiểm thử, đánh giá mô hình:

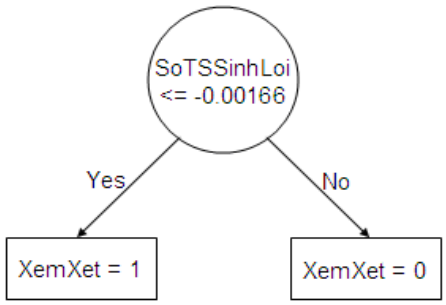
— Áp dụng trên dữ liệu kiểm thử

— Đánh giá độ chính xác khi dùng ma trận chi phí và khi không dùng

Thực hiện trên dữ liệu ngành Thuế, có kết như sau:

Độ chính xác khi không dùng ma trận chi phí và dùng ma trận chi phí là như nhau và bằng 80%.

Cây quyết định như sau:



Hình 3.6 Cây quyết định dùng ODM – Bài toán phân tích tỷ suất

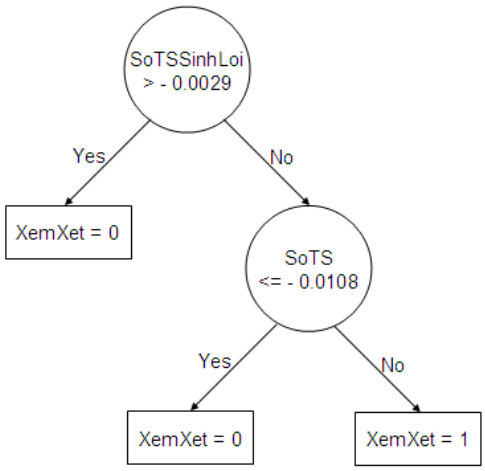
Nhận xét:

Kết quả trên cho thấy: Với những ngành nghề được chọn ở trên đều có một mức chung cho việc phân lớp. Nếu ĐTNT có tỷ suất sinh lợi năm sau giảm so với năm trước ở một mức nào đó thì sẽ phải xem xét lại ĐTNT đó. Ở đây mức phải xem xét là mức -0.00166, nghĩa là tỷ suất sinh lợi của các ngành đang xét nếu năm 2005 giảm đi 0.00166 so với tỷ suất sinh lợi của cùng ĐTNT trong năm 2004, ĐTNT sẽ được xếp vào loại cần xem xét.

Thực tế ĐTNT có tỷ suất sinh lợi giảm ở một mức nào đó, trong khi mức chung của ngành là phát triển, tỷ suất sinh lợi tăng hàng năm thì cần phải xem xét.

Áp dụng cũng số liệu này với công cụ See5 ta có kết quả sau:

Tỷ lệ lỗi là 8%, nghĩa là chính xác 82% - cao hơn so với thực hiện bằng ODM. Cây quyết định như sau:



Hình 3.7 Cây quyết định dùng See5 – Bài toán phân tích tỷ suất

Có thể thấy công cụ demo dựng cây chi tiết hơn, độ chính xác cũng cao hơn. Tuy nhiên với công cụ khai phá trên dữ liệu lớn sẽ có những xem xét để cân đối giữa độ phức tạp của cây với độ chính xác.

Với cây quyết định sinh bằng See5 có thể phát biểu kết quả như sau:

Nếu chênh lệch tỷ suất sinh lợi của ĐTNT so với năm trước giảm đi 0.0029 thì vẫn chưa cần xem xét. Nếu chênh lệch này giảm nhiều hơn 0.0029 thì cần xem xét đến Chênh lệch tỷ suất sinh lợi của ngành.

Nếu tỷ suất sinh lợi của ngành so với năm trước có giảm nhỏ hơn 0.0108 thì ĐTNT không cần xem xét, nếu so với năm trước tỷ suất sinh lợi năm nay giảm hơn 0.0108 thì cần xem xét ĐTNT đó.

3.5.2 Phân lớp ĐTNT theo số liệu của một năm

Xác định nội dung khai phá

So sánh số liệu của ĐTNT trong một năm và so với số bình quân tương ứng của ngành.

Các chỉ tiêu xem xét, lấy từ Báo cáo kết quả kinh doanh của ĐTNT:

— Tỷ suất sinh lợi = (Lợi nhuận thuần kinh doanh + Chi phí lãi vay) / Doanh thu thuần.

— Tổng doanh thu = Doanh thu thuần bán hàng và cung cấp dịch vụ + Doanh thu hoạt động tài chính + Thu nhập khác

— Chi phí = Chi phí tài chính + Chi phí bán hàng + Chi phí quản lý doanh nghiệp + Chi phí khác

Lựa chọn dữ liệu

Số liệu được lấy từ Báo cáo Kết quả hoạt động kinh doanh của ĐTNT.

Mã ngành nghề của ĐTNT được lấy theo dữ liệu ngành nghề.

Tiền xử lý dữ liệu

Lấy các chỉ tiêu cần thiết để tính Tỷ suất Sinh lợi, Tổng doanh thu, Chi phí trong mỗi năm.

Tính toán chỉ tiêu trung bình của ngành: Tỷ suất Sinh lợi trung bình, doanh thu trung bình, chi phí trung bình của ngành trong từng năm.

Cũng thử nghiệm trên cả See5, sẽ lọc lấy một phần nhỏ dữ liệu. Và lấy một số ngành nghề như với bài toán trên (các ngành sản xuất: K70, D26, I60, D22, C14, C10, J65).

Dữ liệu cho xây dựng cây quyết định như sau:

- Mã số thuế (TIN)
- Ngành sản xuất (chỉ lấy mức 3 ký tự) (NGANHSX)
- Tỷ suất sinh lợi (TS)
- Tổng doanh thu (DT)
- Chi phí (CP)
- Trường phân loại xác định ĐTNT có thuộc diện phải xem xét hay không (XEMXET)

Dữ liệu được để trong 2 view tương ứng với 3 bộ dữ liệu để xây dựng, kiểm thử và áp dụng với dữ liệu mới: `tr_So1Nganh_Build_v`, `tr_So1Nganh_Test_v`, `tr_So1Nganh_Apply_v`.

Thiết đặt các tham số và xác định ma trận chi phí:

Ma trận chi phí:

Chi phí	Dự báo cần xem xét 1	Dự báo không xem xét 0
Xem xét (thực tế) 1	0	5
Không xem xét (thực tế) 0	1	0

Chọn sử dụng thuật toán cây quyết định

Tạo mô hình:

Xây dựng cây quyết định từ `tr_So1Nganh_Build_v`.

Kiểm thử, đánh giá mô hình, áp dụng trên: `tr_So1Nganh_Test_v`

Kết quả:

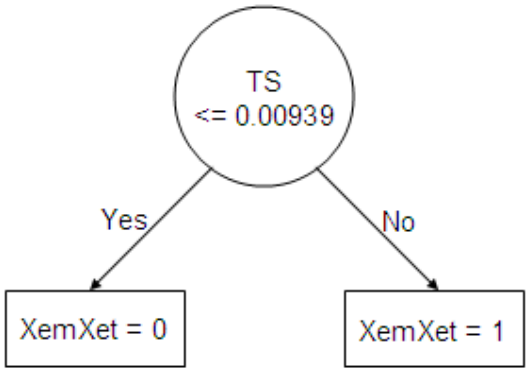
— Áp dụng trên dữ liệu kiểm thử (không dùng ma trận chi phí): đạt độ chính xác 80%. Với kết quả:

Giá trị thực	Giá trị dự báo	Số lượng
0	0	20
1	0	5

— Áp dụng trên dữ liệu kiểm thử (có dùng ma trận chi phí): đạt độ chính xác 96%. Với kết quả:

Giá trị thực	Giá trị dự báo	Số lượng
0	0	19
0	1	1
1	1	5

Cây quyết định như sau:



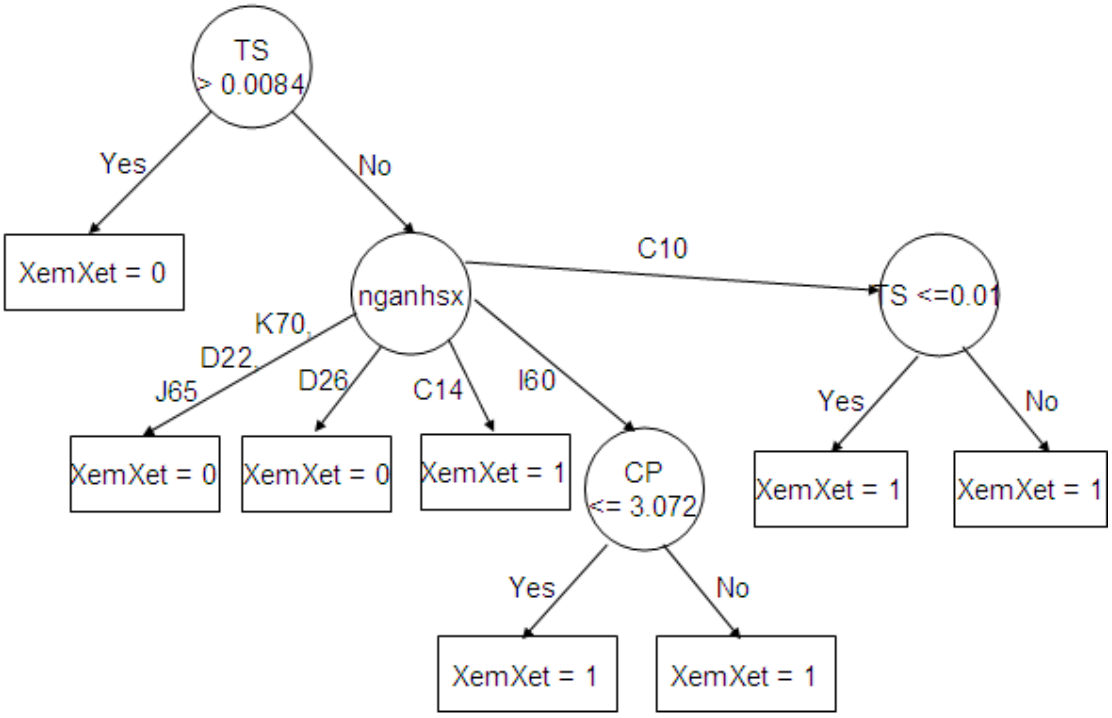
Hình 3.8 Cây quyết định dùng ODM – Bài toán xét số liệu một năm

Nhận xét:

Công cụ khai phá ODM đã dựa vào kết quả và xác định thuộc tính kiểm tra duy nhất là TS (tỷ suất sinh lợi) làm điều kiện cho xây dựng cây quyết định.Với kết quả trên: Với những ngành nghề đang xem xét đều có một mức chung cho việc phân lớp. Nếu ĐTNT có tỷ suất sinh lợi so với tỷ suất sinh lợi chung của ngành là nhỏ hơn 0.00939 thì không cần xem xét ĐTNT đó. Trường hợp ngược lại cần phải xem xét lại ĐTNT.

Áp dụng cũng số liệu này với công cụ See5 ta có kết quả sau:

Tỷ lệ lỗi là 1.3%, nghĩa là chính xác 89.7% - vẫn là cao hơn so với thực hiện bằng ODM. Cây quyết định như sau:



Hình 3.9 Cây quyết định dùng See5 – Bài toán phân tích trong năm

Nhận xét:

Có cùng nhận xét với bài toán trên, xây dựng cây bằng See5 sẽ chi tiết hơn, thuật toán quan tâm xây dựng đúng với mẫu huấn luyện nhất nên sẽ có cây kết quả phức tạp hơn.

Với cây quyết định sinh bằng See5 có thể phát biểu kết quả như sau:

Nếu chênh lệch tỷ suất sinh lợi của ĐTNT so với tỷ suất sinh lợi chung ít hơn 0.0084 thì chưa phải xem xét. Trường hợp ít nhiều hơn 0,0081 so với tỷ suất sinh lợi chung thì cần tiếp tục xem xét. Các xem xét tiếp sau sẽ thực hiện với ngành sản xuất. Nếu ngành trong số K70, D22, I65, và ngành = D36 thì không cần xem xét. Ngành C14 sẽ phải xem xét.

Trường hợp ngành sản xuất là I60 thì cần xét tiếp đến Chi phí (CP). Còn ngành sản xuất là C10 thì xem xét tiếp Tỷ suất sinh lợi chung của ngành (TS).

Thực tế, việc phối hợp nhiều chỉ tiêu và số thống kê trên ngành chính xác, thêm vào các kết quả thực tế đã thanh tra tại các ĐTNT hoặc các nhận định chính xác của những cán bộ thanh tra có kinh nghiệm sẽ cho phép xây dựng được mô hình phân lớp hoàn chỉnh hơn. Mô hình chính xác cao sẽ giúp nâng cao hiệu quả công tác quản lý Thuế.

CHƯƠNG 4. KẾT LUẬN

Với nội dung Nghiên cứu và áp dụng một số kỹ thuật khai phá dữ liệu trên CSDL ngành Thuế Việt Nam, luận văn là bước khởi đầu tìm hiểu các bài toán khai phá dữ liệu, tìm hiểu những vấn đề cần quan tâm khi khai phá dữ liệu để từ đó có thể đưa vào áp dụng trong thực tế.

Trong khuôn khổ luận văn chưa thể thử nghiệm khai phá, áp dụng nhiều kỹ thuật khai phá. Luận văn chỉ dừng lại ở mức áp dụng chủ yếu khai phá luật kết hợp và kỹ thuật phân lớp trên CSDL ngành Thuế. Mặc dù kết quả khai phá chưa mang nhiều ý nghĩa thực tế nhưng cũng đã đem lại ý nghĩa ban đầu của việc áp dụng các kỹ thuật khai phá để phát hiện ra những tri thức từ CSDL.

Những kết quả mà luận văn đã đạt được:

1. Tìm hiểu các chức năng và kỹ thuật cơ bản trong khai phá dữ liệu. Nắm được các trường hợp áp dụng.

2. Do điều kiện thời gian chưa cho phép đi sâu nghiên cứu kỹ tất cả các kỹ thuật khai phá dữ liệu, luận văn mới tập trung tìm hiểu chi tiết đối với chức năng khai phá luật kết hợp và khai phá bằng học cây quyết định. Nắm được các thuật toán, so sánh hiệu năng của các thuật toán, các vấn đề quan tâm khi cải tiến thuật toán khai phá luật kết hợp, các thuật toán mới đảm bảo hiệu năng.

3. Áp dụng thử nghiệm một số khai phá dữ liệu trên CSDL ngành Thuế. Qua đó có được những kinh nghiệm ban đầu khi khai phá tri thức trên dữ liệu thực:

a) Công việc chuẩn bị dữ liệu là một công việc rất quan trọng và mất nhiều thời gian. Thường thì dữ liệu thực luôn có những vấn đề phải xử lý

như dữ liệu thiếu, thậm chí CSDL thiểu hẳn những thông tin quan trọng cần cho khai phá.

b) Việc kết hợp với các chuyên gia phân tích là rất quan trọng để xác định được đúng các thuộc tính dự báo cũng như đưa ra yêu cầu cần thiết về thuộc tính đích và xác định các ngưỡng giá trị quan trọng.

HƯỚNG NGHIÊN CỨU TIẾP THEO

1. Tìm hiểu, nghiên cứu khai thác rộng và sâu hơn các tri thức về lý thuyết cơ bản của khai phá dữ liệu để có thể vận dụng vào thực tiễn chính xác hơn

2. Thử nghiệm và đánh giá kỹ hơn các thuật toán trên dữ liệu lớn

3. Khai phá dữ liệu trên kho dữ liệu với các luật kết hợp đa chiều, nhiều mức

4. Các hướng hiệu chỉnh số liệu

6. Tìm hiểu công cụ hỗ trợ hiển thị kết quả ở dạng đồ hoạ (đồ thị, biểu đồ…)

7. Thuyết phục khởi đầu dự án xây dựng hệ thống phân tích thông tin phục vụ quản lý thuế, đôn đốc nợ và thanh tra kiểm tra. Trong dự án sẽ có sự phối hợp chặt chẽ với chuyên gia phân tích nghiệp vụ trong các bước chuẩn bị khai phá dữ liệu và đánh giá kết quả.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Trương Ngọc Châu, Phan Văn Dũng (2002), *Nghiên cứu tính ứng dụng của khai thác luật kết hợp trong Cơ sở dữ liệu giao dịch*, Trường Đại học Bách Khoa, Đại học Đà Nẵng. http://www.ud.edu.vn/bankh/zipfiles/2_chau_truongngoc.doc
2. Nguyễn An Nhân (2001), *Khai phá dữ liệu và phát hiện luật kết hợp trong Cơ sở dữ liệu lớn*, Luận văn thạc sĩ ngành Công nghệ Thông tin, Trường Đại học Bách khoa Hà Nội.
3. Nguyễn Lương Thục (2002), *Một số phương pháp khai phá luật kết hợp và cài đặt thử nghiệm*, Luận văn thạc sĩ ngành Công nghệ Thông tin, Trường Đại học Bách khoa Hà Nội.

Tiếng Anh

4. Ashok Savasere, Edward Omiecinski, Shamkant Navathe (1995), *An Efficient, Algorithm for Mining Association Rules in Large Databases*, College of Computing Georgia Institute of Technology - Atlanta.
5. H.Hamilton. E. Gurak, L. Findlater W. Olive (2001), *Overview of Decision Trees*
6. Jeffrey D. Ullman (2003), *Data Mining Lecture Notes*, 2003's edition of CS345
7. Jiawei Han and Michelline Kamber (2000), *Data mining: Concepts and Techniques*, Morgan Kaufmann Publishers.

8. Jyothsna R. Nayak and Diane J.Cook (1998), *Approximate Association Rule Mining*, Department of Computer Science and Engineering, Arlington.
9. Mehmed Kantardzic (2003), *Data Mining: Concepts, Models, Methods, and Algorithms*, John Wiley & Sons.
10. Ming-Syan Chen, Jiawei Han, Philip S. Yu (1999), *Data Mining: An Overview from Database Perspective*, Natural Sciences and Engineering Research Council of Canada.
11. Oracle (2003), *Oracle Data Mining Concepts 10g Release 1 (10.1)*, Oracle Corporation.
12. Rakesh Agrawal, John C. Shafer (1996), *Parallel Mining of Association Rules: Design, Implementation and Experience*, IBM Research Report, IBM Research Division Almaden Research Center.
13. Rakesh Agrawal, Ramakrishnan Srikant (1994), *Fast Algorithms for Mining Association Rules*, IBM Almaden Research Center.
14. Ramakrishnan and Gehrke (2002), *Database Management Systems,* McGraw-Hill, *3rd* Edition.

PHỤ LỤC

Một số mã của phần khai phá dữ liệu trên CSDL ngành Thuế:

Khai phá luật kết hợp:

1. Chuẩn bị dữ liệu

```
drop table tr_dondoc;
  create table tr_dondoc as
  (select a.tin, a.nganhsx,
        a.tongDT/12 DT, PhaiNop/12 PN, 0 nopcham
    from tr_tysuat a
    where nam=2005);
--567 recs
update tr_dondoc a
set nopcham = 1
where exists (select tin from tr_nopcham b
     where b.tin = a.tin and
          to_char(b.ngay_bdau,'rrrr')='2005');
commit;
--178 recs
```

EXPORT → IMPORT VAO SH

```
drop table tr_dondoc1;
create table tr_dondoc1 as
(select tin, nganhsx,
decode(sign(dt - 100000000),-1,'VERY SMALL',
decode(sign(dt - 500000000),-1,'SMALL',
decode(sign(dt - 1000000000),-1,'MEDIUM',
decode(sign(dt-5000000000),-1,'LARGE',
'VERY LARGE')))) DT,
      decode(sign(round(PN/1000000) - 5), -1, '5',
decode(sign(round(PN/1000000) - 10), -1, '10',
decode(sign(round(PN/1000000) - 20), -1, '20',
decode(sign(round(PN/1000000) - 30), -1, '30',
```

```
decode(sign(round(PN/1000000) - 50), -1, '50',
decode(sign(round(PN/1000000) - 100), -1, '100',
decode(sign(round(PN/1000000) - 500), -1, '500',
decode(sign(round(PN/1000000) - 1000), -1, '1000',
decode(sign(round(PN/1000000) - 5000), -1, '5000',
     '>5000'))))))))) PN,    nopcham
from tr_dondoc);
```

2. Chuyển về đúng khuôn dạng cho khai phá luật kết hợp

```
drop table tr_dondoc2;
create table tr_dondoc2 as
(select tin, nganhsx, 1 has_it
from tr_dondoc1
union
select tin, dt, 1 has_it
from tr_dondoc1
union
select tin, to_char(pn) pn, 1 has_it
from tr_dondoc1
union
select tin, to_char(nopcham) nopcham, 1 has_it
from tr_dondoc1);

GRANT SELECT ON TR_dondoc2 TO DMUSER;

DROP VIEW TR_dondoc ;
CREATE VIEW TR_dondoc AS
SELECT * FROM sh.tr_dondoc2;

DROP VIEW TR_dondoc_AR;
CREATE VIEW TR_dondoc_AR AS
SELECT TIN,
       CAST(COLLECT(DM_Nested_Numerical(
```

```
            SUBSTRB(nganhsx, 1, 10), has_it))
          AS DM_Nested_Numericals) tinnganhsx
  FROM tr_dondoc
GROUP BY TIN;
```

3. Thiết đặt các tham số

```
BEGIN EXECUTE IMMEDIATE
     'DROP TABLE ar_dondoc_settings';
EXCEPTION WHEN OTHERS THEN NULL; END;
/
set echo off
CREATE TABLE ar_dondoc_settings (
  setting_name  VARCHAR2(30),
  setting_value VARCHAR2(30));
set echo on

BEGIN
  INSERT INTO ar_dondoc_settings VALUES
  (dbms_data_mining.asso_min_support,0.1);
  INSERT INTO ar_dondoc_settings VALUES
  (dbms_data_mining.asso_min_confidence,0.1);
  INSERT INTO ar_dondoc_settings VALUES
  (dbms_data_mining.asso_max_rule_length,2);
  COMMIT;
END;
```

4. Xây dựng mô hình

```
BEGIN DBMS_DATA_MINING.DROP_MODEL('AR_dondoc_nghe');
EXCEPTION WHEN OTHERS THEN NULL; END;
/
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'AR_dondoc_nghe',
   mining_function     => DBMS_DATA_MINING.ASSOCIATION,
```

```
        data_table_name     => 'TR_dondoc_AR',
        case_id_column_name => 'TIN',
        settings_table_name => 'ar_dondoc_settings');
    END;
    /
```

5. Lấy kết quả khai phá

Danh sách frequent itemsets:

```
SELECT item, support, number_of_items
  FROM (SELECT I.column_value AS item,
               F.support,
               F.number_of_items
          FROM
          TABLE(DBMS_DATA_MINING.GET_FREQUENT_ITEMSETS(
                       'AR_dondoc_nghe', 10)) F,
          TABLE(F.items) I
   ORDER BY number_of_items, support, column_value);
```

Danh sách các luật:

```
SELECT ROUND(rule_support,4) support,
       ROUND(rule_confidence,4) confidence,
       antecedent,
       consequent
  FROM TABLE(DBMS_DATA_MINING.GET_ASSOCIATION_RULES
          ('AR_dondoc_nghe', 10))
  ORDER BY confidence DESC, support DESC;
```

Phân lớp, dự báo bằng cây quyết định:

1. Chuẩn bị dữ liệu

create table tr_sinhloi as

(select a.tin, a.nganhsx, sotssinhloi, SoTS, 0 xemxet

```
from tr_so_1DT a, SoNganh b
where a.nganhsx = b.nganhsx);

create table tr_So1Nganh as
(select a.tin, a.nganhsx,  a.nam, (b.ts_nganh - a.tssinhloi) ts,
      (b.DTnganh - a.TongDT) DT,
       (a.ChiPhi - b.ChiPhiNganh) CP, 0 xemxet
from  tr_tysuat a, tr_nganh2004 b
where a.nam=2004 and a.nganhsx=b.nganhsx
union
select a.tin, a.nganhsx,  a.nam, (b.ts_nganh - a.tssinhloi) ts,
      (b.DTnganh - a.TongDT) DT,
       (a.ChiPhi - b.ChiPhiNganh) CP,  0 xemxet
from  tr_tysuat a, tr_nganh2005 b
where a.nam=2005 and a.nganhsx=b.nganhsx);
```

2. Tạo ma trận chi phí

```
DROP TABLE dt_sh_NOP_cost;
CREATE TABLE dt_sh_NOP_cost (
  actual_target_value           NUMBER,
  predicted_target_value        NUMBER,
  cost                          NUMBER);
INSERT INTO dt_sh_NOP_cost VALUES (0,0,0);
INSERT INTO dt_sh_NOP_cost VALUES (0,1,1);
INSERT INTO dt_sh_NOP_cost VALUES (1,0,5);
INSERT INTO dt_sh_NOP_cost VALUES (1,1,0);
COMMIT;
```

3. Thiết lập các tham số

```
DROP TABLE dt_sh_BTC_settings;
CREATE TABLE dt_sh_BTC_settings (
```

```
    setting_name  VARCHAR2(30),
    setting_value VARCHAR2(30));
BEGIN
  -- Populate settings table
  INSERT INTO dt_sh_BTC_settings VALUES
    (dbms_data_mining.algo_name,
     dbms_data_mining.algo_decision_tree);
  INSERT INTO dt_sh_BTC_settings VALUES
    (dbms_data_mining.clas_cost_table_name,
     'dt_sh_NOP_cost');
  COMMIT;
END;
/
```

4. Tạo mô hình

```
BEGIN
DBMS_DATA_MINING.DROP_MODEL('DT_SH_Clas_TS1DT');
EXCEPTION WHEN OTHERS THEN NULL;
END;
/

BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name       => 'DT_SH_Clas_TS1DT',
   mining_function  => dbms_data_mining.classification,
    data_table_name     => 'tr_so_1DT_v',
    case_id_column_name => 'tin',
    target_column_name  => 'xemxet',
    settings_table_name => 'dt_sh_BTC_settings');
END;
```

TÓM TẮT LUẬN VĂN

Khai phá dữ liệu thực sự ngày càng trở nên quan trọng và cấp thiết, nhất là với những nơi nắm giữ lượng dữ liệu khổng lồ. Kho dữ liệu ngành Thuế được lưu giữ qua nhiều năm, khám phá những tri thức tiềm ẩn trong những dữ liệu này chắc chắn sẽ hỗ trợ không nhỏ cho công tác quản lý Thuế. Nghiên cứu những chức năng khai phá dữ liệu và thử nghiệm khả năng áp dụng trên CSDL ngành Thuế chính là mục đích chính của Luận văn.

Qua tìm hiểu những chức năng cơ bản của khai phá dữ liệu, luận văn tập trung hơn vào nghiên cứu các kỹ thuật khai phá luật kết hợp và phân lớp bằng học cây quyết định. Hiểu được các thuật toán hiệu quả gần đây, từ đó nắm được những điểm chính cần quan tâm giải quyết trong mỗi kỹ thuật khai phá, như: Xử lý dữ liệu thiếu, cắt tỉa giảm kích thước, giảm lần duyệt CSDL.

Lựa chọn công cụ Oracle Data Mining (ODM) của Oracle để khai phá tri thức trên CSDL ngành Thuế. Thực nghiệm khai phá luật kết hợp thể hiện mối liên quan giữa ngành nghề kinh doanh của ĐTNT, quy mô doanh nghiệp, doanh thu trung bình, mức thuế phải nộp với ý thức chấp hành nghĩa vụ nộp thuế. Tiếp theo áp dụng phương pháp phân lớp bằng cây quyết định để phân lớp và dự báo trên CSDL ngành Thuế: Phân lớp ĐTNT dựa vào một số chỉ tiêu phân tích (ngành nghề, tỷ suất sinh lợi, tổng doanh thu, chi phí, thuế phải nộp) đưa ra phân loại trên thuộc tính đích trả lời câu hỏi ĐTNT có thuộc diện nghi ngờ vi phạm về Thuế không–là tri thức trợ giúp thanh tra Thuế.

Các tri thức khai phá thực nghiệm chắc chắn còn nhiều thiếu sót, rất mong nhận được góp ý từ các thầy cô và các chuyên gia Thuế. Hy vọng khai phá được hoàn thiện trong dự án khai phá dữ liệu Thuế phục vụ công tác Thanh tra – nơi hội đủ yếu tố thành công: Kết hợp chặt chẽ giữa kỹ thuật với các chuyên gia nghiệp vụ - có kinh nghiệm quý báu làm căn cứ khám phá tri thức.