# Assess_learners Report

Author: Yu Bai
Date: 9/18/2017

**Q1: Overfitting issue in classic Decision Tree with respect to the *leaf_size***

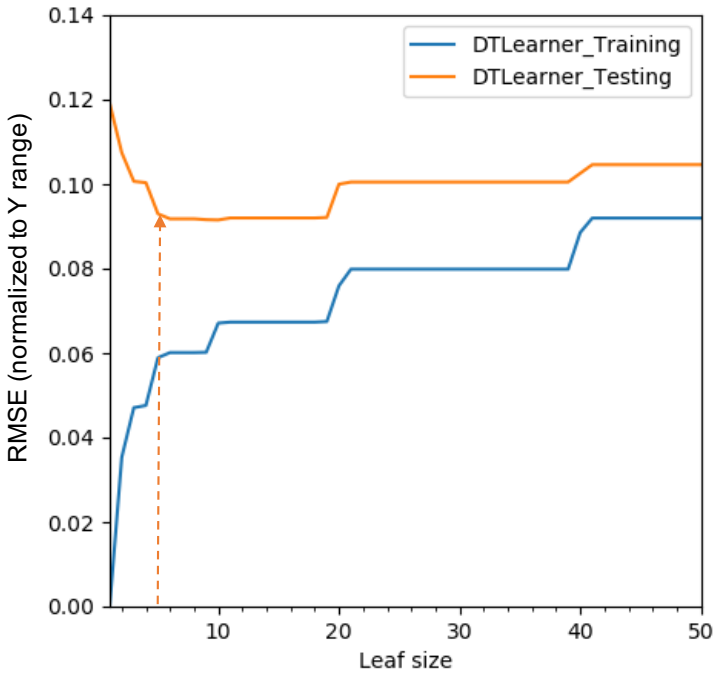
To address whether and how the overfitting occurs regarding the *leaf_size* parameter, we conducted the following experiments:

**Step 1**, a total of 536 samples were read in from the file *istanbul.csv* using the read-in utility provided in the *testlearner.py*. The first 60% (321 samples) and the rest 40% (215 samples) were used as the training and test datasets, respectively.

**Step 2**, a single DTLearner (for classic Decision Tree) was trained using the training dataset described above, subject to a systematic variation of the *leaf_size* from 1 to 50.

**Step 3**, for each *leaf_size*, the performance of the learner in the training and testing datasets was assessed by the root-mean-square errors (RMSE). We normalized the training and the testing RMSE by the range of Y values in the training and testing datasets, respectively. Normalization won't affect the analysis of RMSE with respect to the leaf size. However, it removes the bias due to the different magnitude of Y values such that the comparison between the training and testing RMSEs, when interested, is more fair.

**Figure1**: The impact of the leaf size on RMSE in DTLearner




The experiment results are summarized in Figure 1. Note the DTLearner curves are stair-wise due to its binary split operation: unless the leaf size doubles the same tree structure (i.e. the model) holds and hence the same prediction results. We can see that the DTLearner overfits when the leaf size is <=5 (Figure 1, left panel, leaf size=5 marked by arrow). This region is considered overfitting because although the training RMSE keeps descending, the testing RMSE instead increases. The turning point starts at leaf size=5. From that point on, the smaller the leaf size, the worse the testing RMSE. Clearly

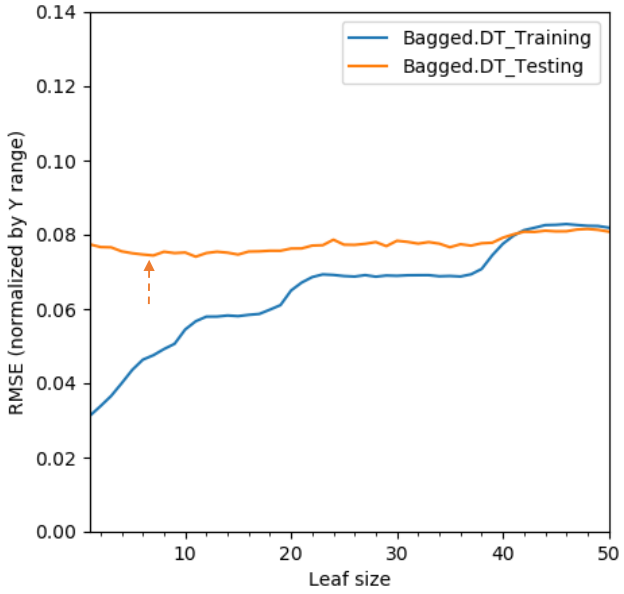
the model performance on the unseen data plunge in this region. This loss of generalizability is, by definition, the overfitting.

It is expected to see tree learners overfit with small the leaf sizes. The smaller the leaf size, the deeper the tree. The model complexity rises such that not only the signal but the noise in the data are fitted. Consequently, the model generalizability decreases.

**Q2: Overfitting in the Bagged DTLearner with respect to the *leaf_size***

For this experiment, we followed almost the same 3-step procedure as described in Q1. The differences are: in Step2, the BagLearner was constructed using the DTLearner as the base learner, and with a fixed number of 20 bags. In Step3, the RMSEs were computed for the bagged learner, not its base learner. To reduce the noise due to the randomness of the boot strapping, we repeated the experiment 10 times and reported the averaged RMSEs here.

**Figure2**: The impact of the leaf size on RMSE in the Bagged DTLearner




In Figure 2, when the training error decreases with smaller leaf size, the testing RMSE of the bagged DTLearner stays rather stable. There might be a slight ascending trend when the leaf size approaches to ~6 and lower (arrow, Figure 2). Nonetheless, the magnitude of the upturn is much smaller than what is observed in Figure 1. In fact, even at the minimum leaf size of 1, the testing RMSE remains below 0.08, whereas it reaches ~0.12 without the bagging in Figure 1. This observation suggests that, given a bag size of 20, the bagging significantly reduces the overfitting.

It's worthwhile to mention that the bagging reduces both the training and testing errors throughout the leaf sizes (except that the training error at leaf size=1 is bigger than 0). For example, all testing RMSEs are about or lower than 0.08 in Figure 2 yet they all are >0.08 in Figure 1. Usually the DTLearner with a large leaf size is under-powered due to the oversimplification of the tree model. Each underpowered tree alone is a weak learner and it produces higher training & testing RMSE. However, with bagging the ensemble of weak learners shows much improved performance. Thus, we see

consistent low RMSEs from small to large leaf sizes (1-50). In other word, we can see here that bagging also helps prevent under-fitting.

**Q3: Quantitative comparison between the DTLearner and RTLearner**

We here focus on two aspects, the prediction accuracy and the computational efficiency, of the two learners, in the absence of bagging. The experiments were conducted over 5 datasets: *3_groups*, *ripple*, *Istanbul*, *winequality-red* and *winequality-white*. For each dataset, the same read-in module in the *testlearner.py* was applied such that 60% and 40% samples were assigned to the training and the test datasets, respectively.

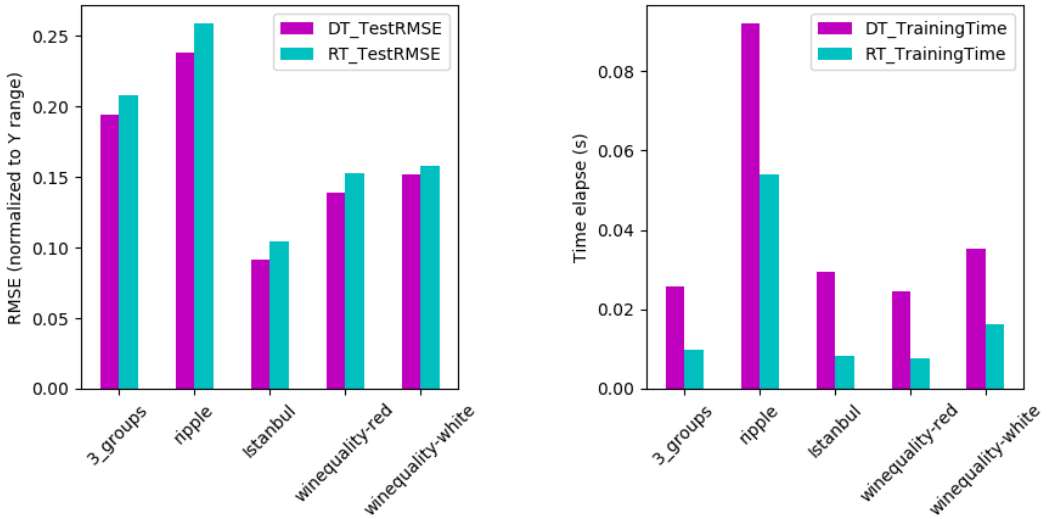
To avoid using ill-constructed learners for this comparison (e.g. overfitted trees), prior to the experiments, we first assessed what leaf sizes were proper to build both learners for each of the data sets. We conducted the same experiments as described in Q1 to define the overfitting regions for the DTLearner and the RTLearner in each dataset. The minimum leaf size outside the overfitting regions of both learners was considered as the best leaf_size without overfitting. The results are shown in Table 1.

For each dataset, we built both the DTLearner and the RTLearner with the same leaf size that doesn't cause overfitting as specified in Table 1, then calculated their RMSEs with respect to the test data as a measure of the prediction accuracy. We also calculated the training time cost for both the learners as a metric for the efficiency. Note that the RTLearner has noisy RMSE & time elapse due to its random nature. To obtain a more stable view, the experiments of the RTLearner were repeated 10 times and the average values were reported here.

**Table 1:** Selection of the leaf sizes with which both tree learners don't overfit

| DataSet | DTLearner overfitting region | RTLearner overfitting region | best leaf_size without overfitting |
|---|---|---|---|
| 3_groups | leaf size <= 5 | leaf size <= 10 | 10 |
| ripple | leaf size <= 2 | leaf size <= 2 | 2 |
| Istanbul | leaf size <= 5 | leaf size <= 9 | 9 |
| winequality-red | leaf size <=35 | leaf size <= 30 | 35 |
| winequality-white | leaf size <= 60 | leaf size <= 55 | 60 |

**Figure3**: Compare the prediction accuracy and the time efficiency of DTLearner and RTLearner




As shown in Figure 3 left panel, the DTLearner consistently has lower testing RMSE values than the RTLearner over all 5 datasets. The decrease in RMSE ranges from 3.6% (*winequality-white* data) to 12.4% (*Istanbul* data). It indicates the single DTLearner has higher prediction accuracy than a single

RTLearner. This result is expected as the DTLearner optimizes the data explanation at each learning step by selecting the X variable that is the most informative in explaining the variation of the Y values (i.e., highest information gain), whereas the RTLearner simply choose a variable randomly that may or may not be relevant to the real signal in Y. So in terms of the prediction performance, the DTLearner is better.

On the other hand, because the selection of the best variable at each step, as well as finding the median value of that variable to ensure the split is as even as possible, the DTLearner is considerably slower than the RTLearner in all 5 datasets (Figure 3, right panel). The time reduction ranges from 41.4% (*ripple* data) to 71.8% (*Istanbul* data). So efficiency-wise the RTLearner is better.

Note the above conclusions are drawn only for the single DTLearner and the single RTLearner. They may change when an ensemble learning strategy is applied, which is not discussed for Q3 here.