UNIVERSIDADE DE ÉVORA

Árvores de Decisão com Pruning

Relatório da 1ªParte do Trabalho Prático

Aprendizagem Automática

2020/2021

Docente: Prof. Teresa Gonçalves e Prof. Luís Rato

Discentes: Leonardo Catarro, nº43025; Diogo Solipa, nº 43071

Licenciatura: Engenharia informática

UNIVERSIDADE
DE ÉVORA

Índice

UNIVERSIDADE
DE ÉVORA

Objetivo

Este trabalho tem como objetivo implementar em Python uma classe para gerar árvores de decisão com “Pruning”, usando as diversas medidas de impureza: Gini, Entropia e Erro e REP (Reduced Error Pruning) como método de poda da árvore.

Análise de Desempenho

- Análise das árvores com e sem Pruning, **não fizemos pruning!**

→Diversas Funções de Homogeneidade/Pureza:

No desenvolvimento deste trabalho, foram usadas as funções da Entropia, Gini e Erro, como formas do cálculo da impureza dos dados fornecidos.

De entre as três, a Entropia será a que possui pior desempenho, daí maior complexidade temporal, isto porque, além da sua complexidade resultante de iteração pelas listas que armazenam os dados, ainda envolve o cálculo com funções logarítmicas, que são computacionalmente complexas

O método que utiliza o cálculo do Gini, poderá ser considerado o intermédio, apenas envolve produtos e cálculo fracionário

Finalmente, o Erro, será o que possui melhor desempenho.

UNIVERSIDADE
DE ÉVORA

As fórmulas matemáticas e respetivas implementações são as seguintes:

(Nota: Implementação de Gini, Entropia e Erro, todas dentro das funções: attributes_impurity(criterion) e impurity_per_type (type_list, criterion)

Nota2: Para cada método de cálculo de impureza dos dados, nós possuímos uma implementação para o cálculo por tipos e outra para os atributos, em funções separadas, mas que complementam uma á outra.

1. Entropia: entropia para dados nominais (envolvem mais que 2 classes)

$$entropy(D) = -\sum_{c=1}^{n} p_c \, log_2 \, p_c$$

```
def attributes_impurity(criterion):
    if criterion == "entropia":
        for element in attribute_list:
            if element.flag == False:
                for i in range(len(element.type_list)):
                    element.attribute_entropy += (element.type_list[i].n_times/total) * element.type_list[i].entropy
```

```
def impurity_per_type(type_list, criterion):
    if criterion == "entropia":
        for element in type_list:
            for i in range(len(element.class_list)):
                if element.n_times != 0:
                    fraction = element.class_list[i].class_ocurrence / element.n_times
                    if fraction == 0:
                        type_list[type_list.index(element)].entropy = 0
                    else:
                        type_list[type_list.index(element)].entropy -= fraction * math.log(fr
        return type_list
```

2. Gini: (para dados nominais)

$$gini(D) = 1 - \sum_{c=1}^{n} p_c^2$$

```
if criterion == "gini":
    for element in attribute_list:
        if element.flag == False:
            for element_type in element.type_list:
                element_type.entropy = (element_type.n_times/total) * 2
                for ocurrence in element_type.class_list:
                    element_type.entropy *= (ocurrence.class_ocurrence/element_type.n_times)
                element.attribute_entropy += element_type.entropy
            element.attribute_entropy = 1 - element.attribute_entropy
            element.attribute_entropy = -element.attribute_entropy
```

Legenda: Função attributes_impurity ()

```
if criterion == "gini":
    for element_type in type_list:
        for ocurrence in element_type.class_list:
            if element_type.n_times == 0:
                element_type.entropy = 0
            else:
                element_type.entropy *= (ocurrence.class_ocurrence/element_type.n_times)
        element_type.entropy = -element_type.entropy
    return type_list
```

Legenda: Função attributes_impurity ()

3. Classe Minoritária (Erro):

$$\min(p, 1-p)$$

```
if criterion == "erro":
    for element in attribute_list:
        if element.flag == False:
            for element_type in element.type_list:
                error = 0
                maximum = 0
                for ocurrence in element_type.class_list:
                    if ocurrence.class_ocurrence > maximum:
                        maximum = ocurrence.class_ocurrence
                for i in element_type.class_list:
                    if i.class_ocurrence != maximum:
                        error += i.class_ocurrence
                element_type.entropy = 1 - (error/total)
                #print(element_type, element_type.)
            element.attribute_entropy += element_type.entropy
```

Legenda: Função attributes_impurity ()

```
if criterion == "erro":
    #Descobrir o ocurrencias da classe maioritaria(classe com mais ocorrencias)
    #Fazer: total - ocurrencias/total
    #Not Implemented yet!
    for element_type in type_list:
        maximum = 0
        #Verifica qual a classe maioritaria
        for ocurrence in element_type.class_list:
            if ocurrence.class_ocurrence > maximum:
                maximum = ocurrence.class_ocurrence
        element_type.entropy = total - (maximum/total)
    return type_list
```

Legenda: Função impurity_per_type ()

UNIVERSIDADE
DE ÉVORA

Problemas Encontrados

Ao longo do desenvolvimento deste trabalho fomos nos deparando e enfrentando alguns problemas.

Inicialmente, o armazenamento dos dados, optamos pela programação orientada por objetos, o que para nós nos pareceu a melhor forma para o armazenamento dos dados. Porém, com o desenvolvimento percebemos que talvez não tenha sido a melhor escolha, mas ainda assim conseguimos desenvolver o que era pedido, com a nossa ideia inicial.

De seguida, e talvez o maior problema enfrentado, foi na parte da função recursiva (recursive_split ()), função esta que constrói a árvore, tratando os dados desde a raiz às folhas. Obrigamos a função a fazer mais iterações após atualizarmos o nó e não necessariamente casos mais generalizados para recursão

Por último, o desenvolvimento do Pruning, onde tivemos bastantes dificuldades, derivado às anteriormente mencionadas, em implementar a poda. Fazendo assim com a nossa árvore apensas seja construída sem Pruning.

UNIVERSIDADE
DE ÉVORA

Conclusão

A dificuldade deste trabalho afetou o seu desenvolvimento, tornando-o mais lento, porém conseguimos chegar ao resultado desejado