Relatório do 2º Projeto

Grupo 5

Descrição do problema:

O problema que nos é dado no contexto do projeto envolve árvores de decisão. Árvores de decisão são modelos estatísticos que utilizam conjuntos de treino para classificarem e preverem futuros dados. Neste projeto são nos dados conjuntos de treino para os quais temos de inferir uma árvore de decisão.

Solução:

A solução base para criar uma árvore de decisão consiste em descobrir para todos os atributos qual destes tem maior ganho de informação e selecionar esse atributo para ser raiz da árvore. Ao encontrarmos o melhor atributo, isto é, o atributo que nos permite classificar mais exemplos de uma só vez, retiramos esse atributo, descobrimos o novo conjunto de dados e corremos o algoritmo novamente para este novo conjunto, recursivamente. Quando chegamos a algum caso terminal, a árvore é construída a partir daí.

Para calcular o atributo que tem maior ganho de informação, procedemos da seguinte forma:

- ➔ Identificamos o número de exemplos que são positivos com classificação verdadeira (*posTrue*) e com classificação falsa (*posFalse*). O número total de positivos no atributo é a soma entre *posTrue* e *posFalse*.
- ➔ Fazemos exatamente o mesmo para os casos negativos, em que o número total de negativos no atributo é a soma entre *negTrue* e *negFalse*.
- ➔ Calculamos o ganho de informação através do cálculo da entropia (mede a quantidade de incerteza numa distribuição de probabilidade).

Fórmula para calcular o ganho de informação:

$$GI = entropia\left(\frac{positivos}{total}\right) - ((\frac{negativos}{total} * entropia(negDivisao) + \frac{positivos}{total} * entropia(posDivisao))$$

Onde: total = positivos + negativos;

negDivisao = $\frac{negTrue}{negativos}$ ou 0 se negativos = 0;

posDivisao = $\frac{posTrue}{positivos}$ ou 0 se positivos = 0;

$$\text{entropia(x)} = -1 * (x * np.log2(x) + (1 - x) * np.log2(1 - x))$$

O atributo escolhido é o atributo com maior valor no cálculo do ganho de informação.

Problemas encontrados no algoritmo base:

1. O primeiro problema que identificámos enquanto testávamos o algoritmo base implementado foi que quando existiam dois ou mais exemplos com a mesma descrição, mas com diferente classificação, isto é, os valores em D são iguais, mas em Y não, o algoritmo não conseguia encontrar uma árvore de decisão consistente com os dados.
2. O segundo problema que identificámos tinha em conta o tamanho da árvore retornada. Isto é, existiam árvores com um menor tamanho, porém o algoritmo retornava uma árvore mais complexa.

O algoritmo base implementado não tinha em conta os problemas mencionados acima.

Soluções para resolver os problemas detetados:

1. A solução que implementámos para o primeiro problema baseou-se no facto de termos avaliado os ganhos de informação para os atributos. Ao avaliarmos estes valores, concluímos que menos de 5% de ganho é bastante pouco para escolhermos esse atributo como o melhor. Assim, o que fazemos quando existe ruído e o ganho é menor que 5% é retornar a classificação que se encontra em maioria. Isto é, se o *array* Y tiver mais falses do que *trues* devolve *false* e vice-versa.
2. A solução que implementámos para o segundo problema baseia-se em reduzir a árvore de decisão através de uma chamada ao algoritmo base para todos os atributos, isto é, calcular uma nova árvore sem escolher o melhor atributo. Se esta nova árvore for menor que a árvore original, então, a árvore retornada é a árvore de menor tamanho.

Análise crítica dos resultados:

Avaliando os resultados do nosso código aos testes fornecidos pelos professores, verificámos que estes resultados são os esperados por nós. Aos testes onde não há ruído o algoritmo base implementado funciona corretamente e como foi calculado por nós previamente, não funciona para os restantes testes. Isto acontece, porque, como identificado anteriormente, há problemas neste algoritmo (ruído, árvores extensas). Assim, para conseguirmos passar a todos os testes fornecidos, implementámos as soluções identificadas por nós.

Ao implementarmos as soluções apresentadas previamente para os problemas detetados, os resultados foram os esperados, uma vez que resolvemos os testes a que falhávamos anteriormente.

Assim, concluímos que os algoritmos implementados por nós são eficazes para inferir uma árvore de decisão que classifique e preveja resultados futuros, que era o problema original deste projeto. Ao observarmos o tempo necessário para inferir a árvore (anexos 1 e 2), concluímos que o nosso código é ótimo.

Catarina Sousa, nº93695
Nelson Trindade, nº93743

Anexos (tabelas com resultados de tempo consoante número de linhas e colunas)

Linhas	Colunas	Tempo (s)
4	2	0.243
8	3	0.233
8	4	0.248

Anexo 1 – tempo decorrido para inferir árvores sem ruído

Linhas	Colunas	Tempo (s)
10000	10	1.1647
10000	12	1.8605
10000	12	1.9863
10000	11	1.0436

Anexo 2 – tempo decorrido para inferir árvores com ruído