Mục tiêu chính của Gradient Descent trong học máy là huấn luyện các mô hình bằng cách giảm thiểu sai số giữa kết quả dự đoán và kết quả thực tế.

Gradient của một hàm luôn chỉ ra hướng mà hàm tăng nhanh nhất. Do đó, việc lấy dấu âm của gradient (như trong công thức cập nhật tham số) là cơ chế trực tiếp đảm bảo thuật toán luôn di chuyển "xuống dốc" trên bề mặt hàm chi phí. Đây không chỉ là một bước thủ tục; đó là đặc điểm xác định cho phép thuật toán lặp đi lặp lại để giảm giá trị của hàm chi phí, cuối cùng dẫn đến một cực tiểu. Nguyên tắc này là nền tảng của toàn bộ quá trình tối ưu hóa.

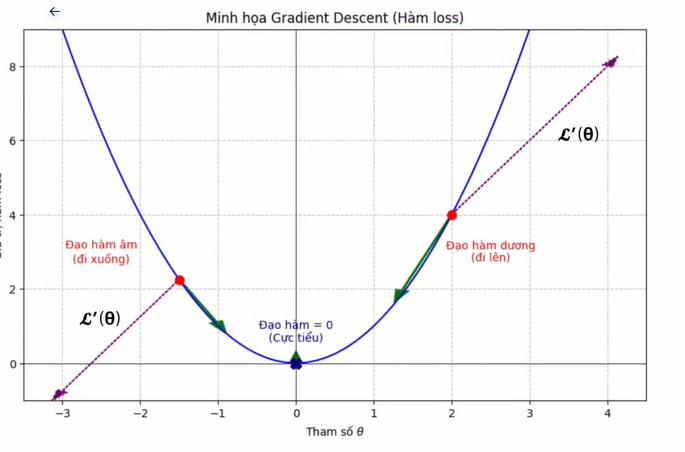
Gradient Descent hoạt động thông qua một loạt các bước lặp để giảm thiểu một hàm mục tiêu, thường là hàm chi phí hoặc hàm mất mát.

**Các thành phần cơ bản trong mô hình học máy:**

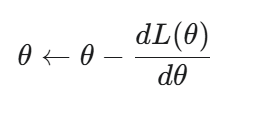
* X: Dữ liệu đầu vào (features)
  + Là thông tin thực tế (ví dụ: diện tích, số phòng ngủ,...)
* θ (theta): Tham số của mô hình
  + Là thứ mô hình cần học để đưa ra dự đoán tốt nhất
* y^ (y hat): Giá trị dự đoán
  + Mô hình dựa vào x và θ để tạo ra dự đoán y^

Làm cho dự đoán y^ gần với y nhất có thể. Để làm được điều này, ta cần:

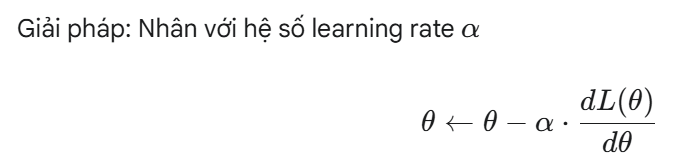
* Định nghĩa một hàm loss (hàm mất mát):
  + L(θ;x, y) Đo lường độ lệch giữa y^ và y
* Tối ưu hóa tham số θ sao cho:
  + L(θ;x, y) càng nhỏ càng tốt
* Quá trình này gọi là tối ưu hóa thường dùng Gradient Descent

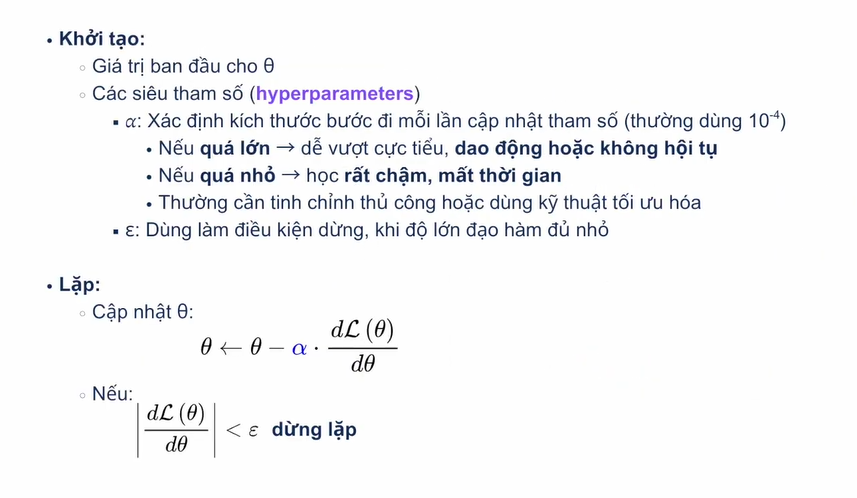


* Nhận định: dấu của đạo hàm "ngược hướng" điểm cực tiểu
* Do đó, để di chuyển về điểm cực tiểu ta phải cập nhật:



* Vấn đề: Khi giá trị đạo hàm quá lớn thì khi cập nhật, θ sẽ đi quá đà.





Cây quyết định (Decision Tree) là một mô hình học máy thuộc nhóm học có giám sát (supervised learning), được sử dụng cho cả bài toán phân loại (classification) và hồi quy (regression).

**Cây quyết định** là mô hình dạng cây, trong đó:

* **Nút gốc (Root Node)**: điểm bắt đầu, chứa điều kiện đầu tiên để phân chia dữ liệu.
* **Nút quyết định (Decision Node)**: nơi kiểm tra điều kiện và chia dữ liệu.
* **Nhánh (Branch)**: kết quả của điều kiện kiểm tra.
* **Nút lá (Leaf Node)**: kết quả cuối cùng (nhãn hoặc giá trị dự đoán).

Mục tiêu: Học các quy tắc đơn giản từ dữ liệu để dự đoán giá trị của biến mục tiêu dựa trên các đặc trưng.

**Nhãn (label)** là giá trị đầu ra mà mô hình cần dự đoán.

* Trong **phân loại**: nhãn là lớp (ví dụ: “spam” hoặc “not spam”).
* Trong **hồi quy**: nhãn là giá trị số (ví dụ: giá nhà, nhiệt độ).
* Mỗi mẫu dữ liệu trong tập huấn luyện đều có một nhãn tương ứng, và cây quyết định học cách phân chia dữ liệu để dự đoán nhãn đó.

## **Cơ chế phân chia nút (Node splitting) 🌳**

Thuật toán bắt đầu từ **nút gốc**, sau đó tiến hành **đệ quy** như sau:

1. **Tính tạp chất (impurity)** hoặc độ *không thuần* của trái (parent node).
2. Với mỗi thuộc tính (feature), thử các ngưỡng (nếu liên tục) hoặc mức giá trị (nếu rời rạc) để phân chia dữ liệu thành các tập con (child nodes).
3. Tính tạp chất cho từng tập con, rồi lấy trung bình có trọng số.
4. Chọn phân chia tối ưu giúp giảm tạp chất lớn nhất.
5. Lặp lại trên từng con cho tới khi dừng (độ sâu, số mẫu tối thiểu, tinh khiết...).

**Điều kiện dừng**:

* Tất cả mẫu trong một nút thuộc cùng một lớp.
* Không còn thuộc tính nào để phân chia.
* Đạt đến độ sâu tối đa của cây hoặc số lượng mẫu tối thiểu tại nút.

**Dự đoán**:

* Khi dự đoán cho một mẫu mới, mẫu sẽ đi qua cây từ nút gốc, qua các nút bên trong theo các điều kiện phân chia, cho đến khi đến nút lá để nhận kết quả.

### **Quy trình phân chia dữ liệu và xây dựng cây (tiếp cận "chia để trị")**

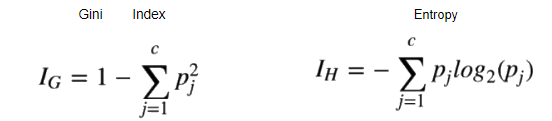
Ý tưởng cốt lõi của cây quyết định là liên tục chia tập dữ liệu thành các tập con nhỏ hơn dựa trên các giá trị của các tính năng đầu vào, từ đó hình thành cấu trúc cây.

tại mỗi bước, thuật toán chỉ chọn giải pháp tốt nhất cục bộ (tức là phân chia tốt nhất tại nút hiện tại) mà không xem xét tác động tổng thể hoặc tối ưu toàn cục của cây. Việc liên tục phân chia để đạt được độ thuần khiết cao nhất trên dữ liệu huấn luyện có thể khiến cây trở nên quá cụ thể, học cả nhiễu và các mẫu không đại diện cho dữ liệu tổng thể. Đây là một yếu tố trực tiếp gây ra vấn đề quá khớp (overfitting)

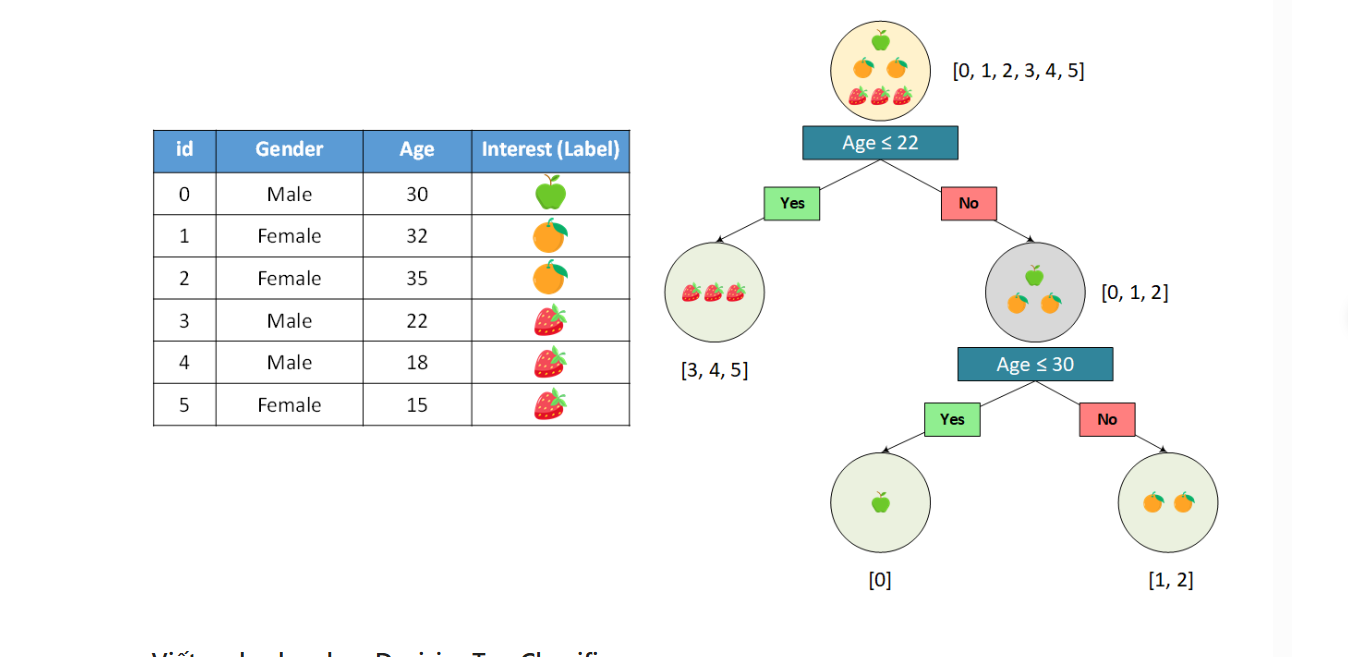
## **Các thuật toán xây dựng Cây Quyết định**

### **Tổng quan về tiêu chí phân chia**

Hai tiêu chí phổ biến nhất là

:

|  |  |
| --- | --- |
| **Gini Index** là một thước đo độ tinh khiết của tập dữ liệu, thường được sử dụng để đánh giá mức độ hỗn loạn giữa các lớp trong một nút của cây quyết định.  C: số lượng lớp trong tập dữ liệu (ví dụ: 2 lớp nếu là bài toán phân loại nhị phân).  pj : tỷ lệ mẫu thuộc lớp *j j* j trong một nút cụ thể.  **Ý nghĩa:**  Giá trị của Gini Index nằm trong khoảng từ **0 đến 1**.  **0**: Tập dữ liệu hoàn toàn tinh khiết, tức là tất cả các mẫu đều thuộc về một lớp duy nhất.  **1**: Tập dữ liệu rất hỗn loạn, các mẫu phân bố đều vào các lớp khác nhau.  Khi xây dựng cây quyết định, mục tiêu là chọn cách phân chia mà làm giảm giá trị Gini Index của các nút con sau khi phân chia. Điều này có nghĩa là các tập con càng tinh khiết càng tốt.  ví dụ : | **Entropy** là một thước đo sự không chắc chắn hoặc độ hỗn loạn trong tập dữ liệu. Nó dựa trên lý thuyết thông tin và được sử dụng để đánh giá mức độ lẫn lộn giữa các lớp.  C: số lượng lớp trong tập dữ liệu.  pj : tỷ lệ mẫu thuộc lớp *j j* j trong một nút cụ thể.  Lưu ý: Nếu *pj=0* , thì *pjlog⁡2(pj) p\_j \log\_2(p\_j)* pj log2 (pj ) được coi là 0 (vì đây là giá trị giới hạn).   * **Ý nghĩa:**   Entropy có giá trị từ **0 trở lên**.  **0**: Tập dữ liệu hoàn toàn tinh khiết, tất cả các mẫu đều thuộc về cùng một lớp, không có sự không chắc chắn.  Giá trị cao hơn cho thấy dữ liệu càng hỗn loạn, tức là các mẫu phân bố rải rác giữa các lớp.  Trong cây quyết định, chúng ta chọn cách phân chia mà làm giảm entropy nhiều nhất, giúp các nút con trở nên tinh khiết hơn. |



Ví dụ dữ liệu đầu vào gồm ba cột **Gender** (giới tính), **Age** (tuổi) và **Interest** (sở thích, tức nhãn cần phân loại). Cây quyết định được xây dựng từ dữ liệu này như trong hình trên, sử dụng thuật toán CART với chỉ số Gini để đánh giá chất lượng phân chia.

Tại nút gốc (root), thuật toán sẽ thử chia dữ liệu theo các thuộc tính khác nhau (giới tính hoặc tuổi) và lựa chọn điều kiện phân chia làm sao để *chỉ số Gini được giảm nhiều nhất*.

Kết quả ví dụ cho thấy thuộc tính **Age** được chọn để phân chia ở ngưỡng 22, chứ không phải thuộc tính **Gender**. Nhánh trái (Age ≤ 22) thu về ba mẫu đều thuộc cùng lớp (cả ba đều “Strawberry”), nên Gini của nhánh trái này bằng 0; nhánh phải còn lại gồm ba mẫu với phân bố hỗn hợp (1 apple, 2 orange) và tiếp tục được chia thêm ở dưới.

**Tại sao chọn Age thay vì Gender?** Ta tính Gini của phép chia theo từng thuộc tính. Phân chia theo **Gender** tạo hai nhóm: nhóm *Male* (3 mẫu) có nhãn {Apple, Strawberry, Strawberry} *GiniGender== 0.444*

Nếu phân chia theo **Age**, ta xem xét các ngưỡng tiềm năng (ví dụ giữa 22 và 30 là ngưỡng 26) rồi tính Gini. GiniAge≤22 = 0.222 Đây là giá trị Gini tổng thấp hơn Do vậy, thuật toán chọn phân chia theo **Age ≤ 22** tại nút gốc, vì **độ tinh khiết Gini được giảm mạnh nhất**

**Chọn ngưỡng 22 và 30:** Ở đây các giá trị tuổi có thể: 15, 18, 22, 30, 32, 35. Thuật toán thử các ngưỡng giữa 15-18 (≈16.5), 18-22 (20), 22-30 (26), 30-32 (31), 32-35 (33.5) và tính Gini tổng. Kết quả cho thấy ngưỡng giữa 22 và 30 (26) cho Gini tổng ≈0.222, tốt nhất. Ta có thể chọn ngưỡng là 22 (vì sau 22 là 30) để được cùng kết quả phân chia: nhóm ≤22 có ba mẫu (Strawberry) và nhóm >22 có ba mẫu còn lại.

Tiếp tục với nhánh phải ban đầu (các mẫu có *Age > 22* tức {30, 32, 35} tương ứng các lá 0,1,2), thuật toán lại xét các ngưỡng giữa 30-32 (31) và 32-35 (33.5). Thử ngưỡng *Age ≤ 30* (tức nhóm trái chỉ chứa mẫu 30; nhóm phải chứa 32,35) cho kết quả **nhánh trái một lớp (Apple) và nhánh phải một lớp (Orange)**, nên cả hai nhánh con đều Gini = 0, Gini tổng = 0. Đây là phân chia tốt nhất. (Nếu thử 31 hoặc 33.5, ta vẫn được Gini tổng = 0.25, kém hơn). Vì vậy **Age ≤ 30** được chọn tại nút này. Nhánh trái (Age ≤ 30) chứa mẫu Apple, nhánh phải chứa hai mẫu Orange, cả hai đều đã thuần nhất về nhãn. Mô hình cây kết thúc ở đây với các nút lá tương ứng label Strawberry, Apple, Orange.

### **Phương pháp ngăn chặn overfitting**

* **Overfitting**: Hiện tượng cây quyết định quá phức tạp, học cả nhiễu trong dữ liệu huấn luyện, dẫn đến hiệu suất kém trên dữ liệu mới.
* **Các phương pháp ngăn chặn**:
  + **Pruning (Cắt tỉa)**: Loại bỏ các nhánh không cải thiện hiệu suất hoặc làm tăng lỗi trên tập kiểm tra.
  + **Giới hạn độ sâu tối đa**: Không cho cây phát triển quá sâu.
  + **Số lượng mẫu tối thiểu tại nút lá**: Đảm bảo mỗi nút lá có đủ số lượng mẫu để tránh phân chia quá chi tiết.

### **5. Ưu điểm và nhược điểm**

#### **Ưu điểm:**

* Dễ hiểu và diễn giải, phù hợp để giải thích các quyết định.
* Không cần chuẩn hóa dữ liệu (như chuẩn hóa khoảng giá trị).
* Xử lý được cả dữ liệu số và dữ liệu danh mục.
* Hỗ trợ khám phá dữ liệu, giúp xác định các thuộc tính quan trọng.

#### **Nhược điểm:**

* Dễ bị overfitting nếu không được kiểm soát.
* Không ổn định: Thay đổi nhỏ trong dữ liệu có thể tạo ra cây hoàn toàn khác.
* Hiệu suất kém với các bài toán có mối quan hệ phức tạp giữa các thuộc tính.