ECS629 ARTIFICIAL INTELLIGENCE
EFTHYMIOS CHATZIATHANASIADIS
150359131

1. INTRODUCTION

This project involves a generic implementation of the ID3 decision tree learning algorithm for classification of datasets with discrete valued attributes. The subsequent sections will describe the 2 core parts of the implementation:

1. Training
2. Classification

2. TRAINING PHASE

The training phase for a particular set of examples was implemented through the method `train()` which constructs the decision tree based on the dataset it receives as input.

2.1 TREE CONSTRUCTION

The `train`() method initiates the call to the method `decisionTreeLearning`() which recursively calls itself until the tree is constructed. At each step, a subset is evaluated and a "parent" node is created using the `TreeNode` object. Further, the "best" question with the highest information gain is selected(see 2.1.3) as the value of the `TreeNode` object. Then `decisionTreeLearning`() loops through all answers of the "best" question and performs the following operations:

- For each answer split(via `split()` method) the examples into a subset containing the examples with that answer.
- For each subset call `decisionTreeLearning()` recursively

After the loop completes, each subset becomes a "child" `TreeNode` object(i.e. subtree). The set of "child" nodes(i.e. subtrees) are assigned to the "parent" `TreeNode` as children.

2.1.1 DATA STRUCTURE

`ArrayList` over array data structure was used for storing the examples because of its dynamic, grow-able nature in size. Knowing in advance(prior split) the size of the subsets is not possible. Hence, the use of a non-static data structure, enhanced the memory efficiency aspects of the split method. In

ID3 Algorithm

addition another `ArrayList` was used to keep track the used attributes at each recursive call.

2.1.2 BASE CASE

The stopping conditions of the recursive `decisionTreeLearning`() method are the following:

A. Example list is empty
B. Examples have same class
C. No attributes/questions left

In both cases A and C, a `TreeNode` leaf object is created with the value of the majority class. Majority is found via the `plurality`() method. In case A where examples list is empty, the `plurality`() method is called on the parent examples, while in case C the `plurality`() method is called on the examples. In case B, a TreeNode leaf object is returned with the class of the examples retrieved via `getClass`() .

2.1.3 ATTRIBUTE SELECTION

For attribute selection the `findBestQuestion`() method was created that loops through questions not asked before and calculates information gain for each question. Then returns the question with the highest information gain.

3. CLASSIFICATION PHASE

Classification is handled by the `classify`() method which receives unlabeled test examples and classifies them based on the decision tree constructed in training phase. In particular, the `classify`() method loops through the test examples and for each example it calls an overloaded tail-recursive version of `classify`(). It starts from the top question of the tree and for each `TreeNode` calls classify() recursively on the child of the `TreeNode` which has answer(i.e. value) equal to the example value for that question. The recursive `classify`() terminates when a leaf node has been reached(i.e. TreeNode has no children). In this case the class is returned.