Project 3: Decision Trees for Classification and Regression

Sarah Wilson swi1s117@jhu.edu
303-921-7225
Engineering Professionals Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA

1. Introduction

Decision trees are tree structures than can be built by machine learning algorithms by training on a data set. A decision tree is a hierarchical structure, consisting of nodes, which can have left or right nodes (children). The root node is the start of the tree and has left and right nodes that would go on to build sub-trees. A node that does not have a left or right child is considered a leaf. Decision trees are built or grown from the training data by deciding which features to build sub-trees on and knowing when to split the data to produce the next node, along with knowing when a leaf node has been encountered. Decision trees can be used in both classification and regression machine learning problems. Decision trees ask "questions" at each node and determine, based on feature observations from the data set, if the left or right node should be taken when traversing the tree. When the leaf node is encountered that will end with the answer for the classification or regression problem.

The problem statement presented in this paper is to understand and implement two major categories of decision tree algorithms. For classification problems the approach explored will be the Iterative Dichotomiser 3 (ID3) algorithm. For regression problems the approach explored will be the Classification and Regression Tree (CART) algorithm. Both ID3 and CART will first be built as standard uni-variate trees, meaning that they will be grown to completion with out intervention. The results of these trees will then be compared with trees that have been edited, for regression problems this will include the process of early stopping, for classification problems this will include the process of reduced error pruning. The uni-variate, early stopping and error pruning trees will be then compared in terms of Classification Error and Mean Square Error for classification and regression problems respectively. k-fold cross validation will be the experimental method used in this report.

The hypothesis of this report is that for the both the classification and regression tasks the trees that are error pruned or early stopped will perform better than trees that are grown uni-variate to completion. The hypothesis assumes that trees grown to full completion will exhibit over fitting to the data. Over-fitting is where in a learning system the model has been so tightly fit to the training data, that results on new un-trained data are highly inaccurate. The hypothesis assumes that in allowing the trees to grow to full completion of nodes that additional patterns may be learned in the training data that might not apply to new data feed into the algorithm. The results from experiments ran on the

provided data sets will be discussed and presented against the outlined hypothesis.

Section 1 has provided the introduction, problem statement and hypothesis in regards to two major decision tree algorithms that will be explored.. Section 2 will provide an in-depth explanation of the ID3 and CART algorithms, how these algorithms will be modified during the early stopping and error pruning processes and discuss each of the 6 data sets used. Section 3 will present the results obtained by variations of ID3 and CART as uni-variate, early stopping and error pruning implementations on the provided data sets. Section 4 will discuss the results that were obtained and compare them to the hypothesis that was outlined in the introduction. This report will conclude in Section 5 with a discussion of lessons learned and areas of possible future work.

2. Algorithms and Experimental Methods

The ID3 algorithm will be used on classification data sets. The ID3 algorithm is a classification algorithm that follows a greedy heurisitic top down approach by selecting the attribute/feature that provides the maximum information gain at each partition of the data set. The key parameters to understand in the ID3 algorithm are Entropy(I), Expected Entropy, Information Gain and Gain Ratio.

Entropy is a measure of uncertainty there is in a data set. An Entropy of 0 means that the data set is pure, meaning it comprised of only one class. An Entropy of 1 or more indicates that there is lots of uncertainty in the data set meaning there is a high level of mixture of multiple different classes contained in the data set.
Expected Entropy is the Entropy only in feature/attribute f_i in the data set.
Information Gain is the Entropy minus the Expected Entropy. The Information Gain provides a metric of how much uncertainty there would be in the data if we removed feature f_i from the equation. What I mean by removing f_i from the equation is that, from this point on in calculating or making decisions about the data in the data set, we would hold feature f_i fixed. As in we would only look from this point on at observations that had a certain value of f_i when making future calculations about Entropy and Expected Entropy. From a building a tree perspective this is logical as we only want to continue down branches that will tell us the most about the data, so the goal is to maximize the entropy that we would have at the next node of the decision tree.

The issue presented by building a decision tree based on information gain alone, is that is we are always looking for the highest gain when making splits in the data set to build new partitions, this can lead to numerous expensive partitions that would force the algorithm to run longer potentially with no real benefit. Instead of splitting on Information Gain alone, the concept of Gain Ratio is introduced. The gain ratio provides a balance of the highest gain while also adding a penalization factor for the splits that would result in a higher number of partitions being built.

Consider a 2 class classification problem we can define the following equations for Entropy, Expected Entropy, Information Gain and Gain Ratio.

Entropy

$$I(p,n) = -\frac{p}{(p+n)} * log(\frac{p}{p+n}) - \frac{n}{p+n} * log(\frac{n}{p+n}) \tag{1}$$

I= Entropy of Partition

π= Current Partition in the Overall Data Set

p_π= Number of Class 1 Observations in the Current Partition

n_π= Number of Class 2 Observations in the Current Partition

Expected Entropy

$$E_\pi(f_i) = \sum_{j=1}^{m_i}(\frac{p_\pi^j + n_\pi^j}{p_\pi + n_\pi} * I(p_\pi^j, n_\pi^j)) \tag{2}$$

E_π= Expected Entropy of Partition

f_i= Feature i of the Data Set

m_i= Size of the Domain of Feature i

π= Current Partition in the Overall Data Set

I= Entropy of Partition

p_π^j= Number of Class 1 Observations in the Current Partition for the jth item in the Domain of Feature f_i

n_π^j= Number of Class 2 Observations in the Current Partition for the jth item in the Domain of Feature f_i

Gain

$$G_\pi(f_i) = I(p_\pi, n_\pi) - E_\pi(f_i) \tag{3}$$

G_π= Gain of Partition

E_π= Expected Entropy of Partition

I= Entropy of Partition

p_π= Number of Class 1 Observations in the Partition

n_π= Number of Class 2 Observations in the Partition

Information Value

$$IV_\pi(f_i) = -\sum_{j=1}^{m_i}(\frac{p_\pi^j + n_\pi^j}{p_\pi + n_\pi} * log\frac{p_\pi^j + n_\pi^j}{p_\pi + n_\pi}) \tag{4}$$

IV_π= Information Value of Partition

m_i= Size of the Domain of Feature i

p_π= Number of Class 1 Observations in the Partition

n_π= Number of Class 2 Observations in the Partition

Gain Ratio

$$GR_\pi(f_i) = \frac{G_\pi(f_i)}{IV_\pi(f_i)} \tag{5}$$

GR_π= Gain Ratio of Partition

G_π= Gain of Partition

IV_π= Information Value of Partition

Note that these equations can be expanded to a **k** Class Classification problems as needed.

The CART algorithm will be used on regression data sets. The CART algorithm is very similar to the ID3 algorithm, with the exception that it used on regression machine learning problems and instead of using the Gain Ratio to determine the best split it instead using the Mean Squared Error. Splits are chosen in CART, such that the split offers the lowest Mean Squared Error on predicted values.

The following steps outline the algorithms. The algorithm uses recursion to build the tree to completion.

Generate Tree
If ID3: Calculate Node Entropy
If CART: Calculate Node MSE
If Node Entropy / MSE is less than ϵ
    ID3: Create a leaf labeled by the majority class in the partition
    CART: Create a leaf labeled with the mean of the partition
    Return
Split Features in Partition
Determine Best Split On Feature in the Partition
    ID3: Set Node as the Feature with the Highest Gain Ratio
    CART: Set Node as the Feature with the lowest Split Error
For each branch based on splits:
Generate Tree (Recursive Call to Self)

Split Features in Partition
If ID3:
    If Feature is Numeric:
        Split partition into two parts based on the middle value
        Determine split with Maximum Gain Ratio
    If Feature is Categorical::
        Split partition in n parts based on the number of discrete options in the features domain
        Determine split with Maximum Gain Ratio
Return Highest Gain Ratio Item

If CART:
Split partition into two parts based on the middle value of the Current Feature
Evaluate the Squared Error of the Left Part to the Classification Reg. Value
Evaluate the Square Error of the Right Part to the Classification Reg. Value
Sum these errors together and divided by the number of observations in the partition
This value becomes your split error
Return feature with the minimum split error

Data Sets

The following data sets were used during the classification and regression tasks for this project.

Breast Cancer

Description:
Task: Classification
Predictor: Diagnosis (Malignant or Benign)
Link:
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

Car Evaluation

Description:
Task: Classification
Predictor: Car Evaluation (Unacceptable, Acceptable, Good, Very Good)
Link:
https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Congressional Vote

Description: 1984 United Stated Congressional Voting Records
Task: Classification
Predictor: Party (Republican / Democrat)
Link:
https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

Albalone

Description: Physical measurements of Albalone
Task: Regression
Predictor: Rings (int)
Link:
https://archive.ics.uci.edu/ml/datasets/Abalone

Computer Hardware

Description: Relative CPU performance data.
Task: Regression
Predictor: PRP
Link:
https://archive.ics.uci.edu/ml/datasets/Computer+Hardware

Forest Fires

Description: Forest Fire burn area data
Task: Regression
Predictor: Area (float)
Link:
https://archive.ics.uci.edu/ml/datasets/Forest+Fires

3. Results

Tables 1-3 display the results from Classification Tasks: the Breast Cancer, Car Evaluation and Congressional Vote data sets while running a full uni-variate tree and reduced error pruning. The number of nodes to prune was determined using the tuning process that was discussed earlier. These tables show the results from the train set and the test set during each fold of the k-fold validation process. The results of each iteration of the tuning process were omitted from this report for brevity. The error used is Classification error and can be described as the Number of Times the Prediction was Wrong / Total Number of Comparisons. This error was averaged across the 5 folds to provide the Average Classification Error against each data set.
Tables 4-6 display the results from the Regression Tasks: the Albalone, Computer Hardware and Forest Fire data sets while running a full uni-variate tree and early stopping. The value ϵ used in the early stopping was determined using the tuning process for regression that was discussed earlier. These tables show the result from the train set and the test set during each fold of the k-fold validation process.The results of the tuning process were omitted from this report for brevity.The error used is Mean Square Error.

Table 1: Car Evaluation: ID3 - Experimental Results

Car Eval: Univariate Tree	
k-fold[1]	NA
k-fold[2]	NA
k-fold[3]	NA
k-fold[4]	NA
k-fold[5]	NA
Average Classification Error Across Folds	

Car Eval: Prunned	
k-fold[1]	NA
k-fold[2]	NA
k-fold[3]	NA
k-fold[4]	NA
k-fold[5]	NA
Average Classification Error Across Folds	

Table 2: Breast Cancer: ID3 - Experimental Results

Breast Cancer: Univariate Tree	
k-fold[1]	NA
k-fold[2]	NA
k-fold[3]	NA
k-fold[4]	NA
k-fold[5]	NA
Average Classification Error Across Folds	

Breast Cancer: Prunned	
k-fold[1]	NA
k-fold[2]	NA
k-fold[3]	NA
k-fold[4]	NA
k-fold[5]	NA
Average Classification Error Across Folds	

Table 3: Congressional Vote: ID3 - Experimental Results

Congressional Vote: Univariate Tree	
k-fold[1]	0.0428
k-fold[2]	0.0571
k-fold[3]	0.0285
k-fold[4]	0.0860
k-fold[5]	0.1010
Average Classification Error Across Folds	0.0631

Congressional Vote: Prunned	
k-fold[1]	NA
k-fold[2]	NA
k-fold[3]	NA
k-fold[4]	NA
k-fold[5]	NA
Average Classification Error Across Folds	

Table 4: Albalone: CART - Experimental Results

Albalone: Univariate Tree	
k-fold[1]	19.6697
k-fold[2]	7.6263
k-fold[3]	13.6392
k-fold[4]	11.4805
k-fold[5]	8.8353
Average MSE Across Folds	12.2502

Albalone: Early Stopped Tree	
	Epsilon = 10
k-fold[1]	20.7040
k-fold[2]	6.5700
k-fold[3]	11.0600
k-fold[4]	6.9710
k-fold[5]	5.9230
Average MSE Across Folds	10.2456

Table 5: Computer Hardware: CART - Experimental Results

Computer Hardware: Univariate Tree	
k-fold[1]	750202.5200
k-fold[2]	544719.5000
k-fold[3]	260921.3636
k-fold[4]	899078.9697
k-fold[5]	786715.6364
Average MSE Across Folds	648327.5979

Computer Hardware: Early Stopped Tree	
	Epsilon = 10
k-fold[1]	750202.5290
k-fold[2]	544719.5000
k-fold[3]	260921.3636
k-fold[4]	899078.9697
k-fold[5]	786715.6364
Average MSE Across Folds	648327.5997

Table 6: Forest Fires: CART - Experimental Results

Forest Fire: Univariate Tree	
k-fold[1]	565.4062
k-fold[2]	606.1958
k-fold[3]	16434.9870
k-fold[4]	729.3448
k-fold[5]	1146.9990
Average MSE Across Folds	3896.5866

Forest Fire: Early Stopped Tree	
	Epsilon = 10
k-fold[1]	564.9116
k-fold[2]	605.5460
k-fold[3]	16428.8700
k-fold[4]	709.6340
k-fold[5]	1151.8463
Average MSE Across Folds	3892.1616

4. Discussion

The hypothesis that was presented at the start of this report for the both the classification and regression tasks the trees that are error pruned or early stopped will perform better than trees that are grown uni-variate to completion.

Looking at the classification data sets, the only data set that was able to run to full completion was the Congressional Vote data set. The Car Evaluation and Breast Cancer data sets, were unable to be fully built, this was most likley due to an implementation error as they never returned from the recursive calls that were used to build the trees. Too much time was spent trying to debug these data sets, such that Prunning was also unable to be implemented in time for submission. This is unfortunate, as no meaningful comparisons can be made towards the support of the hypothesis presented in the Introduction section. However, it does afford some lessons learned that will be applied to next implementation attempts of such algorithms.

Looking at the Regression data sets, the values of ϵ that were evaluated were 0, 0.01, 0.1, 1, 10. During the tuning process it was determined that 10 was the optimal value to use for ϵ and it was observed that there was not large difference between the results obtained for ϵ values such as 0, 0.01, 0.1, 1. One reason for this might that due to range of possible values for the prediction value in the regression data sets, a large ϵ makes a larger impact as it allows for the account of outliers in the data. The Albalone data set was the only one that had measurable improvement, with an Average MSE dropping from 12.5 to 10.2 between the uni-variate tree and the early stopped tree. The Forest Fire and Computer Hardware data sets, did not see measurable improvements between the uni-variate tree and the early stopped tree. The Albalone data set did not include multiple features that needed to be one hot encoded, Sex was the only attributes and it only provided 3 options to one hot encode. The Forest Fire and Computer Hardware data sets on the other hand had multiple attributes that were one hot encoded, and each of those attributes had multiple options that were derived because of this one hot encoding. It is possible that one hot encoding this data was the wrong approach, and alternative methods should have been used, such as dropping those attributes completely. Since one hot encoding offers only 0 or 1 for the options the splits that will be generated when performing the regression might be highly skewed towards on particular branch as the data is classified through the regression tree.

5. Conclusion

Overall there seemed to be some trends in the data that supported the hypothesis, that both the classification and regression tasks the trees that are error pruned or early stopped will perform better than trees that are grown uni-variate to completion. However, it was noted that implementation issues did not allow for the classification data sets to have many meaningful trees built, so no conclusions can be drawn toward the hypothesis for classification. The lesson learned that can be applied is that for recursive problems it is important to have debugging methodology in place that cane be used to determine quickly at what level of the recursion were issue encountered.

The regression data sets were able to be fully grown and implement early stopping. It was noticed that early stopping in some cases did tend to improve the MSE that was obtained fold to fold, but there were other details such as one hot encoding that might have impacted the results for other data sets, that had more data that needed one hot encoding to be applied to.

A suggested area of future work, after fixing basic implementation details, would be to consider other methods of prunning or early stopping. For Classification and Regression machine learning problems, other such methods include Cost-Complexity Pruning.

6. References

1. Alpaydin, E. (2004). Introduction to machine learning (Oip). Mit Press.