Mr.Mitesh Thakkar, Prof.J.S.Shah / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 2,Mar-Apr 2012, pp.588-594

Logistic Model Tree: A Survey

Mr.Mitesh Thakkar [1], Prof.J.S.Shah [2]

[1] Semester IV, M.E. IT, Information Technology Department, Shantilal Shah Engineering College, Bhavnagar, India
[2] Professor and Head, Computer department, L.D. College of Engineering, Ahmedabad, India

Abstract - Tree induction methods and linear models are popular techniques for supervised learning tasks, both for the prediction of nominal classes and numeric values. For predicting numeric quantities, there has been work on combining these two schemes into 'model trees', i.e. trees that contain linear regression functions at the leaves. In Logistic Model Tree (LMT) logistic regression is used instead of linear regression. Their greatest disadvantage is the computational complexity of inducing the logistic regression models in the tree. This issue can be address by using the AIC criterion instead of crossvalidation to prevent overfitting these models. In addition, a weight trimming heuristic is used which produces a significant speedup.

Keywords: *Model tree, logistic regression.*

1 Introduction

Two popular methods for classification are linear logistic regression and tree induction, which have somewhat complementary advantages and disadvantages. The former fits a simple (linear) model to the data, and the process of model fitting is quite stable, resulting in low variance but potentially high bias. The latter, on the other hand, exhibits low bias but often high variance: it searches a less restricted space of models, allowing it to capture nonlinear patterns in the data, but making it less stable and prone to overfitting. So it is not surprising that neither of the two methods is superior in general.

A more natural way to deal with classification tasks is to use a combination of a tree structure and logistic regression models resulting in a single tree. Another advantage of using logistic regression is that explicit class probability estimates are produced rather than just a classification. Logistic Model Tree (LMT) follows this idea.

1.1 Logistic Regression Model

Fitting a logistic regression model means estimating the parameter vectors *βj.* The standard procedure in statistics is to look for the *maximum likelihood* estimate: choose the parameters that maximize the probability of the observed data points. For the logistic regression model, there are no closed-form solutions for these estimates. Instead, we have to use numeric optimization algorithms that approach the maximum likelihood solution iteratively and reach it in the limit. Friedman et al. Propose the LogitBoost algorithm for fitting *additive logistic regression models* by maximum likelihood (Friedman et al, 2000). These models are a generalization of the (linear) logistic regression models.

1. **Start with weights** $W_{ij} = 1/n$, i=1,……,n, j=1,……J, $F_j(x)=0$ and $P_j(x)=1/J\ \forall_j$

2.Repeat for m=1,…,M:

(a) Repeat for j=1,….,J

i. Compute working responses and weights in the j^{th} class

$$z_{ij} = \frac{y^*_{ij} - p_j(x_i)}{p_j(x_i)(1 - p_j(x_i))}$$

$$w_{ij} = p_j(x_i)(1 - p_j(x_i))$$

ii. Fit the function $f_{mj}(x)$ by a weighted squares regression of z_{ij} to x_i with weights w_{ij}

(b) Set $f_{mj}(x) \leftarrow \frac{J-1}{J}(f_{mj}(x) - \frac{1}{J}\sum_{k=1}^{J} f_{mk}(x)),$

$$F_j(x) \leftarrow F_j(x) + f_{mj}(x)$$

(c) $$p_j(x) = \frac{e^{F_j(x)}}{\sum_{k=1}^{J} e^{F_k(x)}}$$

3. Output the classifier argmax $F_j(x)$

Figure 1 LogitBoost algorithm (J classes)

Generally, they have the form

$$Pr(G = j|X = x) = \frac{e^{F_j(x)}}{\sum_{k=1}^{J} e^{F_k(x)}}, \quad \sum_{k=1}^{J} F_k(x) = 0,$$

Where $F_j(x) = \sum_{m=1}^{M} f_{mj}(x)$ and the f_{mj} are (not necessarily linear) functions of the input variables. Figure 1 gives the pseudocode for the algorithm. The variables y^*_{ij} encode the observed class membership probabilities for instance x_i, i.e.

$$y^*_{ij} = \begin{cases} 1 & \text{if } y_i = j, \\ 0 & \text{if } y_i \neq j \end{cases} \quad \text{.....(}$$

y_i is the class label of example x_i). The $p_j(x)$ are the estimates of the class probabilities for an instance x given by the model fit so far.

LogitBoost performs forward stagewise fitting: in every iteration, it computes 'response variables' z_{ij} that encode the error of the currently fit model on the training examples (in terms of probability estimates), and then tries to improve the model by adding a function f_{mj} to the committee F_j , fit to the response by least-squared error. As shown in (Friedman et al., 2000), this amounts to performing a quasi-Newton step in every iteration

1.2 Tree Induction

The goal of supervised learning is to find a subdivision of the instance space into regions labeled with one of the target classes. Top-down tree induction finds this subdivision by recursively splitting the instance space, stopping when the regions of the subdivision are reasonably 'pure' in the sense that they contain examples with mostly identical class labels. The regions are labeled with the majority class of the examples in that region.

Important advantages of tree models (with axis-parallel splits) are that they can be constructed efficiently and are easy to interpret. A path in a decision tree basically corresponds to a conjunction of Boolean expression of the form 'attribute = value' (for nominal attributes) or 'attribute ≤ value' (for numeric attributes), so a tree can be seen as a set of rules that say how to classify instances. The goal of tree induction is to find a subdivision that is fine enough to capture the structure in the underlying domain but does not fit random patterns in the training data.

The usual approach to the problem of finding the best number of splits is to first perform many splits (build a large tree) and afterwards use a 'pruning' scheme to undo some of these splits. Different pruning schemes have been proposed. For example, C4.5 uses a statistically motivated estimate for the true error given the error on the training data, while the CART (Breiman et al., 1984) method cross-validates a 'cost-complexity' parameter that assigns a penalty to large trees.

1.3 The Model

A logistic model tree basically consists of a standard decision tree structure with logistic regression functions at the leaves, much like a model tree is a regression tree with regression functions at the leaves. As in ordinary decision trees, a test on one of the attributes is associated with every inner node. For a nominal (enumerated) attribute with k values, the node has k child nodes, and instances are sorted down one of the k branches depending on their value of the attribute. For numeric attributes, the node has two child nodes and the test consists of comparing the attribute value to a threshold: an instance is sorted down the left branch if its value for that attribute is smaller than the threshold and sorted down the right branch otherwise. More formally, a logistic model tree consists of a tree structure that is made up of a set of inner or non-terminal nodes N and a set of leaves or terminal nodes T.

Figure 2 and Figure 3 depict the corresponding models. At the leaves of the logistic model tree, the functions $F1$, $F2$ determine the class membership probabilities by

$$Pr(G = 1|X = x) = \frac{e^{F_1(x)}}{e^{F_1(x)} + e^{F_2(x)}},$$

$$Pr(G = 2|X = x) = \frac{e^{F_2(x)}}{e^{F_1(x)} + e^{F_2(x)}}.$$

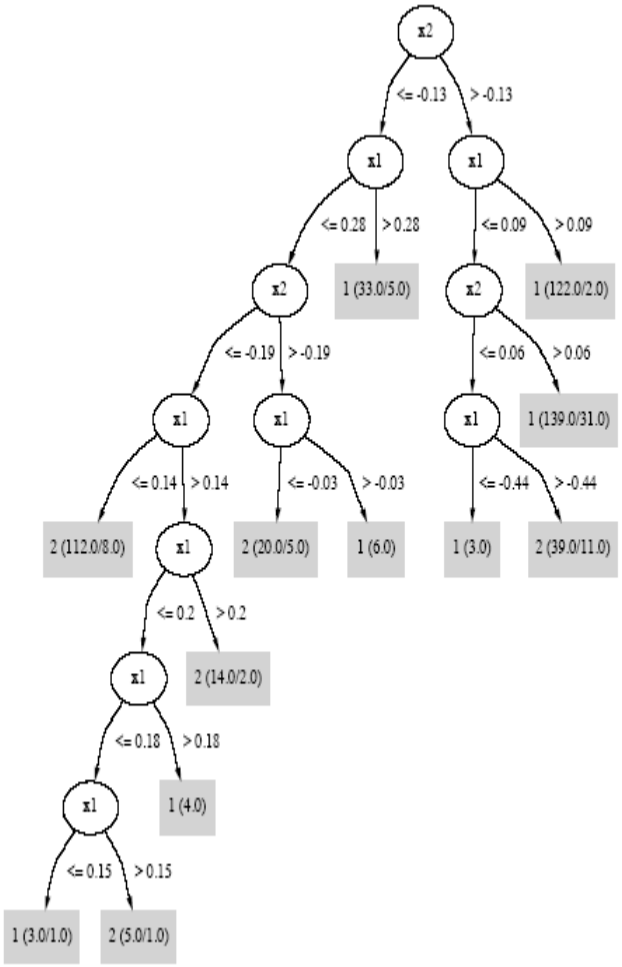

Figure 2. Decision tree constructed by the C4.5 algorithm for the 'polynomial-noise' dataset

The entire left subtree of the root of the 'original' C4.5 tree has been replaced in the logistic model tree by the linear model with

$$F_1(x) = -0.39 + 5.84 \cdot x_1 + 4.88 \cdot x_2$$
$$F_2(x) = \quad 0.39 - 5.84 \cdot x_1 - 4.88 \cdot x_2 = -F_1(x)$$

as shown in Figure 3.

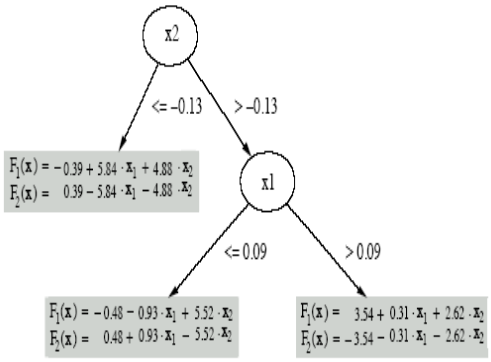

Figure 3. Logistic model tree constructed by the LMT algorithm for the 'polynomial- noise' dataset.

2 Basic LMT Algorithm

Here I listed first the basic "logistic model tree" algorithm by by Niels Landwehr, Mark Hall and Eibe Frank(2004) in details then I will describe "speeding up the logistic model tree" which is modified algorithm of logistic model tree. And also the problem in that later.

2.1 Tree Growing Process

There is a straightforward approach for growing logistic model trees that follows the way trees are built by M5'. This involves first building a standard classification tree, using, for example, the C4.5 algorithm, and afterwards building a logistic regression model at every node trained on the set of examples at that node. Instead, we chose a different approach for constructing the logistic regression functions, namely by incrementally refining logistic models already fit at higher levels in the tree. Assume we have split a node and want to build the logistic regression function at one of the child nodes. Since we have already fit a logistic regression at the parent node, it is reasonable to use it as a basis for fitting the logistic regression at the child.

2.2 Splitting and Stopping Criteria

They implemented two different criteria to select the attribute to split on. One is the C4.5 splitting criterion that tries to improve the purity of the class variable. The other splitting criterion attempts to divide the training data according to the current values of the working responses in the LogitBoost procedure in Figure 1.
But they could not detect any significant differences between either classification accuracy or tree size between the two methods. A disadvantage of splitting on the response is that the final tree structure is less intelligible. Hence they made splitting on the class variable (using the C4.5 splitting criterion) the default option in algorithm.

Tree growing stops for one of three reasons:

i. A node is not split if it contains less than 15 examples. This number is somewhat larger than for standard decision trees; however, the leaves in logistic model trees contain more complex models, which need more examples for reliable model fitting.

ii. A particular split is only considered if there are at least 2 subsets that contain 2 examples each. Furthermore, a split is only considered if it achieves a minimum information gain. This is a heuristic used by the C4.5 algorithm to avoid overly fragmented splits. When no such split exists, we stop growing the tree.

iii. A logistic model is only built at a node if it contains at least 5 examples, because we need 5 examples for the cross-validation to determine the best number of iterations for the LogitBoost algorithm. Note that this can lead to 'partially expanded' nodes, where for some branches no additional iterations of LogitBoost are performed and so the model at the child is identical to the model of the parent.

2.3 Pruning the Tree

As for standard decision trees, pruning is an essential part of the LMT algorithm. They spent a lot of time experimenting with different pruning schemes. Since their work was originally motivated by the model tree algorithm, they

first tried adapting the pruning scheme used by this algorithm. However, we could not find a way to compute reliable estimates for the expected error rate (resulting in an unstable pruning algorithm), hence we abandoned that approach. Instead, we employed the pruning method from the CART algorithm (Breiman et al., 1984).
Figure 4 gives the pseudocode for this algorithm, which call *LMT*.

```
LMT (examples)
{
        root = new Node ()
        alpha = getCARTAlpha (examples)
        root.buildTree (examples, null)
        root.CARTprune (alpha)
}
build Tree (examples, initialLinearModels)
{
        numIterations = CV_Iterations (examples, initialLinea Models)
        initLogitBoost (initialLinearModels)
        linearModels = copyOf (initialLinearModels)
        for i = 1...numIterations
                logitBoostIteration (linearModels, examples)
        split = findSplit (examples)
        localExamples = split.splitExamples (examples)
        sons = new Nodes[split.numSubsets ()]
        for s = 1...sons.length
            sons.buildTree (localExamples[s],nodeModels)
}
CV_Iterations (examples, initialLinearModels)
 {
    for fold = 1...5
          InitLogitBoost (initialLinearModels)
          //split into training/test set
          train = trainCV (fold)
          test = testCV (fold)
          linearModels = copyOf (initialLinearModels)
          for i = 1...200
                  logitBoostIteration (linearModels,train)
                  logErrors[i] += error(test)
    numIterations = findBestIteration (logErrors)
    Return numIterations
}
```

Figure 4 *Pseudocode* for the LMT algorithm.

It calls getCARTAlpha to cross-validate the 'cost-complexity parameter' for the CART pruning scheme implemented in CARTPrune. The method buildTree grows the logistic model tree by recursively splitting the instance space. The argument initialLinearModels contains the simple linear regression functions already fit by LogitBoost at higher levels of the tree. The method initLogitBoost initializes the probabilities/weights for the LogitBoost algorithm as if it had already fitted the regression functions initialLinearModels (resuming Logit- Boost at step 2.a in Figure 1). The method CV_Iterations determines the number of LogitBoost iterations to perform, and logitBoostIteration performs a single iteration of the LogitBoost algorithm (step 2 in figure 1), updating the probabilities/weights and adding a regression function to linearModels.

These ideas lead to the following algorithm for building logistic model trees:

i. Tree growing starts by building a logistic model at the root using the LogitBoost algorithm to iteratively fit simple linear regression functions, using fivefold cross validation to determine the appropriate number of iteration.
ii. A split for the data at the root is constructed. Splits are either binary (for numeric attributes) or multiway (for nominal ones), the splitting criterion will be discussed in more detail below. Tree growing continues by sorting the

appropriate subsets of data to the child nodes and building the logistic models at the child nodes in the following way: the LogitBoost algorithm is run on the subset associated with the child node, but starting with the committee $F_j(x)$, weights w_{ij} and probability estimates p_{ij} of the last iteration performed at the parent node (it is 'resumed' at step 2.a of Figure 1). Again, the optimum number of iterations to perform (the number of f_{jm} to add to F_j) is determined by a fivefold cross validation.

iii. Splitting of the child nodes continues in this fashion until some stopping criterion is met .Once the tree has been built it is pruned using CART-based pruning.

2.4 Issues in LMT

There are several issues that provide directions for future work. Probably the most important drawback of logistic model tree induction is the high computational complexity compared to simple tree induction. Although the asymptotic complexity of LMT is acceptable compared to other methods, the algorithm is quite slow in practice. Most of the time is spent fitting the logistic regression models at the nodes with the LogitBoost algorithm. It would be worthwhile looking for a faster way of fitting the logistic models that achieves the same performance. A further issue is the handling of missing values. At present, LMT uses a simple global imputation scheme for filling in missing values. a more sophisticated scheme for handling them might improve accuracy for domains where missing values occur frequently.

3 Study of Speeding up Logistic Model Tree Induction

Whatever issues are there in Logistic Model Tree (LMT) [1] are addressed by AIC criteria [2] instead of cross-validation to prevent overfitting these models. In addition, a weight trimming heuristic is used which produces a significant speedup.

3.1 Modification to LMT

3.1.1 Weight Trimming:

The idea of weight trimming in association with Logit- Boost is a very simple, yet effective method for reducing computation of boosted models. In our case, only training instances carrying $100 \cdot (1 - \beta)$ % of the total weight mass are used for building the simple linear regression model, where $\beta \in [0, 1]$. Typically $\beta \in [0.01, 0.1]$. We used $\beta = 0.1$. In later iterations more of the training instances become correctly classified with a higher confidence; hence, more of them receive a lower weight and the number of instances carrying $100 \cdot (1 - \beta)$ % of the weight becomes smaller.

3.1.2 Automatic Iteration Termination:

A common alternative to cross-validation for model selection is the use of an in-sample estimate of the generalization error, such as Akaike's Information Criterion (AIC) [3]. They investigated its usefulness in selecting the optimum number of LogitBoost iterations and found it to be a viable alternative to cross validation in terms of classification accuracy and far superior in training time.

AIC provides an estimate of the generalization error when a negative log likelihood loss function is used [3]. Let this function be denoted as *loglik* and let *N* be the number of training instances. Then AIC is defined as

$$AIC = -\frac{2}{N} loglik + 2\frac{d}{N}, \quad \text{.....(}$$

In eq (2) $d = i$, where i is the iteration number. As i increase, the first term in Equation decreases (because LogitBoost performs quasi-Newton steps approaching the maximum log-likelihood) and the second term (the penalty term) always increases. Empirically, this was found to be a good estimate. The optimal number of iterations is the i which minimizes AIC.

Determining the optimal number of iterations in this fashion will be called the *First AIC Minimum* (FAM). SimpleLogistic with FAM is compared with the original cross-validation-based approach. Table 1 shows the training time and classification accuracy for both algorithms on the 13 UCI datasets [2]. FAM consistently produced a significant speedup on all datasets.

Table 1 Training time and accuracy for SimpleLogistic using cross-validation and FAM

Dataset	Training Time		Accuracy	
	SimpleLog (CV)	SimpleLog (FAM)	SimpleLog (CV)	SimpleLog (FAM)
vowel	77.94	6.87	81.98	80.85
german-credit	7.97	0.59	75.37	75.34
segment	50.55	3.42	95.10	94.67
splice	253.96	77.48	95.86	95.87
Kr-vs-kp	57.28	6.69	97.06	96.38
sick	25.40	1.57	96.68	96.50
spambase	119.28	15.74	92.75	92.69

3.2 Issues in Speeding up Logistic Model Tree

Using AIC criteria instead of cross validation to prevent over fitting this model [2].So time training time is 55 times faster than the original LMT algorithm. But it does not significantly affecting classification accuracy. And they have check accuracy only for datasets of relatively low size and dimensionality.

If the AIC is not minimize up to 50 iteration then they have take a minimum iteration 50 as a default. So that number of iteration may not work for the large and high dimensional dataset and may be accuracy is decrease.

4 Conclusion

The Speeding up Logistic Model Tree algorithm, which is modification of Logistic Model Tree (LMT), is 10 to 25 time faster in training time. But it does not significantly affecting classification accuracy. These results were measured on datasets of relatively low size and dimensionality.

5 Future Extension

There are several issues that provide directions for future work. Instead of AIC, modified AIC [4] can be used to minimize the number of LogitBoost iteration. So algorithm can be speedier. At present, LMT uses a simple global imputation scheme for filling in missing values. A more sophisticated scheme for handling them might improve accuracy for domains where missing values occur frequently.

6 References

[1] Niels Landwehr, Mark Hall, and Eibe Frank. "*Logistic model trees"*. Machine Learning, 59(1/2):161–205, 2005.

[2] Marc Sumner, Eibe Frank and Mark Hall **"***Speeding up Logistic Model Tree Induction."*

[3] H. Akaike. "*Information theory and an extension of the maximum likelihood principle.*" In Second Int Symposium on Information Theory, pages 267–281, 1973.

[4] Fjo De Riddef', Rik Pintelon, Johan Schoukens and David Paul Gillikinb "*Modified AIC and MDL Model Selection Criteria for Short Data Records"* (IEEE)

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "*Additive logistic regression: a statistical view of boosting.*"

[6] Marcel Detting and Peter Buhlmann "*Boosting for tumor classification with gene expression data.*" September 5, 2002.

[7] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. AC-19, no. 6, pp. 716–723, Dec. 1974.

[8] Claudia Perlich, Foster Provost, Jeffrey S. Simonoff "*Tree Induction vs. Logistic Regression: A Learning-Curve Analysis*".

[9] Breiman, L., H. Friedman, J. A. Olshen, and C. J. Stone: "*Classification and Regression Trees.* "Wadsworth.1984