Decision Trees

Francisco Ribeiro, Matheus Bissacot, Sérgio Coelho

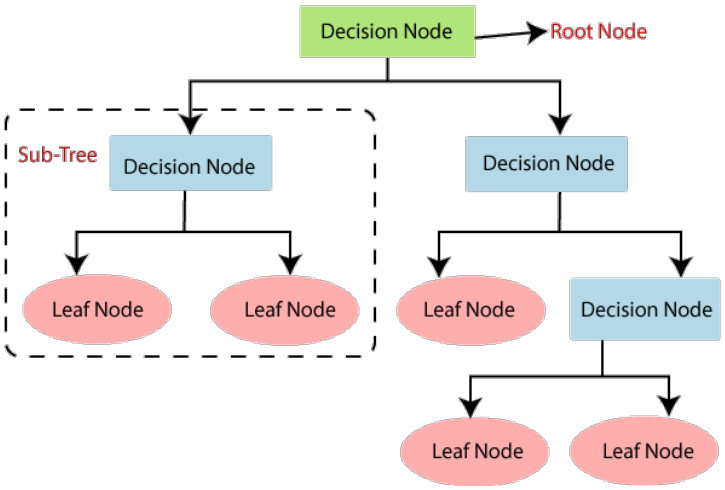
Universidade do Porto

Contents

1 Introdução

1.1 O que é uma árvore de decisão?

Uma árvore de decisão é um modelo hierárquico de suporte à decisão que utiliza uma estrutura em forma de árvore para representar decisões e as suas possíveis consequências, incluindo resultados de eventos aleatórios, custos de recursos e a sua utilidade. É uma forma de apresentar um algoritmo que contém apenas declarações de controlo condicionais.



As árvores de decisão são frequentemente utilizadas na análise de decisões, para ajudar a identificar uma estratégia com maior probabilidade de atingir um determinado objetivo, mas também são uma ferramenta popular em machine learning.

1.2 Qual é a utilidade?

A árvore de decisão é utilizada para construir modelos de classificação e regressão. É utilizada (também) para criar modelos de dados que preveem rótulos de classes ou valores para o processo de tomada de decisão. Os modelos são construídos a partir dos conjuntos de dados de treino fornecidos ao sistema (supervised learning).

Ao utilizar uma árvore de decisão, podemos visualizar as decisões de forma a torná-las mais compreensíveis, o que torna esta técnica bastante popular na área da mineração de dados.

2 Algoritmos de árvores de decisão

2.1 ID3 Iterative Dichotomiser 3

Foi um dos primeiros algoritmos de árvores de decisão e é um dos mais utilizados, inventado por Ross Quinlan, onde gera uma árvore de pesquisa através de um conjunto de dados. Usa uma abordagem com princípios greedy, selecionando o melhor atributo para dividir os dados em cada etapa, e compactar a árvore. É normalmente usado em machine learning e processamento de linguagem natural. O algoritmo segue estes passos:

- primeiro começa com todos os exemplos de treino;
- de seguida, escolhe o teste que melhor divide os exemplos, da mesma classe, ou similares;
- depois de escolhê-lo, cria um nó filho para cada valor possível;
- transporta os exemplos para cada nó filho, tendo em conta o valor do filho;
- repete recursivamente o processo para cada subconjunto criado.

2.2 Algoritmo C4.5

O algoritmo C4.5, inventado também por Ross Quinlan, é uma extensão/aprimoramento do algoritmo ID3, trabalhando com valores indisponíveis e contínuos. Quando existe um atributo sem valor, não é usado para o cálculo da métrica. Quando acaba de construir a árvore, tenta podar os seus ramos não necessários, substituindo por nós folha. Este algoritmo obtém árvores mais simples. As árvores de decisão geradas são conhecidas como classificadores estatísticos porque estas podem ser usadas para classificação.

2.3 CART (Classification and Regression Trees)

CART é um algoritmo versátil que pode ser usado para tarefas de classificação e regressão. Constrói árvores de decisão binárias repartindo recursivamente os dados com base no recurso que fornece a melhor divisão de acordo com um critério específico. O algoritmo segue estes passos:

- primeiro escolhe os valores das variáveis de maneira a que fique a melhor divisão possível;
- de seguida, aplica a divisão aos respetivos filhos;
- quando o algoritmo detetar que não há mais benefício em continuar a dividir, termina.

2.4 Conclusão

Estes são alguns dos algoritmos de árvores de decisão mais conhecido.

Cada algoritmo tem as suas vantagens e desvantagens, e a escolha do algoritmo a usar deve ser feita dependedo do problema em mão (e das características do dataset).

What is the Difference Between ID3 and C4.5 ALGORITHM?

Algorithm	Splitting Criteria of algorithm	Attribute types Managed by algorithm	Pruning Strategy of algorithm	Outlier Detection	Missing values	Invented By
C4.5	Gain Ratio	Manages both Categorical and Numeric value	Error Based pruning is used	Error Based pruning is used	Manages missing values.	developed by Ross Quinlan
ID3	Information Gain	Manages only Categorical value	No pruning is done	No pruning is done	Do not Manages missing values.	invented by Ross Quinlan
CART	Towing Criteria	Manages both Categorical and Numeric value	Cost-Complexity pruning is used	Cost-Complexity pruning is used	Manages missing values.	first published by Leo Breiman in 1984

Read More about C4.5 and ID3

Figure 1: Tabela de Comparação dos algoritmos

3 Implementação

3.1 Linguagem

A linguagem que utilizamos foi Python 3 porque é uma linguagem de programação de alto nível que possui uma sintaxe limpa e legível.

Além disso, Python é uma linguagem versátil, com uma ampla gama de bibliotecas disponíveis. No caso específico da implementação do algoritmo ID3, o uso das bibliotecas padrão do Python, como csv, math e argparse, simplificou a manipulação dos dados e a interação com o usuário através de argumentos de linha de comando.

3.2 organização

O código foi dividido em diferentes arquivos.

- **main.py** é responsável pela execução principal do programa
- **tree.py** contém as funções relacionadas ao algoritmo ID3 e à manipulação da árvore de decisão

Além disso, no código os nomes de variáveis e funções são descritivos. Isso contribui para a legibilidade e compreensão do código.

As seguintes funções desempenham um papel importante na implementação do algoritmo ID3 e na manipulação da árvore resultante, estas são responsáveis por cálculos, seleção do melhor atributo, construção da árvore de decisão e impressão da árvore.

- **calculate_entropy(data)**: Essa função calcula a entropia de um conjunto de dados, com base na distribuição das classes presentes nos dados.
- **is_numerical(attribute_values)**: Essa função verifica se os valores de um atributo são numéricos ou não, retornando `True` caso sejam numéricos e `False` caso contrário.
- **select_best_attribute(examples, attributes, target_attribute)**: Essa função seleciona o melhor atributo para fazer a divisão em um determinado conjunto de exemplos, com base no ganho de informação. Ela considera se o atributo é numérico ou categórico e retorna o atributo selecionado.
- **ID3(examples, attributes, target_attribute)**: Essa função implementa o algoritmo ID3 para construir uma árvore de decisão. Ela recebe um conjunto de exemplos, os atributos disponíveis e o atributo alvo, e retorna a árvore de decisão construída.
- **print_decision_tree(tree, attribute_indentation=")** : Essa função imprime a árvore de decisão em um formato legível. Ela recebe a árvore de decisão e um parâmetro opcional para controlar a indentação dos atributos na impressão da árvore.

3.3 Requisitos

Para ser possível correr o programa é necessário ter python3 instalado.

- Python3

Para instalar Ubuntu/Debian correr o seguinte comando:

```
$ sudo apt-get install python3
```

3.4 Execução

Existe argumentos para a utilização do programa, para ajudar usamos –help:

```
$ python3 main.py -h
```

```
usage: main.py [-h] [-d DATA] [-p]

Decision Trees ID3, please use the following arguments

options:
  -h, --help            show this help message and exit
  -d DATA, --Data DATA  Data to analyze from .csv table.
  -p, --print           Print the tree.
```

Com isto para imprimir a árvore de um ficheiro corre-se o seguinte comando:

```
$ python3 main.py -d ficheiro.csv -p
```

4 Estruturas de dados

Estruturas de Dados Utilizadas:

4.1 Dicionários

4.1.1 "atributos"

É utilizado para associar a cada título de coluna do arquivo CSV um índice correspondente. O dicionário ‘atributos‘ é definido como ‘‘ e é preenchido no loop ‘for‘ com um índice para cada título de coluna do arquivo. Por exemplo, se o arquivo CSV tiver os títulos "title1", "title2" e "title3", o dicionário ‘atributos‘ será preenchido como ‘’title1’: 0, ’title2’: 1, ’title3’: 2‘. O dicionário ‘atributos‘ é usado posteriormente para acessar os índices das colunas dos dados.

4.1.2 "tree"

É utilizado para representar a árvore de decisão construída pelo algoritmo ID3.

O dicionário ‘tree‘ é preenchido recursivamente durante a construção da árvore, onde cada chave do dicionário representa um atributo e o valor correspondente é outro dicionário ou um tuplo.

- Se o valor for um dicionário, significa que o atributo é categórico e possui subárvores associadas a cada valor possível desse atributo.
- Se o valor for um tuplo, significa que o atributo é numérico e contém um ponto de divisão da árvore. A estrutura do dicionário ‘tree‘ segue o formato ‘atributo: subárvore‘.

4.2 Lista de listas

4.2.1 ”data”

É utilizada para armazenar os dados do arquivo CSV.

Cada linha do arquivo é representada como uma lista, e todas as linhas são armazenadas em uma lista maior. Por exemplo, se os dados do arquivo CSV forem ‘[[arg1, arg2, arg3], [arg1, arg2, arg3], ...]‘, a lista ‘data‘ será preenchida da mesma forma. Cada elemento da lista ‘data‘ é uma lista contendo os valores de cada coluna para uma determinada linha do arquivo CSV.

4.3 Counter (‘label_counts‘, ‘majority_label_count‘)

É utilizado para contar as ocorrências das classes na lista ‘class_labels‘.

A classe ‘Counter‘ é uma subclasse de ‘dict‘ que permite contar facilmente as ocorrências de elementos em uma lista. Os contadores são usados para calcular a entropia e contar as classes na construção da árvore de decisão.

5 Resultados

Obtemos os seguintes resultados para as tabelas .csv do enunciado:

5.1 Iris.csv

```
('petallength', 1.9):
        <= 1.9 (Iris-setosa)
                Iris-setosa: 50
        > 1.9:
                ('petalwidth', 1.7):
                        <= 1.7:
                                ('petallength', 4.9):
                                        <= 4.9:
                                                ('petalwidth', 1.6):
                                                        <= 1.6 (Iris-versicolor)
                                                                Iris-versicolor: 47
                                                        > 1.6 (Iris-virginica)
```

```
                                                        Iris-virginica: 1
                                        > 4.9:
                                                ('petalwidth', 1.5):
                                                        <= 1.5 (Iris-virginica)
                                                                Iris-virginica: 3
                                                        > 1.5:
                                                                ('sepallength', 6.95):
                                                                        <= 6.95 (Iris-versicolor)
                                                                                Iris-versicolor: 2
                                                                        > 6.95 (Iris-virginica)
                                                                                Iris-virginica: 1
                        > 1.7:
                                ('petallength', 4.8):
                                        <= 4.8:
                                                ('sepallength', 5.95):
                                                        <= 5.95 (Iris-versicolor)
                                                                Iris-versicolor: 1
                                                        > 5.95 (Iris-virginica)
                                                                Iris-virginica: 2
                                        > 4.8 (Iris-virginica)
                                                Iris-virginica: 43
```

5.2 Restaurant.csv

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

5.3 Weather.csv

```
Weather:
        rainy:
                Windy:
                        FALSE (yes)
                                yes: 3
                        TRUE (no)
                                no: 2
        sunny:
                ('Humidity', 70.0):
                        <= 70.0 (yes)
                                yes: 2
                        > 70.0 (no)
                                no: 3
        overcast (yes)
                yes: 4
```

6 Conclusão

Após a análise e implementação do algoritmo ID3, é importante destacar que foram identificadas algumas limitações e falhas na abordagem. Essas limitações podem afetar a precisão dos resultados obtidos.

No entanto, apesar dessas falhas, a implementação foi capaz de gerar árvores de decisão que parecem tomar decisões corretas com base nos dados fornecidos.

É importante ressaltar que, para obter resultados mais confiáveis, seria necessário lidar com as limitações e aprimorar o algoritmo. Isso pode envolver a consideração de técnicas adicionais, como poda da árvore para evitar overfitting, ou até mesmo explorar outros algoritmos de aprendizado de máquina para comparação.

Apesar das limitações e das possíveis melhorias, a implementação atual ainda oferece uma base sólida para compreender os princípios do algoritmo ID3 e para tomar decisões aparentemente corretas com base nos dados fornecidos.

Com esforços adicionais de aprimoramento e experimentação, é possível avançar na construção de modelos de aprendizado de máquina mais precisos e confiáveis utilizando árvores de decisão.

7 Biliografia

Livro
Artificial Intelligence-A Modern Approach 4rd Ed
Stuart, Russell and Peter, Norvig

Video a explicar arvores de decisao

7.1 Imagens

Tabela de comparacoes de algoritmos