A Novel EagerDT Complexity Approach to Deal with Missing Values in Decision Trees

Sachin Gavankar, Sudhirkumar Sawarkar

Department of Computer Engineering
Datta Meghe College of Engineering
Mumbai University, India.

Email: sachingavankar@yahoo.co.in; Sudhir_sawarkar@yahoo.com

***Abstract* - Incomplete data in decision trees affect classification accuracy. 'Lazy' approaches avoid missing values at testing by considering only attributes whose values are known and hence provide the best accuracy. We propose EagerDT, a variant of Eager Decision Tree, to build a single classification model at training considering the possibility of unknown values at every node in the tree. This removes the problem of missing values like lazy strategy. The biggest advantage of EagerDT over the lazy approach is that it creates a single tree at training. We describe the complexity of EagerDT algorithm and compare it with regular decision trees. We propose various novel approaches to reduce the complexity.**

Keywords - data mining, decision tree, EagerDT, missing values, testing data, complexity.

I. INTRODUCTION

Decision Tree introduced by Quinlan [1] is one of the best classification algorithms. Data mining deals with very large amount of data and data completeness at training and testing is very important for better prediction accuracy. Many approaches have been proposed to reduce the impact of missing values on prediction accuracy. Friedman [2] suggested 'Lazy Decision Tree' which delays the learning till the time we know exactly which attribute values are known. Maytal-Saar [3] explained, "Known value strategy (lazy) is the best strategy to deal with missing values from test data".

In our earlier work we proposed a new eager decision tree algorithm 'EagerDT' [4] which keeps provision for unknowns at every node of decision tree. During tree construction, at every node it creates one additional branch named as 'UAT - Unknown At Test', in addition to the actual branches of the node. At testing, if an attribute value is missing, the algorithm selects the 'UAT - Unknown At Test' branch and asks for the next attribute value. The best part of this algorithm is that the classification model is constructed at training and addresses the problem of missing values in test data.

In this paper we have discussed the complexity of regular decision tree and EagerDT at the time of tree construction and classification. We have proposed few novel methods to reduce the complexity of EagerDT.

II. LITERATURE REVIEW AND LIMITATIONS OF CURRENT TECHNIQUES

Decision Tree is a tree where each node represents a test of an attribute and leaf node provides classification. Test example is classified by starting at root node, testing feature values at each node and sorting down to the appropriate branch till it reaches leaf node which provides classification.

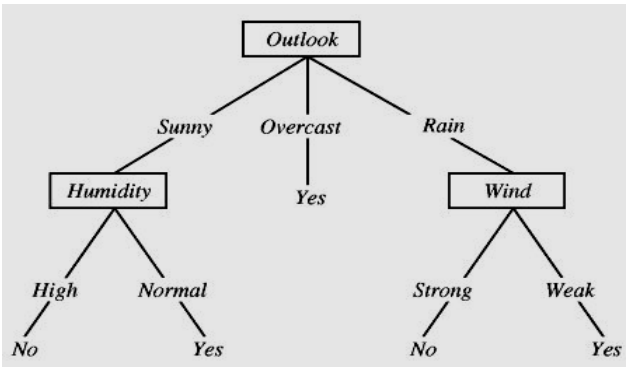

Figure 1. Learned regular Decision Tree for PlayTennis data

TABLE I. TRAINING SET FOR PLAYTENNIS

Day	Outlook	Temp.	Humidity	Wind	PlayTennis
T1	Sunny	Hot	High	Weak	No
T2	Sunny	Hot	High	Strong	No
T3	Overcast	Hot	High	Weak	Yes
T4	Rain	Mild	High	Weak	Yes
T5	Rain	Cool	Normal	Weak	Yes
T6	Rain	Cool	Normal	Strong	No
T7	Overcast	Cool	Normal	Weak	Yes
T8	Sunny	Mild	High	Weak	No
T9	Sunny	Cool	Normal	Weak	Yes
T10	Rain	Mild	Normal	Strong	Yes
T11	Sunny	Mild	Normal	Strong	Yes
T12	Overcast	Mild	High	Strong	Yes
T13	Overcast	Hot	Normal	Weak	Yes
T14	Rain	Mild	High	Strong	No

During tree construction, the feature which gives maximum information gain is selected at root node, possible values are branches from the node. The training examples are split according to the values of selected node attribute and tree construction algorithm is applied recursively to each subset. E.g. as shown in figure 2, for PlayTennis training set, Outlook is the best attribute which provides maximum information gain. It is selected at top and training set is divided into three subsets based on the value of Outlook – Sunny, Overcast and rain.

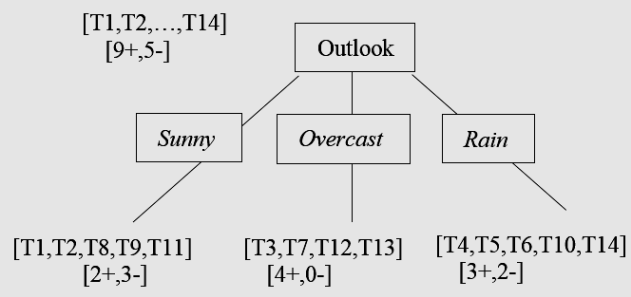

Figure 2. Regular Decision Tree – Distribution of Training examples

The recursive construction of sub trees stopped either in case no attribute left for next node or number of training records are less than the threshold. In both the cases the node is considered as leaf node having class equals to the common value of class in the corresponding training records. The tree once constructed can be considered for classification of testing data.

The complexity [5] in classification tree can be studied at the time of construction of the model and at classification. Once tree has been built, the number of leaves can be calculated from the number of internal nodes.

Let's assume, N be the total number of nodes in the decision tree. Each internal node i corresponds to a test attribute Ai and it takes vi possible values. Number of leaves in the tree can be calculated as:

$$L = \sum_{i=1}^{i=N} vi - (N - 1)$$

In a binary tree, the number of leaves is equal to number of internal nodes + 1. Maximum number of internal nodes in binary decision tree with height h is 2h - 1. In other words, height h is at least log2 (N+1) where binary tree has at most 2h – 1 internal nodes. Similarly, the height h of a binary decision tree, that has at most L = 2h leaves, is at least log2 L.

In case of n-ary decision tree, assume, L: number of leaves in the tree, N: number of internal nodes having values $v1$, $v2$, $v3$,.., vN, Avg v = ($v1$ + $v2$ + $v3$+..+ vN) / N. The height h of the decision tree is at least log$v$ L.

The complexity of decision tree decided by a) the number of attributes and b) the number of training records. Assume m is the number of attributes and n is the size of training set, the complexity to construct decision tree is [6]:

$$O (n x (m + 1) x \log(n))$$

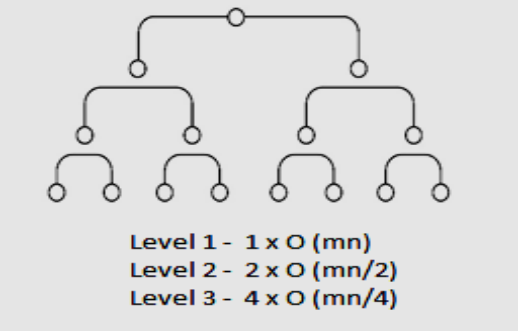

Figure 3. Distribution of training records for balanced tree.

Figure 3 illustrates the method for calculating the complexity during tree construction. Training set is divided into subsets based on the value of the attribute selected. However, the total number of training examples remain same resulting into complexity n x (m + 1) at every layer. Assuming a roughly balanced tree having about log$n$ layers, we see the complexity:

$$O (n x (m + 1) x \log(n))$$

Test example is classified by starting at root, testing attribute values at each node and sorting down to the appropriate branch till it reaches leaf node which provides class value. The worst case complexity is O(h), where h is the height of the tree [6] or O(log$v$ (L)). In case of missing value, multiple paths to be traced and the worst-case complexity becomes:

$$O (L \log_v (L))$$

Since the decision tree deals with huge data at training and testing, data quality is very important from prediction accuracy point of view. There many approaches proposed by researchers [7]. Most of them rely on imputation. Friedman[2] suggested 'Lazy Decision Tree' which delays the learning till we know exactly which attribute values are known. Maytal-Saar [3] explained, "Known value strategy is the best strategy to deal with missing values from test data". The challenge with known value strategy is that classification model (or part) is constructed at the time of testing, resulting into classification time computation and time delay. Alternatively, multiple trees can be computed in advance. Lazy or known value techniques are the best techniques to address the missing value problem in test data. The only disadvantage is that the tree is constructed at testing. In next section we have addressed this problem.

III. THE PROPOSED NEW ALGORITHM - EAGERDT

We presented a novel decision tree algorithm 'EagerDT'[4], which constructs classification model at the time of training and handles all possibilities of missing values at testing.

In EagerDT, at the time of construction of tree for every node a branch called 'UAT- Unknown at Test' is constructed in additional to the other branches as per the possible outcome of the test. This branch will consider all training examples available at the parent node and exclude the test which is selected at parent from attribute list for subtree construction as illustrated in figure 5 and apply algorithm recursively. Finally, it will be a tree same as normal decision tree with additional branch, UAT at each and every node.

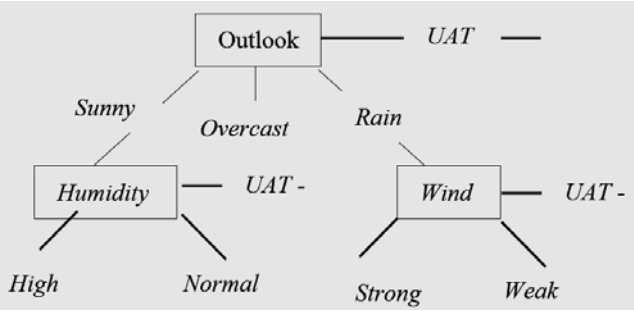

Figure 4. Decision Tree with UAT branches (representative)

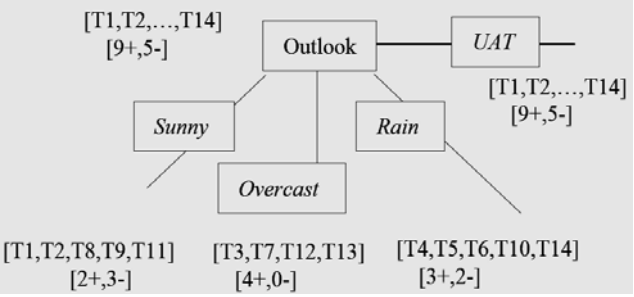

Figure 5. Use of training examples from parent node for construction of UAT branch

Figure 5 shows that training examples from parent node are split according to the value of node attribute. In addition to that all attributes from node are also copied to the UAT branch.

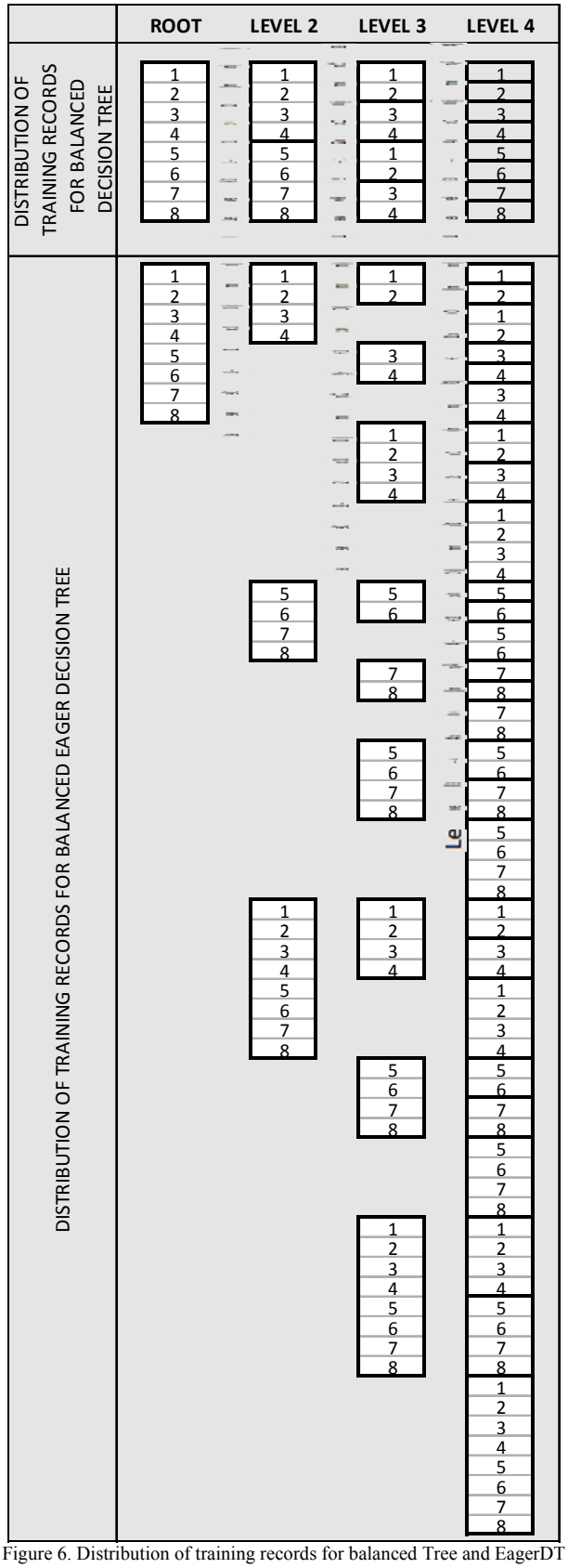

Figure 6. Distribution of training records for balanced Tree and EagerDT

In EagerDT, additional branch is included at every node and assigned complete training set to that branch, in addition to the regular spilt of training examples based on the respective attribute values. This leads to creation of another training set at every level resulting into 2 -fold increase of number of training records at every level of the tree. Figure 6 and figure 7 illustrates the same. The number of training records doubles at every level resulting into complexity n x (m + 1) x 2 Log n.

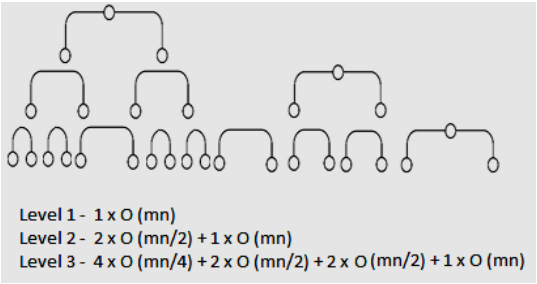

Figure 7. Distribution of training records for balanced EagerDT

Test example is classified by starting at root, testing attribute values at each node and sorting down to the appropriate branch till it reaches to the leaf node which provides class. In case of missing value, it simply selects UAT branch. The worst case complexity is same as that of regular decision tree O(h) where h is the height of the tree or O(logv (L)).

IV. METHODS TO REDUCE EAGERDT COMPLEXITY

As we have seen, the biggest advantage of EagerDT is that it constructs single tree at training time. Since it considers the possibility of unknown entities at every node and constructs sub-tree for UAT branches, complexity at the time of construction is relatively high. However, at the time of classification, it is extremely fast. We propose three approaches to reduce the complexity:

i) EagerDT – UAT Ready: Firstly, we suggest tree construction in normal fashion and store training records at each node without UAT branches, at training time. During testing, if 'Unknown' encounters, respective UAT branch is constructed using the stored training records. Here basic EagerDT framework is in place and UAT branches will be constructed and stored as and when they are required. This might take some time at testing. However, it is less than that of construction of entire tree at testing.

ii) EagerDT – UAT Limited: Many times, values are not known for a specific reason and we know in advance the attributes whose values might not be available. In this case we suggest the construction of UAT branch only for those attributes. All other nodes there is no need to construct UAT branch. This reduces complexity significantly.

ii) EagerDT - UAT Limited + UAT Ready: This is a combination of UAT Limited and UAT Ready approaches. Initially UAT branches are constructed for all the nodes where there are chances of missing values (UAT Limited) and remaining nodes will be equipped with training attributes for UAT Ready approach to construct the UAT branches as and when required.

TABLE II. COMPLEXITY REDUCTION

Approach	At Training	At Testing
EagerDT – UAT Ready	Explicit UAT branches from each node are not constructed. Training records are stored at each Node.	If value is missing, UAT branch is constructed using stored training attributes
EagerDT – UAT Limited	UAT branches are constructed only for attributes which might have missing values. Training records are not stored for remaining attributes	If value is missing for attributes having UAT branch, follow the UAT branch.
EagerDT – UAT Ready + UAT Limited	UAT branches are constructed only for attributes which might have missing values. - Training records are stored for at node for remaining attributes.	If value is missing for attributes having UAT branch, follow the UAT branch. - If value is missing from other attributes, UAT branch is constructed using stored training attributes

Since EagerDT is a single tree constructed at training time, almost all the known pruning algorithms can be directly applied to EagerDT, unlike the Lazy tree where the weakest point was there is no regularization (pruning).

A. Software Used and Methodology

We have written the EagerDT algorithm using Java and tested on various datasets like Breast, Credit, Diabetes and Iris from UCI library. We have modified the algorithm to accommodate the techniques to reduce the complexity.

V. RESULTS AND DISCUSSION

We have considered the commonly referred example of 'Play Tennis' data. There are 4 attributes and 14 training records as shown in Table 1. The constructed EagerDT is shown in figure 8 (partial). Due to the introduction of new UAT branch at every node, the tree complexity is increased considerably.

```
outlook = sunny
| humidity = high: no
| humidity = normal: yes
| humidity = UAT
  | temperature = hot: no
  | temperature = mild
    | wind = strong: yes
    | wind = weak: no
    | wind = UAT: no
  | temperature = cool: yes
  | temperature = UAT
    | wind = strong: yes
    | wind = weak: no
    | wind = UAT: false
outlook = overcast: yes
outlook = rainy
| wind = strong: no
| wind = weak: yes
| wind = UAT
  | temperature = cool: yes
  .........
```

Figure 8. Eager Decision Tree for PlayTennis data

In know value strategy, 2n-1 trees based on various combination of missing features to be constructed. Complexity in terms of number of nodes for regular decision tree, set of all 2n-1 trees (precomputed known value strategy, EagerDT and its complexity reduction approaches EagerDT-UAT Ready & EagerDT-UAT Limited have been listed below.

TABLE III. COMPLEXITY FOR PLAYTENNIS DATA

Description	No. of Nodes (PlayTennis)
No. of Attributes	4
No. of Records	14
No. of Nodes in Regular Tree	3
No. of Nodes in 2n-1 trees	40
No. of Nodes in the EagerDT	20
No. of Nodes in EagerDT – UAT Ready	Initially – 20 Increase in case of missing values
No. of Nodes in EagerDT – UAT Limited	Range 20-40, based on the number of UAT attributes selected

VI. CONCLUSION AND FUTURE WORK

Initially, we reviewed various techniques to address missing values issue in training data as well as in testing data. There are limited number of methods to deal with the missing values at test. Known Value Strategy (Lazy Decision Tree) is the best method, where attributes with missing values are avoided by constructing the tree at testing time using only known attributes. Our novel technique, EagerDT constructs complete tree at training. It has UAT, Unknown At Test, branch at every node to address missing values at testing. At testing if missing value are encountered it simply opts for UAT branch and attributes with the missing value eliminated naturally.

We addressed the complexity of EagerDT at the time of tree construction and classification. The complexity of tree construction is high in EagerDT due to the additional UAT branches. We proposed three approaches to reduce the complexity: i) in the first approach, training records are stored at each node in EagerDT and we construct UAT subtree only in case of missing value at testing (UAT Ready), ii) the second approach does not create an additional branch if there is no possibility of having missing values for the attribute resulting into a huge reduction in the complexity (UAT Limited), iii) alternatively, we start with only those UAT branches that are constructed where there are more possibilities of unknowns and, as we come across additional unknowns, respective UAT branches are added using training records stored at respective nodes.

The EagerDT completely solves the problem of missing values at prediction and the above approaches reduce EagerDT complexity considerably. This work can be extended by applying various decision tree pruning techniques to reduce the complexity and increase the prediction accuracy.

REFERENCES

[1] Quinlan, J. R. (1986). "Induction of decision trees", Machine Learning 1(1): 81–106.

[2] J. Friedman, Y. Yun, and R. Kohavi, "Lazy Decision Tree," Proc. 13th National Conf. Artificial Intelligence, pp. 717-724, 1996.

[3] Maytal Saar-Tsechansky, Foster Provost, "Handling Missing Values when Applying Classification Models", Journal of Machine Learning Research 8 (2007) 1625-1657.

[4] Sachin Gavankar, Sudhir Sawarkar, "Eager Decision Tree", 2nd International Conference for Convergence in Technology- I2CT, Pune, 2017.

[5] Lamis Hawarah, Ana Simonet, Michel Simonet, "The Complexity of a Probablistic Approach to Deal with Missing Values in a Decision Tree", Proceddings of the Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing-SYNASC'06..

[6] J. K. Marin and D. S. Hirschberg. The time complexity of decision tree induction. Technical Report ICS-TR-95-27.

[7] Sachin Gavankar, Sudhir Sawarkar, "Decision Tree: Review of Techniques for Missing Values at Training, Testing and Compatibility", Proc. of the 3rd Int'l Conf on Artificial Intelligence, Modelling and Simulation, AIMS2015, Malaysia, 2015.

[8] Sachin Gavankar, Sudhir Sawarkar, "Decision Tree: Compatibility of Techniques for Handling Missing Values at Training and Testing", International Journal of Simulation: Systems, Science and Technology, Vol-17, No-34, 2016.

[9] J.R.Quinlan, "C4.5 Programs for Machine Learning", Morgan Kaufmann Publications, San Mateo, CA, 1993.

[10] Jiawei Han, Michline Kamber, "Data Mining Concepts and Technique" , Kaufmann Pulications, 2001.

[11] David Hand, Heiki Mannila, Padhraic Smyth, "Principles of Data Mining", The MIT Press.

[12] Tom Mitchell, "Machine Learning", Mc-GrawHill Publications

[13] S. Zhang "Missing is Useful : Missing Values in Cost-Sensitive Decision Trees", IEEE Transactions on Knowledge and Data Engineering, Vol 17, No. 12, 2005.

[14] Shichao Zhang, Xindong Wu, Manlong Zhu, "Efficient missing data imputation for supervised learning", Proc. 9th Int'l Conf. on Cognitive Informatics (ICCI'10), 2010.

[15] A. N. Gulati and S. D. Sawarkar, "A novel technique for multidocument Hindi text summarization," *2017 International Conference on Nascent Technologies in Engineering (ICNTE)*, Navi Mumbai, 2017, pp. 1-6.