SAPIENZA
UNIVERSITÀ DI ROMA

General ID3 Problem Solver

Tiago Daniel Sá Cunha

Machine Learning

7 November 2011

2011/2012

Index

Introduction

Since it was introduced in 1975 by Professor J. Ross Quinlan in the book *Machine Learning*, the ID3 ("Iterative Dichotomizer (version) 3") algorithm has proven to be one of the simplest and most complete algorithms in decision theory. It is based on the Concept Learning System (CLS) algorithm and lead to the development of C4.5, also by Professor Quinlan.

This algorithm provides the possibility to create a decision tree based on a fixed set of examples, in order to classify future samples. The tree outputted by this algorithm represents a simple abstraction to explain all the elements of the set and offers in a clever and intuitive way the overall dependencies among them to better understand the system and prepare a decision.

The abstraction behind ID3 is the concept of decision tree, which classifies a system creating decision nodes, where a certain attribute is evaluated at each step in order to create a set of rules to explain and allow proper decision making upon the system.

With this project its intended to create a program that applies the ID3 algorithm to a standardized text file and output the set of rules based on the decision tree created. It also allows the evaluation of a giving solution throughout the time considering it splits the data between training and test sets.

Decision Tree

Decision tree learning is the discipline to create a predictive model to map the different items in the set and respective target values and associate them in a way that is true to every element. This concept is used in statistics, data mining and machine learning due to its simple and effectiveness.

Decision tree is a abstraction to visually and explicitly represent the dependencies between attributes of elements in a set and their sorting on the tree determines a set of rules which explain and summarize the relations of all items throughout the set.

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance and each branch descending from that node corresponds to one of the possible values for this attribute.

Among the many advantages associated with ID3 are the simplicity of representation, the ability to perform well with large data sets in short time, the capability of use either numerical or alphanumerical data and can be validated using statistical tests.

However, there are some limitations to this concept, considering that a NP-complete problem is being solved by a greedy algorithm (choose of decision node) there are no guaranties to output the optimal decision tree for the problem. There's also a very important factor related with over-complex trees, where there is a over-specialization within the attributes called overfitting that can be solved by tree pruning.

ID3 Algorithm

The ID3 algorithm is a recursive procedure, which in each step there is a evaluation of a subset and there is the creation of decision node, based on metric called Information Gain, until the subset in evaluation if specified by the same combination of attributes and its values.

ID3 searches through the attributes of the training instances and extracts the attribute that best separates the given examples. If the attribute perfectly classifies the training sets then ID3 stops; otherwise it recursively operates on the n (where n = number of possible values of an attribute) partitioned subsets to get their "best" attribute. The algorithm uses a greedy search, that is, it picks the best attribute and never looks back to reconsider earlier choices.

It is presented now a pseudo-code implementation of ID3, as seen on Professor Luca Iocchi's slides for Machine Learning:

Input: Examples, Target attribute, Attributes

1. Create a Root node for the tree
2. if all Examples are positive, then return the node Root with label +
3. if all Examples are negative, then return the node Root with label −
4. if Attributes is empty, then return the node Root with label = most common value of Target attribute in Examples
5. Otherwise
    - A ← the "best" decision attribute for Examples
    - Assign A as decision attribute for Root
    - For each value vi of A
        - add a new branch from Root corresponding to the test A = vi
        - let Examples(vi) be the subset of Examples that have value vi for A
        - if Examples(vi) is empty then add a leaf node with label =most common value of Target attribute in Examples
        - else add the tree ID3(Examples(vi), Target attribute, Attributes−{A})

As we can see, the algorithm is pretty straight forward and, easily implementable. The only step not very explicit is the criteria used to choose the next node. This criteria is based in two metrics called Entropy and Information Gain.

Entropy is a measure in information theory to measure the impurity of a arbitrarily collection of items. For a given set S, being pi the probability of S belonging to class i, we have

$$\text{Entropy(S)} = \sum_{i=1}^{c} -p_i \log_2 p_i$$

As for Information Gain, we can say that is the expected reduction in entropy by splitting the collection S by a given outcome for attribute A, with an associated subset Sv.

$$\text{Gain(S, A)} = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

We can use this measure to rank attributes and build the decision tree where at each node is located the attribute with the highest information gain among the attributes not yet considered in the path from the root.

Project Implementation

After presenting the theoretical bases needed to understand the algorithm and the environment where it fits, we will continue by analyzing the approach used to create a General ID3 Problem Solver program.

The main objective of this project is to apply the algorithm previously explained to a set of items and output the rules that explain the system. It also includes a evaluation method that can be invoked to quantify the accuracy of the tree while it is being build.

System Architecture

Considering the necessary ability for the program to be able to function in any example, there was needed the introduction of a semantics in a external text file. In this way, we have to consider the existence of attributes, identifiers, classes and values for each item for each attribute.

This being said, the first line of the file should have a list containing a tag referring to the identifier, the attributes names and the final class which is the value in analysis, separated by a tab ("\t"). In the following lines the items are values to each attribute, class or identifier in the set.

This text file can, however, have any value for the class (not only Boolean answers) and there is no limit of different type of values for each attribute, nor limit to number of elements in the set.

After loading the necessary data from the text file, the information is organized in a ArrayList<Items> and the attribute names are stored in a ArrayList<String>. There is no need for any other type of organization because being the data stored in the file similarly as a table, it is easy to obtain any value on the table considering the item in evaluation (line on the table) and the attribute (column on the table). For future purposes, there is also created a collection of every different value assigned to each attribute in method countTypesColumn().

At this point it is implemented the recursive part of the algorithm, in order to evaluate each decision node at each time. To avoid termination problems in a purely recursive algorithm, a ArrayList<WaitingNode> was created to store every sub-tree still in need of evaluation and the termination clause assigned was to use a procedure called createAdjacencyTable() until there were no more items to evaluate. This collection was dynamically altered at each iteration of createAdjacencyTable().

This method is the one responsible to implement the algorithm in the same way it was explained before in the theoretical part of this report. It starts by calculating the entropy of the given set and then calculates the Information Gain for every available attribute in the given set. After this step, it is chosen the attribute with higher Gain to be the decision node at this instance of the tree. After the node is found, the different types of possible values are attached (edges in a tree) and calculated the entropy of each subset. If the entropy is zero, a leaf is found and the AdjacencyTable is updated. Otherwise, these subsets are added to ArrayList<WaitingNode> to be calculated in the future.

When all the tree is complete, there is a creation of the rules that explain the system. This are outputted as well as the intermediate Information Gain and Entropies.

The main auxiliary methods are entropySet(), gainAttribute() and createSubSet(). In createSubSet() we visit all the elements and save those that respect all the values given to the attributes so far. In entropySet() we count the number of items that achieve any of the class values in order to obtain the proportion necessary to calculate the entropy (see Entropy formula above). Finally, in gainAttribute() there is a assignment of values for every value not used before on the path since the tree's root, created a sub-set and applied the Information Gain formula. This method outputs only the best decision, being in this case the node with higher Information Gain.

Approach on ID3 Algorithm implementation

In this project, it was able to maintain some ID3 original features like the metrics (Entropy and Information Gain), sub-set re-arranging and the recursive algorithm. However, there are some discrepancies with the original, in the way that it's created a adjacency table instead of a tree abstraction (less memory used) and that the algorithm isn't purely recursive and with this we gained performance by not visit any tree, once we have always present the entire path since the root at any given moment.

Nevertheless, the objective has been achieved and ID3 fundamental procedures and dependencies respected, providing the correct output of the algorithm.

Solution Evaluation

Later on the project life, it was introduced the evaluation metric for the algorithm, in which the original data set is separated in a training set and a verification set. The training set is responsible to create the tree and the evaluation set is used to evaluate at each step of the algorithm it's accuracy.

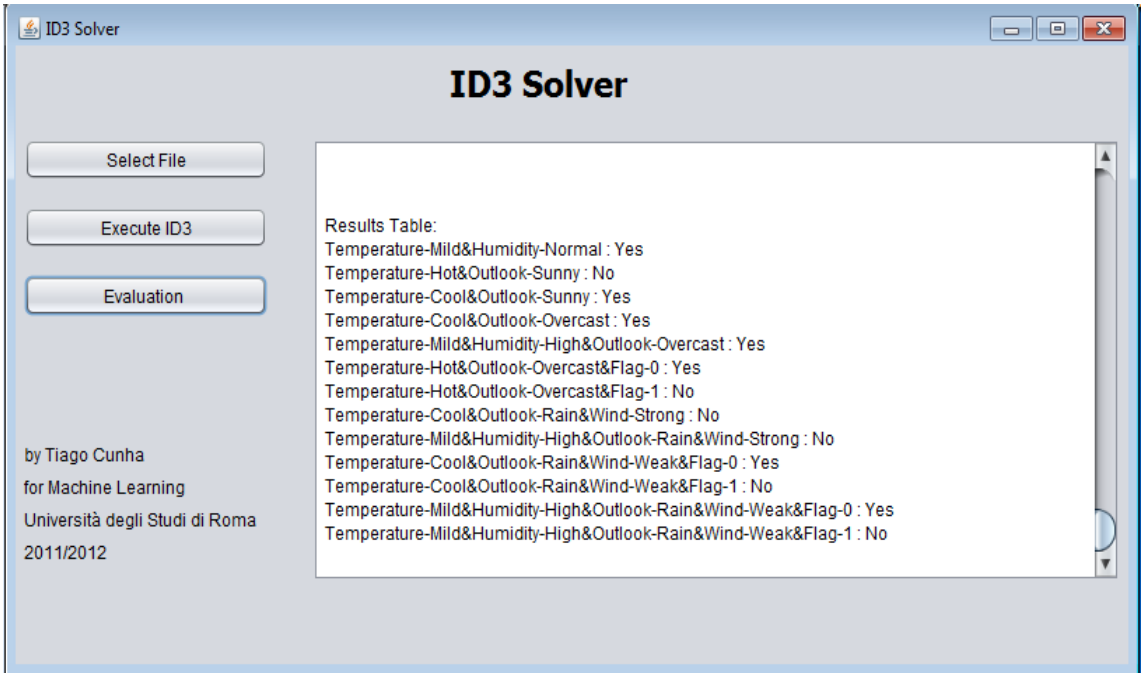
This evaluation occurs each time a new rule is added to the table and its evolution is shown in a 2D graphic when pressed the “Evaluation” button on the program's GUI.

GUI

The Graphical User Interface for this program is very simple. It gives the opportunity for the user to choose the input text file from any place in his computer and makes it possible to execute the ID3 algorithm and evaluate the solution proposed, as seen on the last topic.

When the “Select File” option is chosen it opens another window with a JFileChooser and returns the file path to use afterwards on the algorithm. By clicking the “Execute ID3” button, we are presented in the text area the output of each intermediate Entropy and Information Gain calculation as well as the Set Rules in the end. Finally, it's also possible to invoke a evaluation method explained in the previous topic.

In the following picture it's presented a snapshot of the GUI.



Conclusion

In conclusion, the ID3 algorithm has proven to be very easy and intuitively to implement, as it is general belief. With this project its possible to show the great performance in a large set and even if not provide a perfect decision tree, we can output a set of rules to explain the system.

As for the program itself, it is able to accomplish all the objectives proposed and work in a trustfully way. Also should be mentioned that the graphical interface of the program provides a interface as simple as the algorithm itself.

This way, we can conclude that a algorithm like this and in extension the entire Machine Learning discipline provides a useful and powerful platform of gathering knowledge and should be applied to as many fields of expertise as possible.

References

- *ID3 Algorithm*, Last Updated on 27 October 2011, Available at: http://en.wikipedia.org/wiki/ID3_algorithm
- *Decision Tree Learning*, Last Updated on 5 October 2011, Available at: http://en.wikipedia.org/wiki/Decision_tree_learning
- *CSE5230 Tutorial: The ID3 Decision Tree Algorithm*, Created by David McG., Last Updated on August 26, 2004, Available at: http://www.csse.monash.edu.au/courseware/cse5230/2004/assets/decisiontreesTute.pdf
- *The ID3 Algorithm*, Available at: http://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm
- *Extension and Evaluation of ID3 – Decision Tree Algorithm*, Created by Anand Bahety, Available at: http://www.cs.umd.edu/Grad/scholarlypapers/papers/Bahety.pdf
- *Decision Tree Learning*, Created by Luca Iocchi, Available at: http://www.dis.uniroma1.it/~iocchi/Teaching/ml/slides/3-DecTree_2p.pdf
- *The Machine Learning Dictionary*, Created by Bill Wilson, Last Updated on 7 September 2011, Available at: http://www.cse.unsw.edu.au/~billw/mldict.html#decisiontree