American Journal of Applied Mathematics
2016; 4(6): 330-339
http://www.sciencepublishinggroup.com/j/ajam
doi: 10.11648/j.ajam.20160406.20
ISSN: 2330-0043 (Print); ISSN: 2330-006X (Online)

Modelling of an Extended Brutedl Algorithm for Rule Extraction

Bolanle F. Oladejo, Rukayat Ayomide Erinfolami

Department of Computer Science, Faculty of Science, University of Ibadan, Oyo, Nigeria

Email address:
oladejobola2002@yahoo.com (B. F. Oladejo), erinfolamiayomide@gmail.com (R. A. Erinfolami)

To cite this article:
Bolanle F. Oladejo, Rukayat Ayomide Erinfolami. Modelling of an Extended Brutedl Algorithm for Rule Extraction. *American Journal of Applied Mathematics.* Vol. 4, No. 6, 2016, pp. 330-339. doi: 10.11648/j.ajam.20160406.20

Received: October 8, 2016; **Accepted**: November 22, 2016; **Published**: January 16, 2017

Abstract: Complex predictive models obtain very high predictive performance; however, it is difficult to explain their complex mathematical design. Rule extraction techniques help to understand their designs by generating structures like decision list. BruteDL algorithm generates decision list from a dataset, and also addresses the overlapping rule problem of most decision list learners. However; it does not harness the power of complex predictive model. It also performs poorly with small dataset. Hence, this work aimed to create rule extraction technique by extending BruteDL and to address its poor performance with small dataset. A rule extraction technique named BruteDL-RET (Brute Decision List-Rule Extraction Technique) was modeled and implemented. A finite state automaton was used to model the technique. A functionality to generate supporting training set was included. Artificial Neural Networks (ANN) is chosen as the complex predictive model which serves as the oracle because it decides the class of each example. Decision list was generated using both the predictive model and the dataset it was trained with. The implementation was done using Java programming language. We prove that on the average BruteDL-RET is able to generate more accurate rules than BruteDL. We report on the performance of our model using dataset of UCI repository.

Keywords: Predictive Models, Rule Extraction Techniques, Dataset, Artificial Neural Networks, Oracle

1. Introduction

With the growing popularity of Internet and continuous innovation in storage technology [1], decision makers often have easy access to numerous sources of information they might need about activities related to a decision [2]. According to Breton et al. [3], that approach is not necessarily the optimum approach to support decision makers in their duties. The first reason he stated was that, all available information might not be required to effectively carry out the task at hand. He also stated that processing all this information might exceed human capability. This has therefore resulted to the ubiquity of decision support systems which are based on predictive modeling [2]. Decision support system is the aspect of information systems discipline which is concerned with supporting and improving managerial decision-making activities [4]. Predictive models are used to identify potentially useful patterns in data, or to predict the outcome of some events [2].

A large number of predictive techniques exist, ranging from simple techniques such as linear regression, to complex powerful ones like artificial neural networks [2]. It is noteworthy to state that artificial neural networks have a more suitable inductive bias than competing predictive models [5]. The desired features of complex predictive models are that they provide a methodology to capture non-linear relationships and they obtain better predictive performance. However, even though complex predictive models are superior to the linear alternatives, they are described as 'black-box' because it is enigmatic to estimate their relationships due to their mathematical design. This is an undesirable property especially for practitioners who would want to confirm that the model uses system parameters correctly before choosing to use the model in practice [2], [6]. By extracting rules that mimic the black box (complex models) closely, it will be possible to provide insight into their design [1]. The techniques for doing this are called rule extraction techniques.

Rule extraction is concerned with producing a description of

the hypothesis of an opaque predictive model that is understandable and at the same time closely approximates the predictive model's behavior, given the predictive model and the data on which it was trained [7]. Quite a large number of rule extraction techniques generate either decision tree or decision list as their output because they are more comprehensible than the other representations [5]. Even though decision trees are more common [5], decision lists are more comprehensible [8]. BruteDL is an algorithm for learning decision list (i.e., it generates decision list from a set of examples). BruteDL addressed one of the limitations which are common to decision list learners, the rule overlap problem, by using homogeneous rules (this will be explained later in section 2). BruteDL cannot be classified as a rule extraction technique because it does not make use of any predictive model in its operation but only uses a training set. In this work, a rule extraction algorithm called BruteDL-RET (Brute Decision List-Rule Extraction Technique) which is based on the BruteDL algorithm will be presented to address the limitation of BruteDL algorithm in performing efficiently on small training set. The section 2 of this journal addresses theoretical background, section 3 gives a description of the methodology use for achieving BruteDL-RET, section 4 presents the results of our experiment as well as the findings, section 5 contains the conclusion, section 6 contains the future research and section 7 contains the references.

2. Literature

This section provides theoretical background related to the work. Rule extraction technique has its background in Artificial Intelligence, Machine learning, Data Mining and Knowledge Discovery, hence they will be explained in this section.

2.1. Artificial Intelligence

Intelligence is the computational aspect of the capability to achieve goals. Artificial Intelligence (AI) relates to tasks that involve higher mental processes such as creativity, building analogies, solving problems, language processing, pattern recognition, deduction, classification, optimization, learning, induction, knowledge, etc. [9]. Intelligent behaviors include; perceiving one's environment, learning and understanding from experience, thinking abstractly, using analogies and many more [9]. There are approaches to Artificial Intelligence (AI); the turing test approach (where computers act like human), the cognitive modeling approach (where computers think like humans), the "laws of thought" approach (where computers think rationally) and the rational agent approach (where computers act rationally) These approaches will be explained in detail in chapter 2 [10].

Today, AI has its application in so many areas. In autonomous planning and scheduling, NASA's (National Aeronautics and Space Administration) Remote Agent program was the first on-board autonomous planning program to control the scheduling of operations for a spacecraft which was a hundred million miles from Earth. In game playing, IBM's Deep Blue was the first computer program to beat the world chess champion in a chess match. It bested Garry Kasparov by a score of 3.5 to 2.5 in an exhibition match. In diagnosis, medical diagnosis programs which are based on probabilistic analysis have been able to perform as perfectly as an expert physician in several areas of medicine. These are only few examples of the application of AI today [10].

2.2. Machine Learning

Machine learning is defined as a set of methods that have the capability to automatically discover hidden patterns in data, and adopt the revealed patterns in predicting future data, or carry out other kinds of decision making under uncertainty [11] This is an era of big data where extremely large amount of data is made available every second [11]. Machine learning has two main types; the predictive or supervised learning approach and the descriptive or unsupervised learning approach. There is also a third type which is not so common; the reinforced learning.

In predictive learning approach, the focus is to learn a mapping from inputs x to outputs y, given a labelled set of input-output pairs. D is called the training set, and N is the number of training examples. In the simplest setting, every training input is a D-dimensional vector of numbers, which represents, say, color and weight of a thing. These values are called attributes, features or covariates. Generally, however, it could be a complex structured object, a sentence, an email, an image, a time series, a molecular shape, a graph, etc. The form of the output or response variable can be anything in principle, but most methods take it to be a categorical or nominal variable from some finite set, or that is a real-valued scalar (such as population). When it is categorical, the problem is referred to as classification or pattern recognition and when it is real-valued, the problem is referred to as regression.

In descriptive or unsupervised learning approach, only inputs are given, and the goal is to find interesting patterns in the data. This is sometimes referred to as knowledge discovery. This problem is not as well-defined as supervised learning since we do not know what kind of pattern to look for, and there is no vivid error metric to use.

Reinforcement learning, the third type of machine learning, is useful for learning how to behave when occasional reward or punishment signal is given.

2.3. Data Mining and Knowledge Discovery

According to Mohammed and Wagner [12], "Data mining is the process of discovering insightful, interesting, and novel patterns, as well as descriptive, understandable, and predictive models from large-scale data". Data mining is also popularly known as Knowledge Discovery in Databases (KDD) but in the real sense, data mining is only a part of the knowledge discovery process. The data used in data mining can often be represented as a data matrix, with n rows and d columns, where rows stand for entities in the dataset and columns stand for attributes or properties of interest. The

diagram in Figure 1 below shows data mining to be a step in an iterative knowledge discovery process [12].

The knowledge discovery in databases process is made up of a few numbers of steps starting from collection of raw data to some form of new knowledge. The iterative process is made up of the following steps; Data cleaning, Data integration, Data selection, Data transformation, Data mining, Pattern evaluation, Knowledge representation.

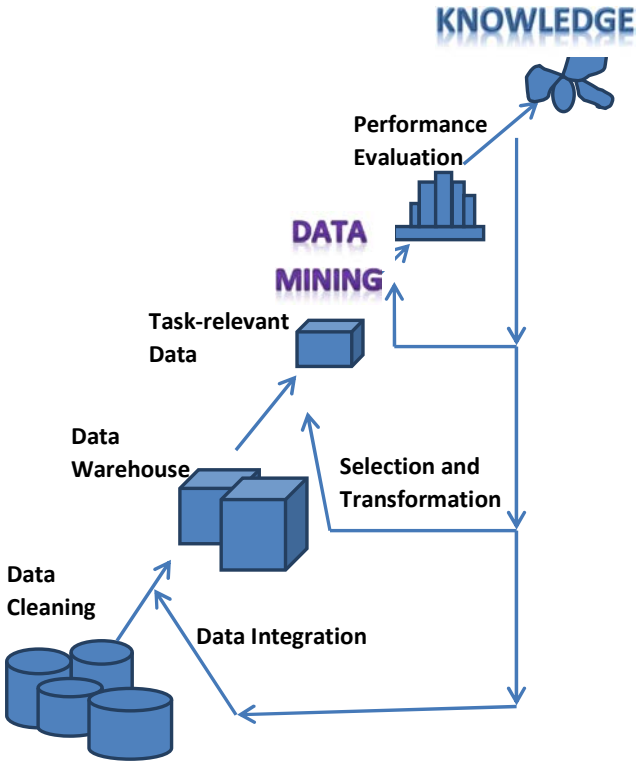

Figure 1. *Data mining as the core of knowledge discovery process [12].*

3. Methodology

BruteDL takes as input a set of training data from which it approximates probability distribution P and determines which rules are homogeneous. It uses LaplaceAccuracy to calculate the approximation of the actual accuracy of a rule [13]. Let r be a rule classifies r_p training examples rightly out of the r_n training examples it matches. Let |G| represent the number of goal classes in the training set. The LaplaceAccuracy of the rule r is given by:

$$LaplaceAccuracy(r) = \frac{r_p+1}{r_n+|G|} \quad (1)$$

Once the estimate of the accuracy of each rule has been defined, it is then possible to confirm if a rule is homogeneous. The accuracy of a homogeneous rule does not change when more conjuncts are added to it. Homogeneity can thus be checked by comparing the Laplace Accuracy of all the rule's specializations. A rule is considered homogeneous if all specializations have roughly the same laplace Accuracy. We check for statistically significant differences in Laplace Accuracy using a χ^2 test.

It is desirable that rules learned by BruteDL do not contain irrelevant conjuncts. An irrelevant conjunct is that conjunct which does not affect the accuracy of a rule. Any rule which does not contain irrelevant conjuncts is thus referred to as minimal. Restricting BruteDL to learning only minimal rules does not affect the class of concepts it can learn since, for every non-minimal homogeneous rule there is a minimal homogeneous rule with similar accuracy and greater coverage that is formed with the use of some subsets of the original rule's conjuncts. The relevance of a conjunct is confirmed by checking if the accuracy of the rule changes when the conjunct is dropped. χ^2 test is again used to ensure that any differences in accuracy that are discovered are significant.

The next section goes into the approach for rule extraction that is discussed in this journal.

4. Modeling of an Extended BruteDL Algorithm

This work initiated the modeling of an extended BruteDL as a result of the limitation of BruteDL which is termed Brute Decision List-Rule Extraction Technique (BruteDL-RET). BruteDL algorithm was adopted as the decision list learning strategy for the algorithm presented in this section because of its ability to address the issue of overlapping rules in most decision list learner. Rules, in the form of decision list, will be extracted from a black box complex predictive model using BruteDL algorithm as the induction strategy. A trained complex predictive model will be passed into the algorithm as an oracle. The idea of oracle was taken from the algorithm developed by Craven [5] which he named "TREPAN". Instead of the class label of an example to be decided by the supplied training set, it is rather decided by the provided predictive model. This way, the decision list generated will be a description of the behavior of the predictive model which served as an oracle. Also, in cases where the provided training is below a given threshold (threshold will be determined by user), supporting training set will be generated. This is in attempt to address BruteDL's inefficiency with small training set thereby ensuring that BruteDL-RET does not inherit that problem from BruteDL.

4.1. Framework of BruteDL-RET

The very first and basic activity to be performed by the user of the system is the training of the artificial neural networks which will be used as the oracle. To do this, the steps involved in the knowledge discovery process are carried out which are: Data cleaning, Data integration, Data selection, Data transformation, Data mining, Pattern evaluation and Knowledge representation. The data mining step was carried out by first setting up a Feed Forward neural network, making use of appropriate number of input layers (the provided training gives this information), desired number of hidden layers and the appropriate number of output layer (the provided training also gives this

information). Afterwards, Back propagation learning method is used to learn useful pattern from the provided training set. At this end of these activities, we would have had our predictive model (oracle) from which rules (in the form of decision list) will be extracted. Figure 2 gives an overall picture of the predictive model development and eventual use. Figure 3 displays a diagram of simple feed forward neural network. The feed forward neural network is made up of three layers: the input, hidden and output layers. The input layer consists of six nodes which represent each attribute represented by the given dataset and the output layer consists of two nodes which represent the two target classes that are available for that particular dataset.

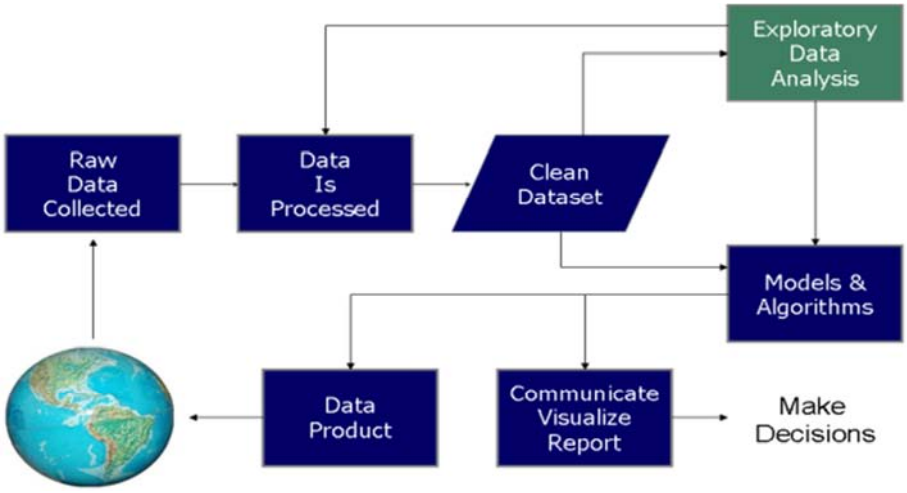

Figure 2. *Framework of the development process of a predictive model and its eventual use [15].*

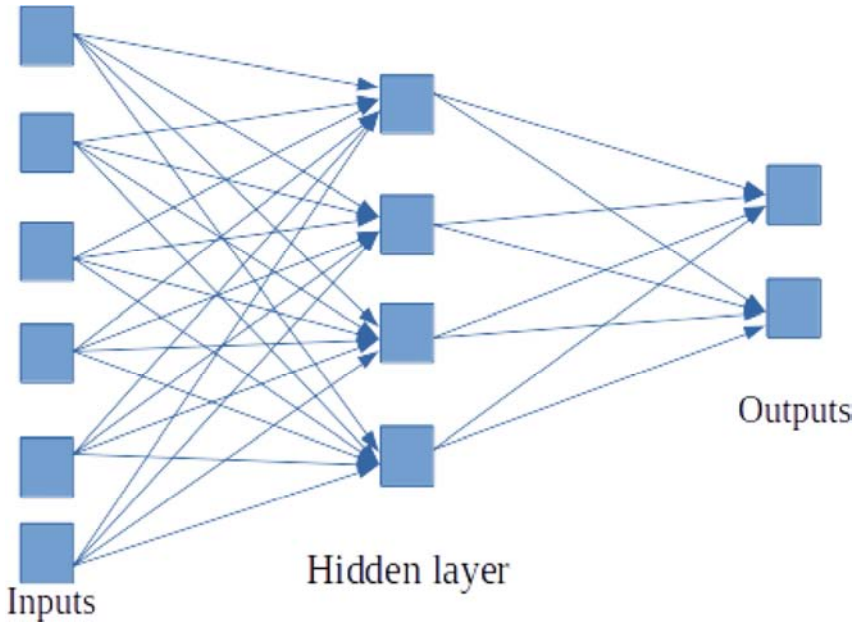

Figure 3. *A simple feed forward neural network [16].*

BruteDL-RET has two major components;

1. The generation of supporting training data
2. Rule extraction process

Inputs to the algorithm are; the training set, the oracle, threshold and number and name of classes. Figure 4 presents a flowchart of the procedure involved in BruteDL-RET. Figure 7 presents the pseudo code for BruteDL-RET.

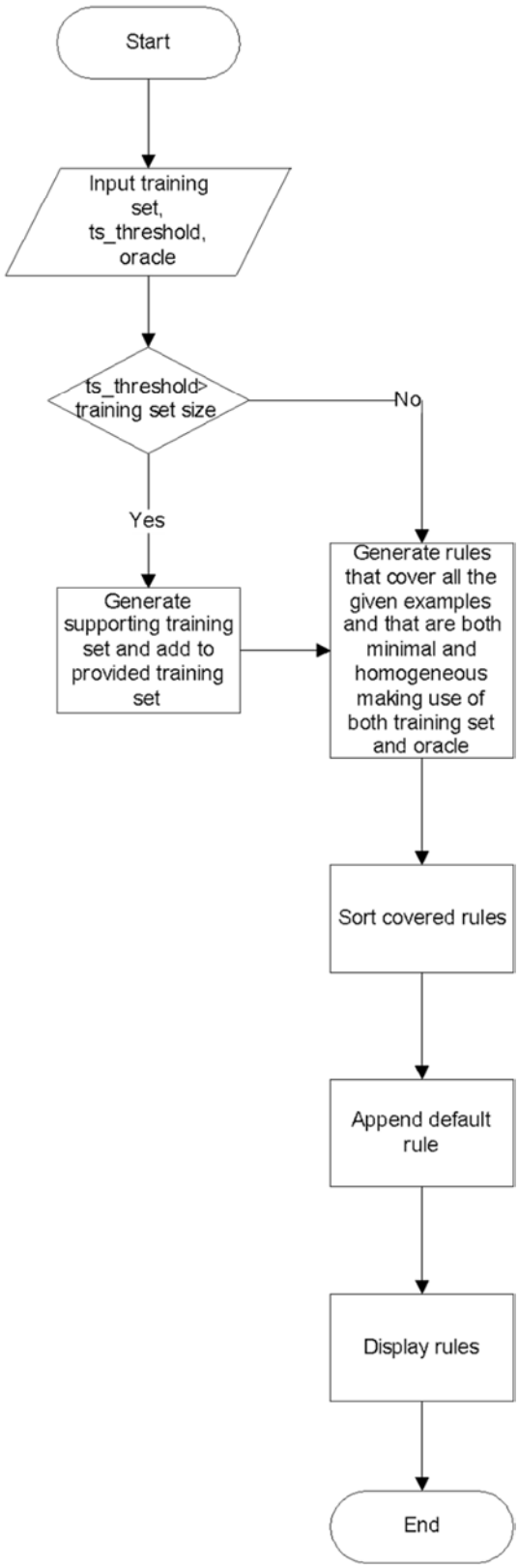

Figure 4. *Flowchart of the operation of BruteDL-RET.*

All the strategies used in BruteDL in generating decision list will still be adopted. The strategies are in three stages;

1. Select the best rule for some examples. Depth first search is used in selecting a single rule from the search space and search is limited to a fixed depth if searching the entire space is too costly [14].

2. Check if that rule is minimal. A minimal rule is one which does not contain irrelevant conjunct [14].

3. Check if that rule is homogeneous.

4.2. Formalization of BruteDL-RET

The finite automaton in figure 5 is a five tuple (Q, Σ, δ, S_0, F) describing the process of transmission in BruteDL-RET from one state to the other.

Q is the set of all states in the automaton, they are represented by circles. Hence:

Q={S_0, S_1, S_2, S_3, S_4, S_5} where:

S_0=initial state where dataset is read and processed

S_1=state where predictive model is trained

S_2=state where additional training set is generated

S_3=state where rule is extracted using BruteDL

S_4=state where the target of an example is either retained or changed depending on what the oracle returns

S_5=state where extracted rules are given

Σ=is the string of valid inputs that brings about change from state to the other.

Q X Σ → Q is the transition function for the automaton

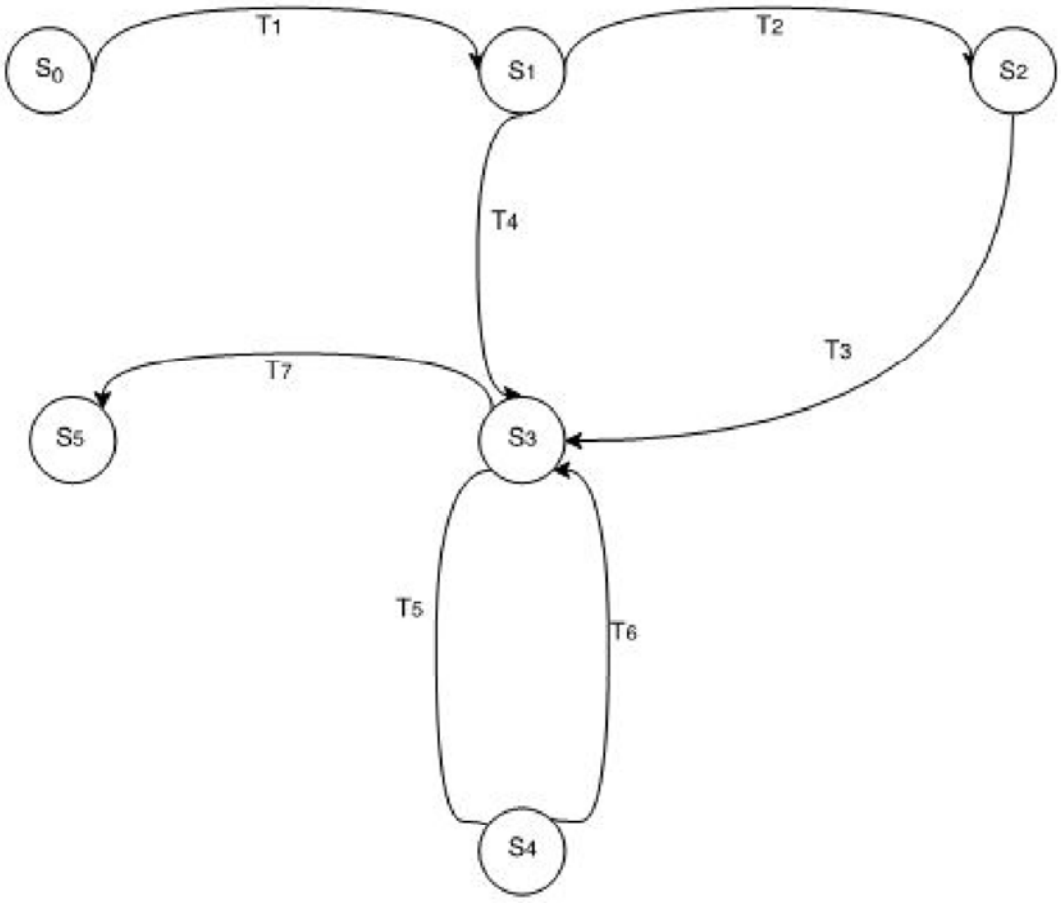

Figure 5. *Finite Automaton of the framework for BruteDL-RET.*

4.3. Generating Supporting Training Data

The supporting data is meant to act as supplement to a small training data. Supporting data will only be generated if the specified threshold is greater than the size of provided training data. Figure 6 presents the algorithm which generates supporting training set. It calls two major functions: CompValueN () and CompValueS (). CompValueN () means "compute value for numeric" and CompValueS () means "compute value for string". CompValueN () generates the supporting dataset for attributes with numeric or floating point values and CompValueS () generates the supporting data for attributes with symbolic attributes, for instance, no, married, etc. Careful consideration has been given to the presented pseudo code to ensure that the generated supporting data examples are replica of the examples in the training data. This is to ensure that the supporting data still represents the domain from which the training data is obtained.

In the supporting training set generation process, user specified dataset is read into the program and values of each attribute for each example in the dataset are grouped under the different target classes where they belong. The number of supporting training set to generate is computed by subtracting the number of given dataset from the user specified threshold, after which the number of examples to generate for each class is determined using the percentage of that class that appears in the given dataset. Data generation is done per attribute; the attribute is first checked if it is a numeric or string attribute. The essence of this is to know whether CompValueN () is to be called or CompValueS ().

4.4. Extracting Rules from an Oracle (i.e. Predictive Model)

The rule extraction process involves generating a decision list using a set of training set which would be used to train a predictive model and the trained predictive model itself. For ordinary decision list learner, in the process of generating a decision list, the training set provided determines the class label of each instance of the training set, however, in this rule extraction process, the oracle makes this decision. Figure 7 gives the pseudocode of the rule extraction process.

5. Results and Findings

Testing was done in four different categories using four different datasets. The four datasets used is a subset of the datasets used by Segal and Etzioni [14] in their work: "Learning Decision Lists Using Homogeneous Rules" where BruteDL algorithm was first presented. As stated in the abstract, the four datasets were obtained from UCI repository. The first category was BruteDL-RET with large dataset (the use of an oracle as well of generating supporting dataset to make up the large dataset), followed by BruteDL-RET with small dataset (the use of an oracle without generating supporting dataset), and then BruteDL with large dataset and finally, BruteDL with small dataset. Tables 1, 2, 3 and 4 presents the result of our experiment. The main essence of testing is to compare the accuracy of BruteDL-RET with that of BruteDL and also to see the impact of supporting data generation. The average percentage accuracy of the decision list generated at each category is computed. The percentage prediction accuracy is calculated as:

$$Accuracy = \frac{N_{cp}}{N_{cp}+N_{ip}} * 100 \quad (2)$$

where:

N_{cp}=Number of correct predictions

N_{ip}=Number of incorrect predictions

```
GenerateSupportingData()
        Inputs:
                No of classes
                Threshold
                Training data

        Algorithm:
        Sort training data according to class
        Note the start and end indices of each class
        Compute no of examples for each class and percentage
        Compute no of supporting data to generate
        Use percentage to compute no of supporting data for each class
        n := No of training data
        For i := 1 to no of classes
                s := no of supporting data to generate for class i
                For a := 1 to no of attributes
                   If (numeric attribute)then
                        h := highest values
                        sm := smallest value
                        CompValueN(sm, h, n, s, a)
                   End
                   Else
                        CompValueS(s, a)
                   End

                End
                For j = 1 to s
                        TD[n + j][target.a] = valueof class i
                End
                n = n + s
        End
End
```

Figure 6. *Pseudo code for the method to generate.*

```
BruteDL – RET()
        Global variables
                Training set, oracle(neural network)

        Algorithm:
        If Length(training set) < threshold
                Generate supporting data and add to training set
        Train oracle with training set
        DecisionList := MakeEmptyDL()
        DecisionList := BruteSearch(MakeEmptyRule( ), oracle)
        Sort DecisionList
        Add default rule to decision list
End

BruteSearch(rule)
        If length(rule) ≥ Maximum length then Exit
        StartTest := FollowingTest(LastTest(rule))
        For test := StartTest to LastTest DO
                newRule := AddConjunct(rule, test)


                If BestForSomeExample(newRule)AND
                        IsMinimal(newrule)AND
                        Homogeneous(newRule, newRule)
                        Then Insert(newrule, DecisionList)
                If Not PruneRule(newrule)
                        Then BruteSearch(newrule)
        End
End

Homogeneous(ParentRule, rule): boolean
        If length(rule) >= MaxLength Then return true
        If rule == ParentRule
                Then StartTest = 1
                Else StartTest = FollowingTest(LastTest(rule))
        For test := StartTest to LastTest DO
                newrule := AddConjunct(rule, test)
                If not SimilarAccuracy(Parentrule, newrule)Then
                        Return False
                Else if not Homogeneous(ParentRule, newrule)Then
                        Return False
        End
        Return True
End

isMinimal(rule): boolean
        For test in Conjunct(rule)Do
                ParentRule := DeleteConjunct(rule, test)
                If SimilarAccuracy(ParentRule, rule)
                        Then return false
        End
        Return True
End
```

Figure 7. *The rule extraction process based on BruteDL algorithm. (Originally from Segal and Etziono, 1994).*

Table 1. *The result from testing with diabetes dataset.*

Algorithm Dataset	BruteDL-RET	BruteDL
Large	94.0	43.0
Small	37.2	46.3

Table 2. *The result from testing with iris dataset.*

Algorithm Dataset	BruteDL-RET	BruteDL
Large	100.0	53.6
Small	100.2	59.3

Table 3. *The result from testing with glass dataset.*

Algorithm Dataset	BruteDL-RET	BruteDL
Large	43.8	43.8
Small	45.0	43.7

Table 4. *The result from testing with voting dataset.*

Algorithm Dataset	BruteDL-RET	BruteDL
Large	100.0	25.0
Small	100.2	28.0

6. Discussion

The use of oracle in the process defined in BruteDL algorithm (the use of oracle transformed BruteDL into a rule extraction technique, hence, the new name BruteDL-RET) impacted the accuracy of the induced decision list. The output of BruteDL-RET is actually the description of the oracle used. From the results in section 5, we can conclude that the oracle has helped in fine-tuning the relationship among the attributes in each given dataset. The reason BruteDL would not perform as good as BruteDL-RET is just that, it is an ordinary decision list learner that induces decision list from a given dataset. BruteDL-RET produces better result because it harnesses the capability of a complex predictive model (the oracle). Also, from the results, it can be said that the use of supporting training set had considerable impact on the overall accuracy of the induced decision list. One the average, BruteDL-RET with large dataset had the highest percentage accuracy.

7. Implication to Research and Practice

In this work, we were able fine-tune the irregularities in the relationship among datasets through the use of a complex predictive model (ANN in this case), thereby ensuring that rules with better predictive accuracy are generated. Also, the supporting training set functionality made it possible to have enough training set to generate rules, thereby improving efficiency even when the user provided training set that is small.

8. Conclusion

This paper presented BruteDL-RET, a rule extraction technique. It adopted the rule induction strategy of BruteDL which addresses the overlapping rule problem of most decision list learners. It also included a functionality to generate supporting training set, which is an attempt to address the address the inefficiency of BruteDL with small dataset. ANN was chosen as the complex predictive model from which rule was extracted. It was shown in the result that on the average, BruteDL-RET outperformed BruteDL.

9. Future Research

It is recommended that in addition to obtaining the percentage accuracy of the decision list generated by both BruteDL-RET and BruteDL, the two algorithms should be tested on some other dataset in order to further validate the obtained accuracy. This will help increase the confidence of practitioners in adopting rule extraction techniques in their decision making activities.

Also, it is recommended that BruteDL-RET be used with other complex predictive models such as SVM (Support Vector Model). In this work, the adopted complex predictive model was ANN.

References

[1] Martens D., Huysmans J., Setiono R., Vanthienen J. and Baesens B. (2008). Rule Extraction from Support Vector Machines: An Overview of Issues and Application in credit scoring. Studies in Computational Intelligence (SCI) 80, 33–63 (2008)

[2] Rikard, K. (2009). Predictive techniques and methods for decision support in situations with poor data quality. University of Boras, School of Business and Informatics, University of Skovde, informatics Research Center, University of Orebro, School of Science and Technology. Örebro University, 2009., 112 p.

[3] Breton R., Paradis, S. and Roy, J. (2002). Command Decision Support Interface (CODSI) for human factors and display concept validation, *International Conference on Information Fusion* pp. 1284–1291.

[4] Arnott, D., Graham, P., Peter, o'Donnell and Gemma, D. (2004). An analysis of decision support systems research: Preliminary Results. Decision Support in an Uncertain and Complex World: *The IFIP TC8/WG8.3 International conference*.

[5] Craven M. (1996). Extracting comprehensible models from trained neural networks. University of Wisconsin-Madison. Published by University of Wisconsin-Madison Department of Computer Sciences

[6] William, A. Young II, Gary, R. W., Maimuna, H. R. and Harry, S. W. II (2011). An investigation of TREPAN utilizing a continuous oracle model, Int. J. Data Analysis Techniques and Strategies, Vol. 3, No. 4.

[7] Johan, H., Bart, B. and Jan, V. (2006). Using Rule Extraction to Improve the Comprehensibility of Predictive Models, Katholieke Universiteit Leuven Department of Decision Sciences and Information Management.

[8] Quinlan J. R. (1986). Simplifying Decision Trees. Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[9] RC Chakraborty (2010). Artificial Intelligence–Introduction. http://www.myreaders.info/html/artificial_intelligence.html, received in 2014

[10] Russell, S. and Norvig, P. (2003). Artificial Intelligence, A modern approach. Pearson Education, Inc., Upper Saddle River; New Jersey.

[11] Kevin, P. M. (2012). Machine Learning–A Probabilistic Perspective. Published by MIT Press, Cambridge, Massachusetts, London, England.

[12] Mohammed, J. Z. and Wagner, M. Jr. (2014). Data Mining and Analysis, Fundamental Concepts and Algorithms. Published by Cambridge University Press 2014.

[13] Peter Clark and Tim Niblett (1989): The CN2 Induction Algorithm. italicThe Turing Institute, 36 North Hanover Street, Glasgow, G1 2AD, U. K. Machine Learning 3: 261-283.

[14] Segal, R. and Etzioni, O. (1994). Learning Decision Lists Using Homogeneous Rules. Department of Computer Science and Engineering, University of Washington.

[15] Prem Nath, https://premnath321.com/author/premnath321/, downloaded in October, 2015

[16] Christos S. and Dimitrios S., http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#An Example to illustrate the above teaching procedure), downloaded in October, 2015.