RELATÓRIO DO 2º PROJETO DE INTELIGÊNCIA ARTIFICIAL 2020/2021

Joana Mendonça, 83597

Filipe Colaço, 84715

07/12/2020

Implementação de Árvores de Decisão

O algoritmo-base utilizado na nossa implementação de Árvores de Decisão foi ID3 (Iterative Dichotomiser 3), desenvolvido e lançado em [Quinlan 1986]. Este algoritmo assenta a sua escolha do nó-base de cada subárvore S na *Information Gain* (IG) obtida ao dividir S mediante cada um dos seus atributos, acabando escolhendo o atributo A com maior $IG(S, A)$.

O cálculo de IG apoia-se na noção de entropia (H); dado o conjunto de dados corrente S, o conjunto de classes X, e a proporção de elementos de classe x para o total de elementos em S $p(x)$, a entropia de S é dada por:

$$H(S) = \sum_{x \in X} -p(x) \cdot \log_2 p(x)$$

Tendo este valor, e considerando S_v como o resultado de particionar S por cada valor v tomado pela classe A a analisar, temos que:

$$IG(S, A) = H(S) - \sum_{v \in A} \left[\frac{|S_v|}{|S|} \cdot H(S_v) \right]$$

Infelizmente, este algoritmo não foi suficiente para obter árvores satisfatórias, dado que a única verificação realizada é a de IG; isto é, mesmo ganhos mínimos fariam com que houvesse uma divisão na árvore, ajustando-se assim esta cada vez mais aos dados fornecidos. Este ajuste em demasia é chamado de *overfitting*, e pode levar a que dados fornecidos à árvore com o objetivo de os classificar sejam mal classificados, visto que a árvore apenas está desenhada tendo em conta os dados de entrada.

Este problema pode ser mitigado generalizando a árvore gerada, por um processo de *pruning* (poda); o objetivo desta poda é remover ramos redundantes, que provém pouco ganho de informação ou que induzem classificações erradas, e substituí-los por nós terminais, ou folhas. Este processo pode ser feito durante a geração da árvore (*pre-pruning*), mas dada a dificuldade de prever quão relevante o ramo será mais abaixo, esta tática não foi considerada. Foi considerado, no entanto, o *post-pruning*, em que a árvore é primeiramente gerada usando um subconjunto dos dados originais, chamado conjunto de treino, e depois podada, usando os restantes dados (conjunto de teste) para verificar a diferença entre ter o ramo presente ou substituí-lo por um nó terminal.

Estratégias de *Pruning*

Para evitar que houvesse conjuntos de treino ou de teste com apenas um tipo de classificação na classe (caso efetivamente houvesse mais que um tipo), a divisão nesses dois subconjuntos foi feita de modo a que fosse garantido que ambos os valores de classificação estivessem presentes nas subclasses.

Umas das técnicas para reduzir o tamanho das árvores foi consistia na eliminação de nós redundantes; se fosse verificado que os dois filhos de um certo nó eram idênticos, esse nó era substituído por um deles (irrelevante qual, pois são iguais). Isto pode ser verificado no final do método `DTree`.

A solução implementada para a poda da árvore gerada é chamada de *Reduced-Error Pruning*; como descrito no final da secção anterior, esta estratégia elimina nós que tenham duas folhas como filhos caso haja uma melhoria na classificação de itens se fosse usado, em vez do nó presente, um nó terminal que classificasse de acordo com a classe maioritária presente àquela profundidade. Para tal, percorremos a árvore do nó mais profundo até à raiz, verificando que alternativa obtinha a menor proporção de *records* mal classificados. Esta poda está implementada no método `pruneTree`.

Estratégias de Redução de *Noise*

Não houve necessidade de uma vasta exploração de métodos para lidar com o *noise* (ruído) do conjunto de dados, visto que as próprias técnicas de poda já limitam o efeito do ruído na construção da árvore, como discutido em [Fürnkranz 1997]. Cremos também que a divisão cuidada do conjunto de dados descrita no início da seção anterior permitiu que o ruído fosse reduzido.

Análise Crítica

Tendo em conta os resultados da execução dos testes fornecidos pelos docentes (`testdecisiontrees`), pudemos concluir que o *pruning* aplicado foi relativamente bem-sucedido, tal como o próprio algoritmo principal de geração de árvores.

Outros Materiais

[Quinlan 1986] Quinlan, J. (1986). Induction of Decision Trees. Machine Learning, 1, 81-106.

[Fürnkranz 1997] Fürnkranz, J. (1997). Pruning Algorithms for Rule Learning. Machine Learning, 27, 139-172.