Homework 1 Response Questions

Nicole Birova, Jack Hanke, Dan Plotkin, Vrishani Shah, Hanna Zelis

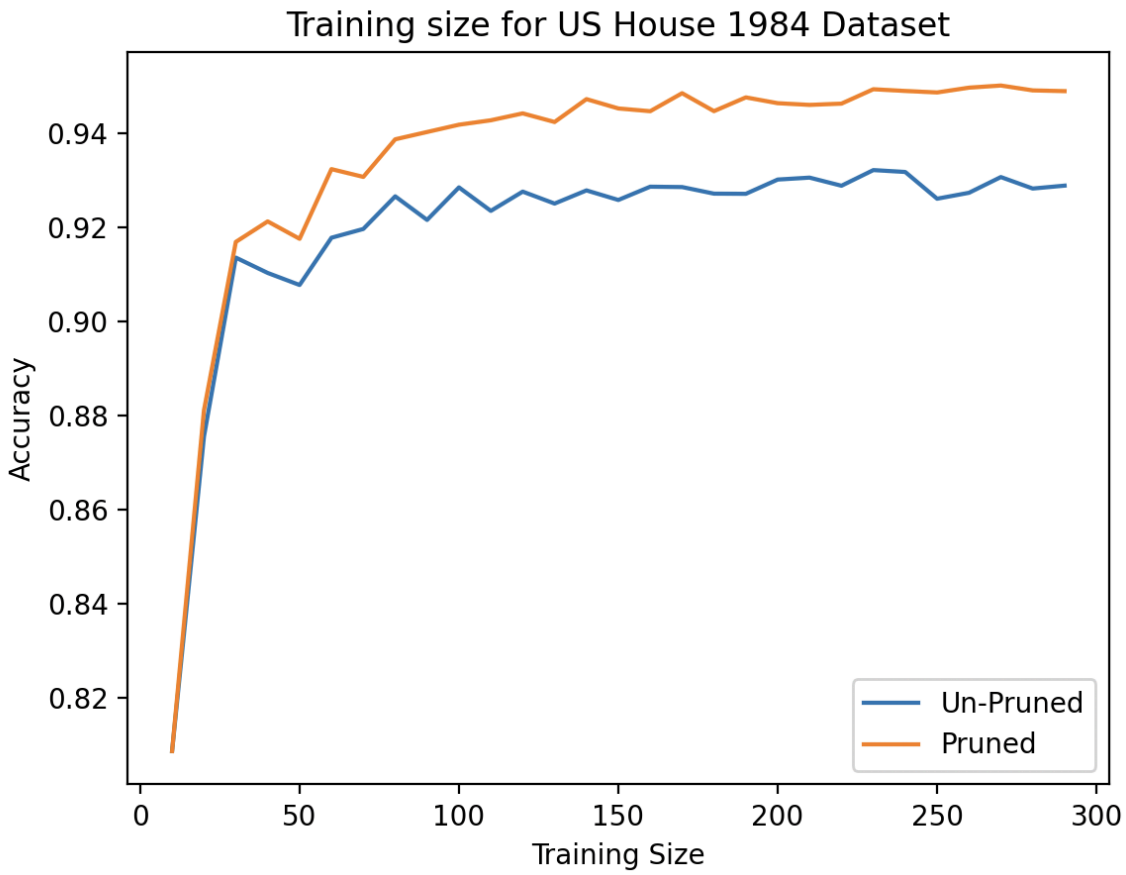
17 October 2024

1. **Did you alter the Node data structure? If so, how and why?**
    a. We altered the Node data structure by adding two additional fields called *self.leaf_eval* and *self.relative_freqs* and setting both to *None*. The purpose of creating *self.leaf_eval* is so when a node becomes a leaf in the decision tree, this stores the class value. It is utilized within our evaluate function as the return value, as when a tree is done being split, the leaf_eval holds the class value, allowing us to easily return the leaf node's class value. Overall, *self.leaf_eval* is used for when a tree traversal hits a leaf node, it stops and outputs the correct class, ensuring the splitting is complete and then outputs the correct class.
    b. Additionally we added *self.relative_freqs*, which assists with pruning by tracking the relative frequencies. During the pruning implemented in our ID3, if the children are removed, a node can be assigned a particular class based on the frequencies stored in this field. Taking this route of pruning helped with improving the accuracy.

2. **How did you handle missing attributes, and why did you choose this strategy?**
    a. We handled missing attributes by finding the mode of each column of each attribute and filling in those '?' attributes (the missing attributes) with the mode of their column. We utilized this strategy because, given categorical attributes, the mode represented most of the population correctly. This was implemented in the ID3 and test functions.

3. **How did you perform pruning, and why did you choose this strategy?**
    a. We performed pruning to the tree by traversing it bottom-up. This meant checking the leaf nodes, and then proceeding to loop over the parents of each child. Considering the parent as a leaf, the method prunes the tree at that node, and checks if the accuracy improves. If accuracy increases or stays the same, the tree moves forward with pruning (having pruned that subtree). If not, it reverts back to the original tree (adding the children back as a subtree). This is the method based on reduced-error pruning, which we chose because it prevents overfitting and improves accuracy while decreasing computation. On running the iterations, we could observe the similarity in accuracy even on pruning the tree, which gave information regarding multiple extra features which were not affecting the accuracy. Pruned trees tended to give better results on the test data compared to the entire tree, this was probably because it was overfitting on the training data.

4. Now you will try your learner on the *house_votes_84.data* and plot learning curves. Specifically, you should experiment under two settings: with pruning and without pruning. Use training set sizes ranging between 10 and 300 examples. For each training size you choose, perform 100 random runs, for each run testing on all examples not used for training (see *testPruningOnHouseData* from *unit_tests.py* for one example of this). Plot the average accuracy of the 100 runs as one point on a learning curve (x-axis = number of training examples, y-axis = accuracy on test data). Connect the points to show one line representing accuracy *with* pruning, the other *without*. **Include your plot in your pdf, and answer two questions:**



   a. **What is the general trend of both lines as training set size increases, and why does this make sense?**
      i. As the training set size increases, the accuracy increases. Because there is more data to test the model, the accuracy increases, hence better generalization. However, once the amount of data reaches a certain point,

the accuracy only increases by a little, which can be seen around the 250 training size, where both the unpruned and pruned dataset starts to plateau.

b. How does the advantage of pruning change as the data set size increases? Does this make sense, and why or why not?

i. Although both the unpruned and pruned accuracy increases as the training size increases, the pruned accuracy starts to improve more than the unpruned accuracy around the 25 training size. This is because the pruned dataset removes unnecessary parts of the decision tree, causing improvement of the decision tree's accuracy.

5. Use your ID3 code to learn a decision tree on *cars_train.data*. Report accuracy on the *cars_train.data*, *cars_valid.data*, and *cars_test.data* datasets. If your accuracy on *cars_train.data* is less than 100%, explain how this can happen. Prune the decision tree learned on cars_train.data using cars_valid.data. Run your pruned decision tree on *cars_test.data*, and explain the resulting accuracy on all three datasets.

a. Dataset 1: cars_train.data

i. If your accuracy on *cars_train.data* is less than 100%, explain how this can happen

1. Accuracy: 1.000 (100%)
2. Explanation: No need to explain if accuracy is less than 100% as we got 100% accuracy. It also may have just been able to perfectly classify all of the instances.

b. Dataset 2: cars_valid.data

i. Accuracy: 0.8000 (80%)

ii. Explanation: We get 80% accuracy on our validation dataset, which was the result of our cars_test.data after pruning. This shows that the cars_valid.data set proved that the pruning implemented worked as appropriate to improve accuracy while ensuring we did not overfit.

c. Dataset 3: cars_test.data

i. Accuracy: 0.6286 (62.86%)

ii. Accuracy after pruning: 0.8000 (80%)

iii. Explanation: When running the cars_test.data, there was an alright accuracy, but since the cars_train.data may have been overfit, the accuracy of cars_test.data was not as good as it could be. So, after implementing pruning, the accuracy of the cars_test.data went up, showing that overfitting was eliminated and that pruning we implemented improved accuracy.

6. Use your ID3 code to construct a Random Forest classifier using the *candy.data* dataset. You can construct any number of random trees using methods of your choosing. **Justify your design choices and compare results to a single decision tree constructed using the ID3 algorithm.**
    a. We created a Random Forest Classifier by taking a subset of our dataset. Specifically, bootstrap sampling was implemented for the rows and random feature selection based on a feature sub size parameter. The number of features and rows in the dataset were chosen randomly. Based on the shortened dataset, with a choice of certain features and certain data rows, the new tree was trained. The number of features and the number of rows to be selected were kept as a user input.
    b. On giving input of 20 trees and 3 features subset size, our accuracy for candy.data comes to be 69.41% and on putting 50 trees and 3 subset size, the accuracy increases to 77.65%. This shows that on increasing the number of weaker trees, the accuracy tends to increase. On putting 20 trees and 7 feature sizes, the accuracy becomes 78.82. Overall we can see, the accuracy improves on increasing the feature size and number of trees.