Fakulteit Ingenieurswese, Bou-omgewing & IT

Faculty of Engineering, Built Environment & IT

EAI 320: Artificial Intelligence

Imtiaz Mukadam

U13083113

Assignment 5: ID3 Algorithm

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Introduction

The ID3, or Iterative Dichtomiser 3, algorithm is a decision building tree procedure that recursively classifies attributes of a data set using Information Gain. Using the data set provided, a decision tree is constructed and is tested against sample data.

Methodology

The ID3 algorithm is implemented in its own class to form a generic procedure that can be used for any data set with any number of attributes. The class contains various methods that involve sorting the data set in preparation for the algorithm and calculating information gains of sub trees before constructing a tree that can be used to find solution decisions with sample data.

The ID3 class also holds an internal Node class that is used to construct the final decision tree.

```
class ID3:
    """ID3 Class that creates a decision tree on the data set provided by 'data_file'.
        the data_set can be changed by calling ID3.changeDataSet()
    """
    class Node:
        """
        basic Node class for the problem, trees will generally be small in test cases.
        """
        def __init__(self, children=None, parent=None, value=None, is_leaf=False):
            self.children = children
            self.parent = parent

            self.value = value
            self.isLeaf = is_leaf
```

Figure 1 Definition of ID3 class

The ID3 class accepts 2 parameters; the first is a path string to the CSV file containing the data set, and the second is a path string to a CSV file containing the names of the attributes.

ID3.py × attributes.csv × restaurant.csv ×

```
Yes, No,  No,  Yes, Some, $$$, No,  Yes, French,  0-10,  Yes
Yes, No,  No,  Yes, Full, $,   No,  No,  Thai,    30-60, No
No,  Yes, No,  No,  Some, $,   No,  No,  Burger,  0-10,  Yes
Yes, No,  Yes, Yes, Full, $,   No,  No,  Thai,    10-30, Yes
Yes, No,  Yes, No,  Full, $$$, No,  Yes, French,  >60,   No
No,  Yes, No,  Yes, Some, $$,  Yes, Yes, Italian, 0-10,  Yes
No,  Yes, No,  No,  None, $,   Yes, No,  Burger,  0-10,  No
No,  No,  No,  Yes, Some, $$,  Yes, Yes, Thai,    0-10,  Yes
No,  Yes, Yes, No,  Full, $,   Yes, No,  Burger,  >60,   No
Yes, Yes, Yes, Yes, Full, $$$, No,  Yes, Italian, 10-30, No
No,  No,  No,  No,  None, $,   No,  No,  Thai,    0-10,  No
Yes, Yes, Yes, Yes, Full, $,   No,  No,  Burger,  30-60, Yes
```

Figure 2 CSV file containing the data set

The data provided to the algorithm is stored in CSV files that first need to be stored and stored before the recursive ID3 algorithm is run.

The algorithm ID3() starts by creating a root node for the tree based on which attribute has the highest information gain from al the examples, it then creates branches for each case of the attribute, and recursively calls ID3() with its own case removed from the example set, which then again finds a root of the highest information gain and adds it to the tree. However, if the example set is pure, a leaf node with its target is created and the recursive call is terminated.

```
def ID3(self):
    self.decisionTree = self.__ID3(self.decisions, self.dataSet, self.root)

def __ID3(self, examples, attributes, node):
    if examples is None:
        return node
    if self.isExamplesPure(examples):
        return examples[0]

    informationPerAttribute = []
    for i in range(self.number_of_attributes):
        informationPerAttribute.append(self.informationGain(examples, i))

    attributeNumber = informationPerAttribute.index(max(informationPerAttribute))
    node = self.__ID3(self.getExamplesByAttribute(attributeNumber),)
    return node
```

Figure 3 recursive ID3 method

For the case of this practical, a separate CSV file containing the names of the attributes is provided to the class in order to create a named decision tree. This is to ensure that the class remains generic to any data set with its own attribute names.

Results

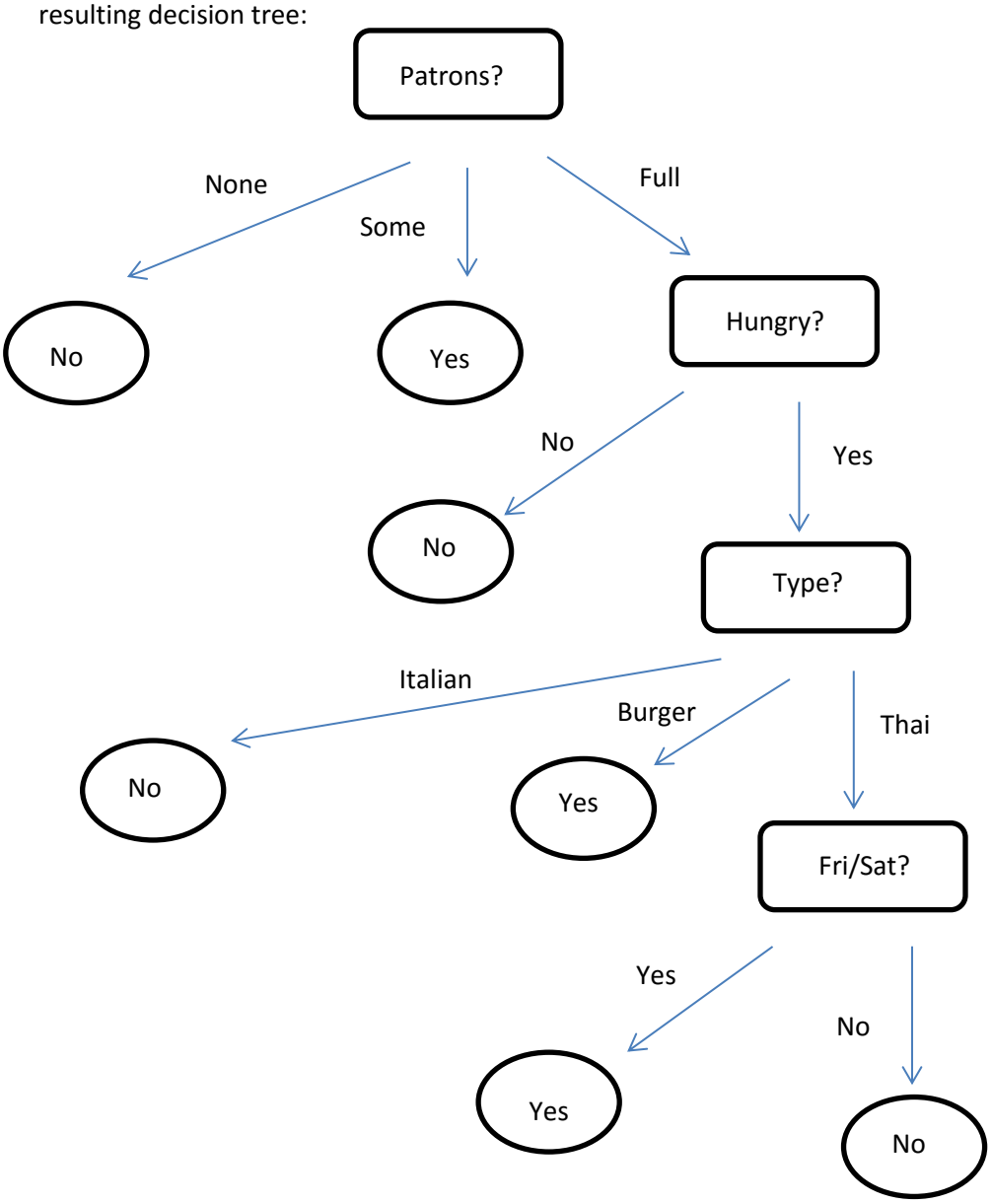
The data set provided by the given “restaurant.csv” file is given as a parameter to the ID3 class and is read and stored in its own List variable within the class. The various stored data is given as:

```
In[1]: x = ID3("restaurant.csv", "attributes.csv")
In[2]: x.data_set()
['Yes', 'No', 'No', 'Yes', 'Some', '$$$', 'No', 'Yes', 'French', '0-10', 'Yes']
['Yes', 'No', 'No', 'Yes', 'Full', '$', 'No', 'No', 'Thai', '30-60', 'No']
['No', 'Yes', 'No', 'No', 'Some', '$', 'No', 'No', 'Burger', '0-10', 'Yes']
['Yes', 'No', 'Yes', 'Yes', 'Full', '$', 'No', 'No', 'Thai', '10-30', 'Yes']
['Yes', 'No', 'Yes', 'No', 'Full', '$$$', 'No', 'Yes', 'French', '>60', 'No']
['No', 'Yes', 'No', 'Yes', 'Some', '$$', 'Yes', 'Yes', 'Italian', '0-10', 'Yes']
['No', 'Yes', 'No', 'No', 'None', '$', 'Yes', 'No', 'Burger', '0-10', 'No']
['No', 'No', 'No', 'Yes', 'Some', '$$', 'Yes', 'Yes', 'Thai', '0-10', 'Yes']
['No', 'Yes', 'Yes', 'No', 'Full', '$', 'Yes', 'No', 'Burger', '>60', 'No']
['Yes', 'Yes', 'Yes', 'Yes', 'Full', '$$$', 'No', 'Yes', 'Italian', '10-30', 'No']
['No', 'No', 'No', 'No', 'None', '$', 'No', 'No', 'Thai', '0-10', 'No']
['Yes', 'Yes', 'Yes', 'Yes', 'Full', '$', 'No', 'No', 'Burger', '30-60', 'Yes']
In[3]: x.number_of_examples
Out[3]: 12
In[4]: print x.attributes
['Alternate', 'Bar', 'Fri/Sat', 'Hungry', 'Patrons', 'Price', 'Raining', 'Reservation', 'Type', 'Time']
In[5]: x.number_of_attributes
Out[5]: 10
In[6]: x.decisions
Out[6]: ['Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'Yes']
```

Figure 4 stored data set

The ID3 algorithm is then executed using x.ID3() which then creates a decision tree using the stored data set.

resulting decision tree:



$$X_1 = \{Alt = Yes, Bar = No, Fri = Yes, Hun = Yes, Pat = Full, Price = \$\$, Rain = Yes, Res = NoType = Thai, Est = 10 - 30\}$$
$$X_2 = \{Alt = No, Bar = No, Fri = Yes, Hun = Yes, Pat = Full, Price = \$\$, Rain = Yes, Res = NoType = Thai, Est = 30 - 60\}.$$

Both Sample X_1 and X_2 return true when tested through the resultant decision tree.

Conclusion

The ID3 algorithm is a fast attribute based decision tree building algorithm, given a data set it easily constructs a decision tree using its information gain. This is accomplished because of it greedy divide and conquer strategy. However, this does not mean that the solutions to samples will necessarily be correct – that is based on how large the training data set is, and whether it includes a large variety of possible outcomes. It shows that if the training data set is large but lacks variety, an overfitting scenario will occur, where the resulting decision tree is only accurate for samples closely resembling the training data set. Since ID3 does not perform any kind of backtracking search, it is crucial that training data set provides variety to prevent a decision tree that is over fitted, which in this case, is the same as getting stuck in some local maxima or minima.