Advanced Institute for Artificial Intelligence

# Árvore de Decisão

Advanced Institute for Artificial Intelligence – AI2

https://advancedinstitute.ai

# Introdução

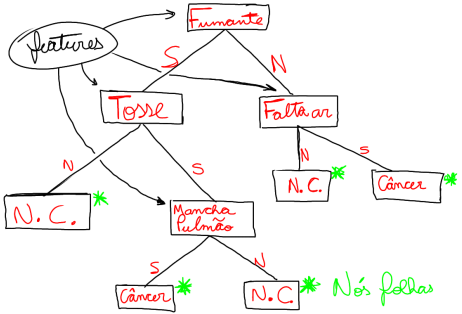
Árvore(s) de Decisão (AD) representa(m) um método de aprendizado baseado em **regras** para a tomada de decisão. Também pode ser chamado de indução de árvore(s) de decisão. É uma abordagem de modelagem preditiva desenvolvida em meados de 1980 com ampla gama de aplicação em estatística, mineração de dados e aprendizado de máquina.

O modelo preditivo baseado em uma árvore de decisão parte das observações dos atributos (representado nos **ramos** da árvore) até as conclusões sobre o valor alvo da amostra em análise (representado nas **folhas**). Nas folhas, o valor alvo pode ser categórico ou numérico, representando problemas de **classificação** ou **regressão**, respectivamente.

Aprender uma AD diz respeito à implementação de um algoritmo **supervisionado**, **não paramétrico**, e baseado em regras, cuja aplicação é uma das mais bem-sucedidas desde sua criação. Seu sucesso se deu devido a várias características atraentes, tais como, simplicidade, **compreensibilidade**, ausência de parâmetros e capacidade de lidar com dados de tipos mistos.

Nosso modelo de AD representa um conjunto de regras que, quando submetidas à observação de uma amostra é capaz de direcionar **hierarquicamente** o caminho a ser seguido na árvore para predizer nossa variável alvo.

Uma Árvore de Decisão é composta por quatro componentes principais, o **nó raiz** (*root*), **nós internos** (intermediários, ou filhos), **nós folhas**, e **ramos**.




Exemplo de Árvore de Decisão.

A principal metodologia para criação, treinamento e classificação com Árvores de Decisão é a CART (*Classification and Regression Trees*). É composta por três etapas:

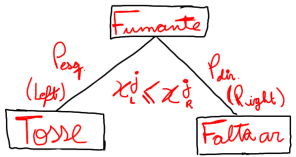
1. Construção da árvore máxima;
2. Escolha do tamanho correto da árvore;
3. Classificação das amostras de teste.

Definição do problema: dado um conjunto de treinamento rotulado com $m$ amostras $\mathcal{X}^1 = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$, queremos classificar corretamente uma amostra $\boldsymbol{x} \in \mathcal{X}^2$.

Objetivo: dada uma amostra $\boldsymbol{x}$ qualquer do conjunto de teste, o seu rótulo $y$ será atribuído de acordo com as decisões em cada nó intermediário da CART, até um nó folha.

A construção da árvore máxima representa a etapa de treinamento (aprendizado). Nela, ocorre um processo de **divisões sucessivas**, de forma recursiva, das variáveis de uma amostra $x_i \in \mathbb{R}^n$, em que o processo só termina quando a amostra $x_i$ é classificada por um nó folha.

A classificação ocorre por meio de uma regra de separação (*splitting rule*), que divide a amostra no nó em duas partes, buscando a **homogeneidade máxima**, exemplo:




Exemplo de Árvore de Decisão.

A homogeneidade máxima de nós filhos é definida por uma **função de impureza** $i(t)$, em que $t$ representa o nó em análise (raiz, filho ou folha).

A impureza do nó raiz ($t_p$) é constante para qualquer divisão $x_L^j \leq x_R^j$, com $j = 1, 2, ..., n$. A máxima homogeneidade dos nós da esquerda e da direita são equivalentes à maximização da função de mudança de impureza:

$$\Delta i(t) = i(t_p) - E[i(t_c)], \tag{1}$$

em que $t_c$ são os nós filhos de $t_p$. Assumindo $P_l$ e $P_r$ as probabilidades da esquerda e da direita, respectivamente, do nó raiz, temos:

$$\Delta i(t) = i(t_p) - P_l i(t_l) - P_r i(t_r). \tag{2}$$

Portanto, cada nó de uma CART resolve a seguinte equação de **maximização**:

$$\underset{x_L^j \leq x_R^j, \forall j \in \mathbb{R}^n}{\arg\max} \quad [i(t_p) - P_l i(t_l) - P_r i(t_r)] \tag{3}$$

A Equação 3 nos diz que a CART vai procurar pelos valores de todas as variáveis $j$ a melhor separação para $x_L^j < x_R^j$ que maximizará a mudança na impureza $\Delta i(t)$.

Resta agora definir uma função de impureza para que a CART possa realizar a medida no aprendizado.

A **impureza de Gini** é uma medida de quantas vezes um elemento escolhido, aleatoriamente, do conjunto $\mathcal{X}^1$ seria rotulado incorretamente se fosse classificado aleatoriamente de acordo com a distribuição de rótulos no subconjunto.

Pode ser calculada somando a probabilidade $p_c$ de uma amostra com classe $c$ $(c = 1, ..., C)$ vezes a probabilidade de um erro $\sum_{k \neq c} p_k$. Além disso, $\sum_{k \neq c} p_k = 1 - p_c$, e portanto, temos:

$$IG(p) = i(t) = \sum_{c=1}^{C} \left( p_c \sum_{k \neq l} p_k \right), \tag{4}$$

e então:

$$i(t) = 1 - \sum_{c=1}^{C} p_c^2. \tag{5}$$

Portanto, os nós serão ranqueados de acordo com a impureza associada a cada variável $x^j$, e substituindo a Equação 5 na Equação 2, temos:

$$\Delta i(t) = 1 - \sum_{c=1}^{C} p_c^2 - P_l i(t_l) - P_r i(t_r) \tag{6}$$

Também podemos utilizar outras métricas, como a **entropia**, para analisar o relacionamento entre as variáveis $x^j$ e a variável alvo (classe), e com isso construir árvores sob metodologias diferentes da CART, como por exemplo, a "C4.5".

A C4.5 leva em consideração o **Ganho de Informação** (GI), utilizando a entropia para a tarefa de classificação. O GI baseia-se na minimização da informação requerida para classificar dados de partições, e reflete a impureza (ou a aleatoriedade) das partições.

A entropia é uma grandeza termodinâmica que mede o grau de **irreversibilidade** de um sistema. É comumente associada ao grau de desordem.

Além disso, a entropia pode ser utilizada para medir o grau de incerteza de uma informação, em que processos com alta entropia nos dizem sobre uma elevada incerteza a respeito da informação que está "circulando"pelo sistema.

Exemplos: xícara de café esfriando; copo quebrado ao cair.

A entropia é definida pela seguinte equação:

$$H(X) = -\sum_{c=1}^{C} p(x_c) \log_2 p(x_c). \tag{7}$$

Cujo gráfico tem forma parabólica.

O ganho de informação pode ser definido como:

$$GI(X) = H(X) - \underbrace{\underbrace{\sum_{a=1}^{m} \frac{\#A_a}{\#A}}_{\text{peso}} * \sum_{c=1}^{C} p(x_c^j) \log_2 p(x_c^j)}_{\text{H(X) utilizando atributo } j}, \tag{8}$$

$H(X)$ representa a entropia para o conjunto de dados $X$ de acordo com a divisão dos nós.

$\#A$ representa a quantidade de dados na divisão utilizando o atributo $j$.

$\#A_a$ representa a contagem de amostras com "valor"$a$, ou seja, todos os valores que a variável $j$ em questão possui nos dados de treinamento.

Em **resumo**:

Os parâmetros de uma AD são: os melhores atributos (variáveis) e o valor dos melhores atributos no melhor local de divisão - o valor no local de divisão que fornece o máximo ganho de informação. As folhas de um nó são os subdomínios.

A seguir, veremos os passos do processo de treinamento.

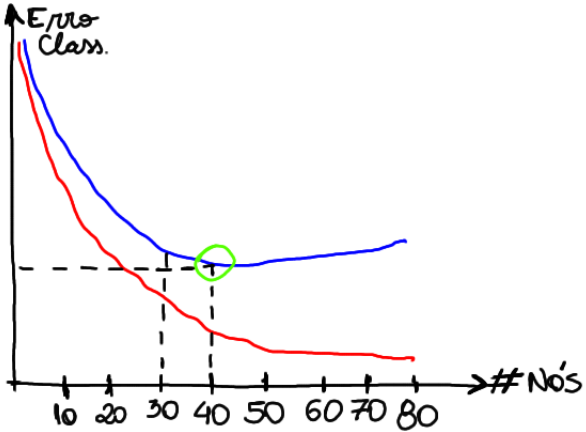
1. Assumindo que os dados são $n$-dimensionais, analise todos os $n$, subespaços unidimensionais (ou seja, uma característica por vez) para procurar as melhores características e dividir as localizações. Se $n$ for alto, então a busca será exaustiva e o processo será computacionalmente caro;
2. Em cada subespaço unidimensional, seu domínio de dados será dividido em cada local (ou seja, $n-2$ locais, se houver $m$ observações) e o ganho de informação resultante da divisão em dois subdomínios será calculado;
3. Para cada atributo, o melhor local de divisão é selecionado com base na divisão que fornece o maior ganho de informação e os valores do atributo nesse local serão registrados;
4. O melhor atributo e o valor do atributo no local da divisão serão atribuídos ao nó que está sendo processado;
5. O domínio de dados é então dividido em dois subdomínios ("SD1" e "SD2") nos locais de divisão;

6. As etapas de 1 a 5 serão repetidas para os subdomínios “SD1” e “SD2’ para obter o melhor atributo, o melhor local de divisão e o melhor valor de divisão do atributo para os novos nós;
7. A árvore de decisão será construída seguindo esses processos até o subdomínio que não precisa de divisão devido ao subdomínio que possui um número significativamente grande de observações de uma única classe.

Os problemas encontrados em um algoritmo recursivo são: obter estimativas confiáveis do erro usado para selecionar as partições, otimizar o erro no conjunto de teste e minimizar o erro no treinamento e a dimensão da árvore.

As AD também sofrem de *overfitting*, com certa facilidade, por conta do algoritmo recursivo de particionamento que gera uma alta dependência entre os atributos para uma correta classificação, visto que são criadas em sua máxima profundidade.

Nas ADs, o *overfitting* é tratado com métodos de poda (*pruning*).




Exemplo de *Overfitting*.

A podagem pode ser realizada de duas formas. Pode ser usada para parar o crescimento da árvore precocemente, chamada de pré-podagem (ou poda descendente), ou pode acontecer com a árvore já completa, chamada de pós-podagem (ou poda ascendente). O processo é descrito como segue:

1. Percorre a árvore em profundidade;
2. Para cada nó de decisão calcule:
   - erro no nó;
   - soma dos erros nos nós descendentes;
3. Se o erro do nó for menor ou igual à soma dos erros dos nós descendentes então o nó é transformado em folha.

Na pós-podagem a árvore é gerada no tamanho máximo e então a árvore é podada aplicando métodos de evolução confiáveis. A sub-árvore com o melhor desempenho será a escolhida.

Este processo pode ser computacionalmente ineficiente pelo fato de gerar uma árvore muito grande e depois esta mesma árvore é reduzida a uma árvore mínima.

Para interromper o crescimento da árvore, verifica-se se a divisão é confiável ou não. Caso seja confiável, interrompe o crescimento da árvore.

Este processo é conhecido como pré-podagem da árvore. A pré-podagem é mais rápida, porém, menos eficiente que a pós-podagem, uma vez que existe o risco de interromper o crescimento da árvore e selecionarmos uma árvore sub-ótima.

Dentre os métodos de poda existentes, destacam-se:

- *Cost Complexity Pruning*;
- *Reduced Error Pruning*;
- *Minimum Error Pruning (MEP)*;
- *Pessimistic Pruning*;
- *Error-Based Pruning (EBP)*;
- *Pessimistic Pruning*;
- *Minimum Description Length (MDL) Pruning*;
- *Minimum Message Length (MML) Pruning*;
- *Critical Value Pruning (CVP)*, etc.

Algumas vantagens das árvores de decisão:

- Simples de entender e interpretar (podem ser visualizadas);
- Requer pouco pré-processamento dos dados. Outras técnicas geralmente requerem normalização, ou variáveis latentes precisam ser criadas, e valores em branco devem ser removidos;
- O custo de usar uma AD para predição é logarítmico no número de amostras usadas para treinar a árvore;
- Capaz de lidar com dados numéricos e categóricos;
- Possível validar um modelo usando testes estatísticos. Isso possibilita a explicação sobre a confiabilidade do modelo.

As desvantagens das árvores de decisão incluem:

- O treinamento pode criar árvores supercomplexas que não generalizam bem os dados de teste;
- ADs podem ser instáveis, pois pequenas variações nos dados podem resultar na geração de uma árvore completamente diferente. Esse problema é mitigado usando árvores de decisão dentro de um *ensemble*;
- ADs não são muito eficientes para extrapolação (regressão a valores contínuos);
- Existem conceitos difíceis de aprender, pois as ADs não os expressam facilmente, como problemas XOR.
- O problema de aprender uma árvore de decisão ótima é conhecido por ser NP-completo.