Relatório 2ª Entrega - Árvores de Decisão

Groupo al059 António Romeu Pinheiro (92427)
Leonor Veloso (92509)

December 6, 2020

1 Algoritmo base

O nosso algoritmo baseia-se fortemente no descrito no livro *Artificial Intelligence - A Modern Approach*, na medida em que o conceito base é testar o atributo mais importante primeiro. Para tal criámos a função **calcGain**, que recebe uma lista de exemplos, de classificações, e a entropia inicial do conjunto e devolve uma lista com os ganhos de todos os atributos.

A função **createdecisiontree** segue, na prática, a seguinte lógica:

O caso em que todos os valores do Y original são 0 ou 1 é trivial, sendo retornada uma árvore com apenas 1 nó e folhas com o valor de Y.

Nos demais é feita uma chamada à função recursiva **createdecisiontree_aux** que considera as possibilidades:

- Os restantes exemplos são todos positivos ou negativos, devolvendo de imediato 1 ou 0
- O ganho máximo é 0 ou há atributos com o mesmo ganho, em que a árvore é dividida, aprofundando a árvore e escolhendo o atributo seguinte
- O ganho máximo é 1, um caso de paragem em que é devolvida uma lista com o atributo e os valores de Y quando o atributo é 0 (esquerda) ou 1 (direita)
- O ganho máximo está entre 0 e 1, em que averiguamos qual o caso favorável do atributo e aprofundamos a árvore correspondente. Caso não haja caso favorável são aprofundadas ambas as árvores

2 Problemas das árvores mais pequenas e soluções

Para reduzir o tamanho das árvores considerámos primeiramente o caso mais óbvio: quando as duas sub-árvores que resultam do corte são iguais. Nesse caso, retornamos apenas uma das árvores, ignorando o índice do nó pai e reduzindo o tamanho da árvore em 1 nó. Outros casos mais problemáticos

e que exigiram soluções ligeriramente mais complexas foram as que figuram nas árvores 1 e 2 (em anexo). Após o corte, as árvores filhas têm o mesmo índice e as mesmas sub-árvores da esquerda/direita. Nesses casos, tomamos como nó-pai o nó-filho repetido, adicionamos as sub-árvores repetidas do lado respetivo, e aprofundamos uma árvore com o índice original no outro lado (essencialmente trocando o pai e o filho de posição hierárquica).

3 Problemas de noise e soluções

Quando se aplica um conjunto de exemplos com noise ao nosso algoritmo base, surgem os seguintes problemas:

- Pode não haver mais atributos para explorar
- No corte da árvore, uma das sub-árvores pode ser vazia

Resolvemos ambos os problemas utilizando um majority vote para classificar exemplos iguais mas com diferentes classificações (que eram a origem dos erros acima encontrados). No primeiro caso, se o majority vote resultar num empate, assumimos o valor 0, refletindo o facto de termos ficado sem atributos para explorar. No segundo assumimos o valor 1, refletindo a possibilidade da árvore ser expandida, dado que o ganho máximo dos atributos não é nulo.

4 Análise crítica dos resultados

Embora o uso de um majority vote tenha funcionado no âmbito deste projeto, em que os atributos só podem tomar 1 de 2 valores, temos noção de que esta abordagem pode não ser tão eficiente noutro tipo de agente (ex: decision-theoretic agent). Uma solução mais abrangente envolveria o cálculo estimado das probabilidades de cada classificação e do desvio, para implementar uma forma de pruning mais complexa, como chi-square.

5 Anexo

Figure 1: Redução à esquerda

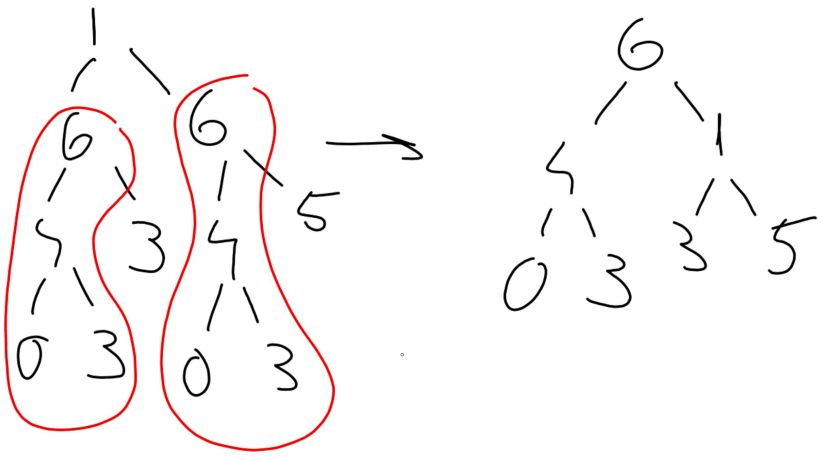

Figure 2: Redução à direita