Post-Train Data Addition To Decision Trees

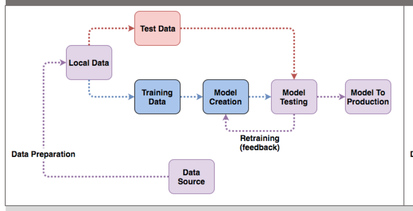
Jeremy Swerdlow
Computer Science, Earlham College

Introduction

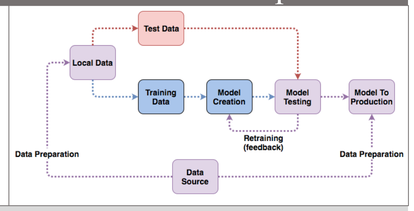
- Decision Trees are among the most powerful techniques in machine learning. They are widely used for prediction problems. However, their widespread use in applications involving Big Data means even though they are powerful, they are expensive to maintain. Creating such models can take hours, even days.
- Decision Trees also become less effective over time, requiring new data to be used to improve their performance. However, in many cases, the data is reused in the development of the new tree. The old and new trees share similar features and data. To combat this cost, many other types of machine learning models have ways to adapt to new data in place, such as recurrent neural networks.
- In an effort to reduce the time cost of these updates, this project presents an algorithm designed to add data to a tree in-place, instead of rebuilding the tree from scratch.

System Diagram

Common Practice



New Concept



- The diagram on the left shows the standard flow of data through the process of training and testing before reaching production.
- On the right, the new system is shown, wherein data is piped directly to the model in production.

Methodology

- The initial step was to develop an algorithm for how to add data to a pre-existing tree. This required understanding of how the tree would originally be created.
- For the base tree, the DECISION-TREE-LEARNING algorithm presented in Russel and Norvig's *Artificial Intelligence: A Modern Approach* was used. It is a foundational algorithm, which is easily implemented in Python.
- With the understanding of how the tree is developed, the ADD-TO-DECISION-TREE algorithm was then created to best utilize the features needed. It is built recursively, similar to how the tree algorithm works.
- Based on how this algorithm works, several changes were made to the implementation of the decision tree to make certain information available.
- Tests were then developed using binary classification metrics.

Psuedocode

Decision Tree Creation Algorithm[1]

function DECISION-TREE-LEARNING(*examples, attributes, parent_examples*) **returns** a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow$ argmax$a \in$ *attributes* IMPORTANCE(*a, examples*)
  tree $\leftarrow$ a new decision tree with root test *A*
  for each value *vk* of *A* **do**
    exs $\leftarrow \{e : e \in examples$ **and** $e.A = vk\}$
    subtree $\leftarrow$ DECISION-TREE-LEARNING(*exs, attributes* $-$ *A, examples*)
    add a branch to *tree* with label ($A = vk$) and subtree *subtree*
  return *tree*

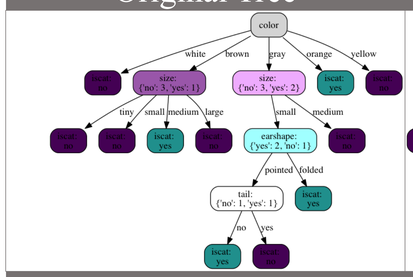
Decision Tree Addition Algorithm

function ADD-TO-DECISION-TREE(*examples, tree*) **returns** a tree

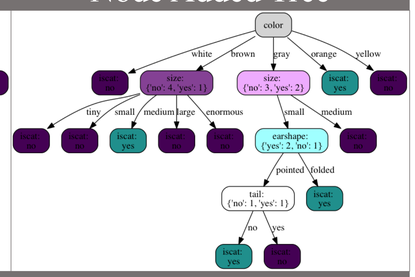
goal = tree.goal
dec = tree.decision
for each value *row* of *examples* **do**
  add *row* to *tree.data*
  if *row[dec]* in *tree.branches* **then**
    if *tree.branches[row[dec]]* is a leaf **then**
      if *row[goal]* is equal to leaf value **then** return *tree*
      else
        exs $\leftarrow \{e : e \in$ *tree.data* **and** *e[dec] = row[dec]*$\}$
        tree.branches[row[dec]] = DECISION-TREE-LEARNING(*exs, tree.remaining_attributes, tree.data*)
    else
      tree.branches[row[dec]] = ADD-TO-DECISION-TREE(*row, tree.branches[row[dec]]*)
  else
    tree.branches[row[dec]] = row[goal]
return *tree*

Example Trees

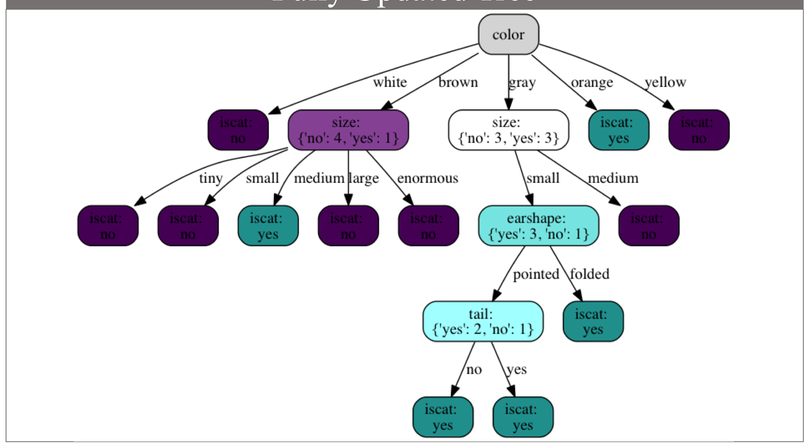
Original Tree



Node Added Tree



Fully Updated Tree



Results

- To see the results of using the proposed algorithm, a series of tests were developed. The tests used an AB method to compare the results of the algorithm against the current practice of rebuilding the tree.
- As decision trees are a machine learning technique, they rely on data to be built. For this project, data came from Kaggle, an online data housing and distribution service, and from David Barbella, Assistant Professor of Computer Science at Earlham College.
- Each test developed used a series of benchmark metrics for classification problems to see how the new model performs in comparison to the control. Certain tests were run to see results when specific elements of the algorithm were reached.
- These data and tests allow for an extensive testing of the algorithm. If the tests show positive improvements in the performance metrics and a decrease in time, the algorithm will be successful.

Conclusion

- The algorithm developed in the process of this research shows decision trees can be successfully updated to contain new data.
- The algorithm allows data to be added regardless of if there are existing nodes within the tree which match its features or not. Regardless of if such new nodes are created, the weights of the tree are updated to match the entire dataset.
- While this result is a positive one, it gives way to several future research questions. Data accuracy is known to decay over time, yet this research is limited only to the editing of trees to add data; it does not explore deleting data from trees. It also adds data one row at a time, instead of attempting to handle it as a batch.
- Additionally, this algorithm is built to work with one of the more basic implementations of decision trees. Further research could go into the creation of similar methods for more complex algorithms.

References & Acknowledgements

References

[1] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall.
[2] "graphviz / graphviz," *GitLab*. [Online]. Available: https://gitlab.com/graphviz/graphviz. [Accessed: 04-Apr-2018].

Acknowledgements

A special thanks to the Earlham College Computer Science Department faculty for their support throughout this research. Without their support this project would not have been successful.
A thanks as well to the System Administrators at Earlham College for the rapid installation of all relevant resources to the work when needed.