# iMauve: A Fast Incremental Model Tree Learner

Denny Verbeeck, Hendrik Blockeel

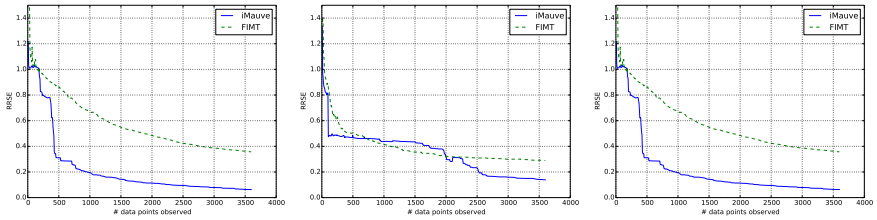
Department of Computer Science, KULeuven,
Celestijnenlaan 200A, 3001 Heverlee, Belgium
{Denny.Verbeeck,Hendrik.Blockeel}@cs.kuleuven.be

We consider the following task: Given a stream of elements of the form $(\mathbf{x}_i, y_i)$, learn a model tree $M : \mathcal{X} \rightarrow \mathcal{Y}$ that predicts $y$ from $\mathbf{x}$. The main difference to the standard learning setting is that data are assumed to arrive sequentially as part of a stream: each data element can be looked at briefly when it arrives, then disappears.

The recent work by Ikonomovska (FIMT) [2] is the current state of the art in learning model trees from streams. Using Hoeffding bounds [1], a split is made only if the splitting heuristic of the best split is better than the second best split with a specified confidence $\delta$. The splitting heuristic FIMT uses is *standard deviation reduction*, as in M5 [3]. Compared to regression trees, which predict a constant value in each leaf, model trees store a linear equation in each leaf that wil be used to make a prediction for instances sorted into this leaf. Thus, when learning model trees from streams, one not only needs to learn a tree structure incrementally, but also the linear regression function in the leaves of this tree. FIMT uses a single perceptron without activation function as the linear model in each leaf. The weights of the perceptron are updated using the Widrow-Hoff rule.

In this work, we propose two main adaptations to Ikonomovska's approach. The first change we propose is to modify the splitting heuristic. As pointed out by Vens and Blockeel [4], the standard deviation reduction heuristic used in many tree learners is sub-optimal when used in combination with linear regression in the leafs. In their work, a heuristic based on simple linear regression, called Mauve, is proposed. This heuristic calculates the reduction in standard deviation of the residuals, assuming a linear model with only the split attribute as a regressor. We have extended Mauve to calculate the residual standard deviation using a full linear least squares regression in all attributes, with no need for random access to the data. The second change we propose is the type of linear regressor used in the leafs of the tree. As the tree grows, the number of data points that each leaf observes decreases exponentially. However the perceptron update rule relies on a large amount of observations to converge the weights of the linear model. As an alternative, we propose a linear least squares regression to estimate the weights of the linear model in each leaf. This approach does not need time to converge, but gives the weights which minimize the sum of squared residuals, given the currently observed data. Like in FIMT, a set of statistics is kept up-to-date when each new data point is observed. The same set of statistics can be used for the split heuristic as well as the linear regression model. We call the resulting algorithm iMauve.

As a result of the adaptations described above we hypothesize: (1) iMauve will learn more compact trees without sacrificing accuracy compared to FIMT, (2) iMauve will reach higher accuracy levels after a lesser amount of observations compared to FIMT, (3) iMauve's higher time complexity will not be prohibitive for problem settings with a reasonable amount of attributes. We set up experiments on different data sets using 10-fold cross-validation. Due to space constraints we here only the results for three datasets, containing 4000 instances each. Paraboloid is a 2-attribute problem, while Lexp and Losc are 5-attribute problems. We measure RRSE during the streaming phase (when the algorithms are receiving data observations). The resulting curves are shown in Figure 1. Tree size and run-time are measured at the end of the run, i.e. when the data set is exhausted. These results are averaged over the 10 runs and shown in Table 1.




Paraboloid Lexp Losc

Fig. 1: Comparison of the evolution of the RRSE for different data sets.

| | FIMT Runtime (s) | iMauve Runtime (s) | FIMT Tree Size | iMauve Tree Size |
|---|---|---|---|---|
| Paraboloid | 0.394 | 0.460 | 22.7 | 20.8 |
| Lexp | 0.713 | 1.963 | 23.7 | 15.2 |
| Losc | 0.725 | 1.431 | 22.9 | 16.5 |

Table 1: Results averaged over 10 runs

Our experiments confirm these hypotheses on almost all data sets used in the comparison. The trade-off made is that the more advanced split heuristic and regression method makes iMauve the slower algorithm of the two in terms of time taken to process each observation. For a small number of attributes, as is the case in the problems shown here, iMauve processes observations at a rate of 2000 to 8000 examples per second. A moderate sized problem, like the Wine quality dataset which has 11 attributes, is still processed at about 1600 observation per second, making iMauve useful for a broad category of applications. iMauve also has a larger memory footprint due to the increased number of statistics that needs to be kept. However iMauve tends to learn smaller and more accurate trees, while needing less observations to accomplish this.

## References


1. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 71–80. KDD '00, ACM, New York, NY, USA (2000)
2. Ikonomovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. Data Mining and Knowledge Discovery 23(1), 128–168 (2011)
3. Quinlan, J.R.: Learning with continuous classes. In: Proceedings of the Australian Joint Conference on Artificial Intelligence. pp. 343–348. World Scientific, Singapore (1992)
4. Vens, C., Blockeel, H.: A simple regression based heuristic for learning model trees. Intell. Data Anal. 10(3), 215–236 (May 2006)