Árvores de decisão

Relatório

João Vivas - número 202108177

Gelson Stalino Varela – número 202109347

Tabela de conteúdos

O que é uma árvore de decisão?

O tema deste trabalho é árvores de decisão. Elas representam uma função que liga um conjunto de atributos a um resultado (uma “decisão”). Para conseguirem fazer uma decisão elas fazem um conjunto de testes que começa na raiz e que vai avançando na árvore até que chegue a uma folha. Cada nó corresponde a um atributo do input e os ramos correspondem aos diferentes resultados do atributo e as folhas correspondem ao resultado da função.

As árvores de decisão são uma forma de prever possíveis consequências, resultados ou custos sobre cenários específicos. Para construir árvores de decisão existem vários algoritmos, mas o principal é o ID3.

Durante o processo de construção da árvore, o ID3 precisa selecionar os atributos mais relevantes, ou seja, aqueles que fornecerão informações mais úteis para tomar as decisões corretas. Existem várias métricas que podem ser usadas pelo ID3 para selecionar esses atributos. Duas das métricas mais comuns são a Entropia e o Ganho de Informação. [ID300][ID301]

1 Ganho de Informação: Essa métrica mede o quanto de informação é ganho ao dividir os dados com base em um determinado atributo.Em outras palavras, quanta incerteza foi reduzida após a divisão definida no atributo. O ID3 utiliza a entropia como medida de incerteza inicial e calcula o ganho de informação como a diferença entre a entropia antes e depois da divisão. Quanto maior o ganho, mais importante é o atributo para a classificação.

2 Índice de Gini: Essa métrica avalia a impureza dos dados ao dividir os exemplos em diferentes classes. Quanto menor o índice de Gini, maior é a pureza da divisão. Essa métrica é útil para problemas de classificação binária, onde se busca minimizar a mistura de diferentes classes nos subconjuntos divididos.

Algoritmo usado

O algoritmo usado no nosso trabalho foi seguido o algoritmo usado no livro “Artificial Intelligence a Modern Approach” [AI00], onde usa uma estratégia greedy dividir para conquistar, onde primeiro escolhe o atributo com mais importância e resolve-o e depois vai recursivamente, escolher os outros atributos por ordem decrescente de importância. O atributo mais importante é aquele que permite chegar às repostas corretas com o menor número de testes. Para escolhermos o atributo com mais importância nós usamos o conceito de entropia que diz que quanto menor a entropia maior o ganho de informação, ou seja, quanto menor a entropia mais informação conseguimos obter de uma variável, logo se torna mais importante.

Implementação

Este projeto foi feito em Python e está dividido em quatro ficheiros ***main.py***, ***util.py***, ***tree.py***, ***decisiontree.py***. O ficheiro executável é o ficheiro ***main.py***. Este ficheiro começa por chamar ler o ficheiro de input com a função *readInput* do ficheiro *util.py*.

Esta função começa por executar uma função auxiliar *read_input* que vai ser a que vai abrir o ficheiro e ler o conteúdo para ser processado na árvore. Primeiro pede que o utilizador insira o nome do ficheiro de input. O programa tenta então abrir o programa com o nome que o utilizador inseriu e lê a primeira linha que vai ter os nomes dos atributos e coloca-os numa lista chamada *attribs*. Como o primeiro atributo normalmente é um ID que não é necessário então descarta-se esse atributo e como o último é a classificação do resultado do exemplo colocamos o nome do último atributo numa variável *classification*. Para cada atributo em *attribs* é colocado como chave num dicionário *attributes* com um conjunto de elementos que por agora vai estar vazio. A função lê então as restantes linhas uma a uma, sempre ignorando a primeira coluna que vai ser para o ID que não é necessário para a árvore. Para cada linha é criado um dicionário chamado *example* que vai guardar os dados do exemplo cuja chave é o atributo. Cada valor na linha é colocado como elemento no dicionário *example* e no dicionário *attributes* e a chave nestes dois dicionários é o atributo correspondente, quando for o último elemento da linha, ou seja, a classificação do exemplo também se coloca a frequência do resultado no dicionário *output_values_frequency* sendo a chave a classificação e o valor da chava a frequência. Quando já se leu o ficheiro todo é retornado quatro variáveis *examples* que vai ser uma lista de dicionários com os dados de cada linha, *attributes* que vai ser um dicionário com todos os resultados presentes no ficheiro de conjunto de treino para cada atributo, *total_examples_in_dataset* o número total de exemplos no ficheiro e *probability_list* que é uma lista das probabilidades das possíveis classificações que um exemplo tem.

Com este dados então para diminuir o tamanho da árvore de decisão na função *readInput* vai discretizar os valores numéricos. Vai primeiro chamar *define_intervals* que vai criar o número de intervalos necessários usando a **fórmula de Sturge** [AI01] e definir quais os valores de cada intervalo. Depois chama *discretize_variable*. Que apenas insere a variável no intervalo correspondente. E depois os atributos numéricos em vez de ter vários valores diferentes vão ter apenas intervalos numéricos. E assim já temos todos os dados para criar a árvore de decisão de forma adequada.

Então agora o programa cria a árvore de decisão com os dados do input usando a estrutura DecisionTree definida em *decisiontree.py*. A estrutura de dados da árvore de decisão vai guardar a árvore que de início vai estar vazia, a classificação que, por defeito, não vai ter qualquer valor, *examples* que guarda os exemplos do ficheiro do conjunto de treino, *attributes* que guarda os atributos do conjunto de

treino, *total_examples_in_dataset* que guarda o número de exemplos no conjunto de treino, a lista de probabilidades de um exemplo tem de ter um certo resultado e a entropia da classificação.

A seguir é executado a função *learn_decision_tree* que vai chamar uma função auxiliar *learnDecisionTree* que recebe como argumentos os exemplos do conjunto de treino, os atributos e o *parent_examples* que de início não vai ter nada. As primeiras três linhas desta função são casos base. Na primeira condição se não haver mais exemplos retorna-se uma árvore com o output mais comum do *parent_example*. A segunda condição verifica se todos os exemplos têm a mesma classificação, se tiverem retorna uma árvore com essa classificação. Na terceira condição diz que se não tivermos mais atributos, é retornado uma árvore com a classificação mais comum do conjunto de exemplos. Se nenhuma destas condições forem cumpridas selecionamos o atributo com maior importância, ou seja, o atributo com maior ganho de informação que é o resultado de aplicar a função *importance* sobre cada um. Criamos então uma árvore com o nome do atributo mais importante. Vamos então pegar todos os valores possíveis que esse atributo pode ter e eliminar a entrada do atributo mais importante do dicionário *attributes*, porque já não vai ser mais necessário. Então para cada valor do atributo mais importante vamos criar várias sub-árvores com os restantes exemplos, separando-os pelo valor que têm no atributo mais importante escolhido. E continuamos a executar de forma recursiva até atingirmos um dos casos base. Quando chegarmos ao fim, ligamos todas as sub-árvores com um ramo com o nome do atributo mais importante do conjunto de exemplos anterior.

Depois imprimimos a árvore no formato especificado no enunciado. E o programa pede para inserir um ficheiro para testar a árvore.

Resultados

Estes foram as árvores que o nosso programa gerou para os diferentes *data sets* que foram fornecidos. O counter de cada ramo é o número que está entre parêntesis à direita dos resultados:

```
<Pat>
        Full:
        <Hun>
                No: No (2)
                Yes:
                        <Type>
                                Italian: No (1)
                                French: No (0)
                                Burger: Yes (1)
                                Thai:
                                        <Fri>
                                                No: No (1)
                                                Yes: Yes (1)
        Some: Yes (4)
        None: No (2)
```

Árvore de decisão 1: restaurant data set

```
<Weather>
        overcast: yes (4)
        rainy:
        <Windy>
                FALSE: yes (3)
                TRUE: no (2)
        sunny:
        <Temp>
                64.0,69.25: yes (1)
                74.5,79.75: yes (1)
                69.25,74.5: no (1)
                79.75,85.0: no (2)
```

Árvore de decisão 2: Weather data set

```
<petalwidth>
        1.6,1.9:
        <sepallength>
                7.0,7.45: Iris-virginica (3)
                4.3,4.75: Iris-virginica (0)
                4.75,5.2: Iris-virginica (1)
                7.45,7.9: Iris-virginica (0)
                6.1,6.55: Iris-virginica (7)
                5.2,5.65: Iris-virginica (0)
                5.65,6.1:
                        <sepalwidth>
                                3.2,3.5: Iris-virginica (0)
                                2.9,3.2:
                                        <petallength>
                                                3.95,4.688: Iris-virginica (0)
                                                3.213,3.95: Iris-virginica (0)
                                                2.475,3.213: Iris-virginica (0)
                                                1.738,2.475: Iris-virginica (0)
                                                5.425,6.163: Iris-virginica (0)
                                                4.688,5.425: Iris-virginica (3)
                                                1.0,1.738: Iris-virginica (0)
                                                6.163,6.9: Iris-virginica (0)
                                2.0,2.3: Iris-virginica (0)
                                2.3,2.6: Iris-virginica (0)
                                3.5,3.8: Iris-virginica (0)
                                4.1,4.4: Iris-virginica (0)
                                3.8,4.1: Iris-virginica (0)
                                2.6,2.9: Iris-virginica (2)
                6.55,7.0: Iris-versicolor (1)
        0.4,0.7: Iris-setosa (2)
        0.7,1.0: Iris-versicolor (7)
        1.9,2.2: Iris-virginica (15)
        1.0,1.3: Iris-versicolor (21)
        2.2,2.5: Iris-virginica (14)
        1.3,1.6: Iris-versicolor (20)
        0.1,0.4: Iris-setosa (48)
```

Árvore de decisão 3: Iris data set

Comentários Finais

Com este trabalho pudemos observar como se constroem árvores de decisão, bem como o seu funcionamento para prever resultados. Também se percebeu a importância que certos atributos têm em relação a outros que permitindo que se a uma decisão mais precisa com base em modelos pré estabelecidos.

Bibliografia

ID300: Augusto Oeiras, Compreendendo algoritmos de Heurísticas ID3, 2020,
ID301: ID3 algorithm
AI00: Stuart Russel, Peter Norvig, Artificial Intelligence A Modern Approach, 2022
AI01: Rodrigo Silva, Distribuição de Frequência - Agrupamento em Intervalos Regulares, 2018