Assignment 2: Improving ID3 by using a genetic algorithm for

feature selection

# COS 314 Artificial Intelligence

## April 2019

Contents

[COS 314 Artificial Intelligence 1](#_Toc6666989)

[April 2019 1](#_Toc6666990)

[Statement of Purpose 3](#_Toc6666991)

[Research Questions 3](#_Toc6666992)

[Research Design 3](#_Toc6666993)

[The GA chromosome 3](#_Toc6666994)

[The GA Operators and Stopping Condition 3](#_Toc6666995)

[Research Findings 4](#_Toc6666996)

[Conclusion 4](#_Toc6666997)

# Statement of Purpose

This report is designed to find the set of features which allows ID3 to obtain the best generalization ability using a Genetic Algorithm (GA).

# Research Questions

This report examines the problem with the ID3 algorithm, its tendency to overfit. This issue can be seen more clearly as the number of features being used becomes larger. A genetic algorithm can solve this by finding the best subset of features to use when creating the ID3 tree. The following factors will be investigated:

1. The effectiveness (accuracy) that the subset has when tested against the test data.
2. The number of features that will be used.
3. Which features will be used.

# Research Design

The run time environment used for the experiment is as follows:

1. Windows 10 64-bit OS
2. Intel i5 7th generation core
3. 20GB of DDR4 RAM

The following restrictions fill be imposed to keep fairness:

1. The GA will only select features that should be used

The following type of data will be used:

0001000101010100000110000111011011000100010100110001011010101111110010100101011110100000011100100001 True

This contains 100 0s and 1s. Each 0 or 1 represents true or false for that specific feature, the pattern ends with the outcome, either true or false.

There are three files that contain this data. The training data will be used to train the algorithm (add features and their data to the tree). The validation data will be used to ensure a level of accuracy. The testing data will be used to test the final tree against.

## The GA chromosome

The GA chromosome that will be used will be structured as follows:

If the number of attributes were 5 and only attribute 2,3 and 5 are being used, a binary array can be created (starting at 0) that would look like: [0, 1, 1, 0, 1], where 0 indicates that a feature will not be used and 1 indicating that it will. Another representation of this is: [1, 2, 4] just representing the bits that have a value of 1. This representation will be used going forward.

## The GA Operators and Stopping Condition

The following GA Operators will be used:

1. A modified version of the crossover operator, this version only selects the bits being used to cross over and not the unused bits. For example, when using [1, 2, 4] and [1, 3] as parents the resulting child would be [1, 2, 3, 4]
2. The mutation operator where a random number between 0 and 99 (representing the 100 attributes) will be added to the parent to create a new child.

These operators are being used as the crossover operator will take all the positive attributes of each parent and combine then, the mutation operator will allow new attributes to be used and testing in further populations.

The following stopping condition will be used:

1. For the initial creation using just the training data, the GA will stop once the tree can represent the training data 99.9999% accurately.
2. For the pruning and adjustments using the validation data, the GA will stop if the tree has an accuracy of 99.99 when tested against the validation data or if it reaches 100 generations, but not before it reaches at least 87% accuracy when tested against the validation data.

# Research Findings

**\_ The progression of the optimization process during the run (how your**

**generalization accuracy changed over generations).**

**\_ The use of a confusion matrix to illustrate the e\_ectiveness of the \_nal**

**decision tree versus a decision tree that uses all the features.**

The following results were obtained:

THE original ID3 using all 100 features

From the data retrieved Breadth first has the best average distance from known optimum with Hill-climbing being the worst. Hill-climbing is the fastest with breadth first being the slowest. With A\* not being relatively slow and returning a path relatively close to the optimal it is a better choice than the other two algorithms, it’s faster than breadth first and has a better solution than Hill-climbing.

# Conclusion

In conclusion the A\* algorithm is a more efficient algorithm, when an efficient heuristic such as the Manhattan distance is used with the cost taken so far.