Báo cáo Môn học Toán cho Khoa Học Máy Tính

Nguyễn Trường Giang
Lớp sinh hoạt: KHTN2024
MSSV: 24520011
Trường đại học Công Nghệ Thông Tin
TP. Hồ Chí Minh, Việt Nam
giangnt.2006.it@gmail.com

28 - 9 - 2025

Tóm tắt nội dung

*Đây là bài báo cáo **assignment 0: Tìm hiểu về Decision Tree** môn học **Toán cho Khoa Học Máy Tính**. Trong bài báo cáo này, tôi sẽ nói về **định nghĩa về Decision Tree**, **công dụng** và **các tiêu chuẩn split** như **gini**, **entropy** cho decision tree. Cùng với đó, tôi sẽ **xây dựng thuật toán Decision Tree** và cách áp dụng Decision Tree. Code demo của bài báo cáo này được để trong link google colab sau: https://colab.research.google.com/github/HiGiangcoder/jupyter_notebook/blob/master/Assignment0_MCS.ipynb*

1 Decision Tree là gì?

Decision Tree (hay còn gọi là cây quyết định) [2] là một mô hình học có giám sát không có tham số (non-parametric supervised learning), được áp dụng cho cả bài toán phân loại (classification) hoặc bài toán hồi quy (regression) bằng cách xây dựng một cấu trúc dạng cây.

Decision Tree có 2 loại [3]: Classification Tree và Regression Tree. Đối với **Classification Tree**, cây được dùng để phân loại dữ liệu thành các nhãn rời rạc. Còn đối với **Regression Tree**, cây này sẽ được sử dụng để dự đoán giá trị số liên tục. Nói một cách khác, Decision Tree có thể xử lý cả 3 dạng dữ liệu: Dữ liệu rời rạc, dữ liệu liên tục, và dữ liệu hỗn hợp.

Cấu trúc của một Decision Tree [2] bao gồm root node (nốt gốc), internal nodes (nốt phân nhánh), branches (nhánh) và leaf nodes (nốt lá).

Root node: Decision Tree bắt đầu từ node này, do đó trong một decision tree, chỉ có duy nhất một root node. Từ node này sẽ có một điều kiện kiểm tra và chia ra các nhánh đến internal node.

Internal node: Là node mà tại đó có một điều kiện kiểm tra, và từ điều kiện kiểm tra đó sẽ có các nhánh con.

Branch: Là kết quả của điều kiện tại internal node (tức là tại internal node, sau khi thực hiện điều kiện kiểm tra, sẽ chia ra làm nhiều nhánh đến các leaf nodes).

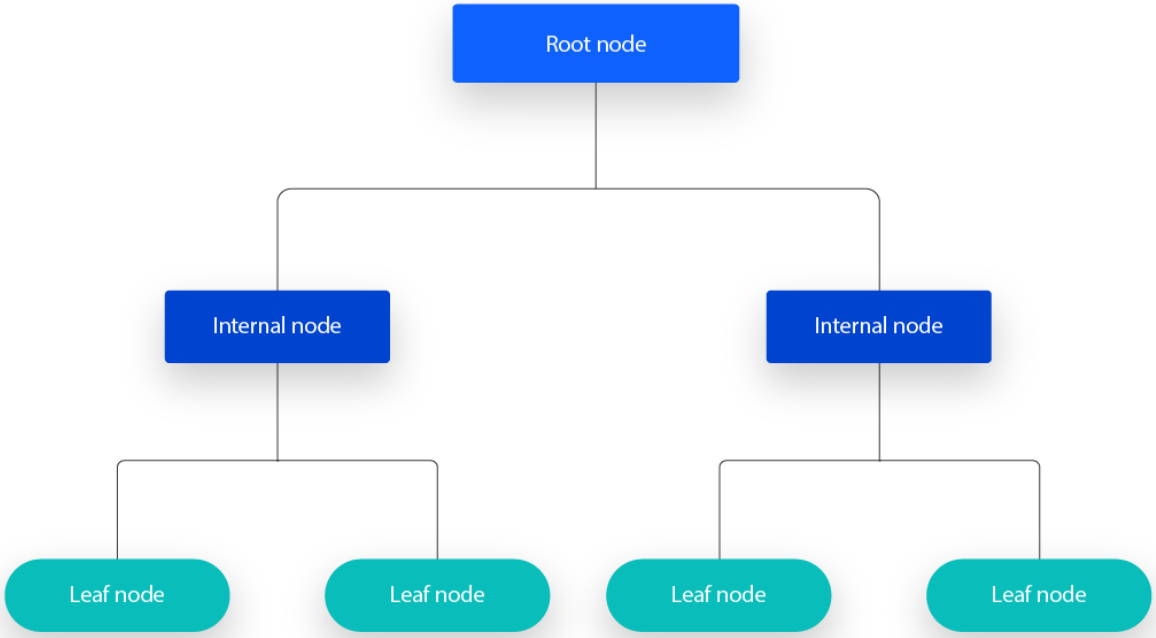
Leaf node: Là node cuối cùng, không có điều kiện kiểm tra như internal node, và tất nhiên cũng sẽ không có nhánh nào xuất hiện từ node này.

2 Công dụng

Decision Tree được dùng rộng rãi do các ưu điểm sau:

Phân loại (classification): Bài toán phân loại sẽ dự đoán nhãn. Ví dụ như bài toán chuẩn đoán bệnh, phân loại khách hàng, hay lọc spam.

Hồi quy (regression): Dự đoán giá trị liên tục, trong đó mỗi node lá sẽ cho giá trị trung bình hoặc giá trị ước tính. Ví dụ như dự đoán giá nhà hay doanh thu.



Hình 1: Sơ đồ Decision Tree

Kết hợp vào mô hình phức tạp hơn: Do mỗi cây dễ giải thích, nhưng hiệu năng có thể thấp, các mô hình tổng hợp giúp cải thiện hiệu quả. Ví dụ như Random Forest, Gradient Boosted Trees (XGBoost, LightGBM…), đều dùng cây quyết định làm thành phần cơ bản.

Khả năng xử lý dữ liệu hỗn hợp: Có thể dùng với các thuộc tính liên tục và rời rạc. Ngoài ra còn xử lý dữ liệu không tuyến tính, tương tác giữa các biến.

Tính giải thích, trực quan. Kết quả dễ hiểu, tức là bạn có thể đọc quy tắc trực tiếp từ cây. Dễ kiểm tra logic, dễ đưa vào hệ thống quyết định (business rules).

3 Các tiêu chuẩn split:

Trong bài toán Decision Tree, có 2 vấn đề cần phải giải quyết là tại mỗi internal node, chúng ta cần phải chọn thuộc tính nào để chia dữ liệu xuống node con và điều kiện đó là gì. Ở phần trình bày cách xây dựng thuật toán tiếp theo, ta sẽ cần sử dụng một số các tiêu chuẩn split để tính toán cách chia dữ liệu xuống node con. Do đó, để thuận tiện hơn trong việc mô tả thuật toán ở phần tiếp theo, ta sẽ bàn về các tiêu chuẩn split.

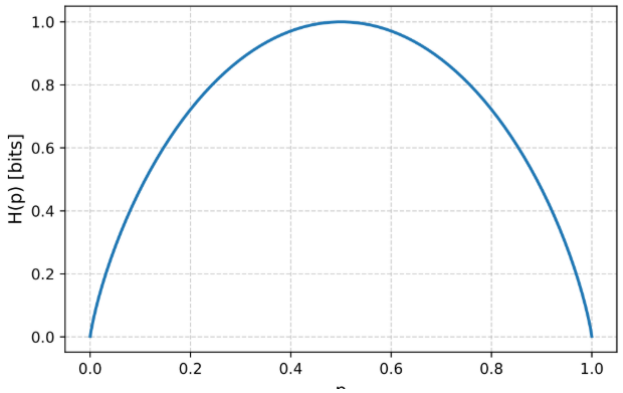
3.1 Entropy và Information Gain:

Entropy trong lý thuyết thông tin được sử dụng để đo lường mức độ hỗn loạn (hoặc độ không tinh khiết) của một tập dữ liệu. Nếu Entropy của một tập dữ liệu có giá trị thấp (tiến đến 0), chứng tỏ tập dữ liệu đó gần như chỉ có một loại duy nhất, ít bị các loại dữ liệu khác lẫn vào. Trường hợp ngược lại, khi giá trị Entropy của tập dữ liệu đó cao (tiến đến 1), chứng tỏ tập dữ liệu đó đang bị trộn lẫn rất nhiều loại dữ liệu khác nhau. Công thức của hàm Entropy được định nghĩa như sau:

$$H(S) = -\sum_{c \in C} p_c \log_2 p_c$$

trong đó, S là tập dữ liệu ban đầu, C là tập các nhãn (classes), p_c là xác suất có nhãn p_c trong S. Hàm Entropy trên sẽ đo được tập dữ liệu đó có khó phân tách hay không, nếu giá trị thu được càng lớn thì tập dữ liệu đó càng khó phân tách.

Infomation Gain được tính dựa trên Entropy, là độ giảm trung bình của entropy sau khi tách theo thuộc tính A. Thuộc tính có giá trị IG càng lớn thì khả năng phân tách càng tốt. Khi phân tách tập mẫu S theo thuộc tính A thành các nhóm con $\{S_v\}$ với $v \in Vals(A)$ (Tập các giá trị (hoặc ngưỡng) của thuộc tính A.), Information Gain được định



Hình 2: Đồ thị của hàm entropy $H(p) = -p\log_2 p - (1-p)log_2(1-p)$ với $p \in (0, 1)$.

nghĩa:

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

trong đó, A là thuộc tính ta định tách, S_v là tập con của S gồm các mẫu có $A = v$.

3.2 Gini Impurity (hệ số "tạp" Gini) và Gini Gain:

Gini Impurity [2] (hay còn gọi là chỉ số tạp Gini) được sử dụng để đo xác suất chọn sai nhãn khi ta lấy ngẫu nhiên một phần tử trong tập dữ liệu và gán nhãn theo phân phối lớp trong tập đó.

Gini Impurity của tập mẫu S được xác định bởi

$$G(S) = \sum_{c \in C} p_c(1 - p_c) = 1 - \sum_{c \in C} p_c^2$$

trong đó, C là tập các nhãn (classes), p_c là xác suất có nhãn c trong tập S, $G(S)$ đo mức "không tinh khiết" hoặc "không thuần" (impurity) trong phân phối nhãn.

Gini Gain được sử dụng để đo độ tốt của phép chia dữ liệu giá trị của hàm Gini Gain được tính sau đây càng lớn, càng chứng tỏ phép chia đó càng tốt. Khi tách S theo thuộc tính A thành các tập con $\{S_v\}$ với $v \in Vals(A)$, Gini Gain có dạng:

$$GG(S, A) = G(S) - \sum_{v \in Vals(A)} \frac{|S_v|}{|S|} G(S_v)$$

trong đó, S_v là tập con của S gồm các mẫu có $A = v$, $GG(S, A)$ (Gini Gain) đo mức giảm impurity thu được nhờ phân tách theo A (tương tự như IG).

3.3 So sánh Entropy và Gini

Điểm giống nhau: Cả hai đều đánh giá chất lượng của phép phân tách dựa trên mức độ "thuần khiết" (purity) của nhãn trong tập dữ liệu. Khi tính toán giá trị của node cha và các node con, cả hai phương pháp đều sử dụng trọng số $\frac{|S_v|}{|S|}$ để phản ánh quy mô của từng nhánh con so với toàn bộ tập dữ liệu.

Điểm khác nhau: Entropy/Information Gain sử dụng hàm logarit để đo lường mức độ bất ngờ (surprise) hay độ hỗn loạn thông tin trung bình, trong khi Gini/Gini Gain không dùng logarit nên có độ phức tạp tính toán thấp hơn. Entropy có xu hướng nhạy cảm hơn với các lớp có tần suất nhỏ (rare classes), do đó có thể phản ánh tốt hơn sự mất cân bằng dữ liệu. Ngược lại, Gini Impurity thường ưu tiên tạo ra các node thuần (pure nodes) và ít dao động hơn trước sự thay đổi nhỏ trong phân bố tần suất của các nhãn. Entropy/IG được áp dụng trong thuật toán ID3, và Gini/GG được áp dụng vào thuật toán CART, đây là 2 thuật toán Decision Tree rất nổi tiếng.

4 Xây dựng thuật toán Decision Tree

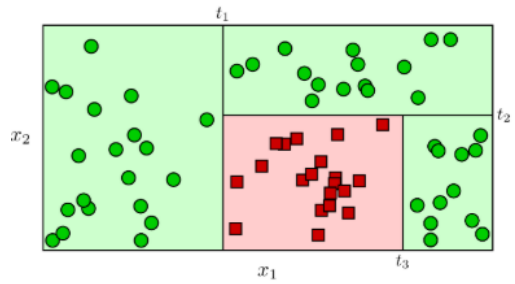
Như chúng ta đã biết, thuật toán Decision Tree có thể áp dụng vào 2 dạng chính là bài toán phân loại và bài toán hồi quy. Do đó, trong bài viết này, ta sẽ chỉ bàn về bài toán phân loại.

Trong phần này, ta sẽ phân tích chi tiết về thuật toán ID3 (Iterative Dichotomiser 3) [1],

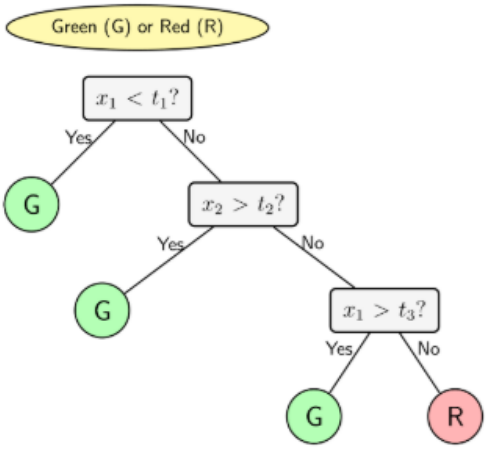
4.1 Ý tưởng

Trong ID3 [3], chúng ta cần xác định thứ tự của thuộc tính cần được xem xét tại mỗi bước. Tức là, tại mỗi bước, một thuộc tính tốt nhất sẽ được chọn ra dựa trên một tiêu chuẩn nào đó (các tiêu chuẩn này ta sẽ bàn sau). Với thuộc tính được chọn đó, ta chia dữ liệu vào các node con tương ứng với các giá trị của thuộc tính đó rồi tiếp tục áp dụng phương pháp này cho mỗi node con. Việc chọn ra thuộc tính tốt nhất ở mỗi bước như này được gọi là cách chọn tham lam.

Trước hết, thế nào là một phép phân chia tốt? Dựa vào trực giác, một phép phân chia tốt nếu như



Hình 3: Ví dụ về cách Decision Tree phân loại 2 nhóm xanh và đỏ



Hình 4: Ví dụ trực quan một mô hình Decision Tree khi phân loại 2 nhóm xanh và đỏ

dữ liệu trong mỗi node con hoàn toàn thuộc một class, khi đó node con này sẽ được coi như là một leaf node, tức là ta không cần phân chia thêm nữa. Ngược lại, nếu dữ liệu trong các node con này vẫn lẫn lộn vào nhau theo một tỉ lệ lớn, ta coi rằng phép phân chia đó chưa thực sự tốt.

Từ nhận xét trên, ta cần có một hàm số đo độ tinh khiết (purity), hoặc độ vẩn đục (impurity) của một phép phân chia. Hàm số này sẽ phải cho giá trị thấp nhất nếu dữ liệu trong mỗi node con là tinh khiết nhất (đều thuộc một class), và sẽ cho giá trị cao nếu mỗi node côn có chứa dữ liệu thuộc nhiều class khác nhau.

Một hàm số có các đặc điểm này và được dùng nhiều trong lý thuyết thông tin là hàm entropy.

ID3 là thuật toán tạo Decision Tree cho bài toán phân loại, dùng **Information Gain** làm tiêu chí chọn thuộc tính.

Algorithm 1: ID3 Decision Tree Learning

Input: Dataset S, set of attributes Attrs
Output: Decision tree T
if *all samples in* S *have same class* C **then**
  return leaf labeled C;
if *Attrs is empty* **then**
  return leaf labeled by majority class in S;
for *each attribute* $A \in Attrs$ **do**
  compute $IG(S, A)$;
$A^* \leftarrow \arg\max_A IG(S, A)$;
node $\leftarrow$ split on A^*;
for *each value* $v \in Vals(A^*)$ **do**
  $S_v \leftarrow$ subset of S where $A^* = v$;
  if S_v *is empty* **then**
    attach leaf labeled by majority class in S;
  else
    child $\leftarrow ID3(S_v, \text{Attrs} \setminus \{A^*\})$;
    attach child to node for branch v;
return node;

4.2 Thuật toán ID3

Mô tả thuật toán: Bắt đầu từ tập mẫu S ở node gốc. Nếu tất cả mẫu cùng class, gán node đó là leaf node; nếu không còn thuộc tính để chia, gán nhãn phổ biến nhất. Ngược lại, tính $IG(S, A)$ cho mọi A, chọn $A^* = \arg\max_A IG(S, A)$ làm thuộc tính chia, tạo node con cho mỗi giá trị $v \in Vals(A^*)$, và lặp lại đệ quy cho từng S_v.

Ghi chú thực tế:

- ID3 chủ yếu xử lý thuộc tính rời rạc; với thuộc tính liên tục cần tạo ngưỡng (ví dụ thử các điểm giữa các giá trị và chọn ngưỡng tối ưu).
- ID3 có xu hướng thiên vị thuộc tính có nhiều giá trị; C4.5 khắc phục bằng *gain ratio*.
- Để tránh overfitting, thường áp dụng pre-pruning (max_depth, min_samples_split, min_samples_leaf) hoặc post-pruning.

4.3 Ví dụ minh họa: Play-Tennis (tính IG cho một thuộc tính)

Bảng dữ liệu (tóm tắt giả định giống ví dụ hay dùng):

Ngày	Outlook	Humidity	Play?
1	Sunny	High	No
2	Sunny	High	No
3	Overcast	High	Yes
4	Rain	High	Yes
5	Rain	Normal	Yes
6	Rain	Normal	No
7	Overcast	Normal	Yes
8	Sunny	High	No
9	Sunny	Normal	Yes
10	Rain	Normal	Yes
11	Sunny	Normal	Yes
12	Overcast	High	Yes
13	Overcast	Normal	Yes
14	Rain	High	No

Ở tập toàn bộ S: có 9 ‘Yes‘ và 5 ‘No‘. Entropy ban đầu:

$$H(S) = -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right) \approx 0.940. \quad (1)$$

Tính Information Gain cho thuộc tính `Humidity` (giả sử hai giá trị: High (7 mẫu), Normal (7 mẫu)):

$$H(Humidity = High) \approx -\left(\frac{3}{7}\log_2\frac{3}{7} + \frac{4}{7}\log_2\frac{4}{7}\right) \approx 0.985, \quad (2)$$

$$H(Humidity = Normal) \approx -\left(\frac{6}{7}\log_2\frac{6}{7} + \frac{1}{7}\log_2\frac{1}{7}\right) \approx 0.592. \quad (3)$$

Trọng số:

$$IG(S, Humidity) = 0.940 - \left(\frac{7}{14}\cdot 0.985 + \frac{7}{14}\cdot 0.592\right) \approx 0.151 \quad (4)$$

Tương tự tính cho các thuộc tính khác (Outlook, Temperature, ...), chọn thuộc tính có IG lớn nhất.

5 Cách áp dụng Decision Tree

5.1 Chuẩn bị dữ liệu

Xử lý giá trị thiếu (imputation): loại bỏ mẫu/thuộc tính nhiều missing, hoặc thay bằng trung bình/median/mode.

Mã hóa biến rời rạc: one-hot hoặc label encoding; chú ý khi dùng cây, label encoding có thể là đủ nhưng one-hot thường an toàn.

Chuẩn hóa/scale: không bắt buộc cho cây, vì cây phân nhánh theo ngưỡng.

Chia dữ liệu: train - validation - test (ví dụ 60 - 20 - 20) để tuning và kiểm tra overfitting.

5.2 Xây dựng cây

1. Chọn tiêu chí split (Entropy/IG cho thuật toán ID3 được trình bày như trên hoặc Gini/GG cho thuật toán CART chưa được trình bày trong bài viết này).
2. Với biến liên tục: xác định tập ngưỡng thử (thường trung điểm giữa các giá trị khác nhau).
3. Chạy đệ quy, theo điều kiện dừng (max_depth, min_samples_split).

5.3 Cắt tỉa cây (Pruning)

Pre-pruning (cắt tỉa sớm). Ý tưởng: ngăn không cho cây phát triển quá sâu hoặc quá phức tạp ngay từ đầu. Cách thực hiện: đặt ra các điều kiện dừng (stopping criteria) trong quá trình xây dựng cây. Ví dụ:

- max depth: giới hạn độ sâu tối đa của cây.
- min samples split: yêu cầu số lượng mẫu tối thiểu để một node được tách nhánh.
- min samples leaf: quy định số lượng mẫu tối thiểu trong một node lá.

Ưu điểm: nhanh, tránh cây phát triển quá mức. Nhược điểm: có thể dừng quá sớm, làm mất đi những phân tách quan trọng.

Post-pruning (cắt tỉa sau). Ý tưởng: xây dựng một cây hoàn chỉnh (full tree) trước, sau đó cắt bỏ các nhánh không cần thiết.

Cách thực hiện: Sử dụng tập validation hoặc kỹ thuật k-fold cross-validation để đánh giá hiệu năng từng nhánh. Loại bỏ (prune) những nhánh mà khi cắt đi thì hiệu năng của cây không giảm, hoặc thậm chí còn tăng.

Ví dụ điển hình: Reduced-Error Pruning, trong đó ta thử cắt từng node con và so sánh độ chính xác, nếu kết quả không tệ hơn thì giữ lại phiên bản cắt tỉa.

Ưu điểm: thường cho cây đơn giản và ổn định hơn so với pre-pruning. Nhược điểm: chi phí tính toán cao hơn vì cần huấn luyện cây đầy đủ và đánh giá lại nhiều lần.

5.4 Dự đoán và sử dụng cây

Đối với một mẫu dữ liệu mới, quá trình dự đoán được thực hiện như sau: bắt đầu từ *root node*, lần lượt xét các điều kiện tại mỗi node và di chuyển theo nhánh phù hợp. Quá trình này tiếp tục cho đến khi gặp một *leaf node*.

- Trong **bài toán phân loại (Classification)**, cây sẽ trả về nhãn lớp tương ứng tại node lá.
- Trong **bài toán hồi quy (Regression)**, cây sẽ trả về giá trị dự đoán (thường là trung bình của các mẫu huấn luyện tại node lá).

5.5 Đánh giá và kiểm tra mô hình

Hiệu quả của Decision Tree cần được đánh giá bằng các thước đo phù hợp với từng loại bài toán:

- **Bài toán phân loại (Classification)**: Accuracy, Precision, Recall, F1-score, ROC-AUC.
- **Bài toán hồi quy (Regression)**: MAE, MSE, RMSE, hệ số xác định R^2.

Ngoài ra, cần kết hợp thêm các kỹ thuật:

- So sánh hiệu suất trên tập **huấn luyện (train)** và tập **kiểm thử (test)** để phát hiện hiện tượng **overfitting**.
- Sử dụng **cross-validation** nhằm đánh giá tính ổn định của mô hình trên nhiều tập dữ liệu khác nhau, giúp giảm độ lệch trong quá trình đánh giá.

6 Kết luận

Decision Tree là mô hình trực quan, dễ hiểu, phù hợp cho nhiều bài toán phân loại và hồi quy; tuy nhiên cần quản lý overfitting bằng pruning hoặc dùng các phương pháp ensemble. Lựa chọn tiêu chí split (Entropy/IG hay Gini/GG) thường không làm khác nhau quá lớn về kết quả cuối cùng nhưng có khác biệt về tính toán và một vài đặc điểm như đã thảo luận.

Tài liệu

[1] Decision trees (1): Iterative dichotomiser 3. Technical report, Machine Learning Cơ Bản.

[2] What is a decision tree. Technical report, IBM.

[3] Đinh Quang Vinh Vũ Yến Linh, Dương Đình Thắng. Bài đọc (25 trang) về cây quyết định. Technical report, AI Vietnam, 2025.