JMLR: Workshop and Conference Proceedings 1:1–4, 2016 Submitted 10/23/2016; Published 10/23/2016

Introduction to Machine Learning - EN 605.449.81 - Lab 4

Max Henry MHENRY22@JHU.EDU

Abstract

In this lab, I implemented two decision tree algorithms, ID3 and CART for regression. These implementations were tuned and tested on three classification data sets and three regression data sets. Experiments show no difference in the performance of pruned and full classification trees, perhaps due to too small of a validation set. Regression tree testing was limited by the variance measure used during the decision tree inducing algorithm. As a result, hyper parameters were tuned on only a few values and their cross-validation error shows this poor performance.

Keywords: ID3 classification trees, regression trees, reduced-error pruning, early stopping

1. Problem Statement

Given three classification data sets and three regression data sets, we were tasked with implementing the ID3 decision tree inducing algorithm and the CART decision tree inducing algorithm. A separate validation set was used to run reduced-error pruning on the ID3 tree from the classification data sets and 5-fold cross-validation was used to tune the early-stopping threshold in the case of the regression tree. I hypothesize that the reduced-error pruning and early stopping trees with perform within 10% of their unpruned counterpart.

2. Decision Tree Induction

Decision trees are used to classify data by learning split attributes at each node of the tree. Interior nodes contain information on the attribute chosen to split the data and the possible values (and therefore branches) that the attribute can take. To classify a particular instance, one starts at the root of the node and compares the instance's split attribute value. Based on this value, a decision is made to follow one branch from this node. This continues until a leaf node (a node with no branches) is encountered. Instead of storing branches, leaf nodes store one class label (or function approximation in the case of regression), and this value is returned as the prediction of the data instance.

Training a decision tree according to the ID3 induction algorithm proceeds in a straightforward way. First a node is created and checks are made on the data instances passed into the function. If all instances are of the same class, the node becomes a leaf node and that class label is stored. In addition, a list of available splitting attributes is kept. If this list is empty the node becomes a leaf node and stores the majority class label. After this the main loop is entered: a split score is calculated for each available splitting attribute and the best one is used to split the data into subsets based on this attribute and its possible values. These splits result in new nodes being create and the ID3 algorithm is run recursively on

© 2016 M. Henry.

each of them.

Determining the split score for a given attribute can be done using an endless list of metrics. Popular methods include information gain, Gini index and train error. In these experiments, gain ratio is used, which is a variant of information gain that takes into account the number of possible values an attribute can take.

3. Tree Pruning

3.1. Reduced-Error Pruning

After a tree is grown to completion, a simple form of pruning is called reduced-error pruning. For reduced-error pruning, one visits each node and replaces it branches with the most population class label of the training instances that pass through that node. Classification error is checked against a validation set. If pruning doesn't hurt the tree, the prune is kept. Pruning continues until performance gets worse.

3.2. Early-Stopping

Early-stopping pruning is another simple pruning algorithm that stops growing the tree after the mean squared error falls below some tunable threshold.

4. Experimental Approach

4.1. Determining Split Point for Numeric Attributes

Attributes that can take a discrete number of values are straightforward to use in ID3. However, a strategy must be applied to numeric attributes to determine on to split on these attributes. The strategy used for these experiments is to sort the instances and create a list of candidate split values. A value is added to the candidate list when, in the sorted list, the class label between two successive instances changes. The mid-point between these instances for the attribute in question is added to the candidate list. After the candidate list is constructed, the gain ratio is calculated for each by splitting the instances in to two groups: less than and equal to the split value and greater than the split value. The split value with the highest gain ratio is chosen to serve as the split value for that attribute so it can be tested against the other available split attributes in the algorithm's main loop.

4.2. Regression Tree Node Impurity

Since gain ratio doesn't directly apply function approximating and regression, a different metric for split score is needed for regression trees. In these cases, I minimize the following equation that looks at variance in the function estimates for each node:

$$I_{var}(f) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2 - (\frac{1}{n_L}^2\sum_{i=1}^{n_L}\sum_{j=1}^{n_L}(x_i - x_j)^2 + \frac{1}{n_R}^2\sum_{i=1}^{n_R}\sum_{j=1}^{n_R}(x_i - x_j)^2)$$

4.3. Experiments

4.3.1. Classification Data Sets

In these experiments, two trees are grown, one to completion and one pruned with reduced-error pruning using a validation set from 10% of the data. The remaining 90% of the data is used for cross-validation to grade tree performance.

4.3.2. Regression Data Sets

In these experiments, two trees are grown, one to completion and one pruned with an early-stopping parameter. This early-stopping parameter was tuned with 5-fold cross-validation.

5. Results

It should be noted that for the regression data sets to complete in a reasonable time, a random subset of 100 instances were used for 5-fold cross validation.

Table 1: Classification Decision Trees

Classification Decision Trees		
	Unpruned	Pruned
Abalone	5.02%	4.89%
Car	14.18%	14.42%
Image	16.19%	17.04%

6. Conclusions

As can be seen in the classification class, performance between pruned and full classification decision trees is about the same. The difference in reported numbers should not be taken to reflect an actual difference in the two implementations. Further investigation showed that even though the reduced-error pruning tree had the option to prune the full tree, at least on these experiments it chose not to do so. Presumably, this is because no pruned tree outperformed the full tree on the validation set. This may be an artifact of the validation set only be the size of 10% of the original data. Further work would focus on data where larger validation sets could be curated.
For the regression data sets, a few things are at play: 1) to meet time complexity, a random subset of only 100 instances were used for each data set and 2) due to the same time

Table 2: Regression Decision Trees

Regression Decision Trees		
	Unpruned	Pruned
CPU	25584.45	99186.851126
Forest Fire	359.216736	7023.803298
White Wine	1.186357	0.517964
Red Wine	0.963492	0.355770

complexity, only four values for the hyper parameter of early stopping were tested. In future work, more hyper parameters should be tested to give a fairer assessment of early stopping.

7. Summary

In these experiments, classification and regression trees were implemented for three classification data sets and three regression data sets. In addition, reduced-error pruning was implemented for the classification trees and early stopping was implemented for the regression trees. For the classification experiments, pruning returned the same performance as not pruning due to the fact that the full tree always out performed the pruned tree. The time complexity introduced by the variance measured used to measure variance for each attribute split made testing the entirety of each regression data set too time consuming, so a subset of 100 instances were used for each. The poor performance of the pruned trees shows that more time should have been spent tuning the early stopping threshold.

8. References

[1] Quinlan, J. Ross. "Induction of decision trees." Machine learning 1.1 (1986): 81-106.
[2] Glenn De'ath, and Katharina E. Fabricius. "Classification and Regression Trees: A Powerful Yet Simple Technique for Ecological Data Analysis." Ecology 81.11 (2000): 3178-192. Web.
[3] UCI machine learning repository: Abalone data set. (1995, December 1). Retrieved October 23, 2016, from https://archive.ics.uci.edu/ml/datasets/Abalone
[4] UCI machine learning repository: Car evaluation data set. (1997, June 1). Retrieved October 23, 2016, from https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
[5] UCI machine learning repository: Image segmentation data set. (1990, November 1). Retrieved October 23, 2016, from https://archive.ics.uci.edu/ml/datasets/Image+Segmentation
[6] UCI machine learning repository: Computer hardware data set. (1987, October 1). Retrieved October 23, 2016, from https://archive.ics.uci.edu/ml/datasets/Computer+Hardware
[7] UCI machine learning repository: Forest Fires data set. (2008, February 29). Retrieved October 23, 2016, from https://archive.ics.uci.edu/ml/datasets/Forest+Fires
[8] UCI machine learning repository: Wine quality data set. (2009, October 7). Retrieved October 23, 2016, from https://archive.ics.uci.edu/ml/datasets/Wine+Quality
[9] ID3 Pseudocode - Swarthmore College. (n.d.). Retrieved October 23, 2016, from https://www.cs.swarthmore.edu/ meeden/cs63/f05/id3.html