Zengchang, Q., & Lawry, J. (2004). ROC Analysis of a Linguistic Decision Tree Merging Algorithm. In Proc. UKCI. (pp. 33 - 42)

ROC Analysis of a Linguistic Decision Tree Merging Algorithm

Zengchang Qin and Jonathan Lawry
Artificial Intelligence Group
Department of Engineering Mathematics
University of Bristol
Bristol BS8 1TR, UK
{*z.qin, j.lawry*}*@bristol.ac.uk*

Abstract

ROC analysis is an important tool for evaluation and comparison of classifiers in imprecise environments (i.e., class distribution and cost parameters are unknown). Area Under the Curve of ROC (AUC) is increasingly being recognized as a better measure for evaluating algorithm performance than accuracy. A bigger AUC value implies a better ranking performance for a classifier. Linguistic decision tree (LDT) is a model based on a random set framework of modelling with words. Classification is made based on probability estimates from all leaves of the tree. In this paper, a branch merging algorithm for LDT model is proposed to generate more compact trees and no significant reduction in AUC values is found.

1 Introduction

Traditionally, the main criterion for evaluating the performance of a classifier is accuracy (percentage of test examples that are correctly classified) or error (percentage of misclassified examples). However, in many situations, not every misclassification has the same consequences when misclassification costs have to be taken into account. Provost *et. al* [11] have demonstrated problems with using accuracy as a metric. It can be irrelevant or misleading when classes are imbalanced or when misclassification costs are unequal. In recent years, Receiver Operating Characteristics (ROC) analysis has been introduced to evaluate machine learning algorithms [11, 12]. Area under the curve (AUC) of ROC is used to measure the quality of ranking for a classifier [5, 13]. It provides tools to compare the classifiers across the entire range of class distributions and misclassification cost. Recently, Ling *et.al* showed that AUC is statistically consistent and more discriminating than accuracy measure. So, it is fair to use AUC rather than accuracy to evaluate a learning algorithm.

Linguistic decision tree (LDT) [14] is a tree-structured model based on a random set framework for "Modelling with Words" [7] referred to as label semantics [8]. In these trees, fuzzy labels such as *small*, *medium* and *large* which are, defined by overlapping fuzzy sets, are used to build the tree. Different from traditional fuzzy or non-fuzzy decision trees, classification is made based on probability estimates from all leaves of the tree and branches of the tree can be interpreted as a set of linguistic rules which support linguistic reasoning and queries [8]. Linguistic ID3 (LID3) [14], the algorithm for learning LDTs, is based on an information heuristics modified from the traditional ID3 [15]. Empirical studies on classification problems showed that the LDT model has better or equivalent accuracy and the best transparency when compared to three other well known classification models [14].

LDTs generated from the LID3 algorithm usually have a large number of branches. However, a large size tree indicates low transparency. To overcome this problem and obtain much more compact trees, a branch merging algorithm is proposed in this paper and the performance for non-merged and merged LDTs are compared according to the AUC measure.

2 Why AUC is a Better Measure?

Traditionally, accuracy, defined as the percentage of instances that are correctly classified, or error, defined as the percentage of incorrectly classified instances, are the measures which are widely used for evaluating performance of a classifier. Using accuracy as a performance measure assumes that the error costs are equal. However, this is not realistic if we consider problems such as medical diagnosis or fraud detection. We need a classification model which minimises the over-

Table 1: Classification by a probability estimator with different thresholds.

Examples	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
$Pr(+\|e_i)$	0.11	0.23	0.25	0.37	0.49	0.58	0.63	0.69	0.84	0.97
$T = 0.5$	$-$	$-$	$-$	$-$	$-$	$+$	$+$	$+$	$+$	$+$
$T = 0.3$	$-$	$-$	$-$	$+$	$+$	$+$	$+$	$+$	$+$	$+$

Table 2: Two classifiers with the same accuracy but different AUC values. This table is inspired by a similar table in [9]

Examples	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
Classifier 1 (CS_1)	$-$	$-$	$-$	$-$	$+$	$-$	$+$	$+$	$+$	$+$
r_i for Classifier 1					5		7	8	9	10
Classifier 2 (CS_2)	$+$	$-$	$-$	$-$	$-$	$+$	$+$	$-$	$+$	$+$
r_i for Classifier 2	1					6	7		9	10

all cost but which does not necessarily minimize error or maximize accuracy. Here, the following two important statistics are considered.

For a binary classification problem, the *true positive rate* (TPR) of a classifier is:

$$TPR = \frac{positive\ correctly\ classified}{total\ positives} \quad (1)$$

The *false positive rate* (FPR) of a classifier is:

$$FPR = \frac{negatives\ incorrectly\ classified}{total\ negatives} \quad (2)$$

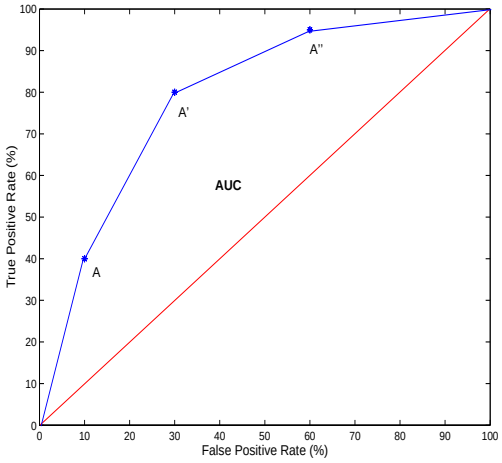

Figure 1: By varying the decision threshold of a probability estimation classifier, a ROC curve is produced.

If we plot FP rate on the X axis and TP rate on the Y axis. A single classification is then represented by a point in this 2D space which is referred to as ROC space. For a probability estimation model, a threshold is needed when doing classification. For example, table 1 shows 10 examples $e_1, \cdots, e_{10}$ classified by a learning model which gives probability estimation in a two-class (i.e. $e_i \rightarrow \{+, -\}$) problem. The probabilities of belonging to class $+$ are sorted from small to large. Different classification results will be given according to the rule of

$$\forall\ i,\ e_i\ \rightarrow \{+\} \quad if : Pr(+|e_i) \geq T \quad (3)$$

by assigning different threshold probabilities T. In table 1, the classification results when $T = 0.5$ and $T = 0.3$ are given. We normally set $T = 0.5$ when we calculate accuracy for a probability estimation model. If we vary the value of T through $[0, 1]$ will result a continuous curve in ROC space which is referred to as a ROC curve.

Suppose the threshold value T is set to 0.5, which gives $e_i \rightarrow \{+\}$ for $i = 1, \cdots, 10$ that satisfies $Pr(+|e_i) \geq 0.5$. However, By varying the threshold , we can generating a trial in ROC space, which is called ROC curve. For example, figure 1 schematically shows a ROC curve by choosing 3 different probability thresholds, where A, A' and A'' are the classification results based on the chosen thresholds. In another words, A classifier results in a ROC curve, which aggregates its behavior for all possible decision thresholds. The quality of the classifier can be measured by the area under the curve of ROC (AUC), which measures how well the classifier separates the two classes without reference to a decision threshold. In other words, AUC represents the quality of ranking of examples by this classifier [5].

Hand and Till [5] present a straightforward approach to calculate the AUC of a classifier based on the ranking of examples based on the their class probabilities. For a binary classification problem with two classes $\{+, -\}$:

$$AUC = \frac{S_+ - n_+(n_+ + 1)/2}{n_+ n_-} \quad (4)$$

where n_+ and n_- are the number of positive and negative examples, respectively. A ranking list is given according to the probabilities of the class +.

$$S_+ = \sum_{i=1}^{n_+} r_i \quad (5)$$

where r_i is the rank of i^{th} positive example in the ranking list. For example, the AUC for classifier 1 and 2 listed in table 1 are:

$$AUC_{(CS_1)} = \frac{(5+7+8+9+10) - 5(5+1)/2}{5 \times 5} = \frac{24}{25}$$

$$AUC_{(CS_2)} = \frac{(1+6+7+9+10) - 5(5+1)/2}{5 \times 5} = \frac{18}{25}$$

We may notice that both classifier 1 and 2 have the same accuracy 80% (8 of 10 examples are correctly classified) and thus they are equally good in accuracy. However, the intuition tells us that Classifier 1 is better than Classifier 2 since Classifier 1 gives a better overall ranking. Consider an intuitive example, suppose the ranking tells us how poisonous ten different kinds mushrooms are, where '−' represents poisonous and '+' edible. Classsifer 2 will classify a very poisonous mushroom as edible. However, Classifer 1 is not that bad by classifying a less poisonous mushroom as edible. So, Classifier 1 is better for the mushrooms classification problem. This can be seen from AUC measure but not the accuracy measure. Ling *et. al* mathematically proved that the AUC measure is consistent and more discriminating than the accuracy measure [9]. The method for calculating AUC for multi-class problems are given in [5], however, in this paper, only two-class problems are considered.

3 Label Semantics

Label semantics [6] was proposed by Lawry to capture the idea of using linguistic expressions to label imprecise concepts. The semantics is based on random set theory although different from earlier work of Goodman and Nguyen [4]. The underlying question posed by label semantics is how to use linguistic expressions to label numerical values. For a variable x into a domain of discourse Ω we identify a finite set of linguistic labels $LA = \{L_1, \cdots, L_n\}$ with which to label the values of x. Then for a specific value $\alpha \in \Omega$ an individual I identifies a subset of LA, denoted D_α^I to stand for the description of α given by I, as the set of words with which it is appropriate to label α. If we allow I to vary across a population V, then D_α^I will also vary and generate a random set denoted D_α into the power set of LA. The frequency of occurrence of a particular label, say S, for D_α across the population then we obtain a distribution on D_α referred to as a mass assignment on labels, more formally:

Definition 1 (*Mass Assignment*)

$$\forall S \subseteq LA, \quad m_x(S) = \frac{|\{I \in V | D_x^I = S\}|}{|V|}$$

For example, given a set of labels defined on a man's age $LA_{age} = \{young(y), middle\text{-}aged(m), old(o)\}$. 3 of 10 people agree that '*middle-aged* is the only appropriate label for the age of 30' and 7 agree 'both *young* and *middle-aged* are appropriate labels'. According to def. 1, $m_{30}(y) = 0.3$ and $m_{30}(y, m) = 0.7$ so that the mass assignment for 30 is

$$m_{30} = \{m\} : 0.3, \{y, m\} : 0.7$$

More details about Mass Assignment theory can be found in [1].

In this framework, *appropriateness degrees* are used to evaluate how appropriate a label is for describing a particular value of variable x. Simply, given a particular value α of variable x, the appropriateness degree for labeling this value with the label L, which is defined by fuzzy set F, is the membership value of α in F. The reason we use the new term 'appropriateness degree' is partly because it more accurately reflects the underlying semantics and partly to highlight the quite distinct calculus based on this framework [8]. This definition provides a relationship between mass assignments and appropriateness degrees.

Definition 2 (*Appropriateness Degrees*)

$$\forall x \in \Omega, \forall L \in LA \quad \mu_L(x) = \sum_{S \subseteq LA : L \in S} m_x(S)$$

Consider the previous example, we can obtain $\mu_m(30) = 0.7 + 0.3 = 1$, $\mu_y = 0.7$. Based on the underlying semantics, we can translate a set of numeric data into a set of linguistic data, where each data value is replaced by mass assignments on appropriate labels. We need to make some assumptions for this translation. The first is the *consonance assumption*, according to which we can determine the mass assignment uniquely from the appropriateness degrees as follows. (For the justification of the consonance assumption in see [8])

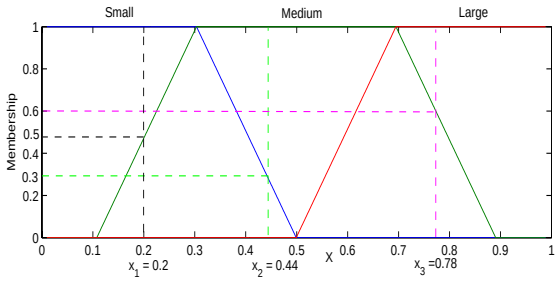

Figure 2: An example of a full fuzzy covering with 3 uniformly distributed trapezoidal fuzzy sets with 50% overlap.

Definition 3 (*Consonance Assumption*)

Let $\{\beta_1, \beta_2, \cdots, \beta_k\} = \{\mu_L(x) | L \in LA, \mu_L(x) > 0\}$ *ordered such that* $\beta_t > \beta_{t+1}$ *for* $t = 1, 2, \cdots, k-1$ *then:*

$$m_x = M_t : \beta_t - \beta_{t-1}, t = 1, 2, \cdots, k-1,$$

$$M_k : \beta_k, \quad M_0 : 1 - \beta_1$$

where $M_0 = \emptyset$ *and* $M_t = \{L \in LA | \mu_L(x) \geq \beta_t\}$ *for* $t = 1, 2 \ldots, k$.

Based on this assumption, there is a unique mass assignment for a given set of appropriateness degree values. For example, given $\mu_{L_1} = 0.3$ and $\mu_{L_2} = 1$, the only unique consonant mass assignment is $\{L_2\} : 0.7, \{L_1, L_2\} : 0.3$, but not $\{L_2\} : 0.8, \{L_1, L_2\} : 0.2, \{L_1\} : 0.1$ or others. However, it is undesirable to have mass associated with the empty set. In order to avoid this, we define a *full fuzzy covering* assumption as follows: Given a continuous discourse Ω, LA is called a full fuzzy covering of Ω if:

$$\forall x \in \Omega, \exists L \in LA \qquad \mu_L(x) = 1$$

In this paper, unless otherwise stated, the fuzzy labels are defined by trapezoidal fuzzy sets with 50% overlap. For example see figure 2 which shows a full fuzzy covering of the universe with three fuzzy labels: *small*, *medium* and *large*.

It is also interesting to note that given assumptions (consonant, full fuzzy covering) on labels we can isolate a set of subsets of LA with non-zero mass assignments. These are referred to as *focal sets*:

Definition 4 (*Focal Set*) *The focal set of LA is a set of focal elements defined as:*

$$\mathcal{F} = \{S \subseteq LA | \exists x \in \Omega, m_x(S) > 0\}$$

Given fuzzy labels defined in figure 2 the following focal elements occur: {small}, {small, medium}, {medium}, {medium, large} and {large}. Since *small* and *large* do not overlap, the set {small, large} cannot occur. We can then always find the unique translation from a given data point to a mass assignment on focal elements, specified by the function μ_L; This is referred to as *linguistic translation (LT)*. For a particular attribute with an associated focal set, linguistic translation is a process of replacing data elements with masses of focal elements of these data. The linguistic translation for $\langle x_1, x_2, x_3 \rangle$ based on the consonance assumption is as follows:

$$\begin{pmatrix} \text{x} \\ \hline 0.2 \\ 0.44 \\ 0.78 \end{pmatrix} \overset{LT}{\rightarrow} \begin{pmatrix} \{s\} & \{s,m\} & \{m\} & \{m,l\} & \{l\} \\ \hline 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0.3 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0.4 \end{pmatrix}$$

4 Linguistic Decision Tree

Linguistic decision tree (LDT) [14] is a tree-structured classification model based on label semantics. The information heuristics used for building the tree are modified from Quinlan's ID3 [15] in accordance with label semantics. The class probability estimation for each branch is evaluated according to the training set. Classification are made by considering the class probabilities across the whole tree.

Consider a database with n attributes and N instances and each instance is labeled by one of the classes: $\{C_1, \cdots, C_m\}$. A linguistic decision tree built from this database can be defined as follows:

$$LDT = \{< B_1, P(C_1|B_1), \cdots, P(C_m|B_1) >, \cdots \\ < B_s, P(C_1|B_s), \cdots, P(C_m)|B_s) >\}$$

where $P(C_t|B)$ is the probability of class C_t given a branch B. A branch with k nodes is defined as:

$$B =< F_1, \cdots, F_k >$$

where, $k \leq n$ and $F_j \in \mathcal{F}_j$ for $i = 1, \cdots, k$. For example, consider the branch:

$$<< \{small_1, medium_1\}, \{large_2\} >, 0.3, 0.7 >$$

in a binary classification problem. This means the probability of class C_1 is 0.3 and C_2 is 0.7 given attribute 1 can be described as *small* & *medium* and attribute 2 can only be described as *large*. In a LDT, one attribute is not allowed to appear more than once in a branch, an attribute which is not currently part of a branch is referred to as a *free attribute*.

4.1 Classification

Basically, fuzzy discretization provides an interpretation between numerical data and linguistic data based on label semantics. The effectiveness of fuzzy discretization may affect the algorithm's performance. In this paper, we will use percentile-based discretization: each attribute universe is partitioned into intervals which each contains approximately the same number of data elements. It is a very intuitive way for generating fuzzy sets.

According to the definition of LDT, if given a branch of a LDT in the form of $B =< F_1, \cdots, F_k >$. Based on a training set with N instances: $DB = \{\mathbf{x_1}, \cdots, \mathbf{x}_N\}^T$ where each instance has n attributes: $\mathbf{x} = \langle x_1, \cdots, x_n \rangle$. The probability of Class C_t $(t = 1, \cdots, m)$ given B can then be evaluated as follows. First, we consider the probability of a branch B given $\mathbf{x}$:

$$P(B|\mathbf{x}) = \prod_{r=1}^{k} m_{x_j}(F_j) \tag{6}$$

$m_{x_j}(F_j)$ for $j = 1, \cdots, k$ are mass assignments of single data element x_j. The probability of class C_t given B can then be evaluated by:

$$P(C_t|B) = \frac{\sum_{i \in DB_t} P(B|\mathbf{x}_i)}{\sum_{i \in DB} P(B|\mathbf{x}_i)} \tag{7}$$

where DB_t is the subset consisting of instances which belong to class t. In the case of $\sum_{i \in DB} P(B|\mathbf{x}_i) = 0$, which can occur when the training database for the LDT is small, then there is no non-zero linguistic data covered by the branch. In this case, we obtain no information from the database so that equal probabilities are assigned to each class.

$$P(C_t|B) = \frac{1}{m} \qquad for \quad t = 1, \cdots, m \tag{8}$$

Now consider classifying an unlabeled instance in the form of $\mathbf{x} = \langle x_1, \cdots, x_n \rangle$ which may not be contained in the training data set DB. First we apply linguistic translation to $\mathbf{x}$ based on the fuzzy covering of the training data DB. In the case that a data element appears beyond the range of training data set $[R_{min}, R_{max}]$, we assign the appropriateness degrees of R_{min} or R_{max} to the element depending on which side of the range it appears. Then, according to the Jeffrey's rule the probabilities of class C_t given a LDT with s branches are evaluated as follows:

$$P(C_t|\mathbf{x}) = \sum_{v=1}^{s} P(C_t|B_v) P(B_v|\mathbf{x}) \tag{9}$$

where $P(C_t|B_v)$ and $P(B_v|\mathbf{x})$ are evaluated based on equations 6 and 7 (or 8), respectively.

4.2 LID3 Algorithm

Linguistic ID3 (LID3) is the learning algorithm proposed for building the linguistic decision tree. Similar to the ID3 algorithm [15], search is guided by an information based heuristic, but the information measurements of a LDT are modified in accordance with label semantics. The measure of information defined for a branch B and can be viewed as an extension of the entropy measure used in ID3.

Definition 5 (*Branch Entropy*) *The branch entropy of a branch B is given by*

$$E(B) = -\sum_{t=1}^{m} P(C_t|B) \log_2(P(C_t|B)) \tag{10}$$

Now, given a particular branch B suppose we want to expand it with the attribute x_j. The evaluation of this attribute will be given based on the expected entropy defined as follows:

Definition 6 (*Expected Entropy*)

$$EE(B, x_j) = \sum_{F_j \in \mathcal{F}_j} E(B \cup F_j) P(F_j|B) \tag{11}$$

where $B \cup F_j$ represents the new branch obtained by appending the focal element F_j to the end of branch B. The probability of F_j given B can be calculated as follows:

$$P(F_j|B) = \frac{\sum_{i \in DB} (B \cup F_j|\mathbf{x}_i)}{\sum_{i \in DB} (B|\mathbf{x}_i)} \tag{12}$$

We can now define the *Information Gain (IG)* obtained by expanding branch B with attribute x_j as:

Definition 7 (*Information Gain*)

$$IG(B, x_j) = E(B) - EE(B, x_j) \tag{13}$$

The goal of tree-structured learning models is to make subregions partitioned by branches be less "impure", in terms of the mixture of class labels, than the unpartitioned dataset. For a particular branch, the most suitable free attribute for further expanding (or partitioning), is the one by which the "pureness" is maximumly increased with expanding. That corresponds to selecting the attribute with maximum information gain.

As with ID3 learning, the most informative attribute will form the root of a linguistic decision tree, and the tree will expand into branches

associated with all possible focal elements of this attribute. For each branch, the free attribute with maximum information gain will be the next node, from level to level, until the tree reaches the maximum specified depth or some other criteria are met.

4.3 Forward Branch Merging

One of the inherent disadvantages for tree induction algorithms is overfitting. Many algorithms for punning the trees back were proposed [10]. Here we present a branch merging algorithm for LDT. By applying the branch merging, LDTs are using developed breath-first search. At each depth, the branches which give similar probabilities for target focal elements are merged into one branch according to a *merging threshold*:

Definition 8 (*Merging Threshold*) *For two adjacent branches B_1 and B_2, if the maximum difference between class probabilities is less than or equal to a given merging threshold T_m, then the two branches can be merged into one branch.*

$$T_m \geq \max_{t=1,\cdots,m}(|Pr(C_t|B_1) - Pr(C_t|B_2)|) \quad (14)$$

The merged branch MB is defined as follows:

Definition 9 (*Merged Branch*) *A merged branch with k nodes is defined as*

$$MB = \langle \mathcal{M}_1, \cdots, \mathcal{M}_k \rangle$$

where $\mathcal{M}_j = \{F_j^1, \cdots, F_j^w\}$ is a set of focal elements such that F_j^i is adjacent to F_j^{i+1} for $i = 1, \cdots, w-1$. The associate mass for $\mathcal{M}_j$ is given by

$$m_x(\mathcal{M}_j) = \sum_{i=1}^{w} m_x(F_j^i) \quad (15)$$

where w is the number of merged adjacent focal elements for attribute j.

Here 'adjacent' means that the fuzzy labels are defined sequentially according to a natural ordering. For the example shown in figure 2, {small} and {small,medium} are adjacent focal elements while {small} and {medium} are not. Then the probability of a merged branch given a data element can be evaluated by

$$P(MB|\mathbf{x}) = \prod_{r=1}^{k} m_{x_j}(\mathcal{M}_j) \quad (16)$$

Based on equations 7, 8 and 15 the following equation is used to calculate the class probabilities given a merged branch.

$$P(C_t|MB) = \frac{\sum_{i \in DB_t} P(MB|\mathbf{x})}{\sum_{i \in DB} P(MB|\mathbf{x})} \quad (17)$$

Table 3: Description of UCI datasets for test.

#	Data	Cases	Features	Mixed
1	Can	286	9	yes
2	CanW	699	9	no
3	HC	303	13	yes
4	HS	270	13	yes
5	HP	155	19	yes
6	Pima	768	8	no

Given a new example $\mathbf{x}$, we can classify it by a merged LDT with s' branches as follows:

$$P(C_t|\mathbf{x}) = \sum_{v=1}^{s'} P(C_t|MB_v)P(MB_v|\mathbf{x}) \quad (18)$$

When the merging algorithm is applied during learning, the adjacent branches meeting the merging criteria will be merged and re-evaluated according to equation 17. Then the adjacent branches after the first round of merging will be examined in a further round of merging, until all adjacent branches cannot be merged further. We then proceed to the next depth. The merging is applied as the tree develops from the root to the maximum depth and hence is referred to as *forward branch merging*.

5 Experimental Studies

In this section, six binary datasets from UCI machine learning repository [2] are tested: breast-cancer (Can), Wisconsin cancer (CanW), heart (HC), heart-Stalog (HS), heptitis (HP) and Indian Pima (Pima). Description about these datasets including the number of examples, the number of attributes (features) and whether the features are mixed features of numerical and nominal, is shown in table 3.

The LDT parameters for each data set are set individually: the number of fuzzy sets used for discretization (N_F) is also shown in table 4. The maximum depth for the Cancer dataset is 2 and other five data sets are 3. These parameter settings are based on a few test-and-trail experiments [14]. For each data set, the examples were equally divided into two subsets, one for training and the other one for test. This is referred to as 50%-50% split experiment. Table 4 shows the average AUC and standard deviation from 10 runs of 50%-50% split experiments by applying merging with the merging threshold T_m ranging from 0 (no merging) to 0.3. The average size of the trees N_R from 10 runs of experiments are also shown in the table, where size of the tree

Table 4: Mean AUC values with standard deviation on six data sets with different merging thresholds.

	$T_m = 0$		$T_m = 0.1$		$T_m = 0.2$		$T_m = 0.3$	
Data	N_F	AUC	N_R	AUC	N_R	AUC	N_R	AUC	N_R
Can	3	73.69 ± 7.73	13	71.45 ± 7.12	11	74.29 ± 8.44	9	81.11 ± 3.25	4
CanW	3	98.76 ± 0.72	44	99.02 ± 0.54	12	98.69 ± 0.59	9	98.99 ± 0.60	7
HC	2	85.36 ± 2.58	35	84.02 ± 3.46	32	84.79 ± 3.39	25	84.32 ± 4.38	17
HS	2	84.41 ± 3.64	29	85.16 ± 2.90	25	81.12 ± 15.05	19	82.36 ± 5.71	11
HP	2	73.26 ± 6.89	19	73.99 ± 5.36	11	74.80 ± 4.83	9	74.25 ± 5.99	7
Pima	2	81.08 ± 0.97	27	81.92 ± 1.79	14	74.74 ± 15.13	5	81.90 ± 4.84	2

is in terms of the number of branches (also the the number of rules can be interpreted from a LDT).

According to the t-test with confidence level 0.9, the AUC values for the merged LDTs are not reduced significantly comparing to the non-merging case. Although for some data sets, (e.g., breast-cancer) the merged trees performs even a little better than non-merged trees. But no statistically significant difference are found. On the other side, the tree sizes are reduced significantly. These facts can be intuitively seen from figure 3: the left-hand figure shows the accuracy comparison and the right-hand figure shows the comparison of the number of branches. The possible reason for this is because the merging algorithm generates self-adapting granularities based on class probabilities. Compared to other methods that discretize attributes independently, merging may generate more reasonable trees with more appropriate information granules. However, this still needs more investigation.

The LDTs obtained can be interpreted into a set of linguistic rules. For example on the Pima Indian dataset, a merged LDT with 6 branches when $T_m = 0.2$ is shown as follows:

$$
\begin{array}{lllll}
1: & \{\{s_2\},\{s_2,l_2\}\} & -- \{s_8\} & & \langle 0.88, 0.12\rangle \\
2: & & -- \{\{s_8,l_8\},\{l_8\}\} & & \langle 0.60, 0.40\rangle \\
3: & \{l_2\} & -- \{s_6\} -- \{s_8\} & & \langle 0.88, 0.12\rangle \\
4: & & & -- \{s_8,l_8\} & \langle 0.31, 0.69\rangle \\
5: & & & -- \{l_8\} & \langle 0.68, 0.32\rangle \\
6: & & -- \{\{s_6,l_6\},\{l_6\}\} & & \langle 0.34, 0.66\rangle
\end{array}
$$

where each attribute (from 1 to 8) is discretized by 2 fuzzy labels: *small* (s) and *large* (l). The class probabilities are shown at the end of each branch. Initially, the tree has expanded to level 3, by applying the merging algorithm, some of branch expanding could be merged back: for example, the first branch:

$$\{\{s_2\},\{s_2,l_2\}\} -- \{s_8\}$$

is from the merging of the following 3 branches:

$$\{\{s_2\},\{s_2,l_2\}\} -- \{s_8\} -- \{s_6\}$$
$$\{\{s_2\},\{s_2,l_2\}\} -- \{s_8\} -- \{s_6,l_6\}$$
$$\{\{s_2\},\{s_2,l_2\}\} -- \{s_8\} -- \{l_6\}$$

For each branch, e.g., the 4th branch:

$$\{l_2\} \ -- \{s_6\} -- \{s_8,l_8\}$$

We can interpret it as a linguistic rule:

$$\neg s_2 \wedge l_2 \ -- \ s_6 \wedge \neg l_6 -- \ s_8 \wedge l_8$$

where $\{l_2\}$ represents that *only the label large is appropriate* or, shortly, only large. This is logically equivalent to the expression $\neg s_2 \wedge l_2$. More details on the logical interpretation can be found in [8]. So, the above linguistic tree can be interpreted as follows:

$$
\begin{array}{lllll}
1: & s_2 -- s_8 \wedge \neg l_8 & & & \langle 0.88, 0.12\rangle \\
2: & -- \ l_8 & & & \langle 0.60, 0.40\rangle \\
3: & \neg s_2 \wedge l_2 \ -- \ s_6 \wedge \neg l_6 & -- \ s_8 \wedge \neg l_8 & & \langle 0.88, 0.12\rangle \\
4: & & -- \ s_8 \wedge l_8 & & \langle 0.31, 0.69\rangle \\
5: & & -- \neg s_8 \wedge l_8 & & \langle 0.68, 0.32\rangle \\
6: & -- \ l_6 & & & \langle 0.34, 0.66\rangle
\end{array}
$$

As we can see from above that each branch is also a linguistic rule based on fuzzy labels. This is also one of the reasons of naming our model as linguistic decision tree.

6 Conclusions

ROC analysis has been recognized as an important tool for evaluation and comparison of classifiers. Some recent work has been done to use ROC analysis study decision trees: Provost and Domingos [13] shows that decision trees can also

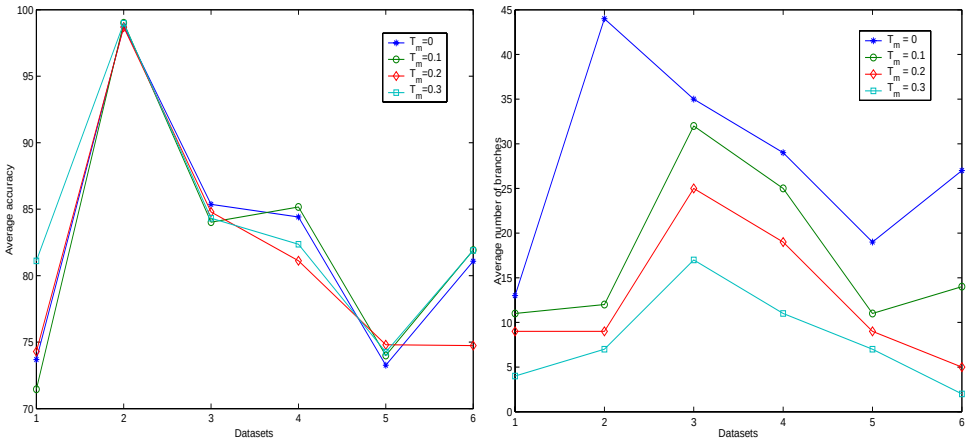

Figure 3: Comparison between non-merged trees and merged trees with T_m ranging from 0.1 to 0.3 on the given test data. Left-hand figures shows the accuracy comparison and right-hand figure shows the comparison of the number of branches.

give good probability estimations in the AUC measure. Ferri *et. al* [3] use the AUC as a heuristics to learning decision trees. Ling *et. al* [9] shows that the AUC measure is more discriminating than the accuracy measure.

In this paper, the AUC measure is used to evaluate a linguistic decision tree merging algorithm. Some initial results for comparing merged LDTs and non-merged LDTs in the AUC measure are presented. By applying this merging algorithm, the tree sizes are significantly reduced while the AUC values are not significantly influenced when merging thresholds ranging from 0.1 to 0.3 based on 6 UCI data sets. Finally, we gave an example of interpreting a merged LDT as a set of linguistic rules.

References

[1] J.F. Baldwin, T.P. Martin and B.W. Pilsworth. *Fril-Fuzzy and Evidential Reasoning in Artificial Intelligence*. John Wiley & Sons Inc, 1995.

[2] C. Blake and C.J. Merz. UCI machine learning repository. *http://www.ics.uci. edu/∼mlearn/ MLRepository.html*.

[3] C. Ferri, P.A. Flach and J. Hernández-Orallo. Learning decison trees using the area under the ROC curve. *Proceedings of the ICML-02*. Sydney, Australia, pp. 139-146, Morgan Kaufmann, 2002.

[4] I.R. Goodman and H.T. Nguyen. Uncertainty Models for Knowledge Based System. North-Holland, Amsterdam, 1985.

[5] D. Hand and R. J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*. Vol. 45, 171-186, 2001.

[6] J. Lawry. Label Semantics: A formal framework for modelling with words. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, LNAI pp. 374-384 Springer-Verlag, 2001.

[7] J. Lawry, J. Shanahan, and A. Ralescu. *Modelling with Words: Learning, fusion, and reasoning within a formal linguistic representation framework*. LNAI 2873. Springer-Verlag 2003.

[8] J. Lawry. A framework for linguistic modelling, *Artificial Intelligence*, 155: pp. 1-39, 2004.

[9] C. X. Ling, J. Huang and H. Zhang. AUC: a statistically consistent and more discriminating measure than accuracy. *Proceedings of IJCAI*, 2003.

[10] C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*. 138: 221-254, 2003.

[11] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. in J. Shavlik, editor, *Prced. of ICML98*, pp. 445-453, 1998.

[12] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*. Vol. 42, 203-231, 2001.

[13] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*. 52, 199-215, 2003.

[14] Z. Qin and J. Lawry. A tree-structured model classification model based on label semantics. *To appear in the Proceedings of IPMU*, 2004.

[15] J.R. Quinlan. Induction of decision trees. *Machine Learning* 1: 81-106. 1986