Inteligência Artificial

3º Trabalho

Árvores de decisão

30 de abril de 2019



David Fernandes - up201605791

Sérgio Araújo - up201608365

Índice

I Introdução …………………………………………….. 2

II Estratégias de Procura

1. Jogos com oponentes …………………………………………….. 2

III Descrição do Problema …………………………………………….. 4

IV Descrição da Implementação …………………………………………….. 4

V Comentários Finais e Conclusões …………………………………………….. 6

VI Referências Bibliográficas …………………………………………….. 7

Introdução

Todos os trabalhos feitos até ao momento nesta unidade curricular têm algo em comum: não possuem qualquer habilidade de intuição. Isto é, todos eles tinham que ir construindo a sua árvore de procura de modo exaustivo ou com uma função de auxilio que poderia não ser mais correta. Tudo isto muda com o algoritmo desenvolvido neste terceiro trabalho – Árvores de Decisão – são capazes de através da observação de casos completos criar uma árvore.

Árvores de Decisão são algoritmos com uma estrutura do tipo de árvore, cada nó representa uma condição a ser testada pelo problema inicial e os ramos deste são as diversas possibilidades de resposta, a chegada a uma folha da árvore devolve uma solução/resposta ao problema inicial.

São criadas através da observação de uma amostra, inferindo o resultado de decisões pela amostra, por isto normalmente são acompanhadas por um valor probabilístico de certeza nos seus ramos e classe.

São usadas em diversos ambientes uma vez que são fáceis de interpretar, exemplos são: nas finanças para prever futuros resultados, na saúde para determinar através de sintomas possíveis doenças.

Algoritmos utilizados em Árvores de Decisão

Há vários algoritmos utilizados em árvores de decisão, como ID3 (*Iterative Dichotomiser 3*), C4.5 (sucessor do ID3), CART (*Classification & Regression Trees*) e entre outros.

**ID3:**

Dando prioridade ao ID3 (já que foi implementado no trabalho), este algoritmo constrói uma árvore de decisão partindo de um conjunto fixo de exemplos e a árvore resultante é usada para classificar futuras amostras.

A ideia base deste algoritmo é construir a árvore utilizando uma pesquisa *greedy* através dos diversos sets de maneira a testar cada atributo em cada nó da árvore. Para tal é necessário selecionar o nó que permitirá construir a árvore mais correta e precisa, isto é feito com o auxílio de uma medida chamada Entropia.

A Entropia é, em teoria, uma medida de incerteza num grupo de dados. Esta dá-nos o grau de ‘desorganização’ dos mesmos.

Dado um conjunto S contendo exemplos positivos e negativos de algum objetivo, o valor da entropia de S é dado por a seguinte fórmula: . P+ e p- correspondem à proporção de exemplos positivos e negativos em S. O valor da entropia é 0 se todos os membros de S pertencerem à mesma classe. Ex: Caso todos os membros sejam positivos (p+ = 1), então p- é 0, o que leva ao que o valor de entropia seja 0, já que -1log2(1) – 0log2(0) = 0.

A entropia é 1 quando existir um número igual de membros positivos e negativos e oscila entre 0 e 1 caso haja um número diferente em ambas as partes.

\*O Monte Carlo (incluído no segundo trabalho) possui algum *learning*.

Sabendo calcular a entropia estão todas as condições reunidas para medir a redução de entropia, após isso decide-se que atributo é adicionado ao nó de decisão. Para minimizar a profundidade da árvore de decisão, o atributo com a maior redução de entropia é a melhor escolha.

Mais precisamente, a informação de ganho (Ganho(S,A)) de um atributo A relativamente a uma coleção de exemplos S é definida por:

Ganho(S,A) = Entropia(S) - ∑()(|Sv|/|S|)\*Entropia(Sv), onde S é cada valor v de todos os possíveis valores do atributo A; Sv corresponde aos subsets de S para cada atributo A com valor v; |Sv| e |S| são o número de elementos em Sv e S respetivamente.

**C4.5:**

Descrição da Implementação

Na implementação do trabalho, optamos pela linguagem JAVA visto que é a que temos mais facilidade.

Para a criação da árvore de decisão seguimos o pseudo-código que está presente no manual sugerido (pag. 702). Assim havia 2 estruturas que eram necessária para o começo do algoritmo uma lista de exemplos e uma lista atributos, em ambos optamos mesmo por uma linkedlist visto que não sabíamos a priori o tamanho da amostra fornecida. A lista de exemplos é formada por Arrays de STRING’s em que cada posição deste é o valor associado ao atributo na mesma posição, para a lista de atributos criamos uma nova classe Atribute, constituinte da lista.

Atribute:

Os atributos desta classe foram lógicos de obter, o nome que representa o nome fornecido, uma lista de valores possíveis para o atributo, obtida por adicionar num lista com características de set todos os valores que apareceram nos exemplos para este atributo, ordenados por ordem alfabética e por fim a posição inicial do atributo no array de strings, necessário manter porque esta varia ao longo das interações

Outra estrutura necessária para o funcionamento do algoritmo, é uma árvore, e por seguimento um nó ( constituinte básico da árvore).

Node:

Os atributos desta classe são um pouco mais versáteis e menos lógicos que a classe acima, uma vez que na árvore temos dois tipos de nó, o nó teste e o no final que fornece uma solução. Em termos práticos é tão simples como ser ou não no folha. Contudo trouxe implicações, o nome é, em caso de folha a solução, no outro o nome do atributo, o count é em caso folha o numero de vezes que este caminho ocorreu nos exemplos, no outro a pos do atributo.

Os restantes são mais concretos sendo estes um array representando em cada posição um ramo da arvore e um array associando a cada ramo um nome.

Comentários Finais e Conclusões

Referências Bibliográficas

* Artificial Intelligence: A Modern Approach
* <https://dzone.com/articles/machine-learning-with-decision-trees>
* https://saiconference.com/Downloads/SpecialIssueNo10/Paper\_3-A\_comparative\_study\_of\_decision\_tree\_ID3\_and\_C4.5.pdf