Journal of Machine Learning Research 1 (2022) 1-5 Submitted 4/00; Published 10/00

Programming Assignment 3

Peter Rasmussenă PRASMUS3@JHU.EDU
Whiting School of Engineering
Johns Hopkins University
Baltimore MD, USA

Editor: Peter Rasmussen

Abstract

This paper discusses the design and execution of the decision tree classification and regression algorithms along with an analysis of the predictors' performance for three classification and three regression datasets. For classification, we used the ID3 algorithm and opted for gain-ratio to split attributes. We employed reduced-error pruning and compared those results to unpruned results. For regression, we used the CART algorithm and selected mean squared error to split attributes. We tuned for various threshold values (theta) on a validation set. For both classification and regression, we performed k-folds cross validation so that our results would be statistically significant. We tested two hypotheses. First, we hypothesized that the pruned models would outperform the unpruned models across each classification dataset. Second: positive values of theta would outperform the default zero-theta (no early stopping). The intuition behind both hypotheses is that pruning and early-stopping, for the classification and regression models, respectively, would tend to mitigate overfitting. Experimental results support the classification hypothesis; more research is needed to assess the regression hypothesis.

Keywords: Decision Trees, Entropy, Mean Squared Error, Classification, Regression

1. Introduction

The decision tree predictor is a nonparametric model "that is a hierarchical data structure implementing the divide and conquer strategy" (Alpaydin, page 2018). At each node, the decision tree finds a locally optimal solution. This greedy heuristic allows the tree to be built without having to exhaustively search all configurations, a problem which - per the lecture notes - would be NP Complete. Although any problem can be formulated iteratively or recursively, recursion is best suited for the decision tree.

As a nonparametric model, the decision tree employs the inductive bias that assumes values near one another will generally have like target values. More interestingly, the decision tree has a preference bias (this is the greedy heuristic) of always selecting the node split that minimizes impurity of the split. Minimization of impurity manifests as gain-ratio maximization in classification and mean squared error minimization in regression.

The rest of this paper is organized as follows. Section 2 provides the problem statement. Section 3 summarizes data preprocessing steps. Section 4 presents the experimental approach. Section 5 presents the results. Section 6 discusses the algorithm's behavior. Section 7 concludes.

2. Problem Statement

The objectives of this assignment are as follows. First, to teach the student the fundamentals of decision trees and how to use them for classification and regression. The second aim of this assignment is to introduce the student to the concept of overfitting and to gain actual experience employing methods commonly used to mitigate this. The third aim is to introduce the student to a new approach to handle numeric data for the decision tree case. Decision tree branches are really categories, and consequently numeric data must be converted into categories - two in this assignment - so the decision tree can process the feature.

3. Preprocessing

We used most of the preprocessing code that we developed from the first and second projects. However, we created a new class to handle to handle both classification and regression data sets. This new class handles numeric (i.e., continuous) and ordinal data differently than was done previously. That is, for numeric features we split features into binary less than / greater than categories. We opted for a different approach from what was recommended in the rubric. For ordinal data, we created splits equal to the number of unique ordinal values in the attribute. This created the maximum number of candidate branches to choose from given binary ordinal splits. For the numeric data, we created quantiles (e.g., 0.1, 0.2, ..., 0.9) and split the numeric data on each quantile. We did not dummy and used categorical data "as-is"; this allowed us to split categorical attributes into the number of branches equal to the number of that attribute's categories. Standardization is unnecessary in decision tree modeling, and so we did not standardize the data. Finally, we dropped "ID" columns, which provide no learnable information for the decision tree.

4. Experimental Approach

As in the first two programming assignments, we utilized K folds cross-validation to reduce overfitting. We first split up the data into two buckets: test-train (80%) and validation (20%). We then split the test-train into five folds. For each fold, we let one fold be the test and the other four folds be the train. For each fold, we tuned using the validation set.

In the case of classification, this meant using the validation set for the pruning of the tree. Specifically, we compared each interior node's gain ratio to that of the node if it were a leaf on the validation data. If the leaf outperformed the interior node's subtree, we made that node a leaf and deleted its children. We stratified each classification set so that classes were balanced across each fold and between the test-train and validation datasets.

For regression, we used the validation set for early stopping. We tested various values of theta, including zero for no early stopping, and compared performance - measured in terms of mean squared error - against the validation set. We specified theta or the threshold using relative error and then converted that - using the training data of the current fold - to mean squared error, which was the value used to perform the splits. This provided a more intuitive way to specify that parameter.

As mentioned above, we opted for a recursive approach for building out the tree. This simplified the code, although getting the hang of the recursion did take some time. We did not thankfully encounter any issues with recursion depth errors, a problem that we have encountered in previous programming assignments involving recursion.

For scoring, we used accuracy for classification and mean squared error for regression.

5. Results

The table below summarizes the classification results by fold and dataset.

dataset_name	fold	pruned_score	unpruned_score
breast-cancer-wisconsin	1	0.953	0.953
breast-cancer-wisconsin	2	0.929	0.905
breast-cancer-wisconsin	3	0.938	0.922
breast-cancer-wisconsin	4	0.967	0.943
breast-cancer-wisconsin	5	0.951	0.943
car	1	0.926	0.971
car	2	0.955	0.990
car	3	0.920	0.965
car	4	0.940	0.968
car	5	0.931	0.980
house-votes-84	1	0.896	0.909
house-votes-84	2	0.962	0.938
house-votes-84	3	0.947	0.921
house-votes-84	4	0.947	0.893
house-votes-84	5	0.988	0.928

We can see that - in terms of accuracy - both the pruned and unpruned models performed reasonably well. However, an F1 score would be ideal for comparing the performance given class imbalances. The following table summarizes performance by pruned and unpruned models across folds. We see that the pruned models tended to outperform the unpruned ones. Interestingly, the car dataset unpruned models outperformed the pruned models across the five folds. We surmise this may be due to the fact that the deeper, unpruned trees were better able to handle the four classes in that dataset. The other two datasets only

have two classes, each, and so perhaps the pruned model did a better job of not overfitting to those datasets.

dataset_name	pruned_score	unpruned_score	mean
breast-cancer-wisconsin	0.948	0.933	0.940
car	0.934	0.975	0.955
house-votes-84	0.948	0.918	0.933
mean	0.943	0.942	0.943

We also investigated the feature importances of two of the models. We wanted to get a better idea of which features, across folds, tended to be more important in terms of gain-ratio. We see in the table below good agreement among the top five feature importances for the car pruned and unpruned datasets. On the other hand, there is difference in the top five feature importances between the breast cancer unpruned and pruned results. This could be random error due to sampling, or perhaps something else.

rank	1	2	3	4	5
breast-cancer__unpruned	uniformity	normal	bland	clump	bare
breast-cancer__pruned	uniformity	clump	bare	bland	normal
car__unpruned	persons	safety	buying	maint	lug
car__pruned	persons	safety	buying	maint	lug

A truncated version of the regression results are provided in the table below, which shows we preformed the training across all five folds, computing the test mean squared error for each threshold (theta) value.

dataset_name	fold	theta_rel	theta_abs	tune_mse	test_mse
abalone	1	0.00	0.000	9.057	8.441
abalone	1	0.02	0.043	9.057	8.441
abalone	1	0.04	0.173	9.057	8.441
abalone	1	0.06	0.390	9.057	8.441
abalone	2	0.00	0.000	9.028	8.207
abalone	2	0.02	0.043	9.028	8.207
abalone	2	0.04	0.174	9.028	8.207
abalone	2	0.06	0.390	9.028	8.207
abalone	3	0.00	0.000	9.013	7.678
abalone	3	0.02	0.044	9.013	7.678
abalone	3	0.04	0.176	9.013	7.678
abalone	3	0.06	0.396	9.013	7.678
abalone	4	0.00	0.000	9.079	8.663
abalone	4	0.02	0.043	9.079	8.663
abalone	4	0.04	0.172	9.079	8.663
abalone	4	0.06	0.388	9.079	8.663
abalone	5	0.00	0.000	9.025	7.304

A more concise summary table, which shows scores by threshold value, is provided below. Unfortunately, the machine results were poor. Subsequent analysis is necessary to determine precisely what the issue is. Another issue with the results is that there was no

discernible effect of the theta on performance, which is a red flag that something is amiss with the algorithm. Additional time would be required to determine precisely what the issue is, but unfortunately that issue remains unresolved as of this writing.

dataset_name	theta_rel	theta_abs	tune_mse	test_mse
abalone	0.03	0.152	9.041	8.059
forestfires	0.03	0.004	2.037	1.964
machine	0.03	50.100	146787.167	144677.820

6. Algorithm Behavior

The decision tree is a nonparametric model that finds a locally optimal solution at the current node. It handles non-linear data well as the combination of features can separate classes and values that are linearly inseparable. As alluded to, the inductive bias of the algorithm is its assumption of local optimality, which it employs to efficiently build a tree that would otherwise take NP Complete time.

7. Hypotheses

The results of the experiments support our hypotheses, although not as strongly for the classification case as we had expected. We thought that the classification models would perform significantly better pruned, but this was so in only two of the three classification datasets. We suspect this may be due to the fact that the car dataset, with its four classes, needs a bigger tree to find its more numerous classes. Notwithstanding the car dataset, the results from the other two models supported our hypothesis.

On the regression side, the results are inconclusive. Unfortunately, (we surmise) an unknown issue with the regression model may have affected the early stopping portion of the algorithm. Because there was no variation across thresholds, the evidence does not support the hypothesis; however, this is again likely due to issues with the model.

8. Conclusion

The objective of this assignment was to learn how to implement another common nonparametric algorithm, this time the decision tree. For the last programming assignment, we learned too late that the car dataset was multi-class; we were ready this time. Hence, we successfully trained classification and regression models on the given datasets.

We learned towards the end of this exercise that the use of an IDE greatly improves efficiency of the coder. Going forward, we will use the IDE from the get-go rather than taking the student's usual approach of building out the algorithms in a Jupyter notebook.

Although we did not employ the multi-class F1 scorer, which we intended to do at the outset of this programming assignment, we hope to implement that feature in the next assignment.

Tuning and training time was much faster for the decision tree than for k nearest neighbors. We used more vectorized equations and built Boolean rulesets in the classification case to quickly predict the desired classes. was too long for the size of the datasets used. Even so, training the regression models - particularly the abalone one - was time consuming.