Inteligência Artificial

Projeto 2 - Grupo 58

Algoritmo Base

ID3 (Iterative Dichotomiser 3)

O ID3 é um algoritmo que serve para inferir uma árvore de decisão a partir de um conjunto de dados.
Recebe como argumentos *examples* - um array numpy L×C em que as L linhas são os exemplos, as primeiras C-1 colunas são as features e a C-ésima coluna é a classificação, *attributes* - uma lista com as features e *default* - a classificação a dar a um conjunto de exemplos se já não restarem mais features para classificar.
O algoritmo é recursivo e, no nosso caso, como para qualquer feature o conjunto de valores possíveis é {0, 1}, então no geral cada chamada à função gera duas novas chamadas, uma para o caso em que a feature em causa é 0 e outra para o caso em que essa feature é 1.

Optimalidade da árvore encontrada

Problema encontrado

O algoritmo base devolve uma árvore com erro 0 mas que, no geral, não é a mais curta (ótima).

Solução encontrada: Remoção de Classificações “Inúteis”

Percorrer a árvore gerada e remover classificações “inúteis”: dado um atributo se a classificação a dar caso esse atributo seja 0 ou 1 for igual, então esse atributo é removido da árvore e faz-se uma “ligação direta”.

Ruído

Problema encontrado

A árvore gerada pelo algoritmo base sofre de overfitting porque o nosso algoritmo continua a escolher atributos pelos quais classificar mesmo que a sua utilidade seja mínima (information gain residual).
Dados exemplos com ruído o nosso algoritmo gerará uma árvore muito comprida para tentar explicá-los “perfeitamente”, ignorando a existência de ruído.

Solução encontrada: Chi-Square Pruning

Percorrer a árvore gerada e remover classificações estatisticamente irrelevantes: dado um atributo aplicamos um teste de significância estatística para decidir se vale a pena continuar a manter aquele ramo ou se o ramo deve ser cortado e, nesse caso, devolvemos a classificação mais frequente.

Análise crítica dos resultados

Correção

Todas as árvores de decisão geradas pelo nosso programa classificam corretamente o input.
Nos testes que corremos, 27 em 27 das árvores tinham erro 0.

Optimalidade

Apesar das terem erro 0, as árvores de decisão geradas pelo nosso programa não são sempre as mais curtas possíveis.
Nos testes que corremos, 1 em 2 das árvores era possível ser encurtada.

Tolerância ao ruído

Todas as árvores de decisão geradas pelo nosso programa toleram bem a presença de ruído nos exemplos.
Nos testes que corremos, 4 em 4 das árvores toleravam bem o ruído i.e. não sofriam de overfitting porque classificavam bem tanto o conjunto-exemplo como o conjunto-teste.