**ỦY BAN NHÂN DÂN TP HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

**Họ và tên sinh viên :**

Phạm Minh Nhật – 3116410179

MSSV : 3116410179

Lớp : DCT1165

Môn học : Khai Phá Dữ Liệu

**Giảng viên hướng dẫn :**

TS. Lê Minh Nhựt Triều

**ĐỀ TÀI**

PHÂN LỚP CÂY QUYẾT ĐỊNH VÀ CÀI ĐẶT THUẬT TOÁN C4.5

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn :

Để hoàn thành chuyên đề báo cáo học phần này trước hết em xin gửi đến quý thầy cô trong khoa CÔNG NGHỆ THÔNG TIN trường Đại Học Sài Gòn lời cảm ơn chân thành.

Cảm ơn thầy cô đã giảng dạy và trang bị những kiến thức quý báu cho em trong suốt quá trình học tập tại trường.

Em xin gửi đến giảng viên hướng dẫn đồ án học phần cho em là thầy LÊ MINH NHỰT TRIỀU đã tận tình hướng dẫn, giúp đỡ em hoàn thành báo cáo này lời cảm ơn sâu sắc nhất.

Vì kiến thức bản thân em còn hạn chế, trong quá trình làm việc, hoàn thiện chuyên đề này em không tránh khỏi những sai sót, kính mong nhận được những ý kiến đóng góp từ thầy.

Xin chân thành cám ơn và kính chúc quý thầy cô, kính chúc ban lãnh đạo dồi dào sức khỏe, công tác tốt.

**Nhận xét của giảng viên hướng dẫn**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

**…………………………………………………………………………**

Lời nói đầu

Ngày nay những dữ liệu nghiệp vụ được con người ta tạo ra trong qua trình hoạt động ngày càng có kích thước càng lớn và có thể chứa những quy luật chưa được khám phá. Từ nguyên nhân đó việc tìm cách làm sao để trích rút từ những dữ liệu đó ra các luật, hay dự đoán những xu hướng tương lai. Các hoạt động thực tiễn, quá trình nghiên cứu khoa học được dựa trên những quy tắc thông minh cua nghiệp vụ, chính vì vậy công nghệ phân lớp và dự đoán dữ liệu ra đời.

Các công nghệ phân lớp dữ liệu ngày nay đang phát triển mạnh mẽ. thu hút sự quan tâm của các nhà nghiên cứu trong nhiều lĩnh vực khác nhau như maketing, thường mại, ngân hàng,…

Lần lượt ra đời các kỹ thuật như : phân lớp Bayesian, K- nearest nieghbor classifier, mạng nơ-ron, thống kế,… và đặc biết là phân lớp cây quyết định. Được xem là một công cụ mạnh, phổ biến và đặc biệt khả năng mở rộng.

Đứng trước nhu cầu này em đã thực hiện bước nghiên cứu tổng quan về công nghệ phân lớp dữ liệu nói chung và phân lớp dữ liệu dựa trên cây quyết định nói riêng và thuật toán C4.5.

Để triển khai thành công cài đặt và thử nghiệm mô hình phân lớp dữ liệu thực tế thì việc tìm hiểu và các thuật toán và đánh giá chúng dựa trên giá trị khoa học, thực tiễn là điều thiết yếu.

Mục lục

[A. TỔNG QUAN VỀ PHÂN LỚP DỮ LIỆU CÂY QUYẾT ĐỊNH 7](#_Toc43991374)

[**I.** **Phân lớp dữ liệu** 7](#_Toc43991375)

[1. Các vấn đề liên quan đến việc phân lớp dữ liệu 7](#_Toc43991376)

[*1.1* *Giai đoạn chuẩn bị dữ liệu cho việc phân lớp* 7](#_Toc43991377)

[*1.2* *So sánh các mô hình phân lớp* 8](#_Toc43991378)

[*1.3* *Các phương pháp đánh giá độ chính xác* 9](#_Toc43991379)

[**II.** **Cây quyết định** 10](#_Toc43991380)

[1. Khái niệm cây quyết định 10](#_Toc43991381)

[2. Ưu điểm của cây quyết định 14](#_Toc43991382)

[**III.** **Ứng dụng của cây quyết định** 15](#_Toc43991383)

[1. Định nghĩa 15](#_Toc43991384)

[2. Các vấn đề khác liên quan đến sử dụng cây quyết định 16](#_Toc43991385)

[*2.1* *Tránh quá vừa dữ liệu* 17](#_Toc43991386)

[*2.2* *Vấn đề với thuộc tính liên tục* 17](#_Toc43991387)

[3. Đánh giá 18](#_Toc43991388)

[*3.1* *Điểm mạnh* 18](#_Toc43991389)

[*3.2* *Điểm yếu* 19](#_Toc43991390)

[4. Xây dựng cây quyết định 20](#_Toc43991391)

[**IV.** **Thuật toán xây dựng cây quyết định** 21](#_Toc43991392)

[1. Tổng quan về tư tưởng 21](#_Toc43991393)

[2. Quy nạp tuyến tính 22](#_Toc43991394)

[3. Cây quyết định tuần tự và thuật toán hóa 23](#_Toc43991395)

[4. Cách đo độ lợi thông tin (Information Gain) 26](#_Toc43991396)

[B. THUẬT TOÁN C4.5 27](#_Toc43991397)

[**I.** **Gain – entropy độ đo lựa chọn thuộc tính tốt nhất** 28](#_Toc43991398)

[**II.** **Xử lí giá trị bị thiếu theo cơ chế riêng** 35](#_Toc43991399)

[**III.** **C4.5 tránh quá vừa dữ liệu** 36](#_Toc43991400)

[**IV.** **Phương pháp chuyển đổi từ cây sang luật** 37](#_Toc43991401)

[**V.** **C4.5 là thuật toán hiệu quả cho những tập dữ liệu vừa và nhỏ** 38](#_Toc43991402)

[C. CHƯƠNG TRÌNH CODE THUẬT TOÁN C4.5 38](#_Toc43991403)

[**I.** **Giới thiệu tổng quát** 38](#_Toc43991404)

[**II.** **Mã chương trình** 38](#_Toc43991405)

[1. Tính entropy 38](#_Toc43991406)

[2. Tính Info Gain 40](#_Toc43991407)

[3. Tính Gain Ratio 41](#_Toc43991408)

[4. Tạo Node 43](#_Toc43991409)

[5. Lập cây 48](#_Toc43991410)

[D. TÀI LIỆU THAM KHẢO 52](#_Toc43991411)

1. TỔNG QUAN VỀ PHÂN LỚP DỮ LIỆU CÂY QUYẾT ĐỊNH
2. **Phân lớp dữ liệu**
   * + - Là một kĩ thuật dựa vào tập huấn luyện và những giá trị hay là nhãn của lớp trong một thuộc tính phân lớp nào đó và nó được sử dụng để phân lớp dữ liệu mới. Bên cạnh đó phân lớp cũng là tiên đoán loại lớp của nhãn, ngoài ra kĩ thuật phân lớp có một hình thức tương tự là kĩ thuật tiên đoán , kĩ thuật tiên đoán khác với phân lớp ở chỗ **phân lớp** chỉ liên quan đến tiên đoán **loại lớp của nhãn** còn kĩ thuật **tiên đoán** mô hình những hàm **đánh giá liên tục.**
       - Kĩ thuật phân lớp được tiến hành bao gồm 2 bướ*c* : Xây dựng mô hình và sử dụng mô hình .
         1. **Xây dựng mô hình :** là mô tả một tập những lớp được định nghĩa trước trong đó mỗi bộ hoặc mẫu được gán thuộc tính về một lớp được định nghĩa xác định bởi thuộc tính nhãn lớp, tập hợp của những bộ này được gọi là tập huấn luyện. Mô hình được biểu diễn là những luật phân lớp, cây quyết định và những công thức toán học.
         2. **Sử dụng mô hình :** Việc sử dụng mô hình phục vụ cho mục đích phân lớp dữ liệu. Trước khi sử dụng mô hình chúng ta cần đánh giá tính chính xác của mô hình trong đó. Đánh giá bằng cách dùng mẫu kiểm tra được so sánh với kết quả phân lớp của mô hình, tập kiểm tra là độc lập với tập huấn luyện .
       - Phân lớp là một hình thức **học được giám sát** tức làmột tập dữ liệu huấn luyện ( quan sát, thẩm định...) , những dữ liệu mới được phân lớp dựa trên tập huấn luyện .
       - Ngược lại với hình thức học được giám sát là hình thức **học không được giám sát** lúc đó nhãn lớp của tập dữ liệu huấn luyện là không được biết đến.
3. Các vấn đề liên quan đến việc phân lớp dữ liệu
   1. *Giai đoạn chuẩn bị dữ liệu cho việc phân lớp*
      * + Trước khi xử lý dữ liệu cho quá trình phân lớp cần trải qua một số công việc không thể thiếu, vì nó mang tính quyết định đến việc có áp dụng mô hình được hay không, ngoài ra nó còn giúp chúng ta trong việc cải thiện độ chính xác, tính hiểu quả, khả năng mở rộng mô hình phân lớp.
        + Quá trình tiền xử lý dữ liệu gồm có các công việc sau:
          1. **Làm sạch dữ liệu:**

Làm sạch dữ liệu liên quan đến việc xử lý với **lỗi (noise)** và giá trị thiếu (missing value) trong tập dữ liệu ban đầu. Noise là các **lỗi ngẫu nhiên hay các giá trị không hợp lệ** của các biến trong tập dữ liệu. Để xử lý với loại lỗi này có thể dùng kỹ thuật làm trơn. **Missing value** là những **ô không có giá trị** của các thuộc tính. Giá trị thiếu có thể do lỗi chủ quan trong quá trình nhập liệu, hoặc trong trường hợp cụ thể giá trị của thuộc tính đó không có, hay không quan trọng. Kỹ thuật xử lý ở đây có thể bằng cách thay giá trị thiếu đó bằng giá trị phổ biến nhất của thuộc tính đó hoặc bằng giá trị có thể xảy ra nhất dựa trên thống kê. Mặc dù phần lớn thuật toán phân lớp đều có cơ chế xử lý với những giá trị thiếu và lỗi trong tập dữ liệu, nhưng bước tiền xử lý này có thể ***làm giảm sự hỗn độn*** rong quá trình xây dựng mô hình phân lớp.

* + - * 1. **Phân tích sự cần thiết của dữ liệu:**

Có rất nhiều thuộc tính trong tập dữ liệu có thể hoàn toàn không cần thiết hay liên quan đến một bài toán phân lớp cụ thể. Ví dụ dữ liệu về ngày trong tuần hoàn toàn không cần thiết đối với ứng dụng phân tích độ rủi ro của các khoản tiền cho vay của ngân hàng nên thuộc tính này là dư thừa. Phân tích sự cần thiết của dữ liệu **nhằm mục đích loại bỏ** những **thuộc tính không cần thiết, dư thừa** khỏi quá trình phân lớp vì thuộc tính đó sẽ làm chậm, phức tạp và gây ra sụ hiểu sai trong quá trình phân lớp dẫn tới mô hình phân lớp không đúng.

* + - * 1. **Chuyển đổi dữ liệu:**

Việc khái quát hóa dữ liệu lên mức khái niệm cao hơn đôi khi là cần thiết trong quá trình tiền xử lý. Việc này đặc biệt **hữu ích** với những **thuộc tính liên tục**. Ví dụ: các giá trị số của thuộc tính thu nhập của khách hàng có thể được khái quát hóa thành các dãy giá trị rời rạc: thấp, trung bình, cao. Tương tự với những thuộc tính rời rạc như địa chỉ phố có thể được khái quát hóa lên thành thành phố. Việc khái quát hóa làm cô đọng dữ liệu học nguyên thủy, vì vậy các thao tác vào/ra liên quan đến quá trình phân lớp sẽ giảm.

* 1. *So sánh các mô hình phân lớp*
* Trong từng ứng dụng cụ thể cần lựa chọn mô hình phân lớp phù hợp. Việc lựa chọn đó căn cứ vào sự so sánh các mô hình phân lớp khác nhau, dựa trên các tiêu chuẩn sau:
  + - * *Độ chính xác dự đoán* (predictive accuracy): Độ chính xác là khả năng của mô hình để dự đoán chính xác nhãn lớp của dữ liệu mới hay dữ liệu chưa biết.
      * *Tốc độ* (speed): Tốc độ là chi phí tính toán liên quan đến quá trình tạo ra và sử dụng mô hình.
      * *Sức mạnh* (robustness): Sức mạnh là khả năng mô hình tạo ra những dự đoán đúng từ những dữ liệu noise hay dữ liệu với những giá trị thiếu.
      * *Khả năng mở rộng* (Scalability): là khả năng thực thi hiệu quả trên lượng lớn dữ liệu của mô hình đã học.
      * *Tính hiểu được* (interpretability): Tính hiểu được la mức độ hiểu và hiểu rõ những kết quả sinh ra bởi mô hình đã học.
      * *Tính đơn giản* (simplicity): Tính đơn giản liên quan đến kích thước của cây quyết định hay độ cô đọng của các luật.
  1. *Các phương pháp đánh giá độ chính xác*
* Để dự đoán chính xác kết quả phân lớp của những dữ liệu tương lai cần ước lượng được độ chính xác của các bộ phân lớp. Độ chính xác còn giúp so sánh các mô hình phân lớp khác nhau. Sau đây là hai phương pháp đánh giá phổ biến là *holdout* và *k-fold cross-validation*. Được dựa trên các phân hoạch ngẫu nhiên từ tập dữ liệu ban đầu.
  + - * **Phương pháp *holdout*** : những dữ liệu đưa ra được phân chia ngẫu nhiên thành 2 phần là: tập dữ liệu đào tạo và tập dữ liệu kiểm tra. Thông thường 2/3 dữ liệu cấp cho dữ liệu đào tạo, phần còn lại cho tập dữ liệu kiểm tra.
      * **Phương pháp *k-fold cross-validation* :** những tập dữ liệu ban đầu được chia ngẫu nhiên thành k tập con (fold) có kích thước xấp xỉ nhau S1,S2,…,Sk. Quá trình phân lớp và test được thực hiện k lần. Tại lần lặp thứ i, Si là tập dữ liệu kiểm tra, các tập còn lại hợp thành tập dữ liệu đào tạo. Có nghĩa là đầu tiên việc phân lớp được thực hiện trên các tập S2, S3,…,Sk sau đó test trên tập S1, tiếp tục quá trình phân lớp được thực hiện trên tập S1,S3,S4,…,Sk sau đó test trên tập S2 và cứ tiếp tục. Độ chính xác là toàn bộ số phân lớp đúng từ k lần lặp chia cho tổng số mẫu cảu tập dữ liệu ban đầu.

1. **Cây quyết định**

* Trong [lý thuyết quyết định](http://vi.wikipedia.org/w/index.php?title=L%C3%BD_thuy%E1%BA%BFt_quy%E1%BA%BFt_%C4%91%E1%BB%8Bnh&action=edit) một cây quyết định (Decision tree) là một [đồ thị](http://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B) của các quyết định và các hậu quả có thể có của nó (bao gồm rủi ro và hao phí tài nguyên). Cây quyết định được sử dụng để xây dựng một [kế hoạch](http://vi.wikipedia.org/w/index.php?title=K%E1%BA%BF_ho%E1%BA%A1ch&action=edit) nhằm đạt được [mục tiêu](http://vi.wikipedia.org/w/index.php?title=M%E1%BB%A5c_ti%C3%AAu&action=edit) mong muốn. Các cây quyết định được dùng để **hỗ trợ quá trình ra quyết định**. Cây quyết định là một dạng đặc biệt của [cấu trúc cây](http://vi.wikipedia.org/w/index.php?title=C%E1%BA%A5u_tr%C3%BAc_c%C3%A2y&action=edit).

1. Khái niệm cây quyết định

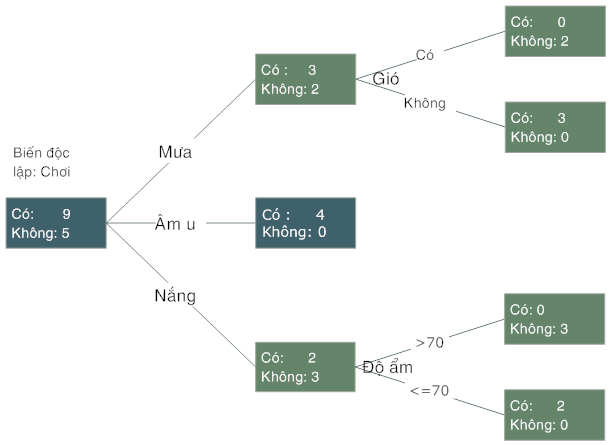
* Cây quyết định (Decision Tree) là một cây **phân cấp** có cấu trúc được dùng để phân lớp các đối tượng **dựa vào dãy các luật** (series of rules). Các thuộc tính của đối tượng (ngoại trừ thuộc tính phân lớp – Category attribute) có thể thuộc các kiểu dữ liệu khác nhau (Binary, Nominal, ordinal, quantitative values) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.
* Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các đối tượng chưa biết (unseen data)
* Cây quyết định là một **flow-chart** giống cấu trúc cây , **nút bên trong** biểu thị một **kiểm tra** trên **một thuộc tính** , **nhánh** **biểu diễn đầu ra** của kiểm tra , **nút lá biểu diễn nhãn lớp hoặc sự phân bố của lớp**.
* Cây quyết định còn có hai tên khác:
  + - * ***Cây hồi quy (Regression tree):*** ước lượng các hàm có giá trị là số thực thay vì được sử dụng cho các nhiệm vụ phân loại. (ví dụ: ước tính giá một ngôi nhà hoặc khoảng thời gian một bệnh nhân nằm viện)
      * ***Cây phân loại (Classification tree):*** nếu x là một biến phân loại như: giới tính (nam hay nữ), kết quả của một trận đấu (thắng hay thua).
* Việc tạo cây quyết định bao gồm 2 giai đoạn**:** Tạo cây và tỉa cây.
  + - * **Để tạo cây** ở thời điểm bắt đầu tất cả những ví dụ huấn luyện là ở gốc sau đó phân chia ví dụ huấn luyện theo cách đệ qui dựa trên thuộc tính được chọn.
      * **Việc tỉa cây** là xác định và xóa những nhánh mà có phần tử hỗn loạn hoặc những phần tử nằm ngoài (những phần tử không thể phân vào một lớp nào đó).
* Việc sử dụng cây quyết định như sau**:** Kiểm tra những giá trị thuộc tính của mẫu đối với cây quyết định.
* Trong lĩnh vực phân lớp, cây quyết định là một kiểu **mô hình dự báo** (predictive model), nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật/hiện tượng. **Mỗi một nút** trong (internal node) tương **ứng với một biến**; đường nối giữa nó với nút con của nó thể hiện một giá trị cụ thể cho biến đó. Mỗi nút lá đại diện cho giá trị dự đoán của biến mục tiêu, cho trước các giá trị của các biến được biểu diễn bởi đường đi từ nút gốc tới nút lá đó.
* Phân lớp bằng cây quyết định cũng là một **phương pháp thông dụng trong** [**khai phá dữ liệu**](http://vi.wikipedia.org/wiki/Khai_ph%C3%A1_d%E1%BB%AF_li%E1%BB%87u). Khi đó, cây quyết định mô tả một cấu trúc cây, trong đó, các lá đại diện cho các phân loại còn cành đại diện cho các kết hợp của các thuộc tính dẫn tới phân loại đó. Một cây quyết định có thể được phân lớp bằng cách chia [tập hợp](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_h%E1%BB%A3p) nguồn thành các tập con dựa theo một kiểm tra giá trị thuộc tính . Quá trình này được lặp lại một cách đệ qui cho mỗi tập con dẫn xuất. **Quá trình** [**đệ qui**](http://vi.wikipedia.org/wiki/%C4%90%E1%BB%87_qui) **hoàn thành** khi không thể tiếp tục thực hiện việc chia tách được nữa, hay khi một phân loại đơn có thể áp dụng cho từng phần tử của tập con dẫn xuất. Một bộ phân loại [rừng ngẫu nhiên](http://vi.wikipedia.org/w/index.php?title=R%E1%BB%ABng_ng%E1%BA%ABu_nhi%C3%AAn&action=edit) (random forest) sử dụng một số cây quyết định để có thể cải thiện tỉ lệ phân loại.
* Cây quyết định cũng là một phương tiện có tính mô tả dành cho việc **tính toán các** [**xác suất có điều kiện**](http://vi.wikipedia.org/wiki/X%C3%A1c_su%E1%BA%A5t_c%C3%B3_%C4%91i%E1%BB%81u_ki%E1%BB%87n).
* Cây quyết định có thể được mô tả như là sự kết hợp của các kỹ thuật toán học và tính toán nhằm hỗ trợ việc mô tả, phân loại và tổng quát hóa một tập dữ liệu cho trước.
* Dữ liệu được cho dưới dạng các bản ghi có dạng:



* + - * **Biến phụ thuộc (dependant variable) y** là biến mà chúng ta cần tìm hiểu, phân loại hay tổng quát hóa.  là các biến sẽ giúp ta thực hiện công việc đó.
* Ví dụ :
  + - * Ta sẽ dùng một ví dụ để giải thích về cây quyết định:
      * Nam là một quản lý của một câu lạc bộ chơi tennis, nam đang có rắc rối chuyện các thành viên đến hay không đến. Có ngày ai cũng muốn chơi tennis nhưng số nhân viên câu lạc bộ lại không đủ phục vụ. Có hôm chẳng ai đến chơi, và câu lạc bộ lại thừa nhân viên.
      * Mục tiêu của Nam lúc này là tối ưu hóa số nhân viên phục vụ mỗi ngày bằng cách dựa theo thông tin dự báo thời tiết để đoán xem khi nào người ta sẽ đến chơi tennis. Để thực hiện điều đó, Nam cần hiểu được tại sao khách hàng quyết định chơi và tìm hiểu xem có cách giải thích nào cho việc đó hay không.
      * Vậy là trong hai tuần, anh ta thu thập thông tin về:
      * Trời(outlook) : nắng(sunny), nhiều mây(clouded) hoặc mưa (rainy). Nhiệt độ (temperature) : Độ ẩm(humidity). Có gió mạnh (windy) hay không.
      * Và tất nhiên là số người đến chơi tennis vào hôm đó. David thu được một bộ dữ liệu gồm 14 dòng và 5 cột.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dữ liệu chơi tennis** | | | | |
| **Các biến độc lập** | | | |  |
| **Quang cảnh** | **Nhiệt độ** | **Độ ẩm** | **Gió** | **Chơi** |
| Nắng | 85 | 85 | không | không |
| Nắng | 80 | 90 | có | không |
| Âm u | 83 | 78 | không | có |
| Mưa | 70 | 96 | không | có |
| Mưa | 68 | 80 | không | có |
| Mưa | 65 | 70 | có | không |
| Âm u | 64 | 65 | có | có |
| Nắng | 72 | 95 | không | không |
| Nắng | 69 | 70 | không | có |
| Mưa | 75 | 80 | không | có |
| Nắng | 75 | 70 | có | có |
| Âm u | 72 | 90 | có | có |
| Âm u | 81 | 75 | không | có |
| mưa | 71 | 80 | có | không |

* + - * Sau đó, để giải quyết bài toán của Nam, người ta đã đưa ra một mô hình cây quyết định.



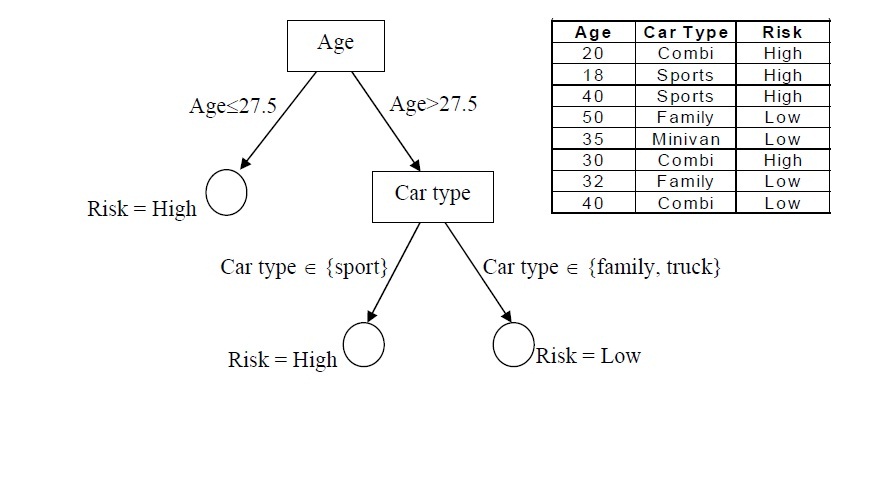
* Cây quyết định là một mô hình dữ liệu mã hóa phân bố của nhãn lớp (cũng là y) theo các thuộc tính dùng để dự đoán. Đây là một đồ thị có hướng phi chu trình dưới dạng một cây. Nút gốc (nút nằm trên đỉnh) đại diện cho toàn bộ dữ liệu. Thuật toán cây phân loại phát hiện ra rằng cách tốt nhất để giải thích biến phụ thuộc, play(chơi), là sử dụng biến outlook. Phân loại theo các giá trị của biến outlook, ta có ba nhóm khác nhau: Nhóm người chơi tennis khi trời nắng, nhóm chơi khi trời nhiều mây, và nhóm chơi khi trời mưa.
* Kết luận thứ nhất: nếu trời nhiều mây, người ta luôn luôn chơi tennis. Và có một số người đam mê đến mức chơi tennis cả khi trời mưa.
* Tiếp theo, ta lại chia nhóm trời nắng thành hai nhóm con. Ta thấy rằng khách hàng không muốn chơi tennis nếu độ ẩm lên quá 70%.
* Cuối cùng, ta chia nhóm trời mưa thành hai và thấy rằng khách hàng sẽ không chơi tennis nếu trời nhiều gió.
* Và đây là lời giải ngắn gọn cho bài toán mô tả bởi cây phân loại. Nam cho rằng phần lớn nhân viên nghỉ vào những ngày trời nắng và ẩm, hoặc những ngày mưa gió. Vì hầu như sẽ chẳng có ai chơi golf trong những ngày đó. Vào những hôm khác, khi nhiều người sẽ đến chơi tennis, anh ta có thể thuê thêm nhân viên thời vụ để phụ giúp công việc.
* Kết luận là cây quyết định giúp ta biến một biểu diễn dữ liệu phức tạp thành một cấu trúc đơn giản hơn rất nhiều.

1. Ưu điểm của cây quyết định

* So với các phương pháp [khai phá dữ liệu](http://vi.wikipedia.org/wiki/Khai_ph%C3%A1_d%E1%BB%AF_li%E1%BB%87u) khác, cây quyết định là phương pháp có một số ưu điểm:
  + Cây quyết định **dễ hiểu**. Người ta có thể hiểu mô hình cây quyết định sau khi được giải thích ngắn.
  + Việc **chuẩn bị dữ liệu** cho một cây quyết định là **cơ bản hoặc không cần thiết**. Các kỹ thuật khác thường đòi hỏi [chuẩn hóa dữ liệu](http://vi.wikipedia.org/w/index.php?title=Chu%E1%BA%A9n_h%C3%B3a_d%E1%BB%AF_li%E1%BB%87u&action=edit), cần tạo các biến phụ (dummy variable) và loại bỏ các giá trị rỗng.
  + Cây quyết định có thể **xử lý** cả **dữ liệu** có giá trị bằng **số** và dữ liệu có giá trị là **tên** thể loại. Các kỹ thuật khác thường chuyên để phân tích các bộ dữ liệu chỉ gồm một loại biến. Chẳng hạn, các luật quan hệ chỉ có thể dùng cho các biến tên, trong khi [mạng nơ-ron](http://vi.wikipedia.org/wiki/M%E1%BA%A1ng_n%C6%A1-ron) chỉ có thể dùng cho các biến có giá trị bằng số.
  + Cây quyết định là một **mô hình hộp trắng**. Nếu có thể quan sát một tình huống cho trước trong một mô hình, thì có thể dễ dàng giải thích điều kiện đó bằng logic Boolean. Mạng nơ-ron là một ví dụ về mô hình hộp đen, do lời giải thích cho kết quả quá phức tạp để có thể hiểu được.
  + **Có thể thẩm định** một mô hình bằng các kiểm tra thống kê. Điều này làm cho ta có thể tin tưởng vào mô hình.
* Cây quyết định có thể xử lý tốt một lượng dữ liệu lớn trong thời gian ngắn. Có thể dùng máy tính cá nhân để phân tích các lượng dữ liệu lớn trong một thời gian đủ ngắn để cho phép các nhà chiến lược đưa ra quyết định dựa trên phân tích của cây quyết định.

1. **Ứng dụng của cây quyết định**
2. Định nghĩa

* Trong những năm gần đây, nhiều mô hình phân lớp dữ liệu đã được các nhà khoa học trong những lĩnh vực khác nhau đề xuất như mạng nơ-ron, mô hình thống kê tuyến tính/ bậc 2, cây quyết định, mô hình di truyền. Trong số những mô hình đó, cây quyết định với những ưu điểm của mình được đánh giá **là một công cụ mạnh, phổ biến** và đặc biệt thích hợp cho phân lớp dữ liệu. Có thể kể ra những ưu điểm của cây quyết định như: xây dựng tương đối nhanh, đơn giản, dễ hiểu. Hơn nữa các cây có thể dễ dàng được chuyển đổi sang các câu lệnh SQL để có thể được sử dụng để truy nhập dữ liệu một cách hiệu quả. Cuối cùng, việc phân lớp dựa trên cây quyết định đạt được sự tương tự và đôi khi là chính xác hơn so với các phương pháp phân lớp khác.
* Cây quyết định là biểu đồ phát triển có cấu trúc dạng cây, như mô tả trong hình vẽ sau:

Hình: ví dụ về cây quyết định.

* Trong cây quyết định:
  + Gốc: là node trên cùng của cây.
  + Node trong: biểu diễn một kiểm tra trên một thuộc tính đơn (hình chữ nhật).
  + Nhánh: biểu diễn các kết quả của kiểm tra trên node trong (mũi tên).
  + Node lá: biểu diễn lớp hay sự phân phối lớp (hình tròn).
* Để phân lớp mẫu dữ liệu chưa biết, giá trị các thuộc tính của mẫu được đưa vào kiểm tra trên cây quyết định. Mỗi mẫu tương ứng có một đường đi từ gốc đến lá và lá biểu diễn dự đoán giá trị phân lớp mẫu đó.

1. Các vấn đề khác liên quan đến sử dụng cây quyết định

* Các vấn đề đặc thù trong khi học hay phân lớp dữ liệu bằng cây quyết định gồm: **xác định độ sâu** để phát triển cây quyết định, xử lý với những **thuộc tính liên tục**, **chọn phép đo** lựa chọn thuộc tính thích hợp, sử dụng tập dữ liệu đào tạo với những **giá trị thuộc tính bị thiếu**, sử dụng các thuộc tính với những chi phí khác nhau, và cải thiện hiệu năng tính toán. Sau đây sẽ đề cập đến những vấn đề chính đã được giải quyết trong các thuật toán phân lớp dựa trên cây quyết định.
  1. *Tránh quá vừa dữ liệu*
* **Thế nào là “quá vừa’ dữ liệu?** Có thể hiểu đây là hiện tượng cây quyết định chứa một số đặc trưng riêng của tập dữ liệu đào tạo, nếu lấy chính tập dữ liệu đào tạo để test lại mô hình phân lớp thì độ chính xác sẽ rất cao, trong khi đối với những dữ liệu tương lai khác nếu sử dụng cây đó lại không đạt được độ chính xác như vậy.
* Quá vừa dữ liệu là một khó khăn đáng kể đối với phân lớp bằng cây quyết định và những phương pháp khác. Đặc biệt khi số lượng ví dụ trong tập dữ liệu đào tạo quá ít, hay có noise trong dữ liệu.
* **Có hai phương pháp tránh “quá vừa” dữ liệu trong cây quyết định:**
  + Dừng cây phát triển sớm hơn bình thường, trước khi tới điểm phân lớp hoàn hảo tập dữ liệu đào tạo. Với phương pháp này, một thách thức đặt ra là phải ước lượng chính xác thời điểm dừng phát triển cây.
  + Cho phép cây có thể “quá vừa” dữ liệu, sau đó sẽ cắt, tỉa cây.
* Mặc dù phương pháp thứ nhất có vẻ trực quan hơn, nhưng với phương pháp thứ hai thì cây quyết định được sinh ra được thực nghiệm chứng minh là thành công hơn trong thực tế. Hơn nữa việc cắt. tỉa cây quyết định còn giúp tổng quát hóa, cải thiện độ chính xác của mô hình phân lớp. Dù thực hiện phương pháp nào thì vấn đề mấu chốt ở đây là tiêu chuẩn nào được sử dụng để xác định kích thước hợp lý của cây cuối cùng.
  1. *Vấn đề với thuộc tính liên tục*
* Việc thao tác với thuộc tính liên tục trên cây quyết định hoàn toàn không đơn giản như với thuộc tính rời rạc.
* Thuộc tính rời rạc **có tập giá trị (domain) xác định từ trước** và là tập hợp các giá trị rời rạc.
  + Ví dụ loại ô tô là một thuộc tính rời rạc với tập giá trị là: {xe tải, xe khách, xe con, taxi}. Việc phân chia dữ liệu vào phép kiểm tra giá trị của thuộc tính rời rạc được chọn tại một ví dụ cụ thể có thuộc tập giá trị của thuộc tính đó hay không: value(A) X với X domain(A). Đây là phép kiểm tra logic đơn giản, không tốn nhiều tài nguyên tính toán.
* Trong khi đó, với thuộc tính liên tục (thuộc tính dạng số) thì **tập giá trị là không xác định trước**. Chính vì vậy, trong quá trình phát triển cây, cần sử dụng kiểm tra nhị phân: value(A)  , với  là hằng số ngưỡng (threshold) được lần lượt xác định dựa trên từng giá trị riêng biệt hay từng cặp giá trị liền nhau (theo thứ tự sắp xếp) của thuộc tính liên tục đang xem xét trong tập dữ liệu đào tạo. Điều đó có nghĩa là nếu thuộc tính liên tục A trong tập dữ liệu đào tạo có d giá trị thì cần thực hiện d-n lần kiểm tra value(A)i với i=1…d-n để tìm ra ngưỡng  tốt nhất tương ứng với thuộc tính đó. Việc xác định giá trị của  và tiêu chuẩn tìm  tốt nhất tùy vào chiến lược của từng thuật toán. Trong thuật toán C4.5, i được chọn là giá trị trung bình của hai giá trị liền kề nhau trong dãy giá trị đã sắp xếp.
* Ngoài ra còn có một số vấn đề liên quan đến phát sinh tập luật, xử lý với giá trị thiếu sẽ được em trình bày cụ thể trong phần thuật toán C4.5 bên dưới.

1. Đánh giá
   1. *Điểm mạnh*

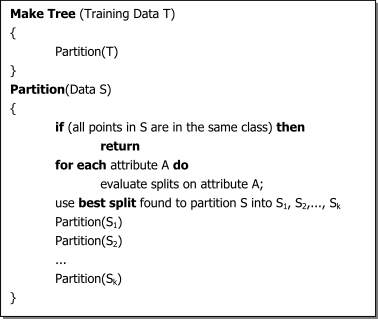
* Cây quyết định có 5 sức mạnh chính sau:
  + *Khả năng sinh ra các quy tắc hiểu được:*
* Cây quyết định có khả năng sinh ra các quy tắc có thể chuyển đổi được sang dạnh English, hoặc các câu lệnh SQL. Đây là ưu điểm nổi bật của kỹ thuật này. Thậm chí với những tập dữ liệu lớn khiến cho hình dáng cây quyết định lớn và phức tạp, việc di theo bất cứ đường nào trên cây là dễ dàng theo nghĩa phổ biến và rõ ràng. Do vậy sự giải thích cho bất cứ một sự phân lớp nào đều tương đối minh bạch.
  + *Khả năng thực thi những lĩnh vực hướng quy tắc:*
* Điều này nghe có vẻ hiển nhiên, nhưng **quy tắc quy nạp** nói chung và cây quyết định nói riêng là lựa chọn hoàn hảo cho những lĩnh vực thực sự là các quy tắc. Rất nhiều lĩnh vực từ di truyền tới các quá trình công nghiệp thực sự chứa các **quy tắc ẩn**, không rõ rang do khá phức tạp và tối nghĩa bởi những dữ liệu lỗi. Cây quyết định là một sự lựa chọn tự nhiên khi chúng ta nghi ngờ sự tồn tại của các quy tắc ẩn, không rõ ràng.
  + *Dễ dàng tính toán trong khi phân lớp:*
* Mặc dù như chúng ta đã biết, cây quyết định có thể chứa nhiều định dạng, nhưng trong thực tế, các thuật toán sử dụng để tạo ra cây quyết định thường tạo ra những cây với **số phân nhánh thấp và các test đơn giản tại từng node**. Những test điển hình là: so sánh số, xem xét phần tử của một tập hợp, và các phép nối đơn giản. Khi thực thi trên máy tính, những test này chuyển thành các toán hàm logic và số nguyên là những toán hạng thực thi nhanh va không đắt. Đây là một ưu điểm quan trọng bởi trong môi trường thương mại, các mô hình dự đoán thường được sử dụng để phân lớp hàng triệu thậm chí hàng tỉ bản ghi.
  + *Khả năng xử lý với cả thuộc tính liên tục và thuộc tính rời rạc.*
* Cây quyết định xử lý được như nhau với thuộc tính liên tục và thuộc tính rời rạc. Tuy rằng với thuộc tính liên tục cần nhiều tài nguyên tính toán hơn. Những thuộc tính rời rạc đã từng gây ra những vấn đề với mạng neural và các kỹ thuật thống kê lại thực sự dễ dàng thao tác với các tiêu chuẩn phân chia trên cây quyết định: mỗi nhánh tương ứng với từng phân tách tập dữ liệu theo giá trị của thuộc tính được chọn để phát triển tại node đó. Các thuộc tính liên tục cũng dễ dàng phân chia bằng việc chọn ra một số gọi là ngưỡng trong tập các giá trị đã sắp xếp của thuộc tính đó. Sauk hi chọn được ngưỡng tốt nhất, tập dữ liệu phân chia theo test nhị phân của ngưỡng đó.
  + *Thể hiện rõ ràng những thuộc tính tốt nhất:*
* Các thuật toán xây dựng cây quyết định đưa ra thuộc tính mà phân chia tốt nhất tập dữ liệu đào tạo bắt đầu từ node gốc của cây. Từ đó có thể thấy những thuộc tính nào là quan trọng nhất cho việc dự đoán hay phân lớp.
  1. *Điểm yếu*
* Dù có những sức mạnh nổi bật trên, cây quyết định vẫn không tránh khỏi có những điểm yếu. Đó là cây quyết định không thích hợp lắm với nhưng bài toán với mục tiêu là dự đoán giá trị của thuộc tính liên tục như thu nhập, huyết áp hay lãi xuất ngân hàng….Cây quyết định cũng khó giải quyết với những dữ liệu thời gian liên tục nếu không bỏ ra nhiều công sức cho việc cài đặt ra sự biểu diễn dữ liệu theo các mẫu liên tục.
  + *Dễ xảy ra lỗi khi có quá nhiều lớp.*
* Một só cây quyết định chỉ thao tác với những lớp giá trị nhị phân dạng yes/no hay accept/reject. Số khác lại có thể chỉ định các bản ghi vào một số lớp bất kỳ, nhưng dễ xảy ra lỗi khi số ví dụ đào tạo ứng với một lớp nhỏ. Điều này xảy ra càng nhanh hơn với cây mà có nhiều tầng hay có nhiều nhánh trên một node.
  + *Chi phí tính toán đắt để đào tạo.*
* Điều này nghe có vẻ mâu thuẫn với khẳng định ưu điểm của cây quyết định ở trên. Nhưng quá trình phát triển cây quyết định **đắt về mặt tính toán**. Vì cây quyết định có rất nhiều node trước khi đến lá cuối cùng. Tại node, cần tính một độ đo (hay tiêu chuẩn phân chia) trên từng thuộc tính, với thuộc tính liên tục phải them thao tác sắp xếp lại tập dữ liệu theo thứ tự giá trị của thuộc tính đó. Sau đó mới có thể chọn được một thuộc tính phát triển và tương ứng là một phân chia tốt nhất. Một vài thuật toán sử dụng tổ hợp các thuộc tính kết hợp với nhau có trọng số để phát triển cây quyết định. Quá trình cắt tỉa cây cũng “đắt” vì nhiều cây con ứng cử phải được tạo ra và so sánh.

1. Xây dựng cây quyết định

* Quá trình xây dựng cây quyết định gồm hai giai đoạn:
  + Giai đoạn thứ nhất **phát triển cây** quyết định:
* Giai đoạn này phát triển bắt đầu từ gốc, đến từng nhánh và phát triển quy nạp theo cách thức chia để trị cho tới khi đạt được cây quyết định với tất cả các lá được gán nhãn lớp.
  + Giai đoạn thứ hai **cắt, tỉa** bớt các nhánh trên cây quyết định:
* Giai đoạn này nhằm mục đích đơn giản hóa va khái quát hóa từ đó làm tăng độ chính xác của cây quyết định bằng cách loại bỏ sự phụ thuộc vào mức độ lỗi (noise) của dữ liệu đào tạo mang tính chất thống kê, hay những sự biến đổi mà có thể là đặc tính riêng biệt của dữ liệu đào tạo. Giai đoạn này chỉ truy cập giữ liệu trên cây quyết định đã được phát triển trong giai đoạn trước và quá trình thực nghiệm cho thấy giai đoạn này không tốn nhiều tài nguyên tính toán, như với phần lớn các thuật toán, giai đoạn này không tốn nhiều tài nguyên tính toán khoảng dưới 1% tổng thời gian xây dựng mô hình phân lớp.
* Do vậy, ở đây chúng ta chỉ tập trung vào nghiên cứu giai đoạn phát triển cây quyết định. Dưới đây là khung công việc của giai đoạn này:
  + Chọn thuộc tính tốt nhất bằng một độ đo đã định trước.
  + Phát triển cây bằng việc thêm các nhánh tương ứng với từng giá trị của thuộc tính đã chọn.
  + Sắp xếp, phân chia tập dữ liệu đào tạo tới node con.
  + Nếu các ví dụ được phân lớp rõ ràng thì dừng.
* Ngược lại: lặp lại bước 1 tới bước 4 cho từng node con.

1. **Thuật toán xây dựng cây quyết định**
2. Tổng quan về tư tưởng

* Phần lớn các thuật toán phân lớp dữ liệu dựa trên cây quyết định có mã giả như sau:



Hình: Mã giả của thuật toán phân lớp dữ liệu dựa trên cây quyết định

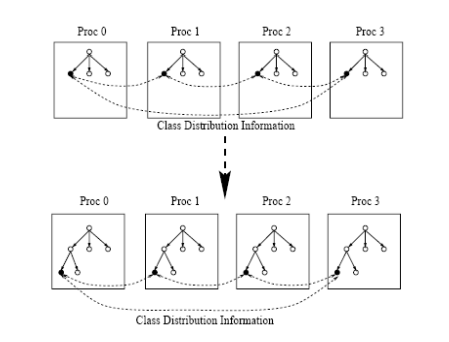
* Các thuật toán phân lớp như C4.5 (của quinlan,1993), CDP (của Agrawal và các tác giả khác, 1993), SLIQ ( của Mehta và các tác giả khác, 1996) và SPRINT ( của Shafer và các tác giả khác, 1996) đều sử dụng phương pháp **Hunt** làm tư tưởng chủ đạo.
* Phương pháp này được Hunt và các đồng sự nghĩ ra vào những năm cuối thập kỷ 50 đầu thập kỷ 60.
* **Mô tả quy nạp phương pháp hunt:**
  + Giả sử xây dựng cây quyết định từ T là **tập training data** và các lớp được biểu diễn dưới dạng tập C= {C1, C2, …, Ck}
  + Trường hợp 1: T chứa các case thuộc về một lớp đơn Cj, cây quyết định ứng với T là một lá tương ứng với lớp Cj.
  + Trường hợp 2 : T chứa các case thuộc về nhiều lớp khác nhau trong tập C. Một kiểm tra được chọn trên một thuộc tính có nhiều giá trị {O1.O2…,On Trong nhiều ứng dụng n thường được chọn là 2 , khi đó tạo ra cây quyết định nhị phân. Tập T được chia thành các tập con T1, T2, …, Tn với Ti chứa tất cả các case trong T mà có kết quả là Oi trong kiểm tra đã chọn. Cây quyết định ứng với T bao gồm một node biểu diễn kiểm tra được chọn, và mỗi nhánh tương ứng với mỗi kết quả kiểm tra đó. Cách thức xây dựng cây tương tự được áp dụng đệ quy cho từng tập con của tập training data.
  + Trường hợp 3 : T không chứa case nào. Cây quyết định ứng với T là một lá, nhưng lớp gắn với lá đó phải được xác định từ những thông tin khác ngoài T. Ví dụ C4.5 chọn giá trị phân lớp là lớp phổ biến nhất tại cha của node này.

1. Quy nạp tuyến tính

* Giải thuật cơ bản (giải thuật tham lam) được chia thành các bước như sau:
  + Cây được xây dựng đệ qui từ trên xuống dưới (top-down) và theo cách thức chia để trị (divide-conquer).
  + Ở thời điểm bắt đầu, tất cả những ví dụ huấn luyện ở gốc.
  + Thuộc tính được phân loại (nếu là giá trị liên tục chúng được rời rạc hóa)
  + Những **ví dụ huấn luyện** được phân chia đệ qui dựa trên thuộc tính mà nó chọn lựa.
  + Kiểm tra những thuộc tính được chọn dựa trên nền tảng của heristic hoặc của một định lượng thống kê.
* **Điều kiện để dừng** việc phân chia:
  + Tất cả những mẫu huấn luyện đối với một node cho trước thuộc về cùng một lớp.
  + Không còn thuộc tính còn lại nào để phân chia tiếp.
  + Không còn mẫu nào còn lại.

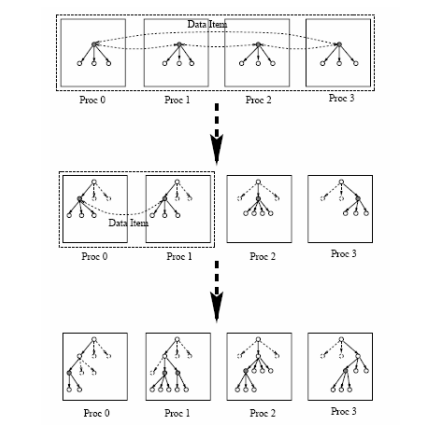
1. Cây quyết định tuần tự và thuật toán hóa

* Song song hóa xu hướng nghiên cứu hiện nay của các phân lớp dữ liệu dựa trên cây quyết định. Nhu cầu song song hóa các thuật toán tuần tự là một nhu cầu tất yếu của thực tiễn phát triển khi mà các đòi hỏi về hiệu năng, độ chính xác ngày càng cao. Thêm vào đó là sự gia tăng nhanh chóng về kích thước của dữ liệu cần phân lớp. Một mô hình phân lớp chạy trên hệ thống tính toán song song có hiệu năng cao, có khả năng phân lớp được những tập dữ liệu lớn hơn từ đó gia tăng độ tin cậy của các quy tắc phân lớp.
* **Phương pháp xây dựng cây đồng bộ:**
  + Trong phương pháp này, tất cả các bộ vi xử lý đồng thời tham gia xây dựng cây quyết định bằng việc gửi và nhận các thông tin phân lớp của dữ liệu địa phương.
  + **Ưu điểm** của phương pháp này là **không yêu cầu việc di chuyển các dữ liệu** trong tập dữ liệu đào tạo. Tuy nhiên, thuật toán này phải chấp nhận chi phí giao tiếp cao và tải bất cân bằng. Với từng node trong cây quyết định, sau khi tập hợp được các thông tin phân lớp, tất cả các bộ vi xử lý cần phải đồng bộ và cập nhật các thông tin phân lớp. Với những node ở độ sâu thấp, chi phí giao tiếp tương đối nhỏ vì số lượng các mục training data được xử lý là tương đối nhỏ. Nhưng khi cây càng sâu thì chi phí cho giao tiếp chiếm phần lớn thời gian xử lý. Một vấn đề nữa của phương pháp này là tải bất cân bằng do cơ chế lưu trữ và phân chia dữ liệu ban đầu tới từng bộ vi xử lý.



Hình: sơ đồ xây dựng cây quyết định theo phương pháp đồng bộ.

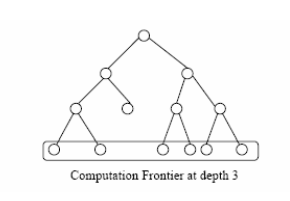
* **Phương pháp xây dựng cây phân hoạch:**
  + Khi xây dựng cây quyết định bằng phương pháp phân hoạch các bộ vi xử lý khác nhau làm việc với các phần khác nhau của cây quyết định. Nếu nhiều hơn 1 bộ vi xử lý cùng kết hợp để phát triển 1 node, thì các bộ vi xử lý được phân hoạch để phát triển các con của node đó. Phương pháp này tập chung vào trường hợp 1 nhóm các bộ vi xử lý Pn cùng hợp tác để phát triển node n. Khi bắt đầu, tất cả các bộ vi xử lý cùng đồng thời kết hợp để phát triển node gốc của cây phân lớp. Khi kết thúc, toàn bộ cây phân lớp được tạo ra bằng cách kết hợp tất cả các cây con của từng bộ vi xử lý.
  + Ưu điểm của phương pháp này là khi một bộ vi xử lý một mình chịu trách nhiệm phát triển một node, thì nó có thể phát triển thành một cây con toàn cục một cách độc lập mà **không cần bất cứ chi phí giao tiếp nào**.
  + Tuy nhiên cũng có một vài nhược điểm trong phương pháp này, đó là :
* Thứ nhất: yêu cầu di chuyển dữ liệu sau mỗi lần phát triển một node cho tới khi mỗi bộ vi xử lý chứa toàn bộ dữ liệu có thể phát triển toàn bộ một cây con. Do vậy dẫn đến tốn kém chi phí giao tiếp khi ở phần trên của cây phân lớp.
* Thứ hai: khó đạt được tải cân bằng. Việc gán các node cho các bộ vi xử lý được thực hiện dựa trên số lượng các case trong các node con. Tuy nhiên số lượng các case gắn với một node không nhất thiết phải tương ứng với số lượng công việc cần phải xử lý để phát triển cây con tại node đó.



Hình: sơ đồ xây dựng cây quyết định theo phương pháp phân hoạch.

* **Phương pháp lai:** 
  + Phương pháp lai có **tận dụng ưu điểm của 2 phương pháp trên**. Phương pháp xây dựng cây đồng bộ chấp nhận chi phí giao tiếp cao khi biên giới của cây càng rộng. Trong khi đó, phương pháp xây dựng cây quyết định phân hoạch thì phải chấp nhận chi phí cho việc tải cân bằng sau mỗi bước. Trên cơ sở đó, Phương pháp lai tiếp tục duy trì cách thức thứ nhất miễn là chi phí giao tiếp phải chịu khi tuân theo cách thứ nhất không quá lớn. Khi mà chi phí này vượt quá một ngưỡng quy định, thì các bộ vi xử lý đang xử lý các node tại đường biên hiện tại của cây phân lớp được chia thành 2 phần (với giả thiết số lượng các bộ vi xử lý là lũy thừa của 2).
  + Phương pháp này cần sử dụng tiêu chuẩn để khởi tạo sự phân hoạch tập các bộ vi xử lý hiện tại, đó là:

 *chi phí di chuyển + Tải cân bằng*



Hình: Sơ đồ xây dựng cây quyết định theo phương pháp lai.

1. Cách đo độ lợi thông tin (Information Gain)

* Information gain là đại lượng được sử dụng để chọn lựa thuộc tính với **information gain lớn nhất**. Giả sử có hai lớp P và N. Cho tập hợp của những ví dụ S chứa p phần tử của lớp P và n phần tử của lớp N. Khối lượng của thông tin, cần để quyết định nếu những mẫu tùy ý trong S thuộc về P hoặc N được định nghĩa như là :

I(p,n) = -[p/(p+n)]log 2 [p/(p+n)] – [n/(p+n)]log 2 [n/(p+n)]

* Giả sử rằng sử dụng thuộc tính A một tập hợp S được phân hoạch thành những tập hợp {S1,S2,..,Sv} . Nếu Si chứa những mẫu của P và ni mẫu của Ni entropy hoặc thông tin mong đợi cần để phân loại những đối tượng trong cây con Si là :

v

E(A) = Σ [(pi+ni)/(p+n)] I(pi,ni)

i=1

* Thông tin nhận được ở nhánh A là: Gain(A) = I(p, n) - E(A)

1. THUẬT TOÁN C4.5

* Với những đặc điểm C4.5 là thuật toán phân lớp dữ liệu dựa trên cây quyết định hiệu quả và phổ biến trong những ứng dụng khai phá cơ sở dữ liệu **có kích thước nhỏ**. C4.5 sử dụng **cơ chế lưu trữ dữ liệu thường trú trong bộ nhớ**, chính đặc điểm này làm C4.5 chỉ thích hợp với những cơ sở dữ liệu nhỏ, và cơ chế sắp xếp lại dữ liệu tại mỗi node trong quá trình phát triển cây quyết định.
* C4.5 còn chứa một kỹ thuật cho phép biểu diễn lại cây quyết định dưới dạng một danh sách sắp xếp thứ tự các luật **if-then** (một dạng quy tắc phân lớp dễ hiểu). Kỹ thuật này cho phép làm giảm bớt kích thước tập luật và đơn giản hóa các luật mà độ chính xác so với nhánh tương ứng cây quyết định là tương đương.
* Tư tưởng phát triển cây quyết định của C4.5 là phương pháp Hunt đã nghiên cứu ở trên. Chiến lược **phát triển theo độ sâu** (depth-first strategy) được áp dụng cho C4.5.
* Mã giả của thuật toán C4.5:

(1) ComputerClassFrequency(T);

(2) if OneClass or FewCases

return a leaf;

Create a decision node N;

(3) ForEach Attribute A

ComputeGain(A);

(4) N.test=AttributeWithBestGain;

(5) if (N.test is continuous)

find Threshold;

(6) ForEach T' in the splitting of T

(7) If ( T' is Empty )

Child of N is a leaf

else

(8) Child of N=FormTree(T');

(9) ComputeErrors of N;

return N

Hình: Mã giải thuật toán C4.5.

* C4.5 có những đặc điểm khác với các thuật toán khác, đó là cơ chế chọn thuộc tính để kiểm tra tại mỗi node, cơ chế xử lý với những giá trị thiếu, việc tránh “quá vừa” dữ liệu, ước lượng độ chính xác và cơ chế cắt tỉa cây.

1. **Gain – entropy độ đo lựa chọn thuộc tính tốt nhất**

* Phần lớn các hệ thống đều cố gắng để tạo ra một cây càng nhỏ càng tốt, vì những **cây nhỏ hơn thì dễ hiểu hơn** và dễ đạt được độ chính xác dự đoán hơn. Do không thể đảm bảo được sự cực tiểu của cây quyết định, C4.5 dựa vào nghiên cứu tối ưu hóa, và sự lựa chọn cách phân chia mà có độ đo lựa chọn thuộc tính đạt giá trị cực đại.
* Hai độ đo được sử dụng trong C4.5 là **information gain** và **gain ratio**.
* RF(Cj, S) biểu diễn tần xuất (Relative Frequency) các case trong S thuộc về lớp Cj.

RF(Cj, S) = 

* Với  là kích thước tập các case có giá trị phân lớp là Cj.  là kích thước tập dữ liệu đào tạo.
* Chỉ số thông tin cần thiết cho sự phân lớp: I(S) với S là tập cần xét sự phân phối lớp được tính bằng:

I(S) = -

* Sau khi S được phân chia thành các tập con S1, S2, …, Sn bởi test B thì **information gain** được tính bằng:

G(S, B) = I(S) - 

* Test B sẽ được chọn nếu có G(S, B) đạt giá trị lớn nhất.
* Tuy nhiên có một vấn đề khi sử dụng G(S, B) ưu tiên test có số lượng lớn kết quả, ví dụ G(S,B) đạt cực đại với test mà từng Si chỉ chứa một case đơn. Tiêu chuẩn gain ratio giải quyết được vấn đề này bằng việc đưa vào **thông tin tiềm năng (potential information)** của bản than mỗi phân hoạch.

P(S,B) = - 

* Test B sẽ được chọn nếu có tỉ số giá trị gain ratio =  lớn nhất.
* Trong mô hình phân lớp C4.5, có thể dùng một trong hai loại chỉ số Information Gain hay Gain ratio để xác định thuộc tính tốt nhất. Trong đó Gain ratio là lựa chọn mặc định.
* **Ví dụ mô tả cách tính information gain:**
  + - **Với thuộc tính rời rạc:**

Bảng 1: bảng dữ liệu training với thuộc tính phân lớp là buys\_computer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rid | Age | Income | Student | Credit\_rating | Class : buys\_computer |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <30  <30  30-40  >40  >40  >40  30-40  <30  <30  >40  <30  30-40  30-40  >40 | High  High  High  Medium  Low  Low  Low  Medium  Low  Medium  Medium  Medium  High  Medium | No  No  No  No  Yes  Yes  Yes  No  Yes  Yes  Yes  No  Yes  No | Fair  Excellent  Fair  Fair  Fair  Excellent  Excellent  Fair  Fair  Fair  Excellent  Excellent  Fair  Excellent | No  No  Yes  Yes  Yes  No  Yes  No  Yes  Yes  Yes  Yes  Yes  No |

* + Trong tập dữ liệu trên: S1 là tập những bản ghi có giá trị phân lớp là yes, S2 là tập những bản ghi có giá trị phân lớp là no. Khi đó:
  + I(S) = I(S1,S2) = I(9,5) = -9/14\*log2 9/14 – 5/14\*log2 5/14 = 0.940
  + Tính G(S, A) với A lần lượt là từng thuộc tính:
  + **A = age**. Thuộc tính age đã được rời rạc hóa thành các giá trị <30, 30-40, và >40.

+ S={S1,S2,S3}

+ S1 (age<30), S2(30-40), S3 (>40).

+ I (S1): s11(yes &<30) =2, s21(no&<30) =3

+ I (S2): s12(yes & 30-40) = 4, s22(no & 30-40 ) = 0

+ I (S3): s13(yes & >40) = 2, s23(no & >40 ) = 3

+ I(S1) = (S11,S21) = **-** 2/5log2 2/5 – 3/5log2 3/5 = 0.971

+ I(S2) = (S12,S22) = -4/4\*log2 4/4 – 0/4\*log2 0/4 = 0

+ I(S3) = (S13,S23) = -2/5\*log2 2/5 – 3/5\*log2 3/5 = 0.971

= 5/14\*I(S1) + 4/14\*I(S2) + 5/14\*I(S3) = 0.694

+ Gain (S,A) = I(S1.S2) - = 0.940 – 0.694 = 0.246

+ P(S,A) = -5/14\*log2(5/14) – 4/14\*log2(4/14) – 5/14\*log2(5/14) = 1.577

+ Gain Ratio(S,A) = 0.246/1.577 = 0.156

* + Thuộc tính income :

+ B = income => S{S1,S2,S3}

+ S1 (B=high), S2(B=medium), S3 (B=low).

+ I(S1) : s11(yes & high) = 2 , s21(no & high) = 2

+ I(S2) : s12(yes & medium) = 4, s22(no & medium) = 2

+ I(S3) : s13(yes & low ) = 3, s23(no & low) = 1

+ I(S1) = I(s11,s21) = -2/4\*log2 2/4 – 2/4\*log2 2/4 = 1

+ I(S2) = I(s12,s22) = -4/6\*log2 4/6 – 2/6\*log2 2/6 = 0.918

+ I(S3) = I (s13,s23) = -3/4\*log2 3/4 – 1/4\*log2 1/4 = 0.811

 = 4/14\*1 + 6/14\*0.918 + 4/14\*0.811 = 0.910

+ Gain ( S,B) = I(S1.S2) - = 0.940 – 0.910 = 0.029

+ P(S,B) = -4/14\*log2(4/14) – 6/14\*log2(6/14) – 4/14\*log2(4/14) = 1.557

+ Gain Ratio(S,B) = 0.029/1.557 = 0.019

* + Thuộc tính student :

+ C = student => S{S1,S2}

+ S1 (C=yes), S2(C=no)

+ I(S1) : s11(yes & yes) = 6 , s21(no & yes) = 1

+ I(S2) : s12(yes & no ) = 3, s22(no & no) = 4

+ I(S1) = I(s11,s21) = -6/7\*log2 6/9 – 1/7\*log2 3/9 = 0.592

+ I(S2) = I(s12,s22) = -3/7\*log2 1/5 – 4/7\*log2 4/5 = 0.985

= 7/14\*0.592 + 7/14\*0.985 = 0.789

+ Gain ( S,C) = I(S1.S2) - = 0.940 – 0.789 = 0.151

+ P(S,C) = -7/14\*log2(7/14) – 7/14\*log2(7/14) = 1

+ Gain Ratio(S,C) = 0.151/1 = 0.151

* + Thuộc tính credit\_rating :

+ D = credit rating => S{S1,S2}

+ S1 (D=fair), S2(D=excellent)

+ I(S1) : s11(yes & fair) = 6 , s21(no & fair) = 2

+ I(S2) : s12(yes & ex ) = 3 , s22(no & ex) = 3

+ I(S1) = I(s11,s21) = -6/8\*log2 6/8 – 2/8\*log2 2/8 = 0.811

+ I(S2) = I(s12,s22) = -3/6\*log2 3/6 – 3/6\*log2 3/6 = 1

= 8/14\*0.811 + 6/14\*1 = 0.892

+ Gain ( S,D) = I(S1.S2) - = 0.940 – 0.892 = 0.048

+ P(S,D) = -8/14\*log2(8/14) – 6/14\*log2(6/14) = 0.985

+ Gain Ratio(S,D) = 0.048/0.985 = 0.049

* + Vậy :

+ A = age : Gain (S,A) = 0.246 , Gain Ratio(S,A) = 0.156

+ B = income: Gain (S, B) **= 0.029 , Gain Ratio(S,B) = 0.019**

+ C = student: Gain (S, C) = 0.151, Gain Ratio(S,C) = 0.151

+ D = credit\_rating: Gain (S, D) = **0.048 , Gain Ratio(S,D) = 0.049**

+ A= age có Gain lớn nhất

=> Thuộc tính age là thuộc tính có độ đo Information Gain lớn nhất.Do vậy age được chọn làm thuộc tính phát triển tại node đang xét.

* + Lưu ý :

+ Tiêu chí Information Gain thường "ưu tiên" chọn những thuộc tính có nhiều giá trị (miền xác định lớn)

+ Spliting entropy, SED(Fi) sẽ lớn khi thuộc tính Fi có nhiều giá trị. Điều này giúp:

Làm giảm Gain Ratio của thuộc tính có nhiều giá trị.

Làm tăng Gain Ratio của thuộc tính có ít giá trị.

+ Ý nghĩa khác: Giảm vấn đề "quá khớp"

* + Ta có bảng age < 30 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rid | Income | Student | Credit\_rating | class |
| 1 | High | No | Fair | No |
| 2 | High | No | Excellent | No |
| 8 | Medium | No | Fair | No |
| 9 | Low | Yes | Fair | Yes |
| 11 | Medium | Yes | Excellent | yes |

* + Ta có bảng age 30-40 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rid | Income | Student | Credit\_rating | class |
| 3 | High | No | Fair | Yes |
| 7 | Low | yes | Excellent | Yes |
| 12 | Medium | No | Excellent | Yes |
| 13 | High | Yes | fair | yes |

* + Ta có bảng age > 40 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rid | Income | Student | Credit\_rating | class |
| 4 | medium | No | Fair | Yes |
| 5 | Low | Yes | Fair | Yes |
| 6 | Low | No | Excellent | No |
| 10 | Medium | Yes | Fair | Yes |
| 14 | Medium | No | Excellent | No |

* + Ta được cây cuối cùng:

Age

Cr- rating

Yes

Student

No

Yes

No

Yes

Hình : cây quyết định cuối cùng

* + R1 : if (age = 30-40) then buy\_computer = yes.
  + R2 : if (age > 40 ) ^ ( cr-rating = fair ) then buy\_computer = yes.
  + R3 : if (age > 40 ) ^ ( cr-rating = excellent ) then buy\_computer = no.
  + R4 : if (age < 30 ) ^ ( income = high ) then buy\_computer = no.
  + R5 : if (age < 30 ) ^ (student = yes) then buy\_computer = yes.
  + R6 : if (age < 30 ) ^ (student = no) then buy\_computer = no.
    - **Với thuộc tính liên tục:**
  + Xử lý thuộc tính liên tục đòi hỏi nhiều tài nguyên tính toán hơn thuộc tính rời rạc. Gồm các bước sau:
  + Kỹ thuật Quick sort được sử dụng để sắp xếp các case trong tập dữ liệu đào tạo theo thứ tự tăng dần hoặc gảm dần các giá trị của thuộc tính liên tục V đang xét. Được tập giá trị V= {v1 , v2,…,vm}
  + Chia tập dữ liệu thành hai tập con theo ngưỡng i= (vi + vi+1)/2 nằm giữa hai giá trị liền kề nhau vi và vi+1 . Test để phân chia dữ liệu là test nhị phân dạng V I . Thực thi test đó ta được hai tập dữ liệu con: V1 = {v1,v2,…,vi} và V2 = {vi+1, vi+2,…,vm).
  + Xét (m-1) ngưỡng i có thể có ứng với m giá trị của thuộc tính V bằng cách tính Information gain hay Gain ratio với từng ngưỡng đó. Ngưỡng có giá trị của Information gain hay Gain ratio lớn nhất sẽ được chọn làm ngưỡng phân chia của thuộc tính đó.
  + Việc tìm ngưỡng (theo cách tuyến tính như trên) và sắp xếp tập training theo thuộc tính liên tục đang xem xét đôi khi gây ra thắt cổ chai vì tốn nhiều tài nguyên tính toán.

1. **Xử lí giá trị bị thiếu theo cơ chế riêng**

* Giá trị thiếu của thuộc tính là hiên tượng phổ biến trong dữ liệu, có thể do lỗi khi nhập các bản ghi vào cơ sở dữ liệu, cũng có thể do giá trị thuộc tính đó được đánh giá là không cần thiết đối với case cụ thể.
* Trong quá trình xây dựng cây từ tập dữ liệu đào tạo S, B là test dựa trên thuộc tính A với các giá trị đầu ra là b1, b2, …, bn. Tập S0 là tập con các case trong S mà có giá trị thuộc tính A không biết và Si biểu diễn các case với đầu ra là b(i) trong test B.
* Khi đó độ đo information gain của test B giảm vì chúng ta không phân được lớp nào từ các case trong S0.

G(S, B) = 

* Tương ứng với G(S, B), P(S,B) cũng thay đổi:

P(S, B) = - 

* Hai thay đổi này làm giảm giá trị của test liên quan đến thuộc tính có tỉ lệ giá trị thiếu cao.
* Nếu test B được chọn, C4.5 không tạo một nhánh riêng trên cây quyết định cho S0. Thay vào đó, thuật toán có cơ chế phân chia các case trong So về các tập con Si là tập con mà có giá trị thuộc tính test xác định theo trọng số .

1. **C4.5 tránh quá vừa dữ liệu**

* Quá vừa dữ liệu là một khó khăn đáng kể đối với phân lớp bằng cây quyết định và những phương pháp khác. Quá vừa dữ liệu là hiện tượng: nếu không có các case xung đột (là những case mà giá trị cho mọi thuộc tính là giống nhau nhưng giá trị của lớp lại khác nhau) thì cây quyết định sẽ phân lớp chính xác toàn bộ các case trong tập dữ liệu đào tạo. Đội khi dữ liệu đào tạo lại chứa những đặc tính cụ thể, nên khi áp dụng cây quyết định đó cho những tập dữ liệu khác thì độ chính xác không còn cao như trước.
* Có một số phương pháp tránh “quá vừa” dữ liệu trong cây quyết định:
  + - * Dừng phát triển cây sóm hơn bình thường trước khi đạt đến điểm phân lớp hoàn hảo tập dữ liệu đào tạo. Với phương pháp này phải ước lượng chính xác thời điểm dừng phát triển cây.
      * Cho phép cây có thể “quá vừa” dữ liệu sau đó sẽ cắt, tỉa cây
* Mặc dù phương pháp thứ nhất có vẻ trực quan hơn, nhưng với phương pháp thứ hai thì cây quyết định được sinh ra được thực nghiệm chứng minh là thành công hơn trong thực tế, vì nó cho phép các tương tác tiềm năng giữa các thuộc tính được khám phá trước khi quyết định xem kết quả nào đáng giữ lại. C4.5 sử dụng kỹ thuật thứ hai để tránh “quá vừa” dữ liệu.

1. **Phương pháp chuyển đổi từ cây sang luật**

* Việc chuyển đổi từ cây quyết định sang luật sản xuất dạng **if-then** tạo ra những quy tắc phân lớp dẽ hiểu, dễ áp dụng. Các mô hình phân lớp biểu diễn các khái niệm dưới dạng các luật sản xuất đã được chứng minh là hữu ích trong nhiều lĩnh vực khác nhau, với các đòi hỏi cả về độ chính xác và tính hiểu được của mô hình phân lớp. Dạng output tập luật sản xuất là sự lựa chọn khôn ngoan. Tuy nhiên, tài nguyên tính toán dùng cho việc tạo ta tập luật từ dữ liệu đào tạo có kích thước lớn và nhiều giá trị sai là vô cùng lớn.
* Giai đoạn chuyển đổi từ cây quyết định sang luật gồm 4 bước:
  + - * **Cắt tỉa:**
  + Luật khởi tạo ban đầu là đường đi từ gốc đến lá của cây quyết định. Một quyết định có L lá thì tương ứng tập luật sản xuất sẽ có L luật khởi tạo. Từng điều kiện trong luật được xem xét và loại bỏ nếu không ảnh hưởng tới độ chính xác của luật đó. Sau đó, các luật đã được cắt tỉa được thêm vào tập luật trùng với những luật đã có.
    - * **Lựa chọn:**
  + Các luật đã cắt tỉa được nhóm lại theo giá trị phân lớp, tạo nên các tập con chứa các luật theo lớp. Sẽ có k tập luật con nếu tập training data có k giá trị phân lớp. Từng tập con trên được xem xét để chọn ra một tập con các luật mà tối ưu hóa độ chính xác dự đoán của lớp gắn với tập luật đó.
    - * **Sắp xếp:**
  + Sắp xếp K tập luật đã tạo ra từ trên bước theo tần số lỗi. Lớp mặc định được tạo ra bằng cách xác định các case trong tập training data không chứa tổng các luật hiện tại và chọn lớp phổ biến nhất trong các case đó làm lớp mặc định.
    - * **Ước lượng, đánh giá:**
  + Tập luật được đem ước lượng lại trên toàn bộ tập training nhằm mục đính xác định xem có luật nào làm giảm độ chính xác của sự phân lớp. Nếu có, luật đó bị loại bỏ và quá trình ước lượng được lặp cho đến khi không thể cải tiến thêm.

1. **C4.5 là thuật toán hiệu quả cho những tập dữ liệu vừa và nhỏ**

* C4.5 có cơ chế sinh cây quyết định hiệu quả và chặt chẽ bằng việc sử dụng độ đo lựa chọn thuộc tính tốt nhất là **information gain**. Các cơ chế xử lý với giá trị lỗi, thiếu và chống “quá vừa” dữ liệu của C4.5 cùng với cơ chế cắt tỉa cây đã tạo nên sức mạnh của C4.5. Thêm vào đó, mô hình phân lớp C4.5 còn có phần chuyển đổi từ cây quyết định sang luật **if- then,** làm tăng độ chính xác và tính dễ hiểu của kết quả phân lớp. Đây là tiện ích rất có ý nghĩa đối với người sử dụng.

1. CHƯƠNG TRÌNH CODE THUẬT TOÁN C4.5
2. **Giới thiệu tổng quát**
   * Chức năng chính: Dựa trên các dữ liệu mẫu sẵn có trong cơ sở dữ liệu, gồm có 5 mẫu .cvs, em chia ra làm dữ liệu làm 2 phần. Phần 1 training data (80%) tức là dùng để áp dụng thuật toán C4.5 để xây dựng nên các tập luật. Từ tập luật đó đối chiếu với các dữ liệu người dùng nhập vào trong testdata (20%) còn lại để đưa ra ra số dự đoán đúng dựa trên các tập luật đã xử lí và khởi tạo cây.
3. **Mã chương trình**
   1. Tính entropy

public TinhC45(IEnumerable<IList<string>> data, Cot decisionColumn)

{

Decisioncot = decisionColumn;

decisionRates = new Dictionary<string, double>();

columnRates = new Dictionary<string, double>();

clRatesVals = new Dictionary<string, IDictionary<string, double>>();

Dulieu = data;

} // hàm khởi tạo

public double GetEntropy()

{

double result = 0d;

foreach (string val in Decisioncot.giatri)

{

double p = decisionRates[val];

if (p != 0 && p != 1)

result += -p \* Math.Log2(p);

}

return result;

} // tính entropy decision

public double GetEntropy(Cot column, string value)

{

Cl = column;

double result = 0d;

foreach (string val in Decisioncot.giatri)

{

double p = clRatesVals[value][val];

if (p != 0 && p != 1)

result += -p \* Math.Log2(p);

}

return result;

} // tính entropy của wind= weak thuộc tính rời rạc

public double GetEntropy(Cot column, string thresholdValue, bool isGreaterThan)//continuous attribute

{

setColumnAndThreshold(column, thresholdValue);

double result = 0d;

foreach (string val in Decisioncot.giatri)

{

double p = clRatesVals[isGreaterThan.ToString()][val];

if (p != 0 && p != 1)

result += -p \* Math.Log2(p);

}

return result;

} // tính entropy lớn hơn, nhỏ, bằng thuộc tính liên tục

* 1. Tính Info Gain

public double GetInfoGain(Cot column)

{

Cl = column;

double result = GetEntropy();

foreach (string val in column.giatri)

{

double p = columnRates[val];

result += -p \* GetEntropy(column, val);

}

return result;

} // tính thuộc tính rời rạc

public double GetInfoGain(Cot column, string thresholdValue)//continuous attribute

{

setColumnAndThreshold(column, thresholdValue);

double result = GetEntropy();

double gt\_p = columnRates[true.ToString()];

double loet\_p = columnRates[false.ToString()];

result += -gt\_p \* GetEntropy(column, thresholdValue, true);

result += -loet\_p \* GetEntropy(column, thresholdValue, false);

return result;

} // tính thuộc tính liên tục

* 1. Tính Gain Ratio

public double GetSplitInfo(Cot column)

{

Cl = column;

double result = 0d;

foreach (string val in column.giatri)

{

double p = columnRates[val];

if (p != 0 && p != 1)

result += -p \* Math.Log2(p);

}

return result;

}

public double GetSplitInfo(Cot column, string thresholdValue)//continuous attribute

{

setColumnAndThreshold(column, thresholdValue);

double result = 0d;

double gt\_p = columnRates[true.ToString()];

double loet\_p = columnRates[false.ToString()];

if (gt\_p != 0 && gt\_p != 1)

result += -gt\_p \* Math.Log2(gt\_p);

if (loet\_p != 0 && loet\_p != 1)

result += -loet\_p \* Math.Log2(loet\_p);

return result;

}

public double GetGainRatio(Cot column)

{

return GetInfoGain(column) / GetSplitInfo(column);

}

public double GetGainRatio(Cot column, string thresholdValue)//continuous attribute

{

return GetInfoGain(column, thresholdValue) / GetSplitInfo(column, thresholdValue);

}

* 1. Tạo Node

public TaoNode(IList<IList<string>> data, IList<Cot> cls) : this(data, cls, new Node())

{ } // hàm khởi tạo

public TaoNode(IList<IList<string>> data, IList<Cot> cls, Node node)

{

this.dulieu = data;

this.Cls = cls;

this.node = node;

thuattoan = new TinhC45(data, cls.Last());

} // hàm khởi tạo

private ColumnWrapper getBranchingColumn()

{

if (Cls.Last().giatri.Count < 2)

return new ColumnWrapper(null);

double gainRatio = double.MinValue;

double? threshold = null;

Cot result = null;

IList<Cot> newCols = new List<Cot>(Cls);

for (int j = 0; j < Cls.Count - 1; j++)

{

//remove columns have 1 value

Cot col = Cls[j];

if (col.giatri.Count < 2)

{

newCols.Remove(col);

}

//calculate gain ratio

else if (col.Isnominal)

{

double gR = thuattoan.GetGainRatio(col);

if (gR > gainRatio)

{

gainRatio = gR;

threshold = null;

result = col;

}

}

else

{

col.giatri = col.giatri.OrderBy(v => v).ToList();

for (int i = 0; i < col.giatri.Count - 1; i++)

{

double th = double.Parse(col.giatri[i]);

double gR = thuattoan.GetGainRatio(col, col.giatri[i]);

if (gR > gainRatio)

{

gainRatio = gR;

threshold = th;

result = col;

}

}

}

}

Cls = newCols;

return new ColumnWrapper(result, threshold);

} // tính gainratio để dùng cột quyết định rẽ nhánh

public Node GetNode()

{

if (node == null)

node = new Node();

ColumnWrapper cw = getBranchingColumn(); // rẽ nhánh ở cột nào

node.cl = cw.Column;

node.Threshold = cw.Threshold;

if (node.cl != null)

{

if (node.cl.Isnominal)

{

IDictionary<string, int> count = new Dictionary<string, int>();

Cot decisionColumn = Cls.Last();

foreach (string val in decisionColumn.giatri)

{

count.Add(val, 0);

}

foreach (IList<string> row in dulieu)

{ count[row[decisionColumn.Number]]++;

}

string mostAppearResult = count.First().Key;

foreach (string val in decisionColumn.giatri)

{

if (count[val] > count[mostAppearResult])

mostAppearResult = val;

}

node.dudoan = row => node.lopcon.ContainsKey(row[node.cl.Number]) ? // gán hàm quyết định cho node

node.lopcon[row[node.cl.Number]].anticipate(row) :

mostAppearResult;

}

else

{

node.dudoan = row => double.Parse(row[node.cl.Number]) > node.Threshold ? // hàm quyết định cho node

node.lopcon[true.ToString()].anticipate(row) :

node.lopcon[false.ToString()].anticipate(row);

}

}

else // không còn cột để rẽ nhánh

{

IDictionary<string, int> count = new Dictionary<string, int>();

Cot decisionColumn = Cls.Last();

foreach (string val in decisionColumn.giatri)

{

count.Add(val, 0);

}

foreach (IList<string> row in dulieu)

{

count[row[decisionColumn.Number]]++;

}

string mostAppearResult = count.First().Key;

foreach (string val in decisionColumn.giatri)

{

if (count[val] > count[mostAppearResult])

mostAppearResult = val;

}

node.dudoan = row => mostAppearResult; // gán bằng giá trị xuất hiện nhiều nhất

}

return node;

} // trả về node

* 1. Lập cây

Node tree = new Node(); // tạo node gốc

stack.Push(new TaoNode(data, cls, tree)); // dùng ngăn xếp để duyệt sâu

while (stack.Any()) // còn giá trị thì xử lí

{

TaoNode nc = stack.Pop(); // lấy giá trị trên cùng

Node node = nc.GetNode(); // trả về node hoàn chỉnh

if (node.cl == null)

continue; // nếu node lá quay lại while

if (node.cl.Isnominal)

{

foreach (string value in node.cl.giatri) // xử lí tính lại column

{

Node child = new Node();

node.themlopcon(value, child); // thêm node con vào

var filteredDataFrom\_nc = nc.dulieu.Where(row => row[node.cl.Number] == value).ToList();

var copiedColumns = new List<Cot>(nc.Cls);

copiedColumns.Remove(node.cl);

var filteredColumnsFrom\_nc = new List<Cot>();

for (int i = 0; i < copiedColumns.Count; i++)

{

var col = copiedColumns[i];

var values = filteredDataFrom\_nc.GroupBy(row => row[col.Number]).Select(c => c.Key).ToList();

values.Remove("?");

if (values.Count > 1 || i == copiedColumns.Count - 1)

{

Cot newCol = col.Clone();

newCol.giatri = values; filteredColumnsFrom\_nc.Add(newCol);

}

} // quá trính xử lí colum

stack.Push(new TaoNode(filteredDataFrom\_nc, filteredColumnsFrom\_nc, child)); // đẩy node con vào

}

}

else

{

/\*greater than threshold\*/

Node child = new Node();

node.themlopcon(true.ToString(), child);

var filteredDataFrom\_nc = nc.dulieu.Where(row => double.Parse(row[node.cl.Number]) > node.Threshold).ToList();

var copiedColumns = new List<Cot>(nc.Cls);

copiedColumns.Remove(node.cl);

var filteredColumnsFrom\_nc = new List<Cot>();

for (int i = 0; i < copiedColumns.Count; i++)

{

var col = copiedColumns[i];

var values = filteredDataFrom\_nc.GroupBy(row => row[col.Number]).Select(c => c.Key).ToList();

values.Remove("?");

if (values.Count > 1 || i == copiedColumns.Count - 1)

{

Cot newCol = col.Clone();

newCol.giatri = values;

filteredColumnsFrom\_nc.Add(newCol);

}

}

stack.Push(new TaoNode(filteredDataFrom\_nc, filteredColumnsFrom\_nc, child));

/\*lesser or equal to threshold\*/

child = new Node();

node.themlopcon(false.ToString(), child);

filteredDataFrom\_nc = nc.dulieu.Where(row => double.Parse(row[node.cl.Number]) <= node.Threshold).ToList();

filteredColumnsFrom\_nc = new List<Cot>();

for (int i = 0; i < copiedColumns.Count; i++)

{

var col = copiedColumns[i];

var values = filteredDataFrom\_nc.GroupBy(row => row[col.Number]).Select(c => c.Key).ToList();

values.Remove("?");

if (values.Count > 1 || i == copiedColumns.Count - 1)

{

Cot newCol = col.Clone();

newCol.giatri = values;

filteredColumnsFrom\_nc.Add(newCol);

}

}

stack.Push(new TaoNode(filteredDataFrom\_nc, filteredColumnsFrom\_nc, child));

} // thêm node con

}

return tree;

1. TÀI LIỆU THAM KHẢO
   * <http://www.cs.umd.edu/~samir/498/10Algorithms-08.pdf>
   * <https://en.wikipedia.org/wiki/C4.5_algorithm>
   * <https://www.academia.edu/1983952/C4._5_algorithm_and_Multivariate_Decision_Trees>