U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Professora Inês Dutra

# Inteligência Artificial

## Terceiro Trabalho: Árvores de Decisão

Abril de 2022

Trabalho realizado por:

João Marrucho - 201804960

Bruno Dias - 201907828

Pedro Leite - 201906697

# Índice

# 1. Introdução

Neste trabalho iremos aprofundar o nosso conhecimento acerca das árvores de decisão. São excelentes ferramentas para nos ajudar a escolher entre várias escolhas. Fornecem uma estrutura altamente eficaz dentro da qual podemos definir opções e investigar os possíveis resultados da escolha dessas opções. Expõem claramente o problema para que todas as opções possam ser desafiadas, permite-nos analisar completamente as possíveis consequências de cada decisão e fornecem uma estrutura para quantificar os valores dos resultados e as probabilidades de alcançá-los. O nosso objetivo será explorarmos as árvores de decisão procurando aplicá-las em certos problemas e reconhecer quais são as suas vantagens e desvantagens.

# 2. Árvores de Decisão

## 2.1. Explicação do que são Árvores de Decisão

As árvore de decisão, ou *decision trees*, estão entre os métodos mais comuns aplicados no *machine learning*. Tais algoritmos subdividem progressivamente os dados em conjuntos cada vez menores e mais específicos, em termos dos seus atributos, até atingirem um tamanho simplificado o bastante para ser rotulado. Para isso é necessário treinar o modelo com dados previamente rotulados, de modo a aplicá-lo a dados novos. São usadas pela sua facilidade de compreensão, podem ser úteis com ou sem dados concretos, novas opções podem ser adicionadas às arvores existentes, entre outras vantagens.

São usadas para classificação ou regressão ou quando há um problema de diversos rótulos. Ou seja, quando as categorias de classificação são múltiplas, e não apenas duas.

## 2.2. Algoritmos de Árvores de Decisão

As árvores de decisão usam vários algoritmos para decidir dividir um nó em dois ou mais sub-nós. A criação de sub-nós aumenta a homogeneidade dos sub-nós resultantes. A árvore de decisão divide os nós em todas as variáveis disponíveis e então seleciona a divisão que resulta em sub-nós mais homogéneos. A seleção do algoritmo também é baseada no tipo de variáveis de destino. Alguns algoritmos usados em Árvores de decisão são: ID3, C4.5 (sucessor do ID3), CART (árvore

de classificação e regressão), CHAID (deteção automática de interação qui-quadrado que realiza divisões em vários níveis ao calcular árvores de classificação) e MARS (*splines* de regressão adaptava multi-variada). O algoritmo usado neste trabalho será o ID3.

# 4. Linguagem, Estrutura de Dados e Funções Auxiliares

## 4.1. Linguagem

A linguagem que decidimos utilizar para este trabalho foi **Python**. Já que trabalhamos com esta linguagem para os projetos anteriores, e achamos mais fácil de utilizar do que as suas concorrentes (C++, Java, etc), devido à sua sintaxe simples e intuitiva e vastos recursos disponíveis online.

## 4.2. Estrutura de Dados

Para armazenar os conteúdos dos ficheiros CSV, decidimos utilizar **Matrizes.**

Definimos duas variáveis, uma com o número de linhas e outra com o número de colunas, que vão definir o tamanho da matriz. E guardamos todos os caracteres numa célula até chegar à vírgula (nova coluna) ou quando chega ao carácter de mudança de linha (nova linha e o número da coluna volta a 0).

## 4.3. Funções Auxiliares

Criamos uma função "**atribute_to_matrix**", que retorna um array com o número de ocorrências de todos as variáveis e o número de variáveis diferentes. Esta função vai-nos ajudar no cálculo da entropia.

Temos uma função chamada "**entropy**", que recebe uma coluna do ficheiro e calcula a sua entropia, de acordo com esta fórmula:

$$\text{Entropy:} \quad H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k).$$

Onde "k" representa o número de variáveis diferentes. O "p" representa o número de vezes que uma variável aparece a dividir pelo "k". Fazemos portanto um loop com tamanho "k", que faz a equação "p*log(p)", para todos os "p's"

da coluna. Os resultados dessas equações são somados e colocados na negativa, resultando assim na entropia final.

Criamos outra função chamada "**importance**", que tem como argumentos dois vetores que dizem, quais colunas e linhas vão ser utilizadas na matriz com o ficheiro e a própria matriz. Esta função calcula a entropia para todas as colunas válidas e retorna o atributo com a menor entropia.

A função "**plur_values**", que tem como argumentos um vetor com as colunas válidas e a matriz. Retorna "*false*" se tiver mais valores negativos que positivos e "*true*" se tiver mais valores positivos que negativos, se a quantidade de valores positivos e negativos for igual, retorna aleatoriamente "*false*" ou "*true*".

Criamos outra função, chamada "**allsame**", que também recebe um vetor com as colunas válidas e a matriz. E verifica se a última coluna da matriz tem sempre o mesmo valor, se sim e se for "Yes" retorna 1, se sim e for "No" retorna 0, se não retorna 2.

A função "**dif_values**", que recebe uma coluna e a matriz. Esta função retorna um vetor com todos os diferentes valores dessa coluna.

A "**atribs_empty**" é uma função, que verifica se há mais algum atributo disponível na matriz.

Também criamos a "**atrib_examples**", que recebe uma matriz, um vetor com as colunas válidas, uma coluna e um valor da coluna. E devolve um vetor com todos os valores das colunas válidas que têm esse valor.

# 5. ID3

## 5.1. Explicação do ID3

O algoritmo ID3 constrói árvores de decisão usando uma abordagem *greedy-search* de cima para baixo. Começa com o conjunto S como nó raiz, em cada iteração do algoritmo, itera através do atributo menos utilizado do conjunto S e calcula a entropia desse atributo. De seguida, seleciona o atributo que tem menor entropia e maior ganho de informação. O conjunto S é então dividido pelo atributo selecionado para produzir um subconjunto de dados. O algoritmo continua a repetir-se em cada subconjunto, considerando apenas atributos nunca selecionados antes.

A entropia é uma medida de aleatoriedade na informação que está a ser processada. Quanto maior a entropia, mais difícil é tirar conclusões a partir dessa informação. A entropia é zero quando a probabilidade é de 0 ou 1 e é máxima quando a probabilidade é 0.5 porque projeta aleatoriedade perfeita nos dados.

## 5.2. Implementação do ID3

A função "**newid3**", recebe como argumentos: uma matriz, um vetor *examples* que define que colunas são válidas, um vetor *atribs* que define que linhas são válidas e um vetor *parent_examples* que corresponde ao *examples* do nó pai.

O algoritmo verifica sempre se o nó corrente é uma folha ou não, entrando para isso numa series de "*if's*".

Caso o tamanho do *examples* seja 0 ou se não houver nenhum atributo disponível na matriz, significa que chegamos ao nó folha e retornamos um valor booleano, que é "*False*" se tiver mais valores negativos que positivos e "*True*" se tiver mais valores positivos que negativos, se a quantidade de valores positivos e negativos for igual, retorna aleatoriamente "*False*" ou "*True*", é calculada com a função "plur_values" explicada anteriormente no ponto 4.3).

Se todos os valores do *examples* forem iguais, retorna "*True*" se esses valores forem positivos e "*False*" se esses valores forem negativos (que é calculada com a função "allsame" também explicada anteriormente no ponto 4.3).

Caso não entre em nenhum dos "*if's*" então é porque não é uma folha e começa o processo de desenvolver uma sub-árvore para ramificação possível do nó a verificar. Definimos uma variável com o argumento de maior importância (que é calculada com a função "importance" explicada anteriormente no ponto 4.3) e inicializamos o nó com essa variável com a importância, chamado root (raíz da sub-árvore a ser criada). Definimos um vetor com todos os valores diferentes do atributo da importância.

Entramos num loop que itera entre cada valor possível desse atributo, onde começamos por definir um vetor com todos os valores das colunas válidas que têm o valor "i" ao usar a função "atrib_examples". Depois retiramos o atributo em questão dos atributos a verificar no futuro, pois este agora já faz parte da árvore. O algoritmo volta a ser chamado de maneira recursiva acabando assim

por desenvolver a árvore para cada nó resultante do atributo principal. Quando tivermos o resultado esse nó e aresta são colocados num array, esse array fica guardado na root, que é o que é retornado quando chegamos ao final.

## 6. Conclusão

O ID3 demonstrou ser de baixo custo e rápido na classificação de *datasets* de pequeno tamanho, mas lento na classificação de *datasets* de grande tamanho. Os dados podem estar demasiado ajustados ou super classificados, se uma amostra pequena for testada. Apenas um atributo de cada vez é testado para tomar uma decisão, classificar dados contínuos pode ter um grande custo computacional, pois muitas árvores devem ser geradas até chegarmos ao nó final. Outra desvantagem do algoritmo é que por ser do tipo *greedy,* acaba por construir uma árvore de decisão pequena, mas nunca chega à árvore pequena possível.

## 7. Referências Bibliográficas

S. Russell, P. Norvig; Artificial Intelligence: A Modern Approach, 3rd ed, Prentice Hall, 2009

Slides da unidade curricular