Basic Hunt's Algorithm for Medical Diagnosis

Implementation and Accuracy Measurements

Jo Inge Arnes
Department of Computer Science
UiT The Arctic University of Norway
Tromsø, Norway
jar005@post.uit.no

***Abstract*—Classification algorithms are used to automatically classify data records. Each record contains a set of independent attributes that can be analyzed by the algorithm to determine a target attribute's value, often called the record's class. From a medical perspective, an example of classification is to determine if a patient meets a certain diagnosis criteria. Medical professionals do this by checking a predefined set of rules that has to be fulfilled for the diagnosis to be met. An ideal classification algorithm have the ability to deduce the criteria and rules automatically, as opposed to only applying predefined criteria.**

For this paper, a basic version of the much-used Hunt's Algorithm [1] was implemented from the ground up, including programs for dataset generation, accuracy evaluation, and interactive classification. The Canadian Consensus Criteria [2] for Myalgic Encephalomyelitis (ME) was used as an example case to generate data sets and explore the accuracy of the algorithm. It should be noted that except for the random record generator, the implementation and programs were generic and not tied to a specific diagnosis.

Keywords— data mining; algorithms; classification; diagnosis criteria; decision support systems

I. INTRODUCTION

Many different classification algorithms exist, utilizing on a variety of techniques. Hunt's algorithm is based on building decision trees[1]. It does this by processing training records that contain both the independent attributes and the target value. The generated decision tree can later be used for fast classification of new records where the target value is unknown. Thus, the algorithm learns by example from previously classified data, and uses this knowledge for future predictions. Fig. 1. Shows an example of a trivial decision tree.

The quality of the training set is important for the resulting decision tree's accuracy. Too few records, missing attribute combinations, noise, little correlation between independent attributes and the dependent target, and other factors can dramatically affect the correctness. Some decision tree's may seem perfectly fine when used with its training set, but perform considerably worse when used with another set of valid records. This phenomenon is named over-fitting; the tree has become specifically tied to the training set. Because of this, it is important to evaluate the decision trees generated by the algorithm against a separate test set.

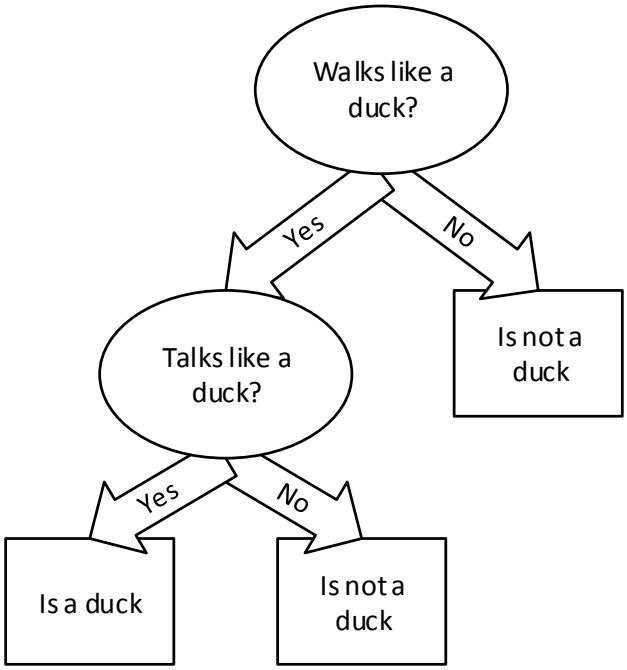

Fig. 1. A Decision Tree

When generating training and test sets, we cannot simply create random records. There has to be some rules behind the data that define which combinations of independent attributes values will classify a record's target value as positive or negative. The purpose of a classification algorithm is to deduce these automatically, for instance manifested as a decision tree. To test the quality of the algorithm, it is also useful to know the system of rules in advance, so that data sets can be generated and the resulting decision trees can be compared to the real set of rules that we know to be correct.

To put the experiments conducted for this paper in a medical context, a real diagnosis was chosen as the set of rules governing the correlation between the independent attributes and the target attribute values. Using a real diagnosis ensured some level of realism and provided a well-understood case, instead of inventing a fictional one.

[1] http://en.wikipedia.org/wiki/Decision_tree

```
Walks like duck?,Talks like duck?,Is duck
0,0,0
0,1,0
1,0,0
1,1,1
```

Code Listing 1: A simple example CSV

The question that this paper explores is this: How accurate are the decision trees compared to the real diagnosis criteria that was used to generate the training sets, given various numbers of training records and percentages of positive and negative classification results? This is interesting to get an idea of how many records would be necessary to create a good-enough decision tree in a real life scenario where the underlying criteria are not well understood, but where some means of predicting an outcome could be helpful for treatment or research purposes.

The diagnosis criteria used as an example in this paper is the Canadian Consensus Criteria for Myalgic Encephalomyelitis. The referred classification algorithm is a basic Hunt's algorithm that was implemented from the ground-up especially for this paper.

II. Design

Hunt's algorithm can be designed in several ways. Firstly, the tree could have binary or many-way splits. The implementation that was made for this paper uses binary splits, which means that a node only can have two child nodes. In addition, the algorithm could have supported attributes of numeric or nominal types, but only Boolean (true/false) was used. There were deliberately made no attempts to optimize the algorithm. Instead the implementation was kept as clean as possible to test the rudimentary features of Hunt's algorithm without modifying parts of its behavior too much.

A crucial part of the algorithm is to divide records into smaller sets for each node in the tree. This is done by comparing the degree of pureness – how homogeneous the sets' target values are – for each candidate attribute that can be used for splitting the records into to smaller groups. There are several mathematical formulas that can be used as impurity functions, e.g. etrophy, gini, or misclassification error. For simple true or false values, the misclassification error formula was considered good enough and chosen because of its performance efficiency.

Accuracy was calculated by simply dividing the number of correctly classified records with the total number of records.

III. Implementation

The full source code for the implementation is available online[2].

The implemented system has four main parts – data set generation, learning algorithm, evaluation algorithm, and a user interface for loading and interactively using decision trees for classification.

A. Data Set Generation

The data sets used for training and testing are generated as records containing values for the Canadian Consensus Criteria attributes. There are thirty-two different attributes, which are grouped in seven different categories. One of these categories has three sub categories. For the criteria to be met, all eight attributes in categories 1, 2, 3, 4 and 7 have to be true, *at least* two out of six attributes of category 5 has to be true, and *at least* two out of three subcategories of category 6 must have one attribute that is true. The three subcategories of category 6 have eight, five and five attributes.

```
{
  "attributeName": "Walks like duck?",
  "trueBranchNode": {
    "attributeName": "Talks like duck?",
    "trueBranchNode": {
      "attributeName": "Is duck",
      "trueBranchNode": null,
      "falseBranchNode": null,
      "leaf": true,
      "value": true
    },
    "falseBranchNode": {
      "attributeName": "Is duck",
      "trueBranchNode": null,
      "falseBranchNode": null,
      "leaf": true,
      "value": false
    },
    "leaf": false,
    "value": false
  },
  "falseBranchNode": {
    "attributeName": "Is duck",
    "trueBranchNode": null,
    "falseBranchNode": null,
    "leaf": true,
    "value": false
  },
  "leaf": false,
  "value": false
}
```

Code Listing 2: A Decision Tree as JSON

In the experiment, the percentage of records meeting the criteria and the number of records was varied.

Records with negative results were generated by randomly choosing values for all attributes. If a record was positive, the record was discarded and a new one was generated instead. This was done because the probability of randomly creating a true record was relatively low.

Records with positive results were generated by setting the eight mandatory attributes to true. For category 5, at least two of six attributes had to be true. This was done by selecting a random number between two and six, inclusive, and then randomly selecting this number of attributes from the six candidate attributes. These attributes were then set to true. Values for category 6 was generated by selecting if two or three subcategories should be true. Then either two subcategories were selected randomly, or all three were selected. For each subcategory, at least one attribute had to be true. The attributes for the subcategories were set in a similar manner as described for category 5.

Java 7 SecureRandom[3] was as the random number generator.

The generated data sets were stored as CSV-files[4]. See code-listing 1 for an example of a simple CSV.

B. Learning Algorithm

This is the actual learning part of the system, which was described in the design section of this paper. The name of a training set CSV is given as input, and Hunt's algorithm is used to generate a decision tree. The decision tree is output to file formatted as JSON[5]. This means that the decision tree is a

[2] https://github.com/mr-speak/simplehunts

[3] https://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html

[4] http://en.wikipedia.org/wiki/Comma-separated_values

[5] http://en.wikipedia.org/wiki/JSON

language agnostic standalone file that can by imported into other programs.

C. Evaluation Algorithm

After a decision tree has been generated, it is important to evaluate it against data set that is different from the one used for training. This is done by loading the decision tree and using it for classifying records from an evaluation set. The result from the decision tree is compared to the correct value that already was present in the record from the evaluation set. This is used to calculate the accuracy of the decision tree.

D. Classification User Interface

A simple text based user interface was also created. It was implemented as a program that takes a decision tree file as input. It then uses the attribute names from the decision tree to ask questions to the user and decide on a result value. In the case where the attribute names matches the ME criteria, these are treated specially, because a map between attribute names and display text will be used. The program is otherwise generic and can use any valid decision tree JSON. See code-listing 2 for an example of a decision tree expressed in JSON format.

IV. Evaluation

To evaluate how well the rudimentary implementation of Hunt's algorithm was able to deduce decision trees matching the predefined set answer (Canada Consensus Criteria), the number of records in the training set and the percentage of records meeting the criteria were varied.

The actual distribution of patients with ME in the general population was not used; neither was the percentage of patients referred to specialist teams for consideration, that end up being diagnosed as actually meeting the criteria. These numbers were not considered relevant in this experiment. The explanation is that a set of records with known target values must always exist to be able to use Hunt's algorithm. The distribution of positive and negative results can thus freely be composed as the data analyst finds suitable. The important goal is to generate the best possible decision tree, not to simulate real distributions.

As an example, let us say that the learning algorithm is only given records with negative results. The resulting decision tree will then never be able to return any positive results, no matter the input. The same goes vice-versa. The interesting question is if using a data set with a very low probability of a diagnosis, will generate a decision tree that is biased towards negative results, and too often will return false negatives.

The number of training records also affect the resulting decision tree. The question is how many records is sufficient to deduce a decision tree for a diagnosis?

In the experiment both the percentage of positive records and the number of records were varied and compared to the same test sets containing 100 000 records each. The reason for using two test sets, one with only positive and one with only negative record results, was to expose decision tree bias.

Selected results are presented in Table I. Not all results are presented in the table. In the experiment sets with 5, 10, 50, 100, 1 000, 10 000, 100 000, and 200 000 records were each generated with a percentage of positive records equal to 0.1, 0.5, 1, 10, 25, 50, 75, 90, 95, 99.5, and 99.9.

TABLE I. Accuracy Comparisons

Training Set Size	*% Positive Records*	*Accuracy True*	*Accuracy False*
5	50%	1.0	0.50292
10	50%	1.0	0.75411
50	50%	1.0	0.94027
100	50%	1.0	0.93949
500	50%	1.0	0.98775
1 000	50%	1.0	0.9995
10 000	50%	0.99813	0.9995
100 000	50%	0.9993	0.9995
200 000	50%	0.99951	0.99951
200 000	0.5%	0.21181	0.9923
200 000	10%	0.85239	0.96253
200 000	25%	0.99854	0.99951
200 000	75%	0.99991	0.9995
200 000	99.5%	1.0	0.9995

The result showed that the decision tree was biased towards returning positive results with a distribution of 50% positive records. The reason is probably that there are much more possible combinations for negative results, and for a decision tree generated with too few records, many combinations would yield false positives. As the number of records were increased, the combined accuracy increased, but the improvement in accuracy seemed to flatten as the numbers grew. Between 1 000 records and 10 000 records, the accuracy actually seemed to drop a little bit. It was also observed that a low percentage of positive records in the training data set, resulted in a low accuracy for positive records, and at the same time a slightly lower accuracy for negative records as well. A very high percentage of positive records, on the other hand, improved the classification of positive results, as well as not degrading the classification of negative records.

V. Related Work

Hunt's algorithm is a well-known algorithm, which have been the basis for many implementations, such as ID3, C4.5 and CART. These are too numerous to mention in detail. The contribution of this paper experimenting with the accuracy of a very basic version of the algorithm in conjunction with medical diagnosis criteria.

VI. Future Work

The proof-of-concept implemented for this paper only supports boolean values, but could be rewritten to support categories and numeric attribute values. A technique for checking information gain to decide whether to further split or

not could be implemented, along with other optimizations and evaluations of these.

VII. Conclusion

In this paper a rudimentary implementation of Hunt's algorithm was experimented with to measure and get an idea of how many records would be required to build a reasonably accurate decision tree in a medical diagnosis criteria context. The Canadian Consensus Criteria for Myalgic Encephalomyelitis was used as an example case to generate data sets and a way to determine the accuracy of the algorithm.

The results of the experiment indicated that an increasing number of training records improved the accuracy, but that the increase in improvement flattened as the numbers grew. It was observed that the percentage of positive records affected the accuracy, and that increasing the percentage of positive records made the decision trees more accurate.

References

[1] Tan, Pang-Ning, Steinbach, Michael, Kumar, Vipin. Introduction to Data Mining (2013) Paperback. Pearson, Chapter 4, 2013.

[2] Carruthers BM et al. 2003. Myalgic encephalomyalitis/chronic fatigue syndrome: Clinical working definition, diagnostic and treatment protocols. Journal of Chronic Fatigue Syndrome 11 (1): 7–36