ID3 Decision trees with and without Reduced Error Pruning (CS605.449 – Project 4)

By Max Robinson

Abstract

In this experiment the performance of the ID3 algorithm for building decision tree models on three different data sets is analyzed and compared to ID3 built models that have undergone Reduced Error Pruning. The models are built and tested on three data sets, Abalone, Car Evaluation, and Image Segmentation. The results of the experiment showed that the all models performed best on the Car Evaluation and Image Segmentation data sets, while performing poorly but better than random on the Abalone data set. When comparing pruned and unpruned models, the unpruned models performed better equally or better across the board. However, the pruned models were able to reduce the size of the model by more than 50% in most cases with within an increase of error rate of around 3% or less.

Problem

The problem being investigated in this experiment is how well the ID3 decision tree building algorithm performs on three different data sets when compared to a decision tree built using ID3 and pruned using Reduced Error Pruning. The three data sets used are the Abalone, Car Evaluation, and Image Segmentation data sets from UC Irvine Machine Learning Repository.

For this experiment, I hypothesize that the ID3 algorithm will build models that perform well on all three data sets. I expect ID3 to perform best on the Car Evaluation data set, and perform the worst on the Abalone dataset. The Car Evaluation data set has the smallest number of both features and classes with all features being categorical features. This should provide the decision tree with fewer total options and make it easier to spit the tree to a point where the majority class for the decisions made is very close to the true distribution. The Ablone dataset on the other hand has eight features and twenty-nine different possible classes. This will make it difficult for the decision tree to get to a point where the majority label for the data is correct most of the time.

When comparing the pruned decision tree to the performance of the unpruned trees, I hypothesize that the pruned trees for the Car Evaluation and Image segmentation will do better than the unpruned trees for either data set. For the Abalone data set, I expect the unpruned tree to outperform the pruned decision tree.

The Car Evaluation data set, and Image Segmentation data sets will split the decision trees well but I believe will be prone to overfitting. The pruning process I believe will help generalize these models and help them to perform better on the test sets. For the Abalone data set, the number of possible classes is very large and as a result I believe that the pruning of the tree will miss important decisions that separate classes that might be very similar.

Algorithm Implementation

ID3

Iterative Dichotomiser 3 (ID3) is an algorithm for building decision trees, originally developed by J.R. Quinlan[1]. When build a decision tree, the goal is to build a tree that correctly classifies the training data, can generalize to perform well on the test data, and is the simple model. ID3 has been found to generally create simple decision trees, but is not guaranteed to produce the simplest decision tree possible [1].

ID3 works by constructing a tree, where each node in the tree is a feature in the data points and the data is then separated into subtrees based on the possible feature values or domain for that feature in the training set. Once an attribute has been used to partition the data, the attribute is no longer used in any subtrees to partition the data.

Once there are either no more features the split the data on, or the data that has been split is homogeneous, subtree creates the class label that will result from that portioning of the data. The class label that is used for a leaf node is the majority class label for the training data that is partitione d into that subtree.

An important aspect of ID3 is the criteria with which an attribute is selected to partition the data. This criterion specifies why one attribute is selected to spit the data over another attribute. This criterion helps to create simple tree verses a complex tree [1]. ID3 originally used a "seat-of-the-pants evaluation function" but later adopted an information based approach that was suggested by Peter Gacs which is based on information gain [1].

This implementation of ID3 uses the information gain to describe the amount of information that is gained by partitioning the data based on a given attribute of the data. The attribute that is selected is attribute that provides the largest information gain. This being the case, this turns ID3 into a greedy search for attributes that provide the largest information gain on a given partition of data with a given set of available attributes.

The information is calculated based on Shannon's entropy and can be described as:

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

Where c_l is a possible class for the a given partition of data.

The expected entropy is defined as follows:

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

Where $c_{pi,k}^{j}$ is the number of examples in the jth subpartition from feature f_i with class c_k.

The information gain is then:

$$gain_{pi}(f_i) = I\left(c_{pi,1}, \dots, c_{pi,k}\right) - E_{pi}(f_i)$$

What this gain equation is describing is the entropy information gained for a given feature f_i for some subset of data is equal to the actual information of all class $c_{pi,k}$ where $c_{pi,k}$ is the number of examples in the subset of data that are of class c_k minus what the expected information is for that feature for that data partition.

For this implementation of ID3, the features being handled could be either categorical, or numerical. Categorical attributes where handled by first storing the possible domain values for an attribute at the very beginning of the learning process, and then using these possible domain values to partition the data. This corresponds to the jth partitioning of data for the expected entropy equation.

For numerical data, the data was portioned in half based on the mean value for a given feature in a given subset of data. In other words, given a sub-partition of data j, and a numerical feature f_1, the partitioning of the data was created by finding the mean for f_1 of the data points in j. This provided a binary split of the data of either less than or equal to the mean, or greater than the mean.

To use a model, a data point is given to the model. Based on the feature values in the data point, the model is traversed accordingly. For instance, if data point $x = [red, car, who]$, and the root node of the tree corresponds to attribute number 0, red, then the tree would be traversed to the subtree that corresponds to attribute 0 being red, as opposed to say blue or gree n. The data point values are used to traverse the tree until a leaf node is hit which contains the predicted class that the data point belongs too.

For numerical features, a simple less than or greater than test is performed against the mean value that was used at that point in the subtree to partition the data. The mean value that was used to partition the training data is stored with the model so that classification can be done without the test data.

Reduced Error Pruning

Reduced Error Pruning (REP) is a post model creation pruning algorithm to reduce the complexity of the decision tree model. Pruning is meant to reduce the complexity of a tree that has grown in complexity while not increasing in accuracy [2]. Since "REP was never introduced algorithmically by Quinlan"[2], and there is not a consensus on if the algorithm is bottom-up or iterative [2], this experiment has implemented a bottom up approach to REP.

REP works by iterating through the non-leaf nodes, and replacing that node with the most likely class for that node. This new tree is then tested against a validation set to see if performance has improved. In this experiment, the performance will be the error rate of the tree. If the pruning has improved the tree, keep the new pruned tree and continue to prune the tree until. If performance is not better or equal revert the tree back to before the node was pruned.

For this implementation, REP starts by finding all of the parents of the current leaf nodes. These nodes are then used as the starting nodes to prune for REP. These nodes are then pruned or not pruned, based on the error rate of the resulting tree. Once all nodes have been evaluated, the next set of parents of leaf nodes are found, and the process is repeated. This continues until no nodes are pruned from the tree.

This implementation has the effect of starting at the bottom of the tree, and pruning nodes up the subtrees. This is consistent with others interpretations of REP using a bottom-up manner, like the interpretation described in Elomaa & Kaariainen's paper, which agrees with other studies with the interpretation as follows [2]:

> Nodes are pruned in a single bottom-up sweep through the decision tree, pruning each node is considered as it is encountered. The nodes are process in postorder. [2]

The only difference is in the implementation for this experiment, nodes are found through repeated searches to the parent of leaves, and nodes can be considered multiple times. This is an inefficiency in this implementation.

Scoring

Error Rate

Error rate is used in this experiment to calculate how many misclassifications occurred when compared to the total number of data points classified. Error rate can be described as

$$errorRate = \frac{\#\ of\ misclassified\ test\ instances}{total\ number\ of\ test\ instances}$$

Number of Nodes per Tree

While not being used as a deciding factor on performance, the number of nodes in a tree should be considered due to the possible constraints on time or hardware that might be needed to use the model. While it is not expected the be an issue for this experiment, it is worth considering the total number of nodes in an unpruned tree as opposed to a pruned tree, and how many nodes can be pruned. This can provide some benefits for constrained systems.

Experimental Approach

The approach for this experiment was twofold. The first part was done through data preprocessing, and the second part was done through how the experiments were run. The primary metric used for evaluation was Error Rate. The number of nodes needed in each model was also calculated and used to compare against the number of nodes in pruned models.

Data Pre-processing

For this experiment there was little data pre-processing that needed to be done. Both the Abalone and Car Evaluation data sets where used as-is.

The segmentation data set was originally two different data sets, a training and test set. For this experiment, the two data sets were merged together into one data set so that cross validation could be done on the entire data set. In addition, all class labels were moved to be the last feature in the feature vector.

Cross Validation and Validation Set

For this experiment, cross validation was used to provide robust and averaged results across different distributions of the data. Five-fold cross validation was used for this experiment. This means that from the entire set of the data, five partitions are created. For each run of k-NN the five folds are rotated through, were four of the folds are merged to create training set of 80% of the data, and one fold is used as the test set, 20%. The partition that is used for the test set is rotated so that every partition is used as the test set once.

The data used for each fold in the cross validation are also stratified for classification problems. This means that the distribution of the data for each fold mirrors that of the distribution of the data in the entire training set. If class A makes up 40% of the data in the entire data set, then class A will make up 40% of the data in each fold for cross validation.

For this experiment, a validation set was also used. Prior to creating the fold for cross validation, a validation set was created, using 10% of the total amount of data in the data set. This validation set was created using stratification as well. This allowed the validation set to also provide a sample of data that matched the true distribution of the data. The validation set remained constant throughout the rotations of the cross validation folds in order to provide consistent pruning for each of the trees pruned.

The 10% validation set was removed from the overall data set regardless of if the experiment was usi ng pruning or not. This allowed each experiment to be run using the same amount of data so that the results could be compared based on the amount of data used to train each model.

ID3 and Pruning

For the final experiment, using cross validation, each data set was used to build models and evaluate the performance of the models twice. First cross validation was used to create a decision tree models based only on the ID3 algorithm and evaluate the performance of the models using the error rates of the models.

Next cross validation was used again to construct models based on the ID3 algorithm. These models were then pruned using Reduced Error Pruning. The pruned model was then evaluated using error rates of the models.

For both pruned models and unpruned models, a new model was created for each fold that was used as a new test set. Since 5-fold cross validation was used, five models were created for each experiment. The evaluation metrics were then averaged across the folds, and the standard deviation between the folds was also calculated.

Results

After the running the experiment as described in the above sections, the error rate for each data set for pruned trees and unpruned trees can be viewed in Table 1. The error rate for the unpruned tree and pruned trees are showed in adjacent rows to each other to compare the results. These results show error rate

Table 1 Average Error rate of ID3 Tree, unpruned and pruned with standard deviation and node counts

Data Set	Pruned	Average Error Rate	SD	Average Node Count Unpruned	Average Node Count Pruned	Average Difference in Node Count
Abalone	False	0.77704	0.00327	703.2	N/A	N/A
	True	0.77401	0.01666	704.6	269.2	435.4
Car Evaluation	False	0.07636	0.01526	313.6	N/A	N/A
	True	0.10255	0.01064	317.0	133.2	183.8

Image Segmentation	False	0.08297	0.00858	337.0	N/A	N/A
	True	0.09835	0.00893	301.8	155.4	146.4

Behavior

The error rates of the decision trees were mostly as expected, but a little more extreme in places.

For the performance of each of the trees individually on the data sets, the Abalone data set was the hardest for the models and all models performed the worst on this data set. Both The unpruned tree had an error rate of 0.77704 with a standard deviation of 0.00327. This means that 60% of the time the error rate will be between .77377 and 0.78031. this is roughly between 77-78% error rate. This error rate was far worse than expected.

When comparing against a random guess, however, this performance is significantly better. There is a total of twenty-nine classes in the data set. If one were to be selected randomly with a uniform distribution of classes, there is a 3.45% chance of randomly guessing the correct class. In other words, a random guess with uniform distribution has a 96.55% error rate. The 77-78% error rate of the model is much better than a random uniform guess. This is still not as good as expected but statistically better than random.

Performance of the models for the Car Evaluation and Image Segmentation were close to expectations. Both pruned and unpruned models for both data sets performing with-in 1 standard deviation of the other corresponding model for the other data set. The Car Evaluation data set was expected to perform better than the Image Segmentation data set, but the difference in performance between them was statistically insignificant.

When comparing pruned versus unpruned models for each data set, the models behaved much different than expected.

For models built for the Abalone data set, the pruned versus unpruned data set performed almost the exact same. The error rates for both were within one standard deviation, using either experiments standard deviation. The interesting thing to note however, is that while the error rate for the pruned model did not decrease, the number of nodes needed to produce the same error rate were much fewer. The pruned model produced the same error rate with on average 435.5 fewer nodes. That is a 61.9% smaller model for no decrease in performance.

Pruned models for the Car Evaluation did worse than the unpruned models. The error rate for the pruned models was about 10.2% error rate, while the unpruned was about 7.6% error rate. For the pruned model, the error rate for the unpruned model is about 2.46 standard deviations away. For the unpruned model, the error rate of the pruned model is about 1.72 standard deviations away. This shows that while it is possible for the two models to perform the same, it is likely that more than 68.5% of the time the unpruned model will perform better than the unpruned model.

Pruned models for the Image Segmentation data set also did worse than the unpruned models. The pruned model had an error rate of about 9.84% while the unpruned model had about an 8.3% error rate. For the pruned model the error rate of the unpruned model is about 1.722 standard deviations away. For the unpruned model the error rate of the pruned model is about 1.793 standard deviations away. This means that again, while it is possible for the models to perform similarly on their respective data sets, it is likely that more than 68.5% of the time unpruned model will perform better.

When comparing the number of nodes needed for the models though, the pruned models used m any fewer nodes than the unpruned models. For Car Evaluation the pruned model had almost 58% fewer nodes than the unpruned model, with about a 2.6% increase in error rate. For Image Segmentation, the pruned model had about 48.5% fewer nodes than the unprun ed model, with about a 1.54% increase in error rate.

Overall for the pruned models, the reduction in the size of the models was very large. If the model was required to be small, and an increase of error rate of 3% or so was acceptable, the pruned models performed well with largely reduced model sizes.

Summary

Overall, the algorithms performed a little outside of expectations when considering accuracy on each of the datasets, but unexpectedly when comparing pruned to unpruned models. The Car Evaluation da ta set performed much better than the Abalone data set as expected, but performed about the same as the Image Segmentation data. When comparing the pruned and unpruned performance, the hypothesis was completely wrong. The pruned models for the Abalone data set performed equally as well as the unpruned models, while the pruned models for the Car Evaluation and Image Segmentation data sets performed significantly worse than the pruned models for each. When considering the size of the models however the pruned models were close to half the size of all of the unpruned models while staying below an additional increase of 3% to the error rate.

References

[1] Quinlan, J R. "Induction of Decision Trees." Machine Learning, vol. 1, no. 1, 1 Mar. 1986, pp. 81–106., doi: 10.1007/BF00116251.
[2] Elomaa, Tapio, and Matti Kääriäinen. "An Analysis of Reduced Error Pruning." Journal of Artificial Intelligence Research, vol. 15, Sept. 2001, pp. 163–187.

Data Sources

Abalone — https://archive.ics.uci.edu/ml/datasets/Abalone
Car Evaluation — https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
Image Segmentation — https://archive.ics.uci.edu/ml/datasets/Image+Segmentation