605.649 Programming Project 3: Decision Trees

Christopher El-Khouri

1. Introduction

As we progress further into the field of machine learning, we are bound to arrive at one of the most commonly used models, Decision Trees. Decision Trees are a type of supervised learning algorithms that can be used for classification and regression.

Classification Trees are decision trees that are used for classification purposes, these trees have a target variable that takes discrete values. The branches are observations about the attributes and the leaves are the different class labels that our data could hold. Regression Trees are decision trees used for regression, similar to classification trees, the branches are observations, but the target variable takes continuous values.

An example of a classification tree is shown below (Colton, 2006):

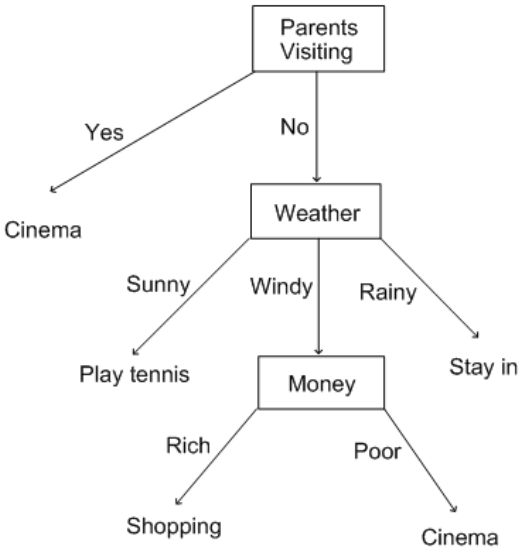

Figure 1: Classification Tree Example

In the figure above, the branches are based on the following attributes: Parents Visiting, Weather, and Money. The leaves are based on the following class labels: Cinema, Play Tennis, Stay in, and Shopping. The decision tree is used by starting from the top, "Parents Visiting" in this case, and traverses until we arrive at a leaf.

In this project, we will be assessing the performance of two Decision Tree algorithms: Iterative Dichotomiser 3 (ID3) and Classification and Regression Tree (CART). ID3 is going to be implemented for classification, and CART is going be implemented for regression. Furthermore, we are going to assess the effects of post-pruning an ID3 tree and early stopping a CART.

ID3 is an algorithm created by J.R Quinlan (1986) typically used for classification. The algorithm begins with the original dataset as the root node. With each, it iterates through

every unused attribute of the dataset and calculates the information gain of that attribute. The algorithm proceeds to select the attribute which results in the largest calculated value of Information Gain. The dataset is then split by the selected attribute to produce subsets of the data and proceeds recursively on each subset, only considering the attributes not previously selected.

CART is an algorithm credited to Breiman et al. (1983) and can be used for either classification or regression. In our project, the splitting criterion used for this algorithm will be the mean-squared error (MSE). Therefore, it will work similar to ID3 in terms of the recursion of attributes, however it will aim to split based on the lowest resulting MSE.

We will be comparing the performance of the two different algorithms on 6 different datasets tackling both classification and regression as well assessing the performance of post-pruning the ID3 trees and early stopping the CART in an effort to reduce overfitting. We hypothesize that the accuracy of the outputs produced by the ID3 algorithms with post-pruning will be superior to the accuracy of the outputs produced without post-pruning. In addition, we hypothesize that the error of the results produced by CART with early stopping will be approximately the same as the results without it with a lower tree size.

The accuracy of our classification problems will be calculated using the classification error, which is the percentage of incorrectly classified points. The accuracy of our regression problems will be calculated using the Mean Squared Error (MSE).

In the next section, we will describe the technical and theoretical basis of our algorithms, in section 3 we will list the datasets involved in this experiment, section 4 will present and compare the results, and in section 5 we will discuss the significance of our experiment and compare the performance with the other algorithms previously encountered in the course.

2. Algorithms and Experimental Methods

ID3 Algorithm

The ID3 Algorithm used in this project is designed on the following basis:

ID3Tree(**S**, **root**) where S is a dataset of features X and target variable (class) y,
root is the root of the tree (initially **None**):
Calculate the Entropy **I** of the dataset S
For each feature X_i:
Calculate the Expected Entropy **E** of X_i
Calculate Information Gain **gain** of X_i
Calculate the Information Value **IV** of X_i
Calculate the Gain Ratio **GR** of X_i
The feature X_i with the highest GR ($\mathbf{X_R}$)is selected as a branch
if root = None:
root = X_R
Else:
insert X_R into root child
For each unique attribute m in X_R: ID3Tree $\left((S - X_{R,m}), \text{root}\right)$
Do recursively until $\mathbf{I} = 0$ or $X = \Phi$

The Entropy **I** of a dataset S with target variable y having k classes c is calculated as follows:

$$I(c_1, \dots, c_k) = -\sum_{l=1}^{k} \frac{c_l}{c_1 + \cdots + c_k} \log_2 \frac{c_l}{c_1 + \cdots + c_k}$$

The Expected Entropy **E** of a feature X_i in S is calculated as follows:

$$E(X_i) = \sum_{j=1}^{m_i} \frac{c_{\pi,1}^j + \cdots + c_{\pi,k}^j}{c_{\pi,1} + \cdots + c_{\pi,k}} I\left(c_{\pi,1}^j, \dots, c_{\pi,k}^j\right)$$

Where:

$c_{\pi,l}^j$: The number of examples in the j^{th} partition from feature X_l with class c_π

m_i: The number of discrete values (partitions) a feature X_i contains

The Information Gain **gain** of a feature X_i in S is calculated as follows:

$$gain_\pi(X_i) = I - E(X_i)$$

The Information Value **IV** of a feature X_i in S is calculated as follows:

$$IV(X_i) = -\sum_{j=1}^{m_i} \frac{c_1^j + \cdots + c_k^j}{c_1 + \cdots + c_k} \log_2 \frac{c_1^j + \cdots + c_k^j}{c_1 + \cdots + c_k}$$

The Gain Ratio **GR** of a feature X_i in S is calculated as follows:

$$GR(X_i) = \frac{gain(X_i)}{IV(X_i)}$$

The algorithm above considers a splitting criterion based on Gain Ratio rather than the typical splitting criterion of Information Gain. Information Gain could be problematic due to the fact that it tends to have a strong bias towards selecting feature attributes with a higher quantity of values. This bias could lead to the creation of more tree partitions and therefore a higher chance that we may incur overfitting. Gain Ratio considers the effect of partitioning and penalizes features that create a larger number of partitions.

The ID3 algorithm is trained by constructing the tree based on a training set. The recursive algorithm shown above is run on the training set until all of the features are considered or until **I** gives a value of 0. An **I** value of 0 indicates that there is only 1 class within a dataset and therefore the tree node corresponding to that would be a leaf representing that class. Once all of the features are considered, a leaf representing the most common occurring class within the resulting dataset will be inserted into the tree. In the case that all the features are considered and there are an equal number of classes remaining in the dataset, the tree does not partition based on the feature.

After the tree has been constructed using the training dataset, the performance is assessed using the training data. The tree is initially constructed without any pruning methods and the non-pruned performance is compared with the post-pruning performance.

Post-pruning is carried out after the tree has been constructed to reduce generalization error and avoid overfitting. This is accomplished by measuring the classification error of the tree after removing vertices from the bottom up to ensure we don't omit any relevant subtrees. In our project, post-pruning is performed by extracting a validation dataset from our original dataset consisting of 10% of the data. The performance of the non-pruned tree is compared to the performance of the pruned tree as we prune the different vertices. The tree that results in the lowest classification error of the validation set is returned as the final pruned tree.

CART Algorithm

The CART algorithm used in this project is designed as follows:

CART(**S**, **root**) where S is a dataset of features X and target variable y (r^t),
root is the root of the tree (initially **None**):
For each feature X_m in S:
Calculate $\mathbf{g_{mj}}$ of X_m
Calculate **MSE** of X_m
The feature X_m with the lowest MSE ($\mathbf{X_R}$)is selected as a branch
if root = None:
root = X_R
Else:
insert X_R into root child
For each unique attribute j in X_R: CART $\left((S - X_{R,j}), \text{root}\right)$
Do recursively until **MSE** = 0 or X = Φ or last remaining X_m has only 1 j

The value $\mathbf{g_{mj}}$ of X_m in S is the mean or median of the values of the target variable y after splitting S on $X_{m,j}$.

The **MSE** of X_m is calculated as follows:

$$MSE_{X_m} = \frac{1}{N_m} \sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}(x^t)$$

Where:

$$b_{mj}(x) = \begin{cases} 1 \; if \; x \in X_{mj} : x \; reaches \; node \; m \; and \; takes \; branch \; j \\ 0 \; otherwise \end{cases}$$

$$N_m : Number \; of \; rows \; in \; S \; prior \; to \; splitting \; by \; X_m$$

$$r^t : Value \; of \; y \; on \; t^{th} \; row$$

The algorithm above considers a splitting criterion based on minimizing the mean squared error and is trained by constructing the tree based on a training set. The recursive algorithm is run on the training set until all of the features are considered or until **MSE** returns a value of 0 which indicates a single value for y, or in the case where the last remaining X_m only has a single j. Once all of the features are considered, a leaf representing the mean of the y-values (r's) or the median is returned depending on the nature of the dataset.

In an effort to reduce overfitting our tree, early stopping can be implemented where we cap the MSE value that results in a split. This requires a small modification to the algorithm above where a split only occurs if the minimum MSE is greater than a certain value. In our project, early stopping is performed by extracting a validation dataset from our original dataset consisting of 10% of the data. Early stopping thresholds of 0.01σ up until σ in increments of 0.01σ were tested on the validation set. The performance of the non-early stopped tree is compared to the performance of the early stopped tree. The tree that results in the lowest MSE of the validation set is returned as the final early stopped tree.

5-fold cross-validation

Our algorithms were fitted and validated using 5-fold cross-validation. After extracting 10% of our dataset for validation purposes, the remaining 90% of the data was used in the 5-fold cross-validation process. The data is split into 5 parts (or folds) and the algorithms are trained on each of the folds accordingly. The performance of the classification algorithms was determined by calculating the classification errors of each of the folds by dividing the number of incorrect predictions over the total size of the folds, then calculating the mean of the errors. The performance of the regression algorithms was determined by calculating the MSE of each of the folds, then calculating the mean of the MSE's.

3. Datasets

The following datasets were used in this project:

1. Breast Cancer
2. Car Evaluation
3. Image Segmentation
4. Abalone
5. Computer Hardware
6. Forest Fires

Dataset 1: Breast Cancer (Classification)

This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The dataset contains 699 points and 2 classes: Benign (denoted by '2') and Malignant (denoted by '4'). Including the class variable 'Class', the data contains a total of 11 features.

There are 16 missing values that need pre-processing and are all under the feature 'Bare Nuclei'. We will fill those missing values by separating the data into the 2 different classes and producing sampled data based on the median of the 'Bare Nuclei' feature of the 2 classes. The median was chosen due to the distribution of 'Bare Nuclei' data shown below:

Table 1: Statistical Description of Bare Nuclei Data by Class

Class=2	
	bare_nuclei
count	444
mean	1.34
std	1.18
min	1
25%	1
50%	1
75%	1
max	10

Class=4	
	bare_nuclei
count	239
mean	7.62
std	3.12
min	1
25%	5
50%	10
75%	10
max	10

Based on the table above, class 2's distribution shows that at least 75% of the data have a value of 1, class 4's distribution shows that at least 50% of the data have a value of 10. The mean values are 1.34 and 7.62 respectively, sampling the data based on the mean values will not give accurate simulations due to the values of 'Bare Nuclei' being round figures and due to the fact that the data is clearly skewed.

The feature attributes were discretized by calculating the median of each of the features within each of the classes. The averages of the medians were then taken as the midpoints and features were generated based on whether each feature column was less than those values or not. Values of "Yes" and "No" were returned and utilized for the construction of our ID3 tree. Below is a sample of the pre-processed data frame used in the algorithm:

Table 2: Dataset 1 Preprocessed

class	clump_thickness <8.5	uniformity_cell_size <4.5	uniformity_cell_shape <4.5
2	Yes	Yes	Yes
2	Yes	Yes	Yes
2	Yes	Yes	Yes
2	Yes	No	No
2	Yes	Yes	Yes
4	Yes	No	No

Dataset 2: Car Evaluation (Classification)

The data is on evaluations of car acceptability based on price, comfort, and technical specifications. The dataset contains 1728 instances, 4 classes, and 7 columns. The attribute values are discrete therefore no discretization was necessary.

Dataset 3: Image Segmentation (Classification)

The instances were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel. The dataset contains 210 instances, 7 classes, and 20 columns. The column "Region-Pixel-Count" was removed during pre-processing since all the rows have a value of "9". Similar to Dataset 1, the features were discretized by calculating the median values of each of the features with each of the classes. Since there are 7 classes, this would result in many features which

would construct a large tree and would consume a great deal of processing time. Therefore, the midpoints of each of the respective features medians were calculated 3 times over in order to result in a maximum of 2 generated discretized features per column.

Dataset 4: Abalone (Regression)
Predicting the age of abalone from physical measurements. The dataset contains 4177 instances and 9 columns. The feature values were discretized by sorting each of the features and taking the median, new features were generated based on whether the respective feature is less than its median resulting in values of "Yes" and "No". In addition, the target variable "Rings" was normalized in order for us to be able to compare the resulting error of our model to our previously developed models.

Dataset 5: Computer Hardware (Regression)
This dataset consists of relative CPU Performance Data. The estimated relative performance values were estimated by the authors using a linear regression method. The results are indicated in the data set as "ERP." The dataset contains 209 instances and 10 columns. Similar to Dataset 4, the features were discretized in the same manner, by taking the median and returning values of "Yes" and "No" whether or not the feature attribute was less than the median. The target variable was normalized in order for us to be able to compare the resulting error of our model to our previously developed models.

Dataset 6: Forest Fires (Regression)
This is a difficult regression task, where the aim is to predict the burned area of forest fires, in the northeast region of Portugal, by using meteorological and other data. The dataset contains 517 instances and 13 columns. The columns "month" and "day" were dropped during pre-processing as it was assumed that the meteorological data would be more productive. Keeping "month" and "day" in our dataset would result in an unnecessarily large tree due to the number of discrete values that those columns could take. The feature values discretized in the same manner as Datasets 4 and 5, and the target variable was normalized. Furthermore, the median was taken instead of the mean during the modeling of the CART algorithm on this dataset as it achieved superior results.

4. Results

Dataset 1: Breast Cancer (Classification)
Modeling the Breast Cancer dataset with our ID3 algorithms we get the following:

Table 3: Classification results for Dataset 1

Dataset 1: Breast Cancer						
Training Fold	Unpruned			Pruned		
	Tree Size	Classification Error		Tree Size	Classification Error	
		Test	Validation		Test	Validation
1	59	4.76	0	27	4.76	0
2	55	4.76	1.43	3	11.11	1.43
3	51	5.6	1.43	3	16.8	1.43
4	67	4.76	1.43	3	12.7	1.43
5	57	5.56	4.29	7	5.56	0
Average	**58**	**5.09**	**1.72**	**9**	**10.19**	**0.86**

Dataset 2: Car Evaluation (Classification)
Modeling the Car Evaluation dataset with our ID3 algorithms we get the following:

Table 4: Classification results for Dataset 2

Dataset 2: Car Evaluation						
Training Fold	Unpruned			Pruned		
	Tree Size	Classification Error		Tree Size	Classification Error	
		Test	Validation		Test	Validation
1	303	7.72	7.51	265	8.04	7.51
2	308	6.11	10.4	282	7.4	10.4
3	310	10.61	9.25	276	10.93	9.25
4	304	6.11	9.83	279	7.07	9.83
5	286	8.36	8.67	246	9.65	8.67
Average	**302**	**7.78**	**9.13**	**270**	**8.62**	**9.13**

Dataset 3: Image Segmentation (Classification)
Modeling the Image Segmentation dataset with our ID3 algorithms we get the following:

Table 5: Classification results for Dataset 3

Dataset 3: Image Segmentation						
Training Fold	Unpruned			Pruned		
	Tree Size	Classification Error		Tree Size	Classification Error	
		Test	Validation		Test	Validation
1	63	7.89	9.52	43	10.53	4.76
2	59	18.42	9.52	51	21.05	9.52
3	57	24.32	23.81	33	21.62	9.52
4	71	10.53	14.29	55	13.16	9.52
5	61	15.79	9.52	53	15.79	9.52
Average	**62**	**15.39**	**13.33**	**47**	**16.43**	**8.57**

Dataset 4: Abalone (Regression)
Modeling the Abalone dataset with our CART algorithms we get the following:

Table 6: Regression results for Dataset 4

Dataset 4: Abalone						
Training Fold	No Early Stopping			Early Stopping		
	Tree Size	MSE		Tree Size	MSE	
		Test	Validation		Test	Validation
1	185	0.73	0.69	165	0.73	0.69
2	187	0.71	0.71	153	0.71	0.70
3	201	0.72	0.71	153	0.72	0.69
4	191	0.74	0.70	159	0.74	0.69
5	191	0.61	0.69	119	0.61	0.69
Average	**191**	**0.70**	**0.70**	**150**	**0.70**	**0.69**

Dataset 5: Computer Hardware (Regression)
Modeling the Computer Hardware dataset with our CART algorithms we get the following:

Table 7: Regression results for Dataset 5

Dataset 5: Computer Hardware						
Training Fold	No Early Stopping			Early Stopping		
	Tree Size	MSE		Tree Size	MSE	
		Test	Validation		Test	Validation
1	71	1.12	0.45	71	1.12	0.45
2	69	1.22	0.46	69	1.22	0.46
3	75	0.54	0.46	39	0.54	0.46
4	69	0.3	0.48	27	0.3	0.48
5	69	0.33	0.48	31	0.33	0.48
Average	**71**	**0.70**	**0.47**	**47**	**0.70**	**0.47**

Dataset 6: Forest Fires (Regression)
Modeling the Forest Fires dataset with our CART algorithms we get the following:

Table 8: Regression results for Dataset 6

Dataset 6: Forest Fires						
Training Fold	No Early Stopping			Early Stopping		
	Tree Size	MSE		Tree Size	MSE	
		Test	Validation		Test	Validation
1	267	0.22	0.11	267	0.22	0.11
2	241	0.29	0.11	43	0.58	0.06
3	215	5.17	0.20	151	5.08	0.14
4	255	0.5	0.1	255	0.5	0.1
5	231	0.2	0.12	41	0.52	0.12
Average	**242**	**1.28**	**0.13**	**151**	**1.38**	**0.11**

5. Discussion

Looking at our classification results above on tables 2-5, our hypothesis does not appear to stand with unpruned ID3 trees outperforming the pruned trees in all of the datasets. In Dataset 1, the average tree size decreased from 58 down to 9, the average error within the training sets doubled but the error within the validation sets halved. In Dataset 2,the average tree size slightly decreased, the average test set error slightly increased and the average validation set error remained the same. In Dataset 3, the tree size decreased (~24%), the test set error increased slightly and the validation set error significantly decreased.

As for the regression results, it seems that in Tables 6 and 7 that the average test error stayed the same while decreasing the average tree size. I would consider this a confirmation of the hypothesis, the early stopped decision trees are as accurate with a smaller tree size. Table 8 is a separate case, we still haven't been able to successfully regress the Forest Fires Dataset, as mentioned in the K-NN project report, it seems that a more thorough EDA could possibly uncover a successful regression method.

6. Conclusion

In conclusion, it seems that our hypothesis regarding post-pruning of the ID3 tree was not confirmed in this experiment. However, I would not rush to completely void it, I believe some fine tuning on the post-pruning method where it does not consider only the validation set may allow us to find the right balance to generate the "perfect" ID3 classification tree. As for the CART with early stopping, besides the Forest Fires Dataset, I would consider our hypothesis confirmed.

Comparing the results of our K-NN type classification algorithms previously done on Dataset 3, both our pruned and unpruned ID3 trees performed better than the C-NN and E-NN models, but the K-NN model performed better. As for the regression models, the Abalone Dataset performed about the same, with CART performing slightly worse than K-NN but slightly better than E-NN. All the K-NN type algorithms significantly outperformed our CART algorithms in the Computer Hardware Dataset. The Forest Fires Dataset had around the same performance.

7. References

Simon Colton.
Lecture 6: Decision Tree Learning.
Artificial Intelligence in Computational Art and Design, School of Interactive Arts and Technology.
Simon Fraser University, British Columbia, Canada, 2006.

John R. Quinlan.
Induction of decision trees.
Machine Learning, 1:81–106, 1986.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen.
Classification and Regression Trees
Wadsworth Publishing Co Inc, Belmont, California, 1983.