# Bài toán phân lớp trong khai phá dữ liệu

| Khái niệm Phân lớp là quá trình gán nhãn cho các đối tượng của 1 dataset dựa trên các đặc tính của đối tượng đó. Quá trình có thể được dùng để hỗ trợ việc xây dựng các mô hình để tự động hóa việc phân lớp các đối tượng  Có 2 loại phân lớp là phân loại nhị phân và phân loại đa lớp.   * Phân loại nhị phân là việc chia các đối tượng vào 2 lớp duy nhất, như ‘spam’ và ‘not spam’ binary classification và multi-class classification. * Phân loại đa lớp là việc chia các đối tượng vào nhiều hơn 2 lớp . |
| --- |
| Các bước của quá trình xây dựng mô hình phân lớp dữ liệu Bước 1: Thu thập dữ liệu   * Thu thập dữ liệu cần thiết cho mô hình từ những nguồn có thể: * Làm khảo sát * Tìm từ các database có sẵn trên Internet     Bước 2: Xử lý dữ liệu   * Thực hiện tiền xử lý cho những dữ liệu thu thập được. Giải quyết các vấn đề như: bị thiếu giá trị, xử lý ngoại lai, chuyển đổi định dạng dữ liệu sao cho phù hợp với việc phân tích     Bước 3: Lựa chọn thuộc tính   * Lựa chọn ra những thuộc tính chính sẽ dùng cho việc phân loại * Có thể dùng các phương pháp như: phân tích tương quan, đo lường tính hữu dụng của thông tin, phân tích thành phần chính     Bước 4: Lựa chọn mô hình   * Ở bước này, ta cần lựa chọn ra thuật toán phân lớp phù hợp với mong muốn của mình * Một số thuật toán có thể sử dụng: decision trees, support vector machines, và neural network     Bước 5: Đào tạo mô hình và đánh giá mô hình   * Những dữ liệu đã xử lý sẽ chia làm 2 loại: tập dữ liệu đào tạo và tập dữ liệu xác thực. Mô hình sẽ được đào tạo bằng tập dữ liệu đánh giá và được đánh giá/ kiểm thử 1 lần nữa bằng tập dữ liệu xác thực |
| Ví dụ Ngân hàng: Giám đốc tín dụng ngân hàng cần phân tích dữ liệu để xác định người vay nào được coi là "an toàn" và người nào "rủi ro" đối với ngân hàng.   * B1: Thu thập dữ liệu:   + Thu thập những thông tin liên quan của người vay như: tuổi, mức thu nhập, nghề nghiệp, lịch sử tín dụng, số tiền vay, thời hạn vay * B2: Xử lý dữ liệu   + Xử lý các vấn đề như thiếu thông tin, thông tin nhiễu/ngoại lai   + Chuẩn hóa thông tin * B3: Lựa chọn thuộc tính   + Xác định các thuộc tính có liên quan nhất góp phần vào việc dự đoán vỡ nợ * B4: Lựa chọn mô hình   + Chọn thuật toán phù hợp dựa trên dữ liệu thu được * B5: Đào tạo và đánh giá mô hình   + Chia lượng thông tin thu được thành 2 phần: tập huấn luyện và tập xác thực     - Huấn luyện mô hình trên tập huấn luyện để tìm hiểu mối quan hệ giữa các thuộc tính và khả năng vỡ nợ của đối tượng     - Đánh giá hiệu suất của mô hình bằng cách tập xác thực |
| Các kỹ thuật phân lớp dữ liệu  * Phân lớp cây quyết định      * Phân lớp Bayesian      * Mô hình phân lớp k-hàng xóm gần nhất      * Mạng nơron      * Các thuật toán di truyền |

# 

# Tìm hiểu về bài toán phân lớp bằng phương pháp cây quyết định, và đưa ví dụ

| Khái niệm cây quyết định:  * Cây quyết định là một cấu trúc dạng cây, trong đó:   + Root node (Gốc): Là nút trên cùng của cây. Nơi bắt đầu quá trình phân chia. Chứa thông tin về thuộc tính đầu tiên được chọn để phân chia dữ liệu.   + Internal Nodes (Nút trong): Là các nút nằm giữa nút gốc và nút lá. Mỗi Internal Node đại diện cho một thuộc tính trong tập dữ liệu và chứa điều kiện để phân chia dữ liệu thành các nhánh   + Branches (Nhánh): Các kết nối giữa các nút, biểu thị sự phân chia dữ liệu dựa trên giá trị của thuộc tính. Chỉ ra các hướng đi từ một nút đến các nút con, thể hiện các kết quả của điều kiện phân chia.   + Leaf Nodes (Nút lá): Các nút không có nút con, nằm ở cuối của cây. Biểu diễn lớp hay sự phân phối lớp.     => Quá trình xây dựng cây quyết định cũng đã được các nhóm tìm hiểu về thuật toán ID3 trong cây quyết định trình bày, vậy nên chúng ta sẽ cùng đi luôn vào phần ví dụ cụ thể. |
| --- |
| Ví dụ về phân lớp bằng cây quyết định: (chỉ cho ảnh vào slide) Bài toán  :  Xét ví dụ trên:  - Hình a với hai class màu xanh lục và đỏ trên không gian hai chiều. Nhiệm vụ là đi tìm ranh giới giúp phân chia hai class này. Hay nói cách khác, ta cần xây dựng một bộ phân lớp để quyết định việc một điểm dữ liệu mới thuộc vào class nào. Quan sát hình ta thấy rằng ranh giới cho hai class trong bài toán này là các đường song song với các trục toạ độ.  Nếu một điểm có hoành độ, < ngưỡng , ta quyết định ngay được rằng nó thuộc class xanh lục. Ngoài ra, hoành độ > ngưỡng , ta quyết định nó cũng thuộc vào class xanh lục. Xét tiếp, nếu hoành độ lớn hơn ngưỡng ta quyết định nó thuộc vào class xanh lục. Các điểm không thoả mãn các điều kiện trên được xếp vào class đỏ. Việc ra quyết định một điểm thuộc class nào được mô tả trên decision tree trên Hình b. |
| 3. Ưu nhược điểm của việc sử dụng cây quyết định (Decision Tree)  1. Ưu điểm:    * Trực quan và dễ hiểu: Mô hình cây quyết định rất trực quan và có thể dễ dàng giải thích bằng cách nhìn vào các nhánh của cây.    * Không yêu cầu chuẩn hóa dữ liệu: Không giống như nhiều thuật toán khác, cây quyết định không yêu cầu chuẩn hóa dữ liệu đầu vào.    * Nhìn chung, các giải thuật cây quyết định cho kết quả có độ chính xác khá cao. 2. Hạn chế:    * Đối với các tập dữ liệu có nhiều thuộc tính thì cây quyết định sẽ lớn (về chiều sâu cả chiều ngang), vì vậy làm giảm độ dễ hiểu.    * Việc xếp hạng các thuộc tính để phân nhánh dựa vào lần phân nhánh trước đó và bỏ qua sự phụ thuộc lẫn nhau giữa các thuộc tính.    * Khi dùng độ lợi thông tin (Information Gain) để xác định thuộc tính rẽ nhánh, các thuộc tính có nhiều giá trị thường được ưu tiên chọn   -> gây ra hiện tượng biased decision tree (cây quyết định thiên vị). Một số đặc điểm của biased decision tree: overfitting (quá khớp với training data), underfitting (không khớp với kiểu mẫu của training data) |
| 4. Phương pháp cây quyết định C4.5:  1. Khái niệm:   C4.5 là một thuật toán học máy để xây dựng cây quyết định được phát triển bởi Ross Quinlan vào những năm 1993. C4.5 là phiên bản cải tiến của thuật toán ID3 trước đó, với khả năng xử lý các thuộc tính có giá trị liên tục và thiếu giá trị (missing values) trong dữ liệu huấn luyện. Thuật toán này sử dụng độ đo đóng góp thông tin (information gain) để chọn thuộc tính phân chia tốt nhất cho mỗi nút trên cây. Ngoài ra, C4.5 còn sử dụng kỹ thuật cắt tỉa cây (pruning) để giảm thiểu hiện tượng quá khớp và tăng khả năng tổng quát hóa của mô hình.   1. Các bước thực hiện:   Thuật toán C4.5 là một phiên bản cải tiến của ID3 và chủ yếu sử dụng cho các bài toán phân loại. Khi sử dụng C4.5, ta sẽ trải qua các bước cụ thể để xây dựng cây quyết định dựa trên dữ liệu đã có. Sau đây là các bước chính, công thức, và phép tính liên quan đến việc sử dụng C4.5.   1. **Tính Entropy cho toàn bộ dữ liệu**: Đây là bước đầu tiên để tính sự bất định (không đồng nhất) của dữ liệu tại một thời điểm. Entropy giúp đánh giá mức độ phân tán của các nhãn lớp. 2. **Tính Information Gain (Lợi ích thông tin)**: C4.5 tìm cách giảm sự bất định (Entropy) tại mỗi bước. Do đó, chúng ta tính toán chỉ số **Information** **Gain (lợi ích thông tin)** cho từng thuộc tính để xác định thuộc tính nào giúp giảm Entropy nhiều nhất. 3. **Tính Gain Ratio (Tỷ lệ lợi ích thông tin)**: C4.5 sử dụng **Gain Ratio** thay vì chỉ sử dụng **Gain** như trong ID3. Gain Ratio được sử dụng để cân bằng giữa các thuộc tính có nhiều hoặc ít giá trị phân lớp, khắc phục thiên vị của chỉ số Gain. 4. **Chọn thuộc tính phân chia**: Thuộc tính có Gain Ratio cao nhất sẽ được chọn làm thuộc tính phân chia tại mỗi bước. 5. **Phân chia tập dữ liệu**: Sau khi chọn thuộc tính phân chia, dữ liệu sẽ được chia thành các nhóm con (dựa trên giá trị của thuộc tính được chọn) để tiếp tục quá trình phân lớp. 6. **Lặp lại quá trình**: Quy trình này sẽ được lặp lại cho đến khi không còn thuộc tính nào để phân chia, hoặc khi tất cả dữ liệu trong một nhánh có cùng nhãn. 7. **Tỉa cây (Pruning)**: Sau khi tạo ra cây quyết định, C4.5 thực hiện bước tỉa cây để loại bỏ các nhánh không cần thiết, giảm thiểu overfitting.  3. Công thức và các phép tính*Entropy* Khái niệm này khá giống với khái niệm entropy trong nhiệt động học của môn Vật lý, đều để đo lường mức độ hỗn loạn hoặc không chắc chắn của một tập dữ liệu. Công thức Entropy cho một gtập dữ liệu với các lớp là:  Trong đó:   * là xác suất của lớp trong tập dữ liệu .   Ý nghĩa của **Entropy** là đo lường mức độ không chắc chắn hay hỗn loạn của một tập dữ liệu. Nếu tất cả các mẫu trong tập dữ liệu thuộc cùng một lớp, thì entropy sẽ là 0 (không hỗn loạn). Nếu các mẫu phân bố đồng đều qua các lớp, entropy sẽ cao hơn, cho thấy mức độ hỗn loạn cao.  Giá trị của entropy nằm trong đoạn [0, 1]  - Entropy = 0: tất cả các mẫu trong tập dữ liệu thuộc cùng một lớp (không hỗn loạn)  - Entropy = 1: các mẫu phân bố đồng đều giữa các lớp Information Gain (Lợi ích thông tin) Lợi ích thông tin cho biết mức độ giảm Entropy khi chia dữ liệu theo một thuộc tính. Công thức tính Gain của một thuộc tính là:  Trong đó:   * là các giá trị có thể có của thuộc tính . * là tập con của chứa các mẫu có giá trị cho thuộc tính .   Information Gain đo lường lượng thông tin mà thuộc tính A cung cấp để phân chia dữ liệu. Một thuộc tính có Information Gain cao sẽ giúp giảm độ hỗn loạn (entropy) nhiều hơn sau khi phân chia, do đó là một thuộc tính tốt để tạo nút quyết định.  Infomation Gain ≥ 0, có nghĩa là việc chia tách một tập dữ liệu dựa trên bất kỳ thuộc tính nào sẽ luôn có xu hướng dẫn đến việc giảm entropy hoặc sự không chắc chắn. *Split Information (Thông tin phân chia)* C4.5 không chỉ dựa vào Gain mà còn tính toán thông tin phân chia để tránh sự thiên vị đối với các thuộc tính có nhiều giá trị. Công thức tính Split Information của một thuộc tính là:  Split Info đo lường xem thuộc tính A có thể phân chia tập dữ liệu thành các tập con nhỏ đến mức nào. Nếu một thuộc tính chia dữ liệu thành quá nhiều tập con nhỏ, Split Info sẽ có giá trị cao, tức là có rất nhiều sự phân tán nhưng không thực sự mang lại nhiều thông tin cho quá trình phân loại. Điều này giúp tránh chọn các thuộc tính có quá nhiều giá trị riêng lẻ (ví dụ, một số định danh hoặc giá trị duy nhất cho mỗi đối tượng). *Gain Ratio (Tỷ lệ lợi ích thông tin)* Gain Ratio là chỉ số được C4.5 sử dụng để quyết định thuộc tính phân chia, nhằm cân bằng giữa Gain và Split Information. Công thức tính Gain Ratio là:  Gain Ratio là một chỉ số cải tiến từ Information Gain, nhằm tránh việc chọn những thuộc tính có nhiều giá trị riêng biệt (như số định danh). Gain Ratio tính đến cả mức độ phân chia dữ liệu do thuộc tính gây ra, giúp đưa ra quyết định tốt hơn.  Thuộc tính có Gain Ratio cao nhất sẽ được chọn làm thuộc tính phân chia. *Tỉa cây (Pruning)* C4.5 sử dụng phương pháp tỉa cây để giảm thiểu việc quá khớp dữ liệu. Tỉa cây giúp đơn giản hóa cây quyết định bằng cách loại bỏ các nhánh không đóng góp đáng kể vào việc phân loại. Phương pháp tỉa cây có thể sử dụng phương pháp **tỉa hậu nghiệm (post-pruning)** hoặc **tỉa trước (pre-pruning)** dựa trên các ngưỡng. |
| **Ví dụ áp dụng thuật toán C4.5:** |
| B1: Ta sẽ tính lần lượt Entropy, information gain, split info và Gain Ratio để  tìm ra nút gốc  B2 Sau đó so sánh các giá trị Gain Ratio của các thuộc tính với nhau và lấy ra  thuộc tính có giá trị gain ratio cao nhất  B3: Lấy thuộc tính đó làm gốc r phân chia các cạnh là giá trị thực của thuộc  tính đó.  B4: Thu gọn bảng dữ liệu dựa trên các giá trị đang xét của thuộc tính. Rồi lặp  lại B1 cho đến khi cây quyết định phân lớp được hết dữ liệu hoặc sử dụng hết thuộc  tính. |
| **Tính entropy(E) của play tennis:**  **Tính entropy(E) của outlook**  **Tính Information Gain(IG) của outlook**  **Tính Split Info(SI)**  **)**  **Tính Gain Ratio(GR)**  **)** |
| Tương tự ta tìm được  ) (Cao nhất)  )  )  )  => Dựa vào hệ số gain ratio của từng thuộc tính thì khi chọn outlook là gốc ta sẽ lấy được thông tin với chính xác cao nhất. Như vậy ta sẽ dựng cây với thuộc tính Outlook làm nút gốc. Đưa ra các cạnh là các giá trị của thuộc tính Outlook. Từ đó ta có cây như sau: |
| Theo như cây được minh họa như trên cùng với tập dữ liệu, ta có thể cho rằng khi Outlook có giá trị là Overcast thì khả năng quyết định chơi Tennis sẽ là Yes. Vì vậy giá trị Overcast sẽ không cần được xét đến ở những lần tính tiếp theo. Tiếp tục tính các hệ số khi xét đến điều kiện của Outlook là Sunny. Từ đó ta rút ngắn được bảng giá trị để xét |
| **Tính entropy(E)** của **play tennis** nhưng không phải của bảng đầu mà là khi **Outlook** = **Sunny**  **Các bước lại lặp lại như phần đầu**  **Gain Ratio(GR)**  )  )  ) |
| Như vậy, ta sẽ chọn thuộc tính Humidity để làm nút nối từ Outlook qua giá trị sunny. Từ đó cây sẽ được tiếp tục phát triển như sau: |
| Cứ lặp lại 4 bước cuối cùng chúng ta sẽ có |

# 