Pré-poda em Árvores de Decisão

Diogo Rodrigues - PG37150, Luís Costa - A74819, and Luis Bouça - PG38933

Universidade do Minho
Departamento de Informática
Campus de Gualtar, Braga

Resumo Árvores de decisão, são sistemas de aprendizagem integrados na família dos algoritmos supervisionados. O objetivo das árvores, é criar um modelo de dados tendo por base a informação existente, permitindo ao utilizador uma fácil interpretação da mesma, e de como ela se relaciona.

A construção de uma modelo de árvores de decisão com capacidade para representar a informação da melhor maneira possível, permite resolver problemas de classificação ou de regressão.

O processo de construção da árvore, envolve conceitos como a impureza e o ganho informativo, com o objetivo de determinar os dados que melhor ganho trazem durante a sua construção.

1 Introdução

Overfitting é um problema que ocorre na contrução de modelos de árvore de decisão e muitos outros modelos de previsão. O overfitting ocorre quando o algoritmo de aprendizagem continua a desenvolver novos casos. O que resulta na redução do erro quando é testado com o conjunto de treino e no aumento do erro do conjunto de testes ou em casos futuros.

Existem duas abordagens para evitar o overfitting em árvores de decisão:

- **Pré-Pruning**: impede que a árvore de decisão continue a crescer antes que se torne completamente adaptada ao dataset de treino.
- **Pós-Pruning**: permite que a árvore de decisão continue a crescer e depois recorre a métodos de remoção de nós e folhas.

Normalmente a segunda abordagem é mais bem-sucedida pois não é fácil estimar com precisão quando é a altura correta para impedir que a árvore continue de crescer porém, neste trabalho, utilizamos apenas técnicas de pré-pruning para reduzir o overfitting de uma árvore de aprendizagem ao seu dataset de treino e assim reduzir o erro de previsão de casos futuros.

No entanto, é necessário ter em consideração, que a implementação abusiva de qualquer uma das abordagens, pode resultar num modelo demasiado simples, ou seja, estamos perante a ocorrência de underfitting, pelo que é necessário que estas técnicas sejam implementadas, tendo em conta os dados e o problema que estamos a tratar.

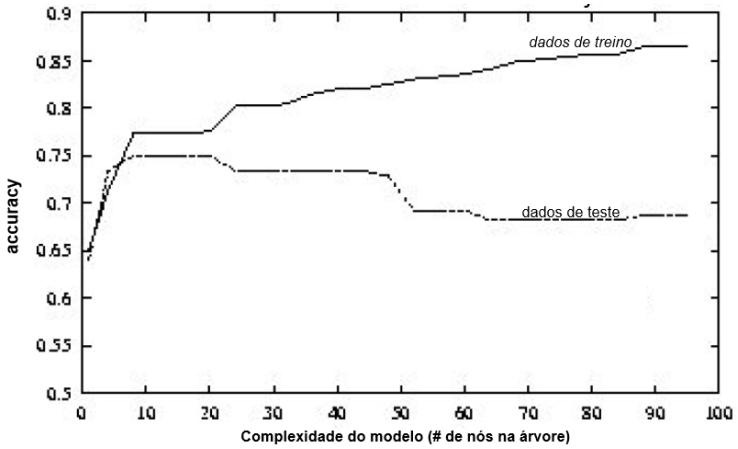


Figura 1: Accuracy / Complexidade

2 Métodos de Pré-Pruning

2.1 Baseado na Profundidade

O objetivo desta técnica de pruning, é podar a árvore assim que ela atingir um tamanho ótimo previamente definido.

Esta técnica parte do principio que a partir de uma certa profundidade, a árvore torna-se demasiado complexa, não trazendo qualquer ganho ao modelo final.

No entanto, determinar o tamanho ideal que a árvore deve ter, é diretamente proporcional à qualidade dos resultados obtidos por esta técnica.

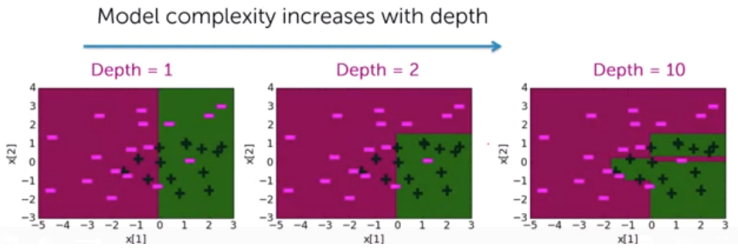
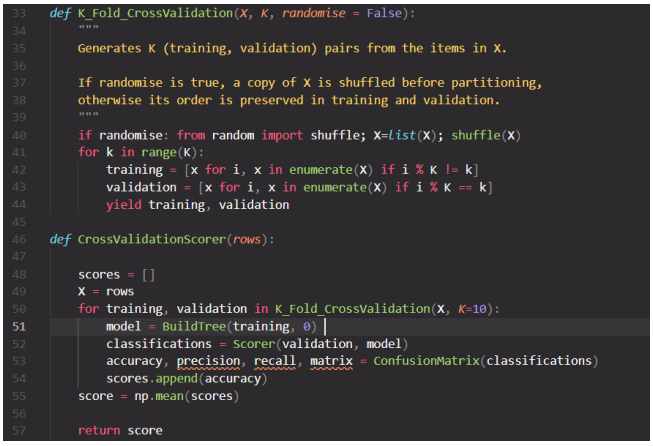


Figura 2: Profundidade / Complexidade

O uso de validação cruzada, permite que seja feita uma validação mais realista ao modelo, sendo neste caso utilizada para determinar o número ideal de profundidade.

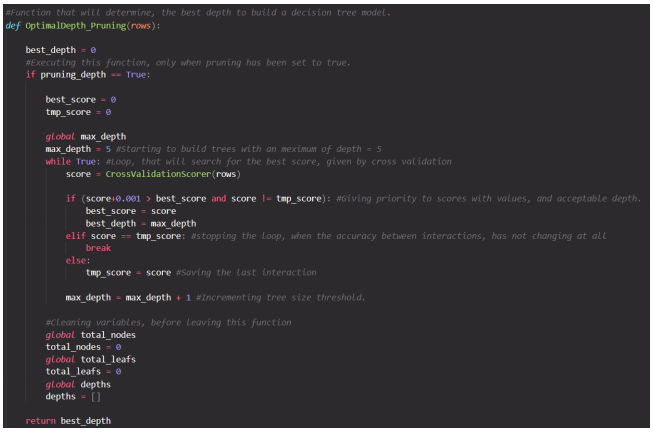


```
def K_Fold_CrossValidation(X, K, randomise = False):
    """
    Generates K (training, validation) pairs from the items in X.

    If randomise is true, a copy of X is shuffled before partitioning,
    otherwise its order is preserved in training and validation.
    """
    if randomise: from random import shuffle; X=list(X); shuffle(X)
    for k in range(K):
        training = [x for i, x in enumerate(X) if i % K != k]
        validation = [x for i, x in enumerate(X) if i % K == k]
        yield training, validation

def CrossValidationScorer(rows):

    scores = []
    X = rows
    for training, validation in K_Fold_CrossValidation(X, K=10):
        model = BuildTree(training, 0)
        classifications = Scorer(validation, model)
        accuracy, precision, recall, matrix = ConfusionMatrix(classifications)
        scores.append(accuracy)
    score = np.mean(scores)

    return score
```

Figura 3: Validação Cruzada

A figura 3 apresenta as duas funções implementadas em python, para realizar validação cruzada sobre dados de treino. Inicialmente a função **K_Fold_CrossValidation**, vai sub-dividir o conjunto de dados em 2 conjuntos de treino e teste, durante 'K' iterações. Os valores escolhidos em cada uma das iterações é referente ao resultado do resto da divisão por 'K'.

A função **CrossValidationScorer**, vai construir e validar os modelos, tendo por base os 'K' conjuntos devolvidos pela função **K_Fold_CrossValidation**, devolvendo uma média final, relativa à precisão dos vários modelos.



```
#function that will determine, the best depth to build a decision tree model.
def OptimalDepth_Pruning(rows):

    best_depth = 0
    #Executing this function, only when pruning has been set to true.
    if pruning_depth == True:

        best_score = 0
        tmp_score = 0

        global max_depth
        max_depth = 5 #Starting to build trees with an maximum of depth = 5
        while True: #Loop, that will search for the best score, given by cross validation
            score = CrossValidationScorer(rows)

            if (score+0.001 > best_score and score != tmp_score): #Giving priority to scores with values, and acceptable depth.
                best_score = score
                best_depth = max_depth
            elif score == tmp_score: #stopping the loop, when the accuracy between interactions, has not changing at all
                break
            else:
                tmp_score = score #Saving the last interaction

            max_depth = max_depth + 1 #Incrementing tree size threshold.

        #Cleaning variables, before leaving this function
        global total_nodes
        total_nodes = 0
        global total_leafs
        total_leafs = 0
        global depths
        depths = []

    return best_depth
```

Figura 4: Profundindade Ótima

A profundidade ótima é determinada pela função **OptimalDepth_Pruning**, apresentada na figura 4. Esta função realiza de forma recursiva, sucessivas cha-

madas à função **CrossValidationScorer**, verificando qual o melhor score obtido, para um tamanho máximo de profundidade, que é estipulado em cada ciclo. O ciclo termina a sua execução, quando a variabilidade do score não sofre qualquer alteração, indicando que a árvore atingiu o seu tamanho máximo. Esta função tem em conta a possibilidade da ocorrência de underfitting no modelo, devido a uma escolha prematura da melhor profundidade, quando ainda existe uma incerteza de como os dados vão ser representados durante a construção do modelo.

3 Baseado na Qualidade

A implementação de uma técnica que se baseia na incerteza dos dados, referentes às instâncias de cada classe que esse nó representa, permite descartar nós que não trazem qualquer ganho na qualidade de informação, tornando assim o modelo mais simples.

Este método de pruning, vai verificar a qualidade da informação em cada nó, comparando essa mesma qualidade, com a qualidade existente na sub-árvore originada por esse nó. O objetivo é avaliar se é vantajoso criar a sub-árvore, dado que ainda existe uma grande incerteza nos dados representados por essa sub-árvore.

```
def ClassificationError_Pruning(true_rows, false_rows):
    classification_error = len(true_rows)/(len(true_rows) + len(false_rows))
    classification_error_left = 0
    classification_error_right = 0

    #Validate each branch
    if len(true_rows) > 0:
        gain, attribute = FindSplit(true_rows)

        if gain != 0:
            true_rows_left, false_rows_left = Partition(true_rows, attribute)
            classification_error_left = len(true_rows_left)/(len(true_rows_left) + len(false_rows_left))

    if len(false_rows) > 0:
        gain, attribute = FindSplit(false_rows)

        if gain != 0:
            true_rows_right, false_rows_right = Partition(false_rows, attribute)
            classification_error_right = len(true_rows_right)/(len(true_rows_right) + len(false_rows_right))

    return classification_error, classification_error_left, classification_error_right
```

Figura 5: Avaliar a Qualidade

A figura 5, mostra a função responsável por obter a qualidade da informação existente no nodo atual, e a qualidade da informação existente numa possível sub-árvore. Essas incertezas vão ser comparadas, determinando assim se é necessário ou não parar a recursividade durante o processo de construção da árvore.

Figura 6: Condição de Paragem

A figura 6 mostra a condição de paragem de recursividade da árvore. De maneira igual à técnica de pruning referida anteriormente, é necessário evitar que o modelo caia numa situação de underfitting. Neste caso a condição é válida quando a árvore atingir uma profundidade superior a 5. Esta condição evita a tentativa de descartar nodos nos primeiros níveis de profundidade da árvore, ou seja, quando ainda existe uma grande incerteza relacionada com a forma de como os dados vão ser distribuídos, à medida que o modelo é construido.

4 Baseado na Significância dos Dados

Implementação de técnica que se baseia na quantidade de dados existentes no nó, ou seja, verifica se a quantidade de instâncias representadas pelo nodo, é significativa o suficiente para o manter ou descartar, tendo como comparação a quantidade de dados existentes em todo o conjunto de treino.

É uma técnica mais flexível e de fácil implementação, que pode ser usada independentemente do problema que tentamos retratar, isto porque, não existe a preocupação do modelo cair em situação de underfitting, uma vez que ele só vai atuar a partir de um certo threshold mínimo, referente aos dados existentes no nó.

```
def MinimumDataPoints_Pruning(total_rows_node):
    global total_rows
    partition_size = (total_rows_node * 100) / total_rows

    if partition_size <= 5:
        return False
    return True
```

Figura 7: Valor de Significância

A figura 7, mostra a função responsável por calcular o valor que os dados num determinado nó têm, comparativamente aos dados existentes no conjunto de treino. Neste caso foi definido um threshold de 5%, isto significa que se os dados representados num determinado nó, tiveram uma representação inferior a 5% face a todo o conjunto de dados, serão descartados.

5 Resultados

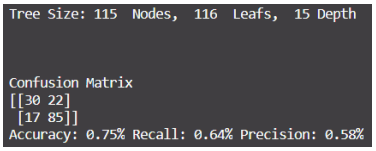


Figura 8: Pruning Desligado

A figura 8, mostra a validação do modelo com 152 instâncias de teste, e os resultados obtidos sob a forma de uma matriz de confusão.

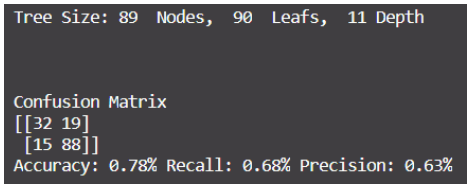


Figura 9: Pruning Baseado na Profundidade

A matriz de confusão apresentada na figura 9, mostra a influencia do pruning na validação do modelo com dados de teste. É possível verificar que existiu um aumento da accuracy, quando comparado à validação sem pruning. A profundidade da árvore gerads, foi considerada a profundidade ideal por parte da validação cruzada.

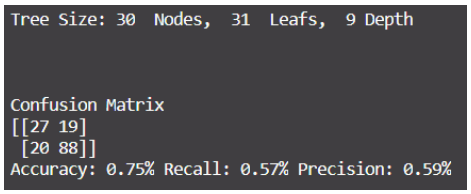


Figura 10: Pruning Baseado na Qualidade

A figura 10, mostra os resultados da execução do pruning, com a segunda técnica referente à qualidade dos dados existentes no nodo. É possível verificar que a árvore obtida é relativamente mais pequena, quando comparada com a árvore sem pruning, e os resultados obtidos são iguais.

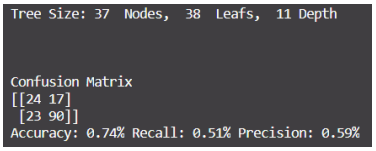


Figura 11: Pruning Baseado na Significância dos Dados

Na figura 11, é possível observar o resultado do pruning, utilizando apenas a técnica baseada no valor de significância dos dados. Neste caso o valor do desempenho foi ligeiramente pior, quando comparado com os restantes resultados, o que pode estar relacionado com o facto do threshold de significância ser demasiado elevado, resultando no descarte de nodos importantes.

6 Conclusão

O desenvolvimento deste trabalho, mostra a importância do pruning nos modelos de árvores de decisão, possibilitando um conhecimento relativo ao funcionamento dos mesmos, suas vantagens e diferenças entre eles.

No caso do pré-pruning, o maior desafio foi ajustar as condições de paragem, de forma as que as mesmas não resultem num modelo demasiado simplificado (**Underfitting**).

Realizar pruning definindo um máximo de profundidade na árvore, foi a técnica que resultou num modelo com melhor desempenho quando exposto a casos do conjunto de testes. No entanto a necessidade realizar validação cruzada com a construção de múltiplos modelos, com o objetivo de determinar a profundidade ideal, exige um poder computacional mais elevado, e dispendioso.

O pruning baseado na qualidade dos dados, foi o que apresentou a árvore com menor profundidade, e com desempenho igual ao de uma árvore sem pruning. Neste caso temos uma árvore bastante mais simplificada, e que pode estar melhor preparada quando validada com uma maior quantidade de casos de teste.

A última técnica implementada, é relativamente mais simples em relação às restantes, pois não existe a preocupação de o modelo poder cair numa situação de underfitting. No entanto a implementação desta técnica implica uma ponderação em relação ao threshold de significância definido, ou seja, o valor de significância a partir do qual vamos descartar os nodos. Questões relacionados com o próprio balanceamento dos dados no dataset, podem também estar relacionadas com o desempenho obtido por esta técnica.

Como trabalho futuro, fica a vontade de implementar técnicas usando uma abordagem de pos-pruning, e combinando os resultados dos vários modelos utilizados em cada técnica, de forma a obter aquela que apresenta um melhor desempenho. No caso de pós-pruning, não existe tanto a preocupação de o modelo poder cair numa situação de underfitting, no entanto exige mais tempo de processamento na criação do modelo. Neste caso o pruning vai ser aplicado após a árvore ter sido criada, numa abordagem bottom-up.