A Criteria based Decision Tree for Classification

John Tsiligaridis
Math and Computer Science
Heritage University
3240 Fort Road, Toppenish, WA, 98948, USA
tsiligaridis_j@heritage.edu

Abstract. *A Decision Tree algorithm (DTA) from data is created using construction criteria and also considering the don't care attributes, for each level of the tree. The DTA goal is to create on-demand a short and accurate decision tree from either data or a stable (or dynamically changing) set of rules. A set of steps provides a decision tree with a definite number of nodes and leaves. The pruning of decision rules case is also examined with the consequences on the accuracy. An improved version of DTA (IDTA) provides smaller DT and eliminates branches by using the criterion of less length (CLL).*

Keywords. Decision Tree, Data Mining

1 Introduction

Decision Trees represent a well-known machine learning technique used to find predictive rules combining numeric and categorical attributes, which raises the question of how associate rules compare to induce rules by a decision tree [1].
In decision trees the input data set has one attribute called class C that takes a value from K discrete values 1,..,K and a set of numeric and categorical attributes A1,..,Ap. [1].
The goal is to predict C given A1,..,Ap. Decision trees algorithms automatically split numeric attributes A_i into two ranges and they split categorical attributes A_j into two subsets at each node. The basic goal is to maximize the class prediction accuracy P(C=c) at a terminal node (also called node purity) where the most points belong to class c and $c \in \{1,..K\}$.
The splitting process is recursively repeated until the end of the data or until there is no improvement of prediction accuracy with a new split.
The final step involves pruning nodes to make the tree smaller and to avoid model overfit.
The output is a set of rules that go from the root to each terminal node consisting of a conjunction of inequalities for numeric variables ($A_i<=x$, $A_i>x$) and set containment for categorical variables ($A_j \in \{x,y,z\}$) and a predicted value c for class C.
In general, DTs have reasonable accuracy and they are easy to interpret if the tree only has a few nodes. Accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.

There are two types of DT: the complete and the incomplete one. In the incomplete there are subtrees where the repetition and replication are included [2]. Repetition is where an attribute is repeatedly tested along a given branch of three , e.i. age, and replication where duplicate subtrees exist within a tree , such as the subtree headed by the node "credit_rating" [2]. Two theorems are developed the first discover any of the two types of a DT in advance given the rules or data, and the second identifies the don't care attributes.

Decision tree is a very popular and practical approach for pattern classification. There are various algorithms for construction decision trees like ID3, C4.5 and CART[2],[3]. The ID3 [4] uses the information gain measure to choose the splitting attribute. To build a decision tree, information gain is calculated for each and every attribute and the attribute with the highest information gain is selected and designated as a root node. ID3 is based on information theory and uses the log function with base 2 for encoded information in bits. DTA can build a DT classifier from a set of data, without the use of the log function. DTA has a processing method similar to the way ID3 works, for finding root and splitting attributes, by utilizing the discovery of maximum values of conditional probabilities instead of the higher information gain. The phases of DTA also cover the case of finding a leaf at the end of a branch without checking whether all the instances are falling under the same class as it happens in ID3. With the CLL the case of extension of a subtree with a new attribute is examined. DTA and IDTA can provide DT with no repetition and replications which are unfavorable for the classification. The C4.5 is an extension to ID3 [5] and uses the criterion of gain ratio which takes into account the number of outcomes produced by the attribute test condition. The attributes with the maximum gain ratio are selected as the splitting attribute. It removes the biasness of information gain when there are many outcome values of an attribute. DTA processing is also similar to the one of C4.5. CART produces binary trees. Gini index measure does not use

probabilistic assumptions like ID3, C4.5. CART [2] uses cost complexity pruning to remove the unreliable branches from the DT to improve accuracy.
In [6] simplification methods of Decision Trees are developed. A new decision tree based classification algorithm, called SPRINT, has been developed in [7] for categorical and continuous attributes in a parallel environment.

2 Model Description

2.1 The DTA

The attributes selection measures, determine how the tuples at a given node are to split. There are two types of attributes that are considered; the "split" or "occupied" and the "free". The first type is already included in the tree while the latter is the attribute that is not yet included. The tree has three types of nodes; a root node, internal nodes and leaf or terminal nodes.

For the construction of the DTA two construction criteria are used: (a) finding the root (max($p(c_i / n)$, n: is the # of tuples) *(criterion 1)*, (b) discovering which branch will be connected with the next node (attribute) using conditional probabilities *(criterion 2)*. The characteristic of the DTA is that the projection of the probabilities of all attributes over a predefined value of C (=ci) is examined.
The DT can be created in the following phases:
Phase 1: Discover the root (i) (from all attributes)

max ($p(E_i)$ = max($\sum_{i=1}^{k} p(A_i)\,p(D_i)$), k= # attributes, for all n , number of tuples

Phase 2: Discover the branches (splitting the root) from the conditional probabilities ($P(B_i)$,for node(attribute) B, so that max ($P(B_i)$) = max ($p(A_i = v_i / P(C=c_i))$) $\neq 0$
If there is a value of attribute, i=k, so that $p(A_k= v_k / P(C=c_k)) \neq 0$ and for all the other, m≠k, $p(A_k= v_k / P(C=c_m)) =0$, then there is a branch with the value of the attribute A_k and node B becomes a leaf node with assigned c_k as a class label.
Phase 3: Discover the next node to be connected (with the root) . For each value i of the occupied attribute A_i find the next free attribute A_j the :

max ($p(E_i)$ = $\sum_{j=1}^{k} p(A_{i,p} = v_{i,p} / A_{j,c} = v_{j,c})$, k= # of values of A_j attribute, for n = # tuples, The term $A_{i,p} = v_{i,p}$ means the attribute A_i with value v_i
It is possible to consider Phase 0, where you can define the value of the category with the maximum value, max{ pr(C=c)}

From all the above the DTA pseudocode is as follows:

```
DTA : Input : training data
Output: decision tree
1. define root node (phase 1)
2. discover the branches from root (phase 2)
while (! end of the attributes)
 {  3. discover the next node (phase 3)
    4. splitting the attribute (phase 2)  }
```

Example 1:
Let's consider the weather example

Weather	parents	money	decision (Example)
Sunny	yes	rich	cinema
Sunny	no	rich	Tennis
…..			
Windy	no	rich	cinema (7*)
…..			

Following the steps of DTA we have:
(a)Discovering root: P(E) = (5/10)*(5/10) + (5/10)*(1/10) = 0.3 *(phase 1)*
(b)Discovering branches: max ($P(B_i)$) =(p(yes parents / C="cinema") , p(yes parents / C="tennis") , p(yes parents / C="shopping") , p(yes parents / C="stay in") = max (5/6,0, 0,0) *(phase 2)*
(c)Discovering next node: p(no parents / weather="sunny") + p(no parents / weather ="windy") + p(no parents / weather="rainy") = 2/3 + 1/3 + 1/3 = 4/3 *(phase 3)*

From the above: (a) parents have the highest value and become the splitting attribute at the root node of the DT, (b) the left branch (for "yes" on parents) the decision is "cinema" no matter what the value of the other two attributes is, (c) the right branch (the "no")of parents is connected with the weather attribute. (d) the process continues with the (3) and (4) in the while loop.
The DT is:

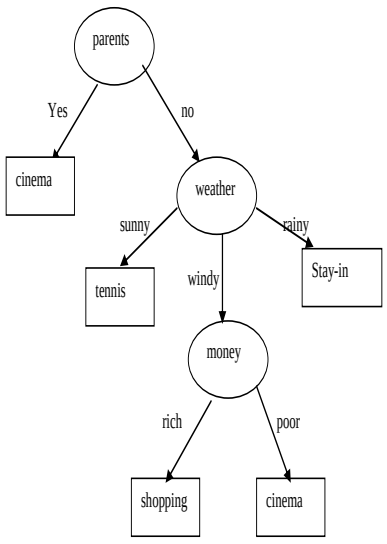

Fig. 1 The Decision tree created by DTA

This DT consists of 3 nodes and 5 leaves with 100% classification accuracy for the data
From DT we can get the rules and some of them are dntca (from "don't care"). The rule set can be created by running the tree.

Example 2:

R1	Cinema<- parents="yes" & weather="dntca" & money="dntca"

From R1 when parents="yes" then D="cinema" and no matter what are the other attributes.
Phase 1 is analogous, discovering the root with the highest value of information gain [2],[3].

2.2 Prunning the rules

We can extract the rules either from the DT by simply finding the paths from root to a leaf node in the form of IF-THEN rules. If the rules are pruned or if we prune the last level then we will have less accuracy. If line 7 is erased: windy, no (parents), rich -> cinema and then accuracy 1 out of 10 , giving predictive accuracy 90% and consists of 2 nodes and 4 leaves. Fig. 2 provides the new DT with the less accuracy using DTA.

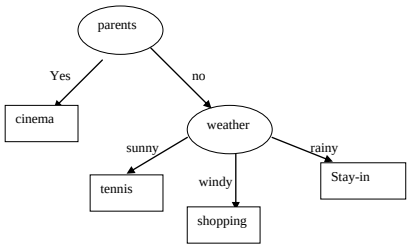

Fig. 2 DT after pruning

2.3 The IDTA

For an attribute (attr1) with value v1, if there are tuples from attr2 that have all the values in relation with v1 (of attr1) then the attr2 is named as: don't care attribute (see Example2, see R1). The way of discovering the existence of don't care attributes is developed by using conditional probabilities and by applying the criterion of less length (P_{CLL}).

It is supposed that all the attributes are conditional independent given the class value $C=c_i$.

From the conditional independence assumption, given the class value and the value a_k of an attribute A_k , we can have:

$P_{CLL} = P(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C=c_i , A_k=a_k) =$

$$\prod_{i=1}^{|A|} p(A_i = a_i \mid C = c_j) \quad (1)$$

In (1) we take into consideration only tuples with $A_k=a_k$. According to the criterion of CLL; if the $P_{CLL} \neq 0$, between two attributes (A1, A2) then A2 is a don't care attribute. The CLL criterion is valid when $P_{CLL} \neq 0$. A branch is eliminated when the $P_{CLA} \neq 0$.
If $P_{CLL} = 0$ new partitions have to be included in the DT.

Example 3: From the left branch of the DT (cinema) we figure out that it is not necessary to have more probable partitions. Here the parent attribute is : parents ="yes" and the child attribute is: weather.
$P_{CLL} = P(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C=c_i) = P$(weather =sunny | C="cinema") * P(weather =windy | C="cinema") * P (weather =rainy | C="cinema") = 1/ 5 * 2/5 * 2/5 ≠ 0. We also get the same result ($P_{CLL} \neq 0$) when we consider the money attribute under the parent attribute.
Hence when parents="yes", there are don't care attributes (weather, money) and the left branch will stop without any extension.

A DT is complete when it has 100% accuracy, (all tuples are qualified). After finding the root and if the criterion CLL for any of the attributes is not valid there is a possibility to have repetitions (incomplete tree) [2].

A theorem can provide the possibility of a complete DT existence in advance, given the data or the rules.

Theorem 1: The CLL criterion can determine the existence of a small DT with the best accuracy (100%, or complete) avoiding repetitions and replications.

Proof: Due to the validity of CLL criterion, repetition is discouraged. Having n-1 remaining attributes out of n, after discovering the root attribute, and the CLL criterion is not valid, for any of n-1 attributes, new partitions have to be included for the DT. A fact that increases the risk to have an incomplete tree (repetitions, replications). •
CLL criterion can minimize the height of the DT.
In case that CLL criterion is valid for many of the attributes the tree becomes of smaller height ('dense') which facilitates the read operation for finding rules.
The validity of CLL can be applied after the creation of each node. Phase 2 (splitting the attribute) from DTA have to be enriched with the CLL criterion so that repetitons or replications are avoided.
If DT can not cover all the data, this is also a problem. CLL can locate this problem, but this is not the focus of this paper. After all of the above, the steps for IDTA are as follows:

```
IDTA : Input : training data
Output: decision tree
//root operation
1. define root node  (phase 1)
2. discover the branches from root  (phase 2a)
   apply CLL  (phase2b)
//node operation
while (! end of the attributes)
 {
 3. discover the next node (phase 3)
 4. //apply CLL (phase 2a)
    if P CLL ≠ 0 (valid)
       { branch eliminated}
    else {splitting attribute}
 }
```

Theorem 2: The CLL can define the don't care attributes.

Proof: From the definition of P_{CLL} and from the parent attribute k and the child attribute m if $P_{CLL} \neq 0$ then the attribute m is a don't care attribute. •

Example 4: Consider the case of getting a loan with attributes: age has_job, own_house, credit_rating.

After finding the "own_house" as a root (phase 1) , the P_{CLA} is computed, in order to discover the branches from root. $P_{CLA} = P(A_1 = a_1,\ldots, A_{|A|} = a_{|A|} \mid C=c_i) = P(\text{age =young , own\_house="y"} \mid C="yes") = P(\text{age =young} \mid C="yes") * P(\text{own\_house="y"} \mid C="yes") \neq 0$. The same with age ="middle" and age="old". So, $P(\text{age =middle} \mid C="yes") * P(\text{own\_house="y"} \mid C="yes") \neq 0$, and $P(\text{age =old} \mid C="yes") * P(\text{own\_house="y"} \mid C="yes") \neq 0$

From the above, the P_{CLA} proves, that it is not appropriate to extend the branch (own_house (with true) -> 'yes') in order to include the "age" attribute. Hence, by applying the P_{CLA} shows that the "age" belongs to the don't care attribute.

3 Some Analytical Results

For ID3 at each node y the gain is calculated from n_y (# of candidate attributes at y), and from m_y (# of examples that reach y. The complexity of choosing an attribute is $O(n_y * m_y)$. For each level i of the tree there are m_i number of examples (bounded by m) and the number of attributes (n-i). So, it takes O(m * (n-i)) to find the splits for all nodes at level i. In the worst case, the tree will be of depth n and the total complexity will be $O(m*n^2)$. DTA has the same complexity with ID3, since the operations are similar. Therefore at each level i the complexity of choosing . an attribe is O(m * (n-i)), as previously. IDTA has the additional step of CLL which increases the complexity of choosing an attribute for a k node of the DTA, by the $O(n_k*m_k)$.

Simulation results, for a size of data of 15 , show that ID3 , DTA, and IDTA have 95.23%, 95.25% and 95.30% accuracy respectively. IDTA comparing with DTA and ID3, has better accuracy since it discourages the repetitions.

4 Conclusion

Two algorithms for discovering DTs have been developed also considering the don't care attributes. The DTA works iteratively with probabilities in order to find the nodes and the branches. The IDTA works more systematically with priority searching of the don't care attributes using the CLL and DTA. Both of them have the purpose of creating a smaller DT without repetition and replication. Moreover, DTA has the same complexity as ID3. The accuracy is better for IDTA due to the construction with CLL. Future work could elaborate more on classification issues with neural networks.

References

[1] C. Ordonez, Comparing Association rules and Decision Trees for Disease Prediction, *HIKM 2006*, Non. 11, 2011, Virginia

[2] J.Han, M. Kamber, J. Pei , *Data Mining Concepts and techniques*, MK, 3ed, 2012

[3] P. Tan, M. Steinbach, V. Kumar, *Introduction to Data Mining* , 2006, Addison Wesley;2006.

[4] J.R. Quinlan, "Induction of decision trees", *Journal of Machine Learning*, 1(1986) pp. 81-106.

[5] J.R. Quinlan,"C4.5 : Programs for machine Learning" , Morgan Kaufmann Publishers, Inc, 1992

[6] J.R. Quinlan, "Simplifyig Decision Trees", Intl. J. Man-Machine Studies 27:221-234, 1987

[7] J.Shafer, R. Agrawal, M. Metha,"SPRINT: A sclabale Parallel Classifier for Data Mining" , in *Proc. of 22nd VLDB Conf.*, Bombay,India, pp.544-555,September 1996.