A Backward Adjusting Strategy and Optimization of the C4.5 Parameters to Improve C4.5's Performance

J. R. Beck[1], M. Garcia[2], M. Zhong[3], M. Georgiopoulos[4], G. C. Anagnostopoulos[5]

[1]School of EECS, University of Central Florida, Orlando, Fl, 32816; JasonRBeck@gmail.com
[2] ECE Department, University of Puerto Rico at Mayaguez, Mayaguez, PR, 00681; mariae_garciaayala@yahoo.com
[3]School of EECS, University of Central Florida, Orlando, GL 32816; myzhong@ucf.edu
[4]School of EECS, University of Central Florida, Orlando, FL; michaelg@mail.ucf.edu
[5]Department of ECE, Florida Institute of Technology, Melbourne, FL, 32901; georgio@fit.edu

Abstract

In machine learning, decision trees are employed extensively in solving classification problems. In order to design a decision tree classifier two main phases are employed. The first phase is to grow the tree using a set of data, called training data, quite often to its maximum size. The second phase is to prune the tree. The pruning phase produces a smaller tree with better generalization (smaller error on unseen data). One of the most popular decision tree classifiers introduced in the literature is the C4.5 decision tree classifier. In this paper, we introduce an additional phase, called *adjustment phase*, interjected between the growing and pruning phases of the C4.5 decision tree classifier. The intent of this adjustment phase is to reduce the C4.5 error rate by making adjustments to the non-optimal splits created in the growing phase of the C4.5 classifier, thus eventually improving generalization (accuracy of the tree on unseen data). In most of the simulations conducted with the C4.5 decision tree classifier, its parameters, *confidence factor, CF*, and *minimum number of split-off cases, MS*, are chosen to be equal 25% and 2, their default values, recommended by Quinlan, the inventor of C4.5. The overall value of this work is that it provides the C4.5 user with a quantitative and qualitative assessment of the benefits of the proposed adjust phase, as well as the benefits of optimizing the C4.5 parameters, *CF* and *MS*.

Introduction

Decision tree is a methodology used for classification and regression. Some of its advantages include the fact that it is easy to understand, and it can be used to predict labels of patterns with numerical as well as categorical attribute values, of which some might be missing. This work deals only with classification problems. A popular decision tree classifier, is the C4.5 decision tree classifier [1]. Typically, in order to design a decision tree classifier, two phases are employed: In the first phase, the tree has to be grown. In this phase a collection of training data is utilized to grow the tree by discovering (using some criterion of merit) a series of splits that divide the training data into progressively smaller subsets that have purer class labels than their predecessor data. In most instances, the growing of the tree stops when we end up with datasets that are pure (i.e., contain data from the same class). At the end of the growing phase, usually an over-trained tree has been produced. In the second phase, the objective is to prune the overgrown (over-trained) tree. Pruning produces less complex and more accurate (on unseen data) decision tree classifiers. Pruning removes parts of the tree that do not contribute to the tree's classification accuracy.

Quinlan, the inventor of C4.5, suggested default values for the C4.5 parameters, *CF* (confidence factor), and *MS* (minimum numbers of split-off cases), equal to 25% and 2, respectively. The *CF* parameter affects the confidence with which error rate at the tree nodes is estimated, with smaller values of *CF* resulting in more aggressive pruning of the tree nodes. The *MS* parameter affects the size of the grown tree by disallowing the formation of tree nodes whose number of cases is smaller than the chosen *MS* parameter; thus higher *MS* parameter values lead to grown trees that are of smaller size. A number of simulations with C4.5 are using the default parameter values for the *CF* and *MS*. In a recent paper ([2]) it was demonstrated that other than the default *CF* values might be more appropriate for designing a higher performing tree (i.e., a better accuracy or smaller size, or better accuracy and smaller size tree). In this work, we also demonstrate, through appropriate experimentation, that changing the default value of the *MS* parameter has the same beneficial effects on the performance of the decision tree classifier. Note that modifying the *MS* parameter to improve C4.5's performance has not been addressed before in the literature.

During the growing phase of a decision tree classifier, like C4.5, if minimum error rate splitting is employed to grow the tree there will be many "optimal" splits and it is unlikely that the true boundaries will be selected; even worse, in many cases none of these "optimal" splits finds the real boundary between the classes. Therefore, it is widely accepted, that another measure of merit, usually referred to as gain (decrease) of impurity, should be used to evaluate the splits during the growing phase. During the pruning phase the estimated error rate is used in order to decide if a branch must be pruned or not. The problem encountered, if a tree is simply grown and pruned, is that the estimated error rate

may not be optimized because the splits decided during the growing phase use as criterion of merit the maximal gain ratio and not the minimal error rate. For this reason, in this paper, we propose adding a third phase in the design of a C4.5 decision tree classifier, called the *backward adjustment phase* with the intent of further improving the performance of the classifier. In the backward adjustment phase the grown tree is used. A re-evaluation and adjustment of the possible splits is done bottom-up, minimizing the estimated error rate. Then, the pruning phase is employed. The tree that is grown and pruned is obviously referred to as C4.5, while the tree that is grown, adjusted, and then pruned is referred to as C4.5A.

In review, this paper presents experimental results that compare C4.5 and C4.5A for the default *MS* and *CF* parameter values, as well as for a variety of other choices of the *CF* and *MS* parameter values. Note that in our experiments we only modify one of the *CF* and *MS* parameter values, at a time, keeping the other parameter value at its default level. These experimental results lead us into a number of conclusions that provide the C4.5 user with a good level of understanding of whether, and how the C4.5 parameter values affect the performance of C4.5 and C4.5A, and furthermore, how C4.5 and C4.5A compare with each other for the same set of parameter settings.

The organization of the paper is as follows: In the *Literature Review* section we discuss pertinent prior work. In the *Backward Adjustment Algorithm* section, we justify the motivation for the backward adjust phase, and we explain in detail the backward adjustment algorithm. In the *Experiments* section, we discuss the type of experimentation conducted, the datasets used in this experimentation, and the results produced. In this section, the important observations from this study are delineated, as an immediate consequence of the results. Finally, in the *Summary and Conclusions* section we summarize our efforts and reiterate the important conclusions of our work.

Literature Review

A number of pruning algorithms have been proposed for decision tree classifiers, to reduce the tree's size. These are: the Minimal Cost-Complexity Pruning (CCP) in CART (Classification and Regression Trees; see [3], page 66), Error Based Pruning (EBP) in C4.5 (see [1], page 37), Minimum Error Pruning (MEP) [4, 5], Reduced Error Pruning (REP), Pessimistic Error Pruning (PEP) (see [6] for both REP and PEP), among others. Some of the pruning algorithms are briefly analyzed and empirically compared in [7].

In this paper we focus on three approaches to improve the performance of an induced tree that is produced by an existing, popular decision tree algorithm, the C4.5 algorithm. In addition to introducing, justifying, and implementing an adjust phase, to improve C4.5's performance, we argue for the usefulness of changing the default parameters (*MS* and *CF*) of the C4.5 algorithm. The important message that this paper delivers is that the adjust phase, and the modification of the C4.5 parameters allows for the creation of a population of trees that have different degrees of merit (accuracy on unseen data, and tree sizes). Furthermore, in the databases tested, there were many cases of non-default C4.5 parameter values that the produced C4.5 and C4.5A trees were not only of much smaller size, than the default C4.5 and C4.5A trees, but they achieved this feat without sacrificing the default tree's accuracy.

Backward Adjustment Algorithm

Motivation

During the growing process the splits that were chosen may not have been the error optimal splits. This happens because during the growing process the data is separated using the maximized gain ratio measure (see [1] for a definition of the maximized gain ratio). This split finds a good boundary and future splits may further separate these classes. For example, consider **Figure 1**, which shows two class distributions (Gaussian distributions) that overlap. When C4.5 makes the first split, the maximized gain ratio split is chosen to separate part of class 1 from class 2. Nevertheless, if there is no future split under this node or all future splits are pruned, it is clear that this split is not the Bayes optimal split (the one minimizing the probability of error; see **Figure 1** for an illustration of this fact).

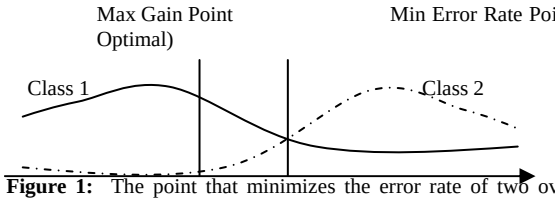

Figure 1: The point that minimizes the error rate of two overlapping Gaussian populations may not maximize the gain ratio.

Adjustment of a Tree

For the decision node pertaining to Figure 1 it is clear that it is beneficial to adjust the split. In general, after a decision node is adjusted, its ancestor nodes may also be improved by the same adjusting method. Therefore, this adjusting process attempts to improve the accuracy of the decision nodes in a bottom-up order. The algorithm for this backwards adjusting phase is shown in Figure 2. There are two options to adjust the tree: before the pruning phase or after the pruning phase. In this paper, we chose to adjust the tree before the pruning phase.

Strictly speaking, after one pass of the adjusting algorithm, the leaves may cover different training examples

because the splits in their ancestor nodes may have changed. We could repeat the same adjusting process again from the bottom decision nodes, based on the updated leaf coverage. Nevertheless, our experiments showed that a second run of the adjusting process does not offer significant improvements compared to the first pass, and therefore in our experiments we apply only one pass of the adjusting algorithm to the tree nodes.

```
algorithm AdjustTree (t, S)
   input: a tree rooted at node t, training set S
   output: the adjusted tree (modified from the input)

   if t is a decision node then
                    Divide S into subsets {S_i} using the current split;
                    foreach child node t_i of t do
                              Recursively call AdjustTree (t_i,S_i);
                    end
      foreach possible split s_k in t do
           Set the split of t to s_k;
           E_k = EstimateError (t,S);
      end
      Set the split of t to s_k, k = arg min E_k;
      RemoveEmptyNodes(t,S).
   end
```

Figure 2: The pseudo code for the backwards adjusting algorithm.

Experiments

In this section we describe the experimental design, the databases used for the experimentation, as well as the results of these experiments.

Experimental Design

The experimentation aims at comparing two measures of tree performance: (a) the accuracy of the tree on the test set, and (b) the size of the tree created. Quite often, these measures of merit are contradictory, in the sense that the highest accuracy tree is not necessarily the smallest size tree, and vice versa. Our experiments compare the tree produced through the growing and pruning phase of C4.5, simply referred to as C4.5, and the tree produced through the growing, adjust and pruning phase, referred to as C4.5A. Furthermore, our experiments allow modification of the default C4.5 parameters, the confidence factor, *CF* (default value of 25%) and the minimum numbers of split-off cases, *MS* (default value of 2). Note that modifying the *MS* parameter to improve C4.5's performance has not been addressed before in the literature. In our experiments the *CF* parameter assumes values of 75%, 50%, 25%, 10%, 5%, 1%, .1%, .01%, and .001% . Also, in our experiments, the *MS* parameter ranges from the default number of 2 and increases, at steps of 10, up to a number equal to 15% of the number of cases. In our experiments we only modified one of the *CF* and *MS* parameter values, at a time, keeping the other parameter value at its default level.

In order to perform our experiments each database is separated in two groups; a training set and a test set. To reduce the randomness in our reported tree accuracy and size results, the data are partitioned in twenty different ways in a training set and a test set. Then, the reported tree accuracies and sizes are the average accuracies and sizes. The partitioning of the data in training and test sets was accomplished as follows: We divided each database into 20 subsets of equal size. In the i^{th} (i = 0,1,2,…,19) partitioning, we use subsets i mod 20, (i + 1) mod 20,…, (i+ m -1) mod 20 for training and the others for testing, where m is predefined (for example, if m=1, the training ratio is 5%; if m=10 the training ratio is 50%). This approach guarantees that each example appears exactly m times in the training set and exactly (20 - m) times in the test set. In our experiments the training ratio was chosen equal to 50%.

```
algorithm EstimateError (t, S)
   input: a tree rooted at node t, training set S
   output: estimated error

        leafError = error if node was made a leaf using binomial
estimate;

   if t is a leaf node then
        return leafError;
        else
                Divide S into subsets {S_i} using the current split;
                foreach child node t_i of t do
                        sumError += EstimateError (t_i,S_i);
                end

     if sumError < leafError then
        return sumError;
     else
        return leafError;
   end
```

Figure 3:. The pseudo code for the estimate error function.

In our experiments, we used a combination of artificial and real datasets. The real datasets are available from the Univ. of California at Irvine's (UCI) machine learning repository [8], and they are: *Abalone* (4177 data-points), *Satellite* (6435 data-points), *Segment* (2310 data-points), and *Waveform* (5,000 data-points). The artificial datasets are Gaussianly generated data, of dimensionality 2, belonging to six different classes. The centers of the Gaussians are located at the edges of a hexagon, centered at (0, 0), with one of the edges located on the positive x-axis. The standard deviations of these Gaussians are chosen to be such that the Bayesian classifier's optimal performance on these datasets is 15% (this dataset is referred to as *g6c15*), and 25% (this

dataset is referred to as *g6c25*). The number of data-points generated for these artificial datasets was chosen equal to 5000.

Results and Observations

As emphasized above, we experimented with C4.5 and C4.5A, for many different *CF* and *MS* parameter values, and for six datasets. The volume of the results is significant and consequently they cannot all be presented here. In the following we provide a list of important observations from these experiments and a collection of appropriately selected tables and graphs that verify the validity of these observations.

Observation 1: The accuracy of C4.5A is always better than the accuracy of C4.5 for the same set of parameter values (*MS* and *CF*). The improvement in accuracy attained by C4.5A is database dependent and parameter dependent, and in most instances it is small. There are cases though, where improvement in accuracy for C4.5A can be as high as 6% (observed for the g6c15 dataset).

Observation 2: The size of the trees created by C4.5A and C4.5, for the same set of parameter values (*MS*, *CF*), is approximately the same.

Observation 3: Keeping the *MS* parameter at its default level (*MS* = 2), and modifying the *CF* parameter from its default value of 25% creates C4.5 and C4.5A decision trees that are quite often of much smaller size than the default C4.5 and C4.5A trees (*MS* = 2, and *CF* = 25%), without reducing the tree's accuracy. In the experiments conducted, decreasing the *CF* value decreased monotonically the size of the C4.5 and C4.5A produced trees.

Observation 4: Keeping the *CF* parameter at its default level (*CF*=25%), and modifying the *MS* parameter, from its default value of 2 creates C4.5 and C4.5A decision trees that are quite often of much smaller size than the default C4.5 and C4.5A trees (*MS*=2, and *CF*= 25%), without reducing the tree's accuracy. In the experiments conducted, increasing the *MS* value decreased monotonically the size of the C4.5 and C4.5A produced trees.

Database	Best CF	Error Rate	Num Nodes
g6c15	5%	17.3 (17.3)	44.8 (63.3)
g6c25	5%	26.5 (26.9)	47.5 (94.8)
Abalone	1%	36.7 (39.1)	52.6 (376.7)
Satellite	1%	14.2 (15.0)	161.6 (328.5)
Segment	25%	4.9 (4.9)	59.5 (59.5)
Waveform	1%	23.5 (24.5)	165.3 (323.6)

Table 1: Error Rates and sizes of C.4.5A for the best found *CF* value and the default *CF* value (the default *CF* error rates and sizes are in parentheses); *MS* =2.

Database	Best MS	Error Rate	Num Nodes
g6c15	42	18.4 (17.3)	18.8 (63.3)
g6c25	22	26.5 (26.9)	32.1 (94.8)
Abalone	32	36.5 (39.1)	29.7 (376.7)
Satellite	12	14.5 (15.0)	105.3 (328.5)
Segment	22	8.3 (4.9)	17.9 (59.5)
Waveform	22	23.6 (24.5)	54.6 (323.6)

Table 2: Error Rates and sizes of C.4.5A for the best found *MS* value and the default *MS* value (the default *MS* error rates and sizes are in parentheses); *CF*=25%.

Table 1 above, compares the error rate and size of C4.5A for the default *CF* and the best found *CF* value, while Table 2, above, does the same for the default *MS* and the best found *MS* value found.

Observation 5 (Overall Observation): From Tables 1 and 2, it is clear that non-default *CF* and *MS* values produce much smaller network sizes, while occasionally they improve the error rate as well. If one's objective is to create a population of C4.5 trees that have different accuracies and sizes, and quite often better accuracies and sizes, than the default C4.5 tree (i.e., the one using the default *MS*=2, and *CF*=2 parameter values), it is worth experimenting with different *MS* and *CF* values, as well as with the adjust version of C4.5 (C4.5A).

Figure 4 shows the performance of the C4.5 and C4.5A for the Abalone dataset as the *MS* parameter value changes (ranges from the default number of 2 and increases, at steps of 10, up to a number equal to 15% of the number of cases). Furthermore, Figure 5 shows the performance of C4.5 and C4.5A for the Abalone dataset as the *CF* parameter changes (75%, 50%, 25%, 10%, 5%, 1%, .1%, .01%), while *MS* is kept at its default setting of 2. Figure 4, verifies the validity of Observations 1, 2, 3, and 5. In particular, at the knee of both curves (C4.5 and C4.5A) we observe that by choosing an *MS* parameter value equal to 52, the size of the tree is reduced from approximately 400 nodes to approximately 25 nodes (**more than order of magnitude reduction in size**), while the accuracy of the tree on the test set is **improved by at least 2 percentage points.** Furthermore, Figure 5 verifies the validity of Observations 1, 2, 4, and 5. In particular, at the knee of both curves (C4.5 and C4.5A) we observe that by choosing the *CF* parameter value equal to 1% the size of the tree is reduced from approximately 400 nodes to approximately 50 nodes (**an order of magnitude reduction in size**), while the **accuracy of the tree is improved by approximately 2 percentage points**. In a similar fashion, Figures 6 and 7, corresponding to the satellite dataset, justify observations 1, 2, 3, 5, and 1, 2, 4, 5, respectively. Finally, Figure 8 (g6c15 results for different *MS* parameter values) also verifies Observations 1 and 5 (we see in Figure 8 an improvement in accuracy of the C4.5A tree compared to the accuracy of the C4.5 tree of approximately 6 percentage points).

Summary and Conclusions

In this work we have focused on C4.5, one of the classical and most often used decision tree classifiers. We introduced an innovative adjustment of tree splits strategy that led to a C4.5 algorithm modification, called C4.5A, involving a growing, adjust and pruning phases. We explained the motivation of this adjustment phase through an example and qualitative arguments. The quantitative justification of the adjustment phase was provided through experimentation with a number of datasets. In our experiments we also compared the performance (accuracy on unseen data and tree size) of C4.5 and C4.5A for various values of the C4.5 parameters (*MS*, and *CF*), beyond their default values of *CF*=25% and *MS*=2. The major observations are: (a) C4.5A improves C4.5's accuracy for all parameter values, slightly in most cases, but significantly in a few cases, (b) Using *MS* parameter values different than the default value (=2) produces C4.5 and C4.5A trees that are of smaller size (quite often much smaller size), while at the same time improves the tree's accuracy; for the databases tested, results showed that *MS* values equal to a small factor times 10 produce good trees, (c) Using *CF* parameter values different than the default value (=25%) produces C4.5 and C4.5A trees that are of smaller size (quite often much smaller size), while at the same time improves the tree's accuracy; for the databases tested, results showed that *CF* values around 5% produce good trees, (d) The C4.5 adjustment phase, and the C4.5 parameter modification allows the user to obtain decision trees with different merits (accuracy and size), and quite often of higher merits (accuracy and size) than the default C4.5.

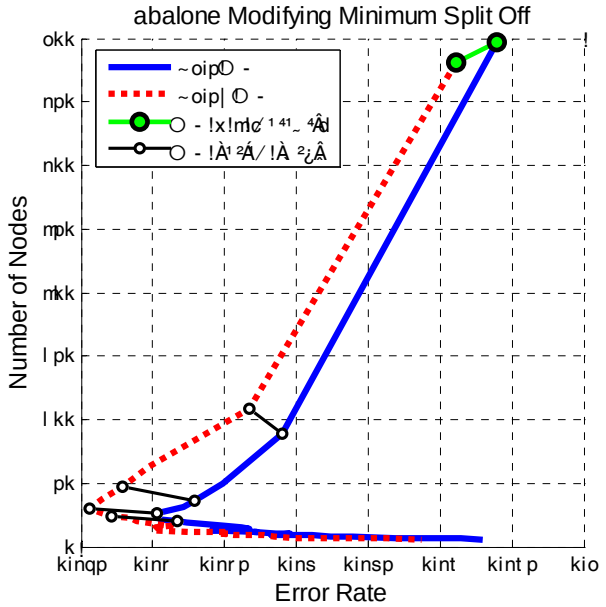

Figure 4: Number of tree nodes versus tree error rate for the abalone dataset, when modifying the minimum number of cases split-off (*MS*) during C4.5 growing. When the value of *MS* increases, the number of nodes in the tree drops significantly along with the error rate. At the knees of the curves, corresponding to an *MS* parameter value of 52, the number of tree nodes produced is significantly reduced (from around 400, when the default *MS* parameter is used, to less than 50, when *MS* equals 52), while the error rate is reduced by at least two percentage points.

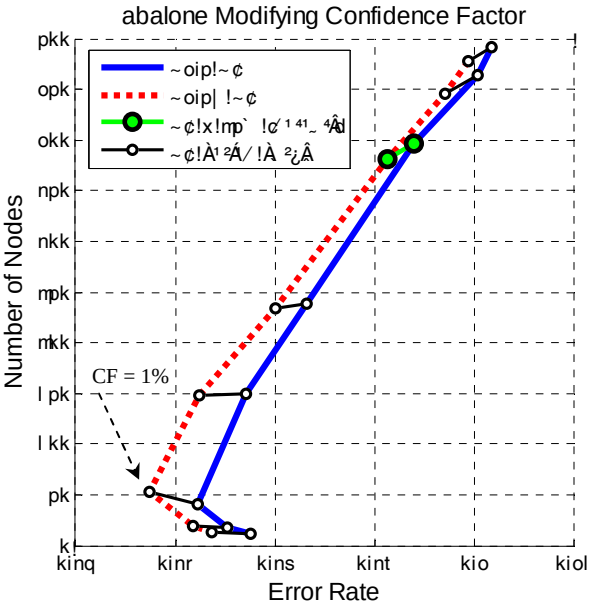

Figure 5: Number of tree nodes versus tree error rate, when modifying the confidence factor (*CF*) during C4.5 pruning for the abalone dataset. When the value of *CF* decreases, the number of nodes in the tree drops significantly along with the error rate. At the knees of the curves, corresponding to the *CF* parameter value of 1%, we see a significant reduction of tree sizes (from around 400, when the default *CF* parameter is used, to around 50, when *CF*=1% is used), while the error rate is also decreased by at least 2 percentage points.

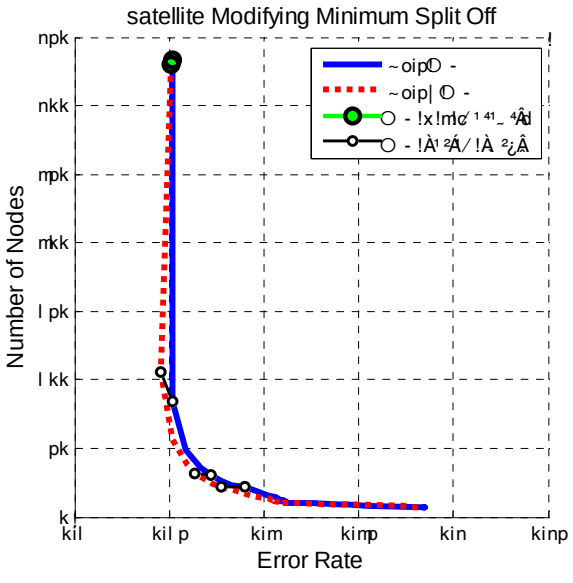

Figure 6: Number of tree nodes versus tree error rate for the satellite dataset, when modifying the minimum number of cases split-off (*MS*) during C4.5 growing. The difference in error is not easily seen because of

the complete range of *MS* plotted for. What is evident in this graph is the knee in the curve at approximately 100 nodes. The significant reduction in tree size from the default value is caused by increasing *MS* from 2 to 12. A size reduction of approximately 225 nodes is achieved with no apparent penalties in error rate.

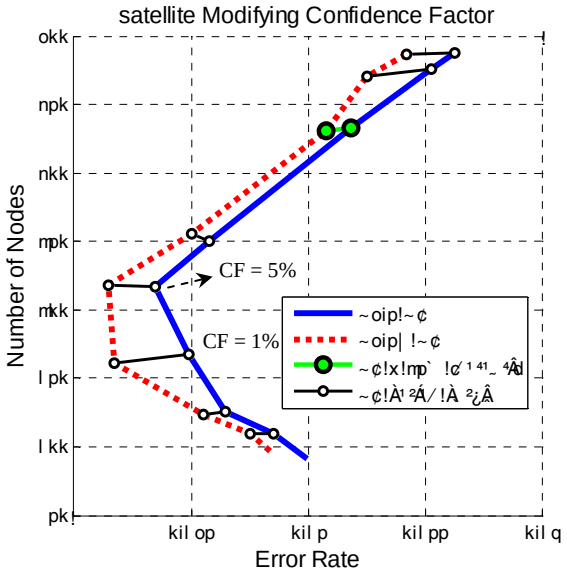

Figure 7: Number of tree nodes versus tree error rate, when modifying the confidence factor (*CF*) during C4.5 pruning for the satellite dataset. Modifying the confidence factor from the default of 25% to 5% produces a tree on the blue (solid) curve (shown at the knee of the curve) with approximately 225 nodes. On the red (dotted curve) curve the knee is for a *CF* value of 1% and has approximately 150 nodes at the knee. Interestingly, *CF* values of 10%, 5%, 1%, .1% and .01% all provide error rates and tree sizes less that the ones observed for the default parameter value.

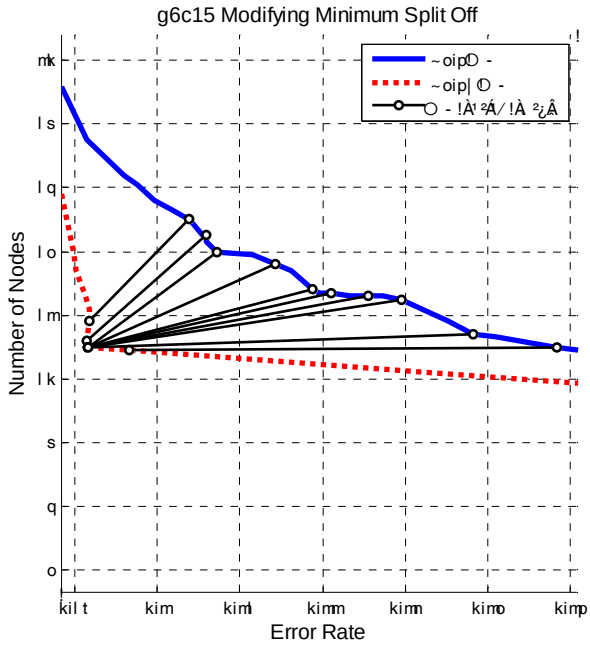

Figure 8: Number of tree nodes versus tree error rate for the g6c15 dataset (only a portion of the produced curves is shown in this figure to accentuate the point that C4.5A improves accuracy for some datasets and C4.5 parameter values). In the Figure, the C4.5A_MS points (red dotted curve) are highly concentrated at the knee of the C4.5A MS plot. On the contrary, their corresponding C4.5_MS points on the C4.5 MS plot (blue solid curve) show a consistent increase in error rate. It is along this section of the plot where the greatest differences in error rate between C4.5_MS and C4.5A_MS are seen.

Acknowledgment

This was work was supported by NSF grants: 0341601, 0647018, 0717674, 0717680, 0647120, 0525429, 0203446.

References

[1] Quinlan, J. R., 1993. *Programs for Machine Learning*. Morgan Kaufmann.

[2] Hall, L. O., Bowyer, K. W., Banfield, R. E., Collins, R. 2003. *International Journal of Artificial Intelligence Tools*, 12(3): 249-264.

[3] Breiman, L., Friedman, J.H., Olshen, R. A., Stone, C. J. 1984. *Classification ad Regression Trees*. Wadsworth.

[4] Cestnik, B., 1991. On estimating probabilities in tree pruning. In Proceedings of the European Working session in Machine Learning (EWSML-91). New York, NY.

[5] Niblett, T., Bratko, I. 1986. Learning decision rules in noisy domains. In Proceedings of the Expert Systems 1986, the 6th Annual Technical Conference on Research and Development on Expert Systems III, 25-34. Brighton, United Kingdom.

[6] Quinlan, J. R. 1999. Simplifying Decision Trees. *International Journal of Human-Computing Studies*, 51:497-510.

[7] Esposito, F., Malerba, D., Semeraro, G. 1997. A Comparative Analysis of Methods for Pruning Decision Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19: 476-491.

[8] Newman, D. J., Hettich, S., Blake, C. L., Merz, C. J., 1998. UCI Repository of Machine Learning Databases, University of California, Irvine, CA