Knowledge Based Decision Tree Construction with Feature Importance Domain Knowledge

MD. Ridwan Al Iqbal, Saiedur Rahman, Syed Irfan Nabil, Ijaz Ul Amin Chowdhury
Department of Computer Science, American International University-Bangladesh
stopofeger@yahoo.com, saied@aiub.edu,{nabilsayed, ijazulamin}@yahoo.com

***Abstract*—Decision Tree is a widely used supervised learning algorithm due its many advantages like fast non parametric learning, comprehensibility and son. But, Decision Tree require large training set to learn accurately because, decision tree algorithms recursively partition the data set that leaves very few instances in the lower levels of the tree. In order to address this drawback, we present a novel algorithm named Importance Aided Decision Tree (IADT). It that takes Feature Importance as an additional domain knowledge. Additional domain knowledge have been shown to enhance the performance of learners. Decision Tree algorithm always finds the most important attributes in each node. Thus, Importance of features is a relevant domain knowledge for decision tree algorithm. Our algorithm uses a novel approach to incorporate this feature importance score into decision tree learning. This approach makes decision trees more accurate and robust. We presented theoretical and empirical performance analysis to show that IADT is superior to standard decision tree learning algorithms.**

***Index Terms*—Decision Tree, Domain Knowledge, Feature Importance**

I. Introduction

Decision Tree algorithms are widely used in real world. There are many advantages to Decision Tree such as fast non-parametric learning with few assumptions and they are easily adaptable to most learning tasks. Moreover, the generated Decision Trees are highly comprehensible by human practitioners. Decision Trees can be converted to rules or intuitively visualized. They are especially used in a business setting and in data mining software [1].

Decision Trees works by dividing data sets into smaller and more homogenous subsets. Therefore, It works in a divide and conquer approach. This recursive division of data sets means the subsets on lower levels have very few instances. In some data sets, the instances on a particular node can be as low as 2-5 instances [7]. It is not possible take a reliable decision with such small statistically insignificant data.

Therefore, Decision Trees tend to over fit in the lower levels of the tree. This problem is even more pronounced in small data sets. This is why Decision Trees are normally outperformed by other learners such as *Neural Networks* [6], *Support Vector Machines*[9].

It has been shown that additional domain knowledge improves performance of Machine learning algorithms. Therefore, utilizing domain knowledge would improve the accuracy of Decision Trees. However, two problems must be addressed to properly utilize domain knowledge.

1) What should be the form of the input domain knowledge?
2) How to consider both domain knowledge and training data so that the performance is maximized?

We surmised that the foremost task of a *Decision Tree* is to calculate the importance of an attribute in predicting the value of the class. In each node the attribute that increases the predictability or reduces the uncertainty of the class is chosen to split on. So, the relationship of attributes with the class is the most important knowledge required in a *Decision Tree* algorithm. Instead of treating all features equally, if the learners treat features based on their importance and use this knowledge of importance in learning, then they have been shown to perform better [10], [5].

Use of Feature Importance to extend machine learning has been explored before for Neural Networks. IANN (Importance aided Neural Network) [5] extended Multilayer Perceptrons to use Feature Importance values. Domain knowledge was provided as feature weights, a real value in [0,1] range to represent the importance of a feature. IANN performed better than many empirical learning algorithms and also required significantly less training data to perform well.

In this paper, we propose an algorithm called *Importance Aided Decision Tree (IADT)* that takes domain knowledge in the form of Attribute Importance Score. This score is then used to construct *Decision Tree* in such a way so that it conforms to both the Attribute Importance Score and training data. We use a hybrid approach so that both forms of information could be utilized effectively. Especially in the lower levels of the tree where less data is available, the domain knowledge is given more weight than the score calculated from training data.

We establish theoretically that *IADT* is probabilistically a better Tree Construction algorithm than traditional methods that uses only training data. We also tested IADT extensively in both synthetic learning problems and real world data sets. We also test on an actual application of IADT in the Bank Loan candidate screening problem. The detailed performance analysis shows the advantage of our algorithm over traditional forms of learners.

II. Background

A. Decision Trees

For simplicity, we are only considering binary classification. However, it can be extended to higher feature space trivially. Suppose, we are given an input of $(x_1, y_1) \dots (x_n, y_n)$ samples over the input space $X \epsilon [0,1]^d$ & $Y \epsilon [0,1]$, the goal is to induce a classifier $F(X) \to Y$ based on the samples. Decision Tree algorithm is one such classifier.

Decision-tree learning involves constructing a tree by recursively partitioning the training examples. Each Node in the Tree represents a query on the value of a particular attribute. Based on this value, a path to a child is taken where another query is presented. Each time a node is added to the tree, some subset of the training examples are used to pick the logical test at that node. All of the training examples are used to determine the test for the root of the tree. After this test has been picked, it is used to partition the examples, and the process is continued recursively[8].

One of the key aspects of any decision-tree algorithm is selecting the test for partitioning the subset of training examples that reaches a given node. The partition is done based on attribute selection criteria that measures the attribute importance. Therefore, finding the most influential attribute is the foremost task of a decision tree inducer. At each node, the decision tree algorithm chooses the attribute with the highest Attribute Selection Score $S(X)$. Thus, $I_{best} = \underset{x}{\operatorname{argmax}} S(X)$

There are many attribute selection measures that are used for Decision Trees e.g Information Gain [8], Gini Index [3]. Information Gain in particular is the most popular Attribute selection measure being used in the popular ID3 and C4.5 tree learning algorithms. It uses a Information theoretic measure Entropy.

B. Learning with Feature Importance Domain Knowledge

In this problem setting, we are provided an additional form of information as input, *Feature Importance Score*. Therefore, the new problem setting will be given $(x_1, y_1) \dots (x_n, y_n)$ samples over the input space $X \epsilon [0,1]^d$ & $Y \epsilon [0,1]$ as well as *Feature Importance Score* $I_1 \dots I_d$, the goal is to induce a classifier $F(X, I) \rightarrow Y$.

The Feature Importance values represent the amount of Influence a feature has over the class. It can be more precisely defined as the expected probability of knowing the class value $Y = y_i$ if the value of attribute X is known.

Definition 1: Feature Importance I_X of a Feature X is the expected probability of Y given X.

$$I_X = E[Pr(Y = y_i | X)] \qquad \forall i \tag{1}$$

Based on this definition, we can surmise that I_X can provide useful insight for choosing the best attribute in decision tree construction.

III. Importance Aided Decision Tree

A. Derivation with true Importance Value

The key idea behind *IADT* is to use a hybrid approach that combines the power of both inductive learning and domain knowledge. Our intention is to enhance the performance of the learner by using Domain knowledge, not to supplant the training data set all together. Therefore, we would go for a middle ground that uses both information equally. As mentioned before, finding the best attribute is the central step in a decision tree. So, our algorithm modifies Attribute Selection Measure.

Let us consider an algorithm IADT-0, that selects attributes based on the weighted average of both Attribute Selection Score $S(X)$ and Feature Importance score I_X. This new Scoring function $S_I(X)$ will use both information equally based on the weight p.

$$S_I(X) = (1-p).S(X) + p.I_X$$

The correctness of using this modified attribute selection measure is evident from the fact that, $S(X)$ is a score that measures the probability of an attribute X influencing target variable Y. Similarly, I_X also measures the probability of knowing the class if X is known. The difference between the two scores is that Feature Importance assumed to be the true Importance value of the attribute while S(X) is an estimate calculated from the data set. Therefore, Our assumption is that I_X is the true Importance value and has no error. But, S(X) is dependent on estimation error.

The estimation error of S(X) can be bounded by using a statistical measurement known as the Hoeffding Bound. In Probability Theory, Chernoff-Hoeffding Bound limits how much an estimation will differ from its expected value[4], [2]. Consider a real-valued random variable X whose range is R (e.g., for a probability the range is 1, and for an information gain the range is $logc$, where c is the number of classes). Suppose we have made n independent observations of this variable, and computed their mean $\bar{x}$ The Hoeffding inequality states that, with probability $1-\delta$, the true mean of the variable is at least $\bar{x} - \epsilon$, where,

$$\epsilon = \sqrt{\frac{R^2}{2n} \ln\left(\frac{1}{\delta}\right)} \tag{2}$$

Therefore, the probable estimation error ϵ of S(X) will also be:

$$Error_S = \epsilon = \sqrt{\frac{R^2}{2n} \ln\left(\frac{1}{\delta}\right)}$$

However, for $S_I(X)$, we are taking the weighted average of $S(X)$ and I_X, therefore the error will also be the weighted total of these two terms.

Total error is:

$$\begin{aligned} Error_{S_I} &= (1-p).\epsilon + 0 = (1-p).\epsilon \\ &= (1-p).\sqrt{\frac{R^2}{2n} \ln\left(\frac{1}{\delta}\right)} \end{aligned} \tag{3}$$

We are calculating weighted average where $p < 1$. Hence, $Error_{S_I} < \epsilon$. The error bound for the new Attribute Selection measure is lesser than the error bound for $S(X)$.

So, it is possible to achieve same accuracy with lesser training data using this new Attribute Selection Measure. But, *IADT-0* suffers from an immense drawback. That is, we are making an assumption that the input Importance value I_X is the true Importance value. We are assuming there is no error on this value, which is difficult to achieve in practice. In a real world problem setting, Feature Importance value will be either provided by human experts or they can be calculated from some other data set or problem. This importance value will also be an estimate of the true Feature Importance with associated error.

B. Derivation with Approximate Importance Value

Let us define I_X^* to be the Input Feature Importance given to our algorithm. Assume, I_X^* to be close to I_X by a bound τ, where τ is a small value. We are not bounding how small τ will be however. Therefore, $I_X - I_X^* < \tau$.

Based on this definition, the error bound for $S_I(X)$ will be:

$$E_{S_I} = (1-p).\epsilon + p.\tau$$

Error bound for $S(X)$ is not fixed on every node of the tree. It depends on the number of training instances available in a particular node. As the tree is being built, the data set is divided; child nodes receive less data and thus increasing the chances of error. Especially, in the top levels of the tree, number of instances available is high. Value of ϵ will be much less in those nodes, but value of τ will remain as is. If $\tau > \epsilon$ on the top levels, then instead of reducing the chances of error, it will actually increase the error possibility. Therefore, *IADT-0* can not be assumed to have lesser error bound.

In order solve the shortcomings of *IADT-0*, we developed *IADT*. This algorithm uses the same scoring function $S_I(X)$, but it changes the way both terms are weighted in $S_I(X)$.

The key idea in *IADT* is that, in the upper levels, only training data is sufficient to select the best attribute with strong statistical confidence. As the tree goes down the lower levels, uncertainty increases as the data is reduced on each node. We can use Feature Importance value in those lower levels. Therefore, value of weight parameter p will depend on the number of samples available in a given node. As we go down the lower levels, p will be increased and thus increasing the weight of I_X. This will reduce the uncertainty in the lower levels where very few training instances are available. We define p based on the ratio of number of instances in a node n_t, and total number instances in the data set n.

$$p = 1 - \frac{n_t}{n} \quad \& \quad S(X) = (1-p).S(X) + p.I_X$$

Based on this error bound for a particular node:

$$\begin{aligned} E^t_{S_I} &= (1 - 1 + \frac{n_t}{n})\epsilon + (1 - \frac{n_t}{n})\tau \\ &= \frac{\sqrt{n_t}}{n}\sqrt{\frac{R^2}{2}\ln\left(\frac{1}{\delta}\right)} + (1 - \frac{n_t}{n}).\tau \end{aligned} \tag{4}$$

The advantage of this error bound is that, error will always be minimized. In the top levels ϵ is less so $S(X)$ is given more weight. On the other hand, ϵ will be high on lower levels where S(X) will have less weight and I_X will have more weight.

The steps of IADT are summarized in Algorithm 1.

Algorithm 1 Importance Aided Decision Tree

```
Input: Dataset $D = (x_1, y_1) \ldots (x_n, y_n)$ , Feature Importance $I = I_1 ... I_d$
Output: Learned Tree $T$
  1) If all of the instances belong to the same class $C_i$, Stop
     and return the leaf node with class $C_i$ .
  2) Find the split $I_{best} = \underset{x}{\mathrm{argmax}}\, S_I(X)$ that best classifies
     the instances using $S_I(X) = (1-p).S(X) + p.I_X$
  3) Divide dataset using $I_{best}$. Each split divided the data
     into subsets. Calculate $p = 1 - \frac{n_t}{n}$ for each subset
  4) For each subset, repeat steps to construct the decision
     tree
```

IV. Empirical Evaluation

A. Synthetic Data sets

We tested first with artificially generated data sets that allow more flexibility about the concepts thus providing a more detailed empirical evaluation. The comparison was done between IADT and C4.5.

We randomly generated arbitrary decision trees of various size and complexity. 50 decision trees were generated that had trees with gradually increasing tree size. Tree size mostly followed the normal distribution that ranged from 100 leaves to 2000 leaves. The depth also varied from 2 to 15. There were 100 binary attributes in all the concepts. Then based on these concepts, we generated instances using a normal distribution of random mean and variance. Then various levels of noise were added ranging from 0% to 30%. The noise levels means that there $n\%$ chance of being assigned an arbitrary value as the class. The testing was performed with a $20k$ test set generated from the same distribution.

We first show the accuracy of averaged over all the concepts over number of instances given as training set at Figure 1. The results show that C4.5 performs much poorly compared IADT. The performance gap is highest initially. But as the training set size increases, the gap is reduced.

We show how the performance fares when varying concept size in Figure 2. The training set size was 10k while the test set size was kept at 20k like before. Like the previous test, IADT performs better than C4.5. IADT is highly robust to complex problems and requires far less training data than C4.5. Figure 3 shows the average number of nodes in the trees induced by each of the learners. It can be seen C4.5 produces far more nodes than IADT.

Figure 4 shows how the algorithms respond to noise. It compares several runs on 10 concepts of similar size, but with increasing levels of noise added to the training sets. the training sets used has 10k instances. IADT has better accuracy than C4.5 and the gap is increased as more noise is added. This shows IADT is less prone to noise than C4.5.

B. Experiments on UCI data sets

We tested our algorithm with several benchmark data sets from the University of California, Irvine (UCI) data set repository. The datasets were divided in two parts, importance generation set and experimentation set. In order to have a valid Importance knowledge, we decided to calculate the Importance of features using 30% of the data set. The rest of the data set was used for the experiments. This was done for all the datasets except promoters which has an associated Expert knowledge. The Importance values for Promoters dataset was derived from that knowledge. These Importance values were given to IADT. The algorithms were then experimented using 10-folds cross validation on the experimentation set. Experiments are averaged over 20 iterations. The results are shown in Table I. The results show clear difference between IADT and standard Decision Tree learner. IADT outperforms C4.5 in all data sets. The performance gap is around 10% on some datasets. This is a significant increase in accuracy over C4.5.

C. Experiment on Bank Loan Screening Problem

We also tested our algorithm in a real world problem setting. The Loan Screening is a common and important task in banks. The goal is to detect applicants that are suitable for having a loan of a particular amount. There are many factors to this problem that might affect the response; such as income, past record, background and so on. These factors interact non-trivially to affect whether the applicant is suitable for the loan or not. Each instance will be composed of user information such as past credit records, income, family background, job information and so on. We have collected past records of accepted and rejected candidates from a bank. The collected data set had 1000 instances. There were redundant information such as name, address and other personal information unrelated to this problem. Those information were discarded and we composed each record with 20 attributes.

These 20 attributes present a complete financial profile of each candidate. We then took expert opinion from three actual operators to identify the important attributes. The experts were given a choice to rate the features with importance score of $1-5$. These scores were then normalized between 0-1 to generate input Feature Importance Score I_X. We then performed actual classification experiment. We performed 10-fold cross validation on the full dataset. We also performed an incremental training testing. 30 of the data set was kept aside

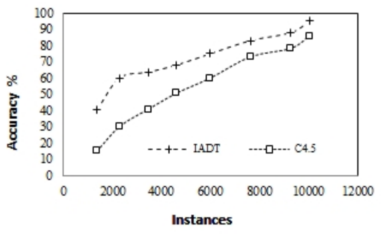

Fig. 1. Accuracy % over Number of instances

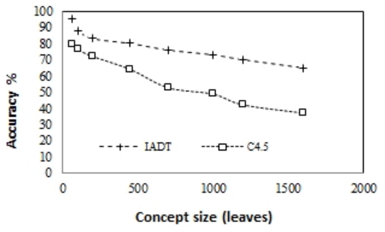

Fig. 2. Accuracy % over Concept size (Leaves)

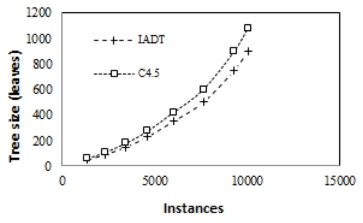

Fig. 3. Tree size over Number of instances

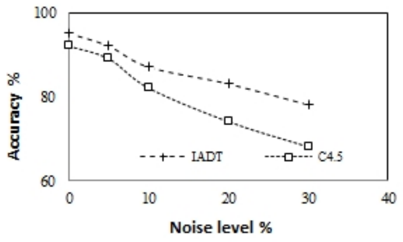

Fig. 4. Accuracy % over Noise level %

Methods	Promoters	Brst. Canc.	Hrt Dis.	Vote	Monks 1	Monks 2	Monks 3
IADT	94.83	95.57	87.83	96.35	100.0	92.84	100
C4.5	85.93	88.34	79.12	93.19	98.34	75.94	98.39

TABLE I
ACCURACY ON DIFFERENT DATASETS

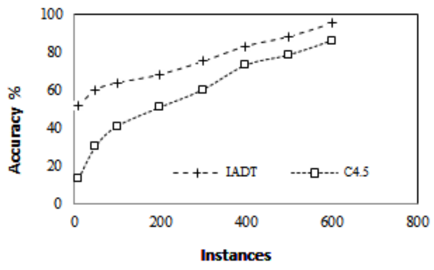

Fig. 5. Accuracy on Bank Loan Screening data set

as test set. The rest of the data was used to incrementally test the performance of the algorithm providing incrementally higher number of training data. This would generate a learning curve that shows how fast our algorithm learns compared to classic decision tree.

The results show the superiority of our algorithm over C4.5. IADT attained accuracy of 94.52% during 10-fold cross validation while C4.5 only gained 86.48%. The performance gap is quite high among classification tasks. Moreover, the learning curve in Fig. 5 shows how quickly IADT has learned compared to classic decision tree. Our algorithm learns much faster and attains higher accuracy when less data is presented. Even when no data is given our algorithms attains accuracy of 53.37% which is very high compared to 13.83% that C4.5 attains. This high is reduced when more instances are presented but IADT retains its lead over standard decision tree.

V. CONCLUSION

We have proposed a well performed approach of incorporating feature importance into Decision Tree learning. The performance of such a learner shows feature importance aided learners can achieve superior performance over ordinary inductive learners. Extra knowledge could be transferred to IADT to attain higher performance. This approach of incorporating feature importance into learners is worthy of further development. Possible future applications of this algorithm will be areas where related machine learning problems are being solved or where expert knowledge is available. The future research areas can be modifications of existing popular empirical learners so that they utilize feature importance. Importance aided algorithms maybe developed for algorithms such as Support Vector Machines or Bayesian classifiers. Current machine learning algorithms rely too much on training examples. Incorporating more and more domain knowledge or using the knowledge from related problem area is the way for improvement. Our proposed method shows how improvements can be had from such methods.

REFERENCES

[1] A., CLAYTON, J. . G. E. S. *A practical guide to knowledge acquisition.* Addison-Wesley, 1991.
[2] ALON, N., AND SPENCER, J. *The Probabilistic Method.* Wiley, New York:, 1992.
[3] BREIMAN, L., FREIDMAN, J. H., OLSHEN, R. A., AND STONE, C. J. Classification and regression trees.
[4] HOEFFDING, W. Probability inequalities for sums of bounded random variables&quot;. *J. Amer. Statist. Assoc.*, 13–30.
[5] IQBAL, R. A. Empirical learning aided by weak knowledge in the form of feature importance. CMSP'11, IEEE.
[6] MITCHELL, T. M. *Artificial neural networks.* McGraw-Hill Science/Engineering/Math, 1997, pp. 81–126.
[7] MITCHELL, T. M. *Machine Learning.* McGraw-Hill, 1997a.
[8] QUINLAN, J. R. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA,, 1993.
[9] VAPNIK, V. N. *Statistical Learning Theory.* Wiley, New York,, 1998.
[10] ZHANG, L., AND WANG, Z. Ontology-based clustering algorithm with feature weights. *Journal of Computational Information Systems 6*, 9 (2010).