# Enhanced SPRINT Algorithm based on SLIQ to Improve Attribute Classification

Benchie Maribao[1], Bobby Gerardo[2], and Bartolome T. Tanguilig, III[1]

***Abstract*—**Classification is the most commonly applied data mining technique that had been the focus on research extensively in the past due to its significance. The classification algorithm is complex, and it is an open problem. Though classification is a well-studied problem for the past years, no such classifiers are perfect, and those drawbacks can be used as a basis for the researchers for the enhancement of such classifier. SPRINT is one of them, and the issue to be resolved are rewriting and resorting of attributes in classification. This paper presented the Enhanced SPRINT algorithm based on SLIQ pre-sorting technique to address the identified drawbacks in rewriting and resorting of attribute lists in each node to improve attribute classification. In experimental evaluation, it shows that the performance of the proposed algorithm had shortened the time taken for attributes classification as compared to SPRINT algorithm.

***Keywords*—**Classification Algorithm, Data Mining, Decision Tree, SPRINT, SLIQ.

## I. Introduction

CLASSIFICATION is the most commonly applied data mining technique that is effective for data mining analysis. This can be used to describe and extract models from data classes and predict future data [1]. The analysis and forecasts of these data provide good decision support in various industries. Classification algorithms have used for publications like churn prediction, fraud detection, artificial intelligence, medical aspects and even in academic institutions adapted for predicting or forecasting performance rating, and decision support [2]. There are many classification algorithms available such as Ruled-based classifier, neural networks, support vector machines and naïve Bayes classifiers, but the most commonly used is Decision Tree [3].

A decision tree is widely used as the practical method for inductive inference over supervised data and represents a procedure for classifying continuous and categorical data [2]. It is constructed fast, and models are simple and easy to understand [4]. Decision trees are easily converted into SQL statements [1] and finally, decision tree classifiers obtain better accuracy compared with other classification methods [5].

SPRINT is a decision tree classifier. It is fast, and scalable. In constructing the tree, it partitioned the data sets recursively using breadth-first greedy technique until each partition belongs to the same leaf node. It uses two data structure: attribute list and histogram which is a disk resident that make SPRINT capable in handling huge of databases and removes the data memory restrictions on data. It also handles both continuous and categorical attributes [6][7]. Compared to other classifiers, it is more effective. Its weakness is in the process of classifying attribute lists where it iteratively rewrites and resorts attribute lists in each node. This process takes time for the SPRINT to classify attribute lists [8].

This problem serves as the basis to enhance SPRINT algorithm based on SLIQ pre-sorting technique to answer the rewriting and resorting of attribute lists during the classification process, and to test the performance of the enhanced algorithm in terms of time for classifying attribute lists with the SPRINT algorithm.

The rest of the paper is organized as follows: In Section 2, it covers the related literatures which discuss the decision tree, SPRINT, and SLIQ algorithm. Section 3, discuss the sequence on how SPRINT algorithm is improve based on SLIQ pre-sorting technique. Section 4, the experimental evaluation of the proposed algorithm and SPRINT algorithm, and concluded summary in Section 5.

## II. Related Literatures

This section discusses the general structure of decision tree, functionalities and capabilities of SPRINT and a brief description of SLIQ as the basis for the improvement of the proposed algorithm.

### *A. Decision Tree*

Most decision tree classifiers perform classification in two (2) phases, the growth phase and prune phase [4][10]. In the growth phase, the tree is built by recursively partitioning the data until each partition belongs to the same leaf or node "pure". The form of split used to partition the data depends on the type of the attribute used in the split. The Splits for a *continuous attribute* $\mathbf{A}$ are the form $value(\mathbf{A}) \leq \mathcal{X}$ where $\mathcal{X}$ is a value in a domain of $\mathbf{A}$ and the splits for a *categorical attribute* $\mathbf{A}$ are the form $value(\mathbf{A}) \in \mathbf{X}$ where $\mathbf{X} \subset domain(\mathbf{A})$[6]. A binary split is being used, but it can be extended to handle other technique of splitting. If the tree has been fully grown, it is pruned to reduce the size of tree and overfitting to improve predictive accuracy [12].

[1] Technological Institute of the Philippines

*B. SPRINT*

SPRINT stands for Scalable Parallel Classifier. It is fast and scalable. In constructing the tree, it partitioned the data sets recursively using breadth-first greedy technique until each partition belongs to the same leaf node. It handles both continuous and categorical attributes. It also uses two data structure the attribute list and histogram.

The attribute list initially creates for each attribute, and those entries are called *attribute records*, which contains attribute value, a class label, and the index of the record (*rid*). The initial lists for continuous attributes are sorted by attribute value once when first created. The initial lists are associated with the root of the classification tree. As the tree is grown the attribute lists belonging to each node are partitioned and associated with the children.

Two histograms are associated with each decision tree node for continuous attributes. These are denoted as $\boldsymbol{C}_{above}$ and $\boldsymbol{C}_{below}$, which can be used to capture the class distribution of the attribute records at a given node. The $\boldsymbol{C}_{below}$ maintains the distribution for attribute records that have been processed, while $\boldsymbol{C}_{above}$ maintains it for those that have not processed. For categorical attributes one histogram associated with a node. However, only one histogram is needed, and it contains the class distribution for each value of the given attribute.

*Finding Split-points*

In growing phase, the goal of each node is to determine the split point that "best" divides training records belonging to each leaf. The split point value depends on how the classes are being separate. The splitting index that used to evaluate the goodness of the split is the *gini index*. For the dataset $\mathbf{X}$ containing examples from **n** classes, *gini*($\mathbf{X}$) is defined as $gini(\mathbf{X}) = 1 - \Sigma p^2_j$ where $\mathbf{p_j}$ is the relative frequency of *class* **j** in $\mathbf{X}$. if $\mathbf{X}$ is split into two (2) subsets $\mathbf{X}_1$ and $\mathbf{X}_2$, the divided data index is given with the formula as $gini_{split}(\mathbf{X}) = \mathbf{(n1/n)}gini(\mathbf{X_1}) + \mathbf{(n2/n)}gini(\mathbf{X_2})$ [6].

In finding the best split point for a node, every attribute lists is scan, and split is being evaluate based on the attributes. The attribute with the lowest value of split point on the gini index is then be used to split the node. For the continuous attributes, the mid-points between two (2) consecutive attributes in the dataset is the split candidate points. In determining the attribute split for a node, the histogram $\mathbf{C}_{below}$ is initialized to zeros whereas $\mathbf{C}_{above}$ is initialized with the class distribution for the node in all records. For the root nodes, class distribution is acquired during the sorting process while the other nodes, the class distribution are acquired once the node is created. As attribute records are process, the record of $\mathbf{C}_{below}$ and $\mathbf{C}_{above}$ are updated. After each record has been processed, a split between values (attribute records) have and have not yet seen is evaluated. Note that $\mathbf{C}_{below}$ and $\mathbf{C}_{above}$ have all necessary information to compute the *gini index*. The attribute for each candidate split-points are evaluated in a single sequential scan of the equivalent attribute list. During scanning, if a wining split point was found, it is automatically saved and the $\mathbf{C}_{below}$ and $\mathbf{C}_{above}$ histograms are de-allocated before processing the next attribute [6].

*Performing the Split*

When the best split point has been found for a node, it performs the split by creating a child node and the attribute records will be divided between them. Splitting the attribute list of the winning attribute is straightforward. It scans the lists, apply the split test, and then move the records to two new attribute lists-one for each new child [6].

*C. SLIQ*

SLIQ stand for a Supervised Learning in Ques. It can handle both numeric and categorical attributes. In growing phase, it uses the pre-sorting technique to reduce cost in evaluating numeric attributes. This process is integrated with a breadth-first tree growing technique. It also uses a fast sub-setting algorithm for determining splits for categorical attributes. In pruning phase, the algorithm is based on the Minimum Description Length principle [11].

*Pre-Sorting Technique*

When finding the best split in decision tree node, sorting time is the main factor to consider [12]. The initial technique used in SLIQ is to eliminate the resorting of data at each node. Instead, the training data are sorted just once at the beginning of the tree growth phase.

To attain this pre-sorting technique, it uses the following data structures. In the training data, it creates a separate list (class list) for each attribute and serves as the class labels attached to the examples. An entry in an attribute list has two fields; an attribute value and an index into the class list. An entry of the class lists also has two fields; a class label and a reference to a leaf node of the decision tree. In the training data, the *i*th entry of the class list corresponds to the *i*th example. Each leaf node represents a partition of the training data. The partition is define by the combination of the predicates on the path through the node to the root. Thus, the class list can identify the partition to which an example belongs. Figure 1 shows the illustration of the data structures, before and after pre-sorting [12].

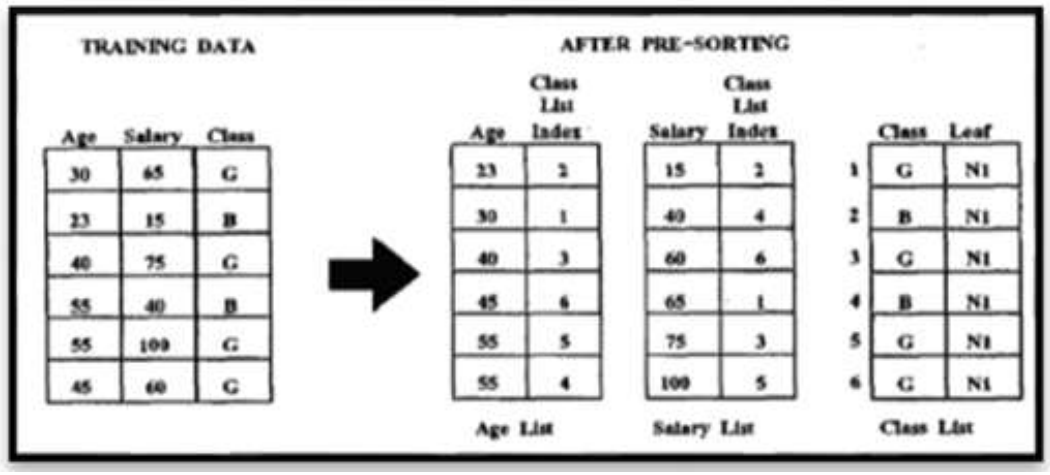


Fig. 1: Example of Pre-sorting Technique

## III. Enhanced SPRINT Algorithm

As discussed in the previous section, SPRINT uses two data structures for the growth phase. The attribute list and histogram. It initially creates an attribute list for each attribute; the attribute value, a class label, and the index of the record. These values were acquired in an entry called attribute records. The initial lists for continuous attributes are sorted by attribute value once when first created. The initial lists are associated with the root of the classification tree. As the tree is grown the attribute lists belonging to each node are partitioned and associated with the children. The classification of attribute lists is recursively rewrites and resorts attribute lists in each node. This process takes time for the SPRINT to classify attribute lists.

The pre-sorting technique of SLIQ will be used as a basis for the development of the new algorithm and can be named as Enhanced SPRINT Algorithm. The proposed algorithm composed of three (3) functions (PreSorting, EvaluateSplit, and MakeTree).

**The Enhanced SPRINT Algorithm is as follows:**

**PreSorting ( )**

1. In Training Data, determine all the attributes then store the values in an array.
   - 1.1 Create an Attribute list (value, ID, and class) for each attribute of the training data.
   - 1.2 Create a class list for each attribute
   - 1.3 Iterate all the training samples.

**repeat**

**EvaluateSplit ( )**

2. Evaluate the best split-point using ***Gini Index***
   - 2.1 Determine all the Gini Index of the attributes of the Training Data
   - 2.2 Get the smallest Gini Index
   - 2.3 Determine the ***equivalent row*** then get the **Midpoint**

**MakeTree ( )**

3. Split the attributes based on the Midpoint values

**until** (no record to be split)

**end**

### 3.1 PreSorting ( )

The **PreSorting ( )** covers the pre-sorting technique for attribute classifications. Based on the Training Datasets, it will determine all the attributes and store in an array. It creates an attribute list for each attribute and a class list then training samples will iteratively be done.

### 3.2 EvaluateSplit ( )

The **EvaluateSplit ( )** deals for the evaluation of the best split point using Gini Index. The Gini Index is used to evaluate the "goodness" of the alternative splits for an attribute. For the dataset $\mathbf{X}$ containing examples from **n** classes, $gini(\mathbf{X})$ is defined as $gini(\mathbf{X}) = 1 - \Sigma p^2_j$ where $p_j$ is the relative frequency of *class* **j** in $\mathbf{X}$. if $\mathbf{X}$ is split into two (2) subsets $\mathbf{X}_1$ and $\mathbf{X}_2$, the divided data index is given as $gini_{split}(\mathbf{X}) = \mathbf{(n1/n)}gini(\mathbf{X}_1) + \mathbf{(n2/n)}gini(\mathbf{X}_2)$.

### 3.3 MakeTree ( )

The **MakeTree ( )** covers the splitting of the attribute for each leaf nodes.

## IV. Experimental Evaluation

The important metric for evaluating classifier performance is classification accuracy, time and the size of the decision tree. For a decision tree classifier it must produce a compact, accurate tree in short classification time [6]. SPRINT algorithm is being enhanced to address the drawbacks, the rewriting and resorting for classifying attribute lists in each node. Therefore, the evaluation of the performance of enhanced SPRINT algorithm will only focus on the classification time metric.

In order to determine the performance of SPRINT and the Enhanced SPRINT in terms of time for classifying attribute lists, an experimental evaluation was made to compare the performance using synthetic datasets. The SPRINT and Enhanced SPRINT algorithms were tested on the same platform using XAMPP (Php and MySQL) and Google Chrome as a web browser. To obtain a fair comparison, the test were performed in the Laptop with an Intel Core i7Ghz processor, 4 GB RAM, 64 bit OS (Windows 10) and 1 terabyte hard disk with 779 GB disk space.

*Datasets*

A synthetic database was being used as a point of reference containing datasets ranging from 500 to 1,000 records. Each record in this synthetic database consists of six (6) attributes which are shown in Table 1. This dataset was used to enable a thoughtful evaluation on the performance of the algorithms in terms of time in classifying attribute lists in the datasets.

TABLE 1: DESCRIPTION OF ATTRIBUTES FOR SYNTHETIC DATA

| Attribute | Value |
|---|---|
| Name of Employee | Name of Employees |
| Rank | 12 to 30 |
| Gender | Male, Female |
| Designation (equivalent units) | 0, 3, 6, 9, 12, 15 |
| Subjects (equivalent units) | 0 to 39 |
| FTE | 6 to 39 |

*Enhanced SPRINT Algorithm Performance*

In comparing the response time of SPRINT and Enhanced SPRINT on training sets of various sizes, Training sets used 500 to 1,000 records. This range was selected to examine how well SPRINT and Enhanced SPRINT performs on attribute lists classification. The result shows that Enhanced SPRINT Algorithm consumed a short period of time in attribute list classifying as compared to SPRINT Algorithm. The time to execute the classification of datasets was measured in seconds. The results are shown in Table 2, and figure 2 illustrate the graphical representation.

TABLE 2: TIME FOR ATTRIBUTE LISTS CLASSIFICATION

| Dataset | SPRINT Algorithm | Enhanced SPRINT Algorithm |
|---|---|---|
| 500 records | 13.73 sec | 12.91 sec |
| 700 records | 20.21 sec | 17.74 sec |
| 1,000 Records | 31.67 sec | 25.83 sec |


Fig. 2: Graphical Representation of Time for Attribute Lists Classification

## V. CONCLUSION

With the growth in the field of data mining, there is a need for algorithms for building classifiers to consider accuracy, classification time and the size of the decision tree. For a decision tree classifier, it must produce a compact, accurate tree in short classification time.

SPRINT algorithm is a decision tree classifier that was being enhanced, to addressing the drawbacks in rewriting and resorting of attribute lists in each node for classification. Based on the experimental evaluation result, Enhanced SPRINT algorithm take a shortened time in attribute lists classification as compared to SPRINT algorithm..

## REFERENCES

[1] GHan, J. 2001. Data Mining Concepts and Techniques[M].2001.Beijing: Machinery Industry Press.

[2] Anyanwu, M., and Shiva, S.2009. Application of Enhanced Decision Tree Algorithm to Churn Analysis.2009 International Conference on Artificial Intelligence and Pattern Recognition (AIPR-09), Orlando Florida.

[3] Tan, P., Steinbach, M. et. al .2006. Introduction to Data Mining

[4] Quinlan, J.R. 1993. C4.5 Programs for Machine Learning. Morgan Kaufman

[5] Michie, D., Spiegelhalter, D.J. et.al. 1994. Machine Learning Neural and Statistical Classification. Ellis Horwood.

[6] Shafer, J., Agrawal, R., et. al, 1996. SPRINT: A scalable parallel Classifier for Data Mining [C]. Proceedings of 22nd J International Conference on Very Large Database, Mumbai(Bombay), India

[7] Anyanwu, M., and Shiva, S. Comparative Analysis of Serial Decision Tree Classification Algorithms. International Journal of Computer Science and Security, (IJCSS) Volume(3): Issue(3)

[8] Haider, A.A., and Asghar, S. 2013. A Survey of Logic Based Classifiers. International Journal of Future Computer and Communication, Vol. 2, No.2, April 2013

[9] Rokach, L. and Maimon, O. Data Mining and Knowledge Discovery Handbook.

[10] Breiman, L., Friedman, J.H., et.al. 1984. Classification and Regression Trees. Wadsworth, Belmont.

[11] Rissanen, J. 1989. Stochastic Complexity in Statistical Inquiry. World Scientific Publication Co.

[12] Mehta, M., Agrawal, R., et al. 1996. SLIQ: A fast, scalable classifier for data mining. In. Proc. Of the Fifth Int'l Conference on Extending Database Technology