Relatório

Árvores de Decisão

Trabalho realizado por: Mafalda Aires (202106550)

Gonçalo Monteiro (202105821)

Relatório 1

Árvores de Decisão 1

Introdução 3

Algoritmos para indução de árvores de decisão 4

ID3 (Iterative Dichotomiser 3) 4

C4.5 4

CART (Classification and Regression Trees) 4

O que têm em comum? 5

ID3 5

Implementação 6

Linguagem escolhida 6

Estruturas de Dados Usadas e Organização do Código 6

Results 8

Comentários Finais e Conclusões 10

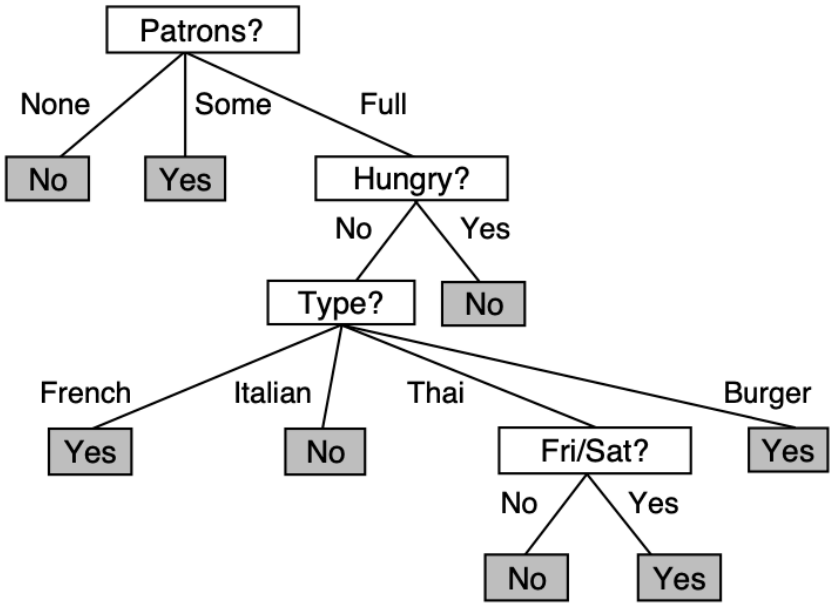
Introdução

Uma árvore de decisão é um modelo de machine learning que utiliza como estrutura uma árvore (Tree).

Face a um dataset, tendo em conta os atributos dados no input e a classe "target", constrói-se uma árvore de decisão.

- Cada nó da árvore de decisão representa um teste a um atributo. Este atributo é escolhido mediante uma função de avaliação (atributos com melhor avaliação serão escolhidos primeiro).
- Cada ramificação representa o resultado do teste ao atributo, que representa os valores únicos que o atributo(nó) possui.
- As folhas da árvore representam o valor da classificação final. São folhas aqueles nós onde todos os registos do dataset correspondente ao nó tem a mesma "target" classe, ou quando já não existem mais atributos para explorar, retornando os valores que representam a maioria.

Construída a árvore de decisão será possível tomar decisões ou fazer previsões percorrendo o caminho apropriado começando pelo nó raiz e seguindo os ramos até chegar a uma folha, onde estará a classificação da decisão ou previsão.



Algoritmos para indução de árvores de decisão

ID3 (Iterative Dichotomiser 3)

No final dos anos 1970 e início dos anos 1980, J. Ross Quinlan, um investigador em aprendizado de máquina, desenvolveu um algoritmo de árvore de decisão conhecido como ID3 (Iterative Dichotomiser 3).

O ID3 apenas consegue lidar com atributos discretos. Requer que os dados de treino estejam completos, pois não consegue lidar com valores ausentes. Utiliza o ganho de informação como função para selecionar o melhor atributo para ser testado. Para além disso, não possui mecanismos de "pruning" incorporados.

C4.5

J. Ross Quinlan apresentou C4.5, um sucessor de ID3.

O C4.5 consegue lidar com atributos discretos e contínuos, assim como dados com valores ausentes.

A função que avalia os melhores atributos é a "normalized information gain" que tem em consideração o número de valores distintos em cada atributo, o que evita a preferência por atributos com muitos valores. Inlcui mecanismos de "pruning".

CART (Classification and Regression Trees)

Em 1984, um grupo de estatísticos (L. Breiman, J. Friedman, R. Olshen e C. Stone) publicou o livro "Classification and Regression Trees" (CART) que descreve a geração de árvores de decisão binárias.

O CART lida com atributos contínuos, mas adota uma abordagem de divisão binária para atributos contínuos. Também lida com valores ausentes

e utiliza uma medida de impureza, Gini, para avaliar a qualidade dos atributos e realizar as divisões da árvore. Também possui mecanismos de"pruning".

O que têm em comum?

Os três algoritmos adotam uma abordagem "greedy". As árvores de decisão são construídas de forma recursiva, de cima para baixo, num processo de "divide and conquer". O conjunto de treino é dividido recursivamente em subconjuntos à medida que a árvore é construída.

Para além disso, os algoritmos são relativamente simples.

ID3

Dado um Training Set, o algoritmo avalia todos os atributos a ser testados através de uma função que mede o ganho de informação associado ao atributo. A função ganho de informação mede a entropia do conjunto de dados antes do teste ao atributo ser feito e depois do teste ser feito. Quanto maior for o módulo da diferença entre estes dois, maior será o ganho de informação.

Dito isto, o algoritmo escolhe o atributo com maior ganho para representar a "root" da árvore e remove esse atributo da lista de atributos a ser testados. O resultado do teste dá origem a "branches"(ramos) que representam os valores únicos do atributo que lhes deu origem. À medida que isto acontece, o conjunto de dados vai-se dividindo em porções originando novos "sub" Training Set's. Todo este processo é repetido recursivamente para os "sub" Set's e para a lista de atributos que ainda não foram testados.

O algoritmo termina com "leafs" (folhas), que são nós que retornam o valor da classificação. Estes nós definem-se como "leafs" quando todos os

exemplos do “sub” Set possuem a mesma classificação, ou quando não há mais atributos a testar (neste caso, a maioria determinará a classificação).

Implementação

Linguagem escolhida

A linguagem usada para a realização deste trabalho foi Python.

Estruturas de Dados Usadas e Organização do Código

read_csv(file)

Em primeiro, construímos uma função “read_csv(file)” que lê um file.csv e passa para uma matriz. Esta função também converte string para os seu floats correspondentes, para que seja possível trabalhar com este valores na discretização.

DataFrame

Num ficheiro DataFrame.py contruimos uma classe DataFrame. Em DataFrame, o dataset é guardado em forma de matriz assim como algumas características da matriz (rows, cols, attributes,size).

Esta classe tem como principal objetivo facilitar a manipulação do dataset (como se fosse uma classe pandas).

É nesta classe que estão definidas as funções que calculam a entropia e o ganho de informação para a seleção de atributos, assim como as funções que são responsáveis pela discretização de atributos:

- função que seleciona os atributos que têm que ser discretizados;
- função que calcula o ganho de informação para splitting points;
- função que descobre os melhores split points para um atributo;
- função que processa a discretização de um atributo;

- função que discretiza todos os atributos necessários.

Nota: Este algoritmo não lida com dados contínuos. Pelo contrário, a discretização é considerada neste caso como um pré-tratamento de dados.

Node

Num ficheiro Node.py construímos a classe Node.

Esta classe guarda os exemplos do Dataset que pertencem a cada nó, o atributo a ser testado se não for “leaf” e a classificação se for “leaf”.

Aqui, os “branches” e os nós filhos de cada “branch” são usados com auxílio de um dicionário. As chaves representam o valor do “branch” e o valor único da chave é o nó filho que advém do resultado do teste do nó pai.

Id3

Num ficheiro id3.py encontra-se a definição das funções “learn_decision_tree (examples, attributes, parent_examples)” que segue o pseudo-algoritmo da Id3 do livro (“Artificial Intelligence A Modern Approach-Four Edition” capitulo 19.3 página-678) e a função “print_tree(node,ident=””) que recebe a “root” da árvore resultante.

TestFrame

Num ficheiro TestFrame.py definimos a classe TestFrame. Esta classe tem como objetivo facilitar o tratamento de datasets que possuam atributos contínuos.

Com as funções:

- _continuo_to_discreto(atributo,split_list);
- _discretize_attribute(atributo);
- _discretize()

é possivel enquadrar os valores contínuos do Test-set nos intervalos que foram definidos no Trainning-set.

Main

No ficheiro main.py é onde vai ser processado tudo o que construímos.

Recebemos um ficheiro para o Training-Set, discretizamo-lo se for necessário, aplicamos a função learning_decision_tree() e fazemos print da árvore resultante.

Se um segundo ficheiro for submetido, este será lido como um Test-set para realizar previsões e testar o modelo id3 obtido. Depois, iniciamos as previsões cujo o resultado fica guardado numa lista “y_pred” pela mesma ordem que foi dado no input.

Results

Para o dataset **restaurant.csv** , o resultado obtido foi:

```
(base) gmonteiro@Air-de-Goncalo final dt % python3 main.py restaurant.csv

ID3 Tree Visualization:

                <Alt>
                    Yes:
                        <Bar>
                            No:
                                <Fri>
                                    No:
                                        <Hun>
                                            Yes:
                                                <Pat>
                                                    Some:
                                                        Yes (1)
                                                    Full:
                                                        No (1)
                                        Yes:
                                            <Hun>
                                                Yes:
                                                    Yes (1)
                                                No:
                                                    No (1)
                            Yes:
                                <Fri>
                                    Yes:
                                        <Hun>
                                            Yes:
                                                <Pat>
                                                    Full:
                                                        <Price>
                                                            $$$:
                                                                No (1)
                                                            $:
                                                                Yes (1)
                    No:
                        <Bar>
                            Yes:
                                <Fri>
                                    No:
                                        <Hun>
                                            No:
                                                <Pat>
                                                    Some:
                                                        Yes (1)
                                                    None:
                                                        No (1)
                                            Yes:
                                                Yes (1)
                                        Yes:
                                            No (1)
                            No:
                                <Fri>
                                    No:
                                        <Hun>
                                            Yes:
                                                Yes (1)
                                            No:
                                                No (1)
```

Para o dataset **weather.csv** , o resultado obtido foi:

```
(base) gmonteiro@Air-de-Goncalo final dt % python3 main.py weather.csv

_____________________
ID3 Tree Visualization:
_____________________

                <Weather>
                    sunny:
                        <Temp>
                            [84.0,+inf[:
                                no (1)
                            ]-inf,84.0]:
                                <Humidity>
                                    [82.5,+inf[:
                                        no (2)
                                    ]-inf,82.5]:
                                        yes (2)
                    overcast:
                        yes (4)
                    rainy:
                        <Temp>
                            ]-inf,84.0]:
                                <Humidity>
                                    [82.5,+inf[:
                                        <Windy>
                                            FALSE:
                                                yes (1)
                                            TRUE:
                                                no (1)
                                    ]-inf,82.5]:
                                        <Windy>
                                            FALSE:
                                                yes (2)
                                            TRUE:
                                                no (1)
```

Para o dataset **iris.csv** , o resultado obtido foi:

```
(base) gmonteiro@Air-de-Goncalo final dt % python3 main.py iris.csv

_____________________
ID3 Tree Visualization:
_____________________

            <sepallength>
                ]-inf,5.4]:
                    <sepalwidth>
                        [3.3,+inf[:
                            Iris-setosa (25)
                        ]-inf,3.0]:
                            <petallength>
                                ]-inf,4.8]:
                                    <petalwidth>
                                        ]-inf,1.6]:
                                            Iris-setosa (14)
                                        [0.8,+inf[:
                                            Iris-virginica (1)
                        ]3.0,3.3]:
                            Iris-setosa (12)
                [5.5,+inf[:
                    <sepalwidth>
                        [3.3,+inf[:
                            <petallength>
                                ]-inf,4.8]:
                                    <petalwidth>
                                        ]-inf,1.6]:
                                            Iris-setosa (4)
                                [2.5,+inf[:
                                    Iris-virginica (5)
                        ]3.0,3.3]:
                            <petallength>
                                ]-inf,4.8]:
                                    Iris-versicolor (6)
                                [2.5,+inf[:
                                    <petalwidth>
                                        ]-inf,1.6]:
                                            Iris-versicolor (1)
                                        [0.8,+inf[:
                                            Iris-virginica (12)
                        ]-inf,3.0]:
                            <petallength>
                                ]-inf,4.8]:
                                    <petalwidth>
                                        ]-inf,1.6]:
                                            Iris-versicolor (28)
                                        [0.8,+inf[:
                                            Iris-virginica (2)
                                [2.5,+inf[:
                                    <petalwidth>
                                        ]-inf,1.6]:
                                            Iris-virginica (6)
                                        [0.8,+inf[:
                                            Iris-virginica (27)
                ]5.4,5.5]:
                    <sepalwidth>
                        [3.3,+inf[:
                            Iris-setosa (2)
                        ]-inf,3.0]:
                            Iris-versicolor (5)
```

Comentários Finais e Conclusões

Concluimos que o Id3 é um algoritmo muito simples e eficiente quando estamos a lidar com dados discretos. Porém, para dados contínuos, somos obrigados a pré-processar os dados para obtermos uma boa classificação.