Speeding Up Hoeffding-Based Regression Trees with Options

Elena Ikonomovska ELENA.IKONOMOVSKA@IJS.SI
Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

João Gama JGAMA@LIAAD.UP.PT
LIAAD/INESC - Porto and Faculty of Economics - University of Porto

Bernard Ženko BERNARD.ZENKO@IJS.SI
Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

Sašo Džeroski SASO.DZEROSKI@IJS.SI
Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

Abstract

Data streams are ubiquitous and have in the last two decades become an important research topic. For their predictive non-parametric analysis, Hoeffding-based trees are often a method of choice, offering a possibility of any-time predictions. However, one of their main problems is the delay in learning progress due to the existence of equally discriminative attributes. Options are a natural way to deal with this problem. Option trees build upon regular trees by adding splitting options in the internal nodes. As such they are known to improve accuracy, stability and reduce ambiguity. In this paper, we present on-line option trees for faster learning on numerical data streams. Our results show that options improve the any-time performance of ordinary on-line regression trees, while preserving the interpretable structure of trees and without significantly increasing the computational complexity of the algorithm.

1. Introduction

Numerical data streams can be found in many real-world problem domains. Meteorological and financial data, computer and sensor networks, web applications, are just a few of the most representative ones. In all these situations, the data flows continuously and dynamically at high speed, which demands efficient solutions for any-time learning.

Algorithms for incremental learning of classification (Domingos & Hulten, 2000; Gama et al., 2003; 2004), regression and model trees (Potts & Sammut, 2005; Ikonomovska et al., 2010) represent a well established line of research on any-time prediction. Although the main idea behind these approaches is that splitting decisions can be derived on a subsample rather than on the whole training dataset at each tree level, they differ in their computation complexity. Hoeffding-based tree learners (Domingos & Hulten, 2000) have been recognized as the most efficient in terms of processing speed per example, although their learning might be slow, which results in lower any-time accuracy at the beginning.

The Hoeffding-based trees tend to be less accurate in situations where several attributes appear to be equally discriminative. In such tie-situations, the split selection method based on the Hoeffding bound might never decide, which attribute is significantly better. To solve this problem, Hoeffding-based tree learners typically use a tie-breaking mechanism based on a user-defined threshold, which implicitly specifies the amount of data that has to be observed in order to decide on one of the competitive attributes. Setting this threshold requires knowledge of the problem domain. An additional side effect is that the splitting decision will be significantly delayed, which results in a lower any-time accuracy during this delay.

We propose to solve this problem using option trees, which can include option nodes (options) in addition to ordinary split nodes. The main motivation is that

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

introducing option nodes removes the need for selecting the best splitting attribute. As a result, the greedy search that uses only one step look-ahead and makes decision trees unstable learners is replaced with a more robust search procedure[1]. The introduction of option nodes does not significantly increase the computational complexity of Hoeffding-based algorithms, and option trees are shown to outperform regular trees in terms of prediction accuracy (Kohavi & Kunz, 1997).

Improving the accuracy of Hoeffding trees was the main motivation for the work of Pfahringer et al. (2007) who propose an algorithm for adding options to Hoeffding-based classification trees. Their results show that the additional structure improves the accuracy in almost all the datasets used. What distinguishes their work from ours, besides their trees being classification ones and our regression ones, is that their method for adding options is employed after a split has already been chosen. On the other hand, we opt for a faster improvement of the any-time accuracy, which is a very important property of predictive algorithms for real-time dynamic systems.

Our main idea is to introduce options only when splitting decisions are ambiguous, which will avoid excessive and unnecessary tree growth and reduce memory consumption. Our experimental results support the ideas presented above and show an improvement of the any-time accuracy of Hoefding-based regression trees. Our analysis also confirms previously obtained conclusions for batch learners (Kohavi & Kunz, 1997), which show that option nodes are most useful near the root of the tree and act as a method for variance reduction (in terms of a bias-variance decomposition of the error).

The reminder of the paper is organized as follows. Section 2 surveys the related work. Our algorithm for learning on-line regression trees with options (ORTO) is presented in Section 3. Section 4 describes the evaluation methodology, and Section 5 presents the experimental results. The last section concludes and gives some directions for future work.

2. Related Work

Option trees are similar to regular decision trees, but can also contain *option nodes* (or *options*) in addition to the standard splitting nodes (Buntine, 1992; Kohavi & Kunz, 1997). In Figure 1, we give an example of a

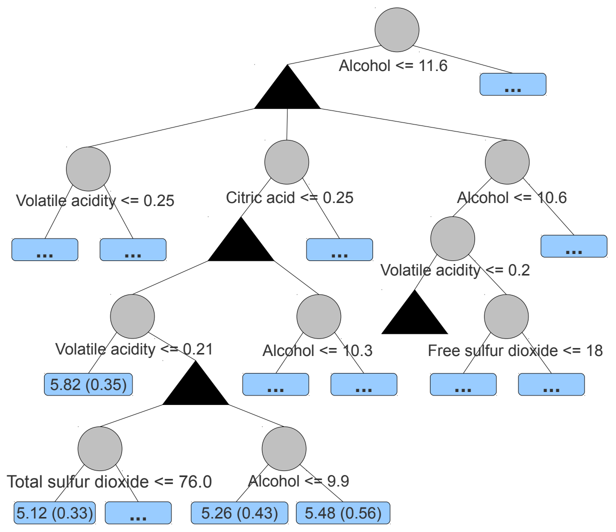

Figure 1. An option tree for the Wine Quality dataset.

partially constructed option tree for the Wine Quality data set from the UCI repository (Frank & Asuncion, 2010). The triangular nodes represent the option nodes, which implement an OR function. As a result, an example arriving at an option node follows all of its branches, ending at multiple leaves.

2.1. Options for Any-time Decision Trees

The work of Pfahringer et al. (2007) is the first that explores options as an extension to Hoeffding trees for on-line classification. Their approach is somewhat different from the original (batch) option trees (Buntine, 1992), mainly because option nodes are not introduced in the split selection process, but only at existing splits as a re-evaluation step triggered on predefined time intervals. A new option is introduced only if the best unused attribute looks better than the current split. Predictions of options are combined by averaging.

Following Pfahringer et al. (2007), another variant of option trees on streams (Liu et al., 2009) is based on the CVFDT algorithm (Hulten et al., 2001) in which alternate trees are initiated every time a new splitting test is found to be better than the existing one. The main difference is that not all alternate trees are used for prediction, only the most recent option that offers the highest accuracy. An option thus reflects the most recent and best performing knowledge.

The disadvantage of both above mentioned methods is the way option nodes are introduced. To be able to perform a re-evaluation, sufficient statistics must be stored in all internal nodes, which implies an increase in memory consumption and computation. In

[1]The greedy split evaluation criterion typically used for learning trees prefers attributes that score high in isolation. As a result, it may overlook combinations of attributes, leading to better solutions globally (Kohavi & Kunz, 1997).

addition, because the Hoeffding bound is conservative, the new sub-trees will be introduced with some delay, which will at least temporarily decrease the accuracy or delay the improvement. In our approach, on the other hand, we introduce options before a split is confirmed and therefore eliminate the delay.

2.2. On-line Model Trees

Many sophisticated methods can be found in the literature for solving classification tasks on streams, but only a few exist for regression tasks. To the best of our knowledge, there exist only two algorithms for on-line learning of regression and model trees. In the algorithm of Potts & Sammut (2005) for incremental learning of linear model trees the splitting decision is formulated as hypothesis testing. The split least likely to occur under the null hypothesis of non-splitting is considered the best one. The linear models are computed using the RLS (Recursive Least Square) algorithm that has a complexity, which is quadratic in the dimensionality of the problem. This complexity is then multiplied with a user-defined number of possible splits per numerical attribute for which a separate pair of linear models is updated with each training example and evaluated.

In the second approach by Ikonomovska et al. (2010) the authors propose an incremental algorithm FIMT-DD for any-time model trees learning from evolving data streams with drift detection. The FIMT-DD algorithm is able to incrementally induce model trees by processing each example only once, in the order of their arrival. Splitting decisions are made using only a small sample of the data stream observed at each node, following the idea of Hoeffding trees.

3. On-line Regression/Model Trees with Options (ORTO)

We begin with a brief description of the ORTO algorithm and then discuss the relevant points in more detail. The pseudo code is given in Algorithm 1.

The algorithm starts with an empty leaf and reads examples from the stream in the order of their arrival. Each example is traversed to a leaf where the necessary statistics are maintained, such as $\sum y_k$, $\sum y_k^2$ per splitting point. For each numerical attribute we use an extended binary search tree, which enables sorting unique values on the fly as well as efficient one-pass computation of the variance reduction for every possible split point (Ikonomovska et al., 2010).

Newly arriving examples are passed down the corresponding branch of a decision node given the outcome of the splitting test. At an option node, newly arriving examples are passed down along all the options. The procedure repeats recursively until the end of the stream.

Algorithm 1 Pseudo code for the ORTO algorithm.

```
 1: ORTO ($n_{min}$)
 2: Begin with an empty Leaf (Root)
 3: repeat
 4:   Read the next example
 5:   Traverse the example to a Leaf
 6:   Update statistics in the Leaf
 7:   if (Examples seen in a Leaf = $n_{min}$) then
 8:     Find the best split per attribute
 9:     Rank the attributes
10:     if (!Ambiguity and !Prepruning) then
11:       Make a split on the best attribute
12:     else
13:       $k \leftarrow$ CountCompetitiveSplits(Leaf)
14:       Make $k$ splits
15:     end if
16:   end if
17: until End of the stream.
```

The most important parts of the algorithm are the criterion for introducing option nodes and the prediction rule, which we discuss in more detail below.

3.1. The criterion for evaluating ambiguity

Let A_1, A_2, A_3, …, A_d be the attribute ranking obtained at line 9 of Algorithm 1. Attributes are ranked according to the variance reduction estimated for the best possible splitting point for each attribute.

Let us further consider the ratio of the variance reduction $VR(\cdot)$ values of any attribute from the set A_2, A_3, …, A_d and the best one A_1 (e.g., $VR(A_2)/VR(A_1)$) as a real-valued random variable r with range $R \in [0, 1]$. The Hoeffding bound can be used to obtain high confidence intervals for the true average ($\overline{r}_{true}$), i.e., $\overline{r} - \varepsilon \leq \overline{r}_{true} < \overline{r} + \varepsilon$, where $\overline{r} = \frac{1}{n}\sum_{i=1}^{n} r_i$ and ε is the value of the Hoeffding bound, calculated as $\varepsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$. If, after observing n_{min} examples, the inequality $\overline{r} + \varepsilon < 1$ holds, then $\overline{r}_{true} < 1$, meaning that the best attribute observed over a portion of the stream is truly the best attribute over the entire stream.

The described procedure is the splitting criterion of the FIMT-DD algorithm (Ikonomovska et al., 2010). To allow the introduction of option nodes we make the following modification: if, after observing n_{min} examples, the inequality $\overline{r} + \varepsilon < 1$ holds, a normal split is performed, otherwise, an option node is introduced with splits on all the attributes (A_i) for which the inequality $VR(A_i)/VR(A_1) > 1 - \varepsilon$ is satisfied.

In other words, when inequality $\overline{r} + \varepsilon < 1$ does not hold, it means that the observed best attribute A_1 is approximately equally discriminative as the second best A_2 and will not necessarily remain the best if more examples are observed. In general, it competes with one or several other attributes that rank high enough. Therefore, instead of waiting for more evidence to be observed, we broaden the greedy search, soften the criterion and accept all competitive attributes as promising directions for the search procedure.

To choose a more sensible threshold that will not allow excessive tree growth, instead of a multiplicative option factor (Kohavi & Kunz, 1997) (where the options are the attributes that rate at least a multiplicative factor of say 0.7 of the $VR(A_1)$) we use the Hoeffding bound and make the following assumption: All the attributes for which the inequality $\overline{r} + \varepsilon < 1$ does not hold are approximately equally discriminative on the observed sample of data. The fact that this inequality does not hold expresses the lack of the necessary confidence for discarding those attributes as non-useful.

A possible shortcoming of the described criterion is an excessive growth of the tree. For this reason, it is crucial to have some form of a restriction on the number of options or the number of trees, which will control tree growth and the memory allocation. Kohavi & Kunz (1997) show that option nodes are most useful near the root of the tree. For that reason we follow their approach and reduce the number of options with the depth of the tree as $o = k \cdot y^{level}$, where k is the option factor equal to the number of competitive attributes chosen using the inequality $VR(A_i)/VR(A_1) > 1 - \varepsilon$, and y is a decay factor. The root is at level 0. Thus, o will in this case represent a decaying option factor, which will be used as an alternative to the basic algorithm (with option factor k).

Due to the use of a probability bound in the split selection process, incremental trees are less unstable than batch trees. However, our experience is that this approach introduces a significant delay in the growth of the tree, which in effect affects the accuracy and the speed of learning. With the modified splitting criterion, the tree is allowed to grow faster while at the same time, multiple options can overcome the one-step look-ahead problem of standard greedy search.

The prediction rule. Having option nodes in the tree structure results in multiple paths and thus multiple predictions for examples that pass through option nodes. The standard prediction rule for regression is to average all the possible predictions. The example is traversed to an option node where the average prediction will be computed from all of its options.

We have further examined an additional alternative by using the prediction from the currently *best tree*. This requires a definition of the notion of a best tree at a given time point t. To be able to obtain a fast prediction we assume that the best tree at time t is composed from the options that have the smallest weighted prequential mean-squared error $wPreqMSE(T_i,\ t)$ observed at time point t.

The weighted prequential error $wPreqMSE(T_i,\ t)$ is an improved variant of the prequential error originally proposed by Dawid (1984) and using a fading factor as proposed by Gama et al. (2009). This measure gives an estimate on the most recent accuracy of the trees. It is computed by an incremental test-then-train procedure for every alternative at an option node, i.e., every example is first used for testing and then for training. The error is accumulated at each time step.

The formula for computing the weighted[1] prequential mean squared error $wPreqMSE(T_i,\ t)$ of a tree T_i at a time point t is given below:

$$\begin{aligned} wPreqMSE(T_i, t) &= \frac{wPreqSE(T_i, t)}{wNum(T_i, t)} = \\ &= \frac{f \cdot wPreqSE(T_i, t-1) + SqErr(t)}{f \cdot wNum(T_i, t-1) + 1} \end{aligned} \tag{1}$$

where f is the fading factor, $wPreqSE(T_i,\ t)$ is the weighted prequential squared error and $wNum(T_i,\ t)$ is the weighted number of examples.

Whenever a new option node is introduced, the sums are initialized to $wPreqSE(T_i, 0) = 0$ and $wNum(T_i, 0) = 0$ for all the alternatives. The squared error $SqErr(t)$ is computed simply as:

$$SqErr(t) = (y_t - Pred(T_i, t))^2, \tag{2}$$

where y_t is the true value for the example seen at time point t and $Pred(T_i, t)$ is the prediction by the tree alternative i using the *BestTree* prediction rule.

To make a distinction between the two prediction rules of ORTO we will use the following acronyms: ORTO-A (by averaging), and ORTO-BT (the best tree constructed from the option tree).

[1]The term weighted is used due to the use of a fading factor f. Namely, if we recursively substitute the right hand-side member of the equation using the same formula, we will get the sum of weighted squared errors. The weight for the oldest one will be f^t. The value of f should be around 0.9997 (< 1).

4. Experimental Setup

The following section explains how we designed the experiments to evaluate the above described algorithm.

Evaluation methodology. The any-time accuracy is measured trough a test-then-train interleaved evaluation procedure in which we use the weighted prequential mean squared error (Equation 1). To get reliable estimates on the final mean squared error, we use the ten-fold cross-validation procedure for all of the stationary real-world datasets. We measure trees' complexity in terms of number of leaves, and option nodes (for ORTO-A and ORTO-BT). We further perform a comparison of all the algorithms in terms of the allocated memory (in MB) and the learning time (in seconds).

To obtain a bias-variance profile of the different methods, we use the standard bias-variance decomposition of the squared loss (Domingos, 2000). As proposed by Bouckaert (2008), we use the ten-fold cross-validation sampling procedure for estimating the decomposed mean squared error, which has been reported to provide the most reliable estimates. Instead of sampling from the folds, we use all the samples in order to obtain larger training sets.

For all of the performed experiments we set the common parameters to the following values. The confidence parameter is set to $\delta = 10^{-6}$ and $n_{min} = 200$ (the minimal number of examples before a split evaluation is done). The decay factor y is set to 0.9 according to the results in (Kohavi & Kunz, 1997).

Datasets. In order to get a clear picture on the improvement of the any-time accuracy and evaluate the speed-up obtained by using options we decided to use the nine largest available "benchmark" datasets from the UCI Machine Learning Repository (Frank & Asuncion, 2010), the Delve Repository[2] and the StatLib System's site[3], on which no concept drift has been reported.

5. Experimental results

In this section, we discuss the results obtained using the above evaluation methodology.

5.1. Evaluation of the any-time accuracy

In Figure 2 we show the weighted prequential mean squared error (upper plot) and the tree size (number of leaves, lower plot) computed for the first 3500 examples over 10 folds of the Pol dataset. The behavior of the algorithms is very similar for the rest of the datasets. The upper plot shows that the prequential mean squared error of the option trees, both with averaging and best tree prediction rules, is lower than that of the regular regression tree, especially in the beginning of the learning. This was the exact purpose of introducing the option nodes. This effect decreases as the learning proceeds, i.e., as the tree grows and improves its approximation of the regression surface. In the final stages of learning the errors of option trees ORTO-BT and ORTO-A are quite similar, while FIMT-DD regression trees have larger error. In general, for those problems for which the regression surface can be modeled using the available number of training examples, all three approaches give comparable errors at the end.

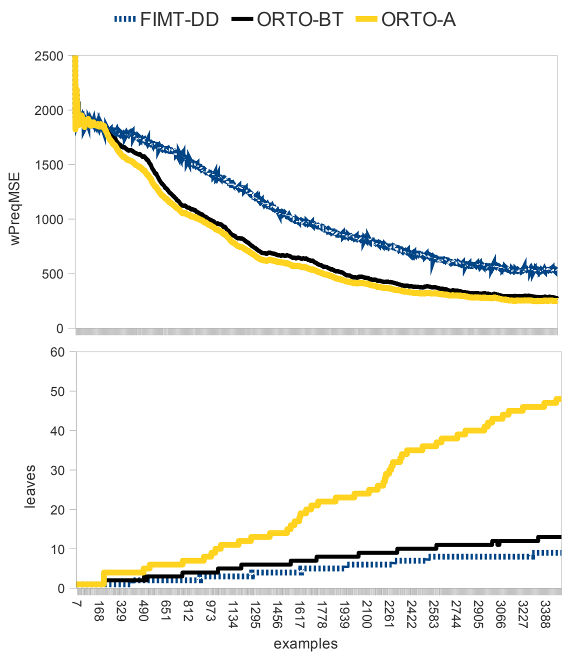

Figure 2. Weighted prequential mean squared error and number of leaves for FIMT-DD, ORTO-BT, and ORTO-A over the first 3500 examples of the Pol dataset.

Table 1 gives the results of a 10-fold cross-validation on the stationary real-world datasets. If we compare the tree of FIMT-DD and the best tree that can be constructed from the non-constrained option tree ORTO-BT, we can see that in 6 out of 10 datasets test ORTO-BT has a smaller mean squared error and more leaves, although this is not statistically significant by the Friedman statistical test with the Bonferroni-Dunn post-hoc test (at p=0.10). However, for those datasets

[2]http://www.cs.toronto.edu/ delve/data/datasets.html
[3]http://lib.stat.cmu.edu/index.php

Table 1. Results from 10-fold cross-validation on stationary real-world datasets. Comparison of FIMT-DD, ORTO-A, and ORTO-BT. The final MSE and size (number of leaves/rules) of models are given. Results for which ORTO-A has smaller MSE than FIMT-DD are given in red. Results for which ORTO-BT has smaller MSE than FIMT-DD are boldfaced.

	FIMT-DD		ORTO-A		ORTO-BT	
Dataset	MSE	Size	MSE	Size	MSE	Size
Abalone	7.45 ± 1.25	7.80	**6.27 ± 0.85**	255.0	**6.41 ± 0.79**	14.2
Cal Housing	5.09e+9 ± 0.42e+9	50.0	**4.69e+9 ± 0.26e+9**	810.0	**4.66e+9 ± 0.23e+9**	66.19
Elevators	25.0e-6 ± 2.0e-6	23.10	**18.0e-6 ± 1.0e-6**	850.7	**20.0e-6 ± 1.0e-6**	54.20
House 8L	1.24e+9 ± 0.16e+9	51.5	**1.23e+9 ± 0.14e+9**	964.29	1.26e+9 ± 0.14e+9	429.29
House 16H	1.68e+9 ± 0.22e+9	46.7	**1.60e+9 ± 0.17e+9**	927.09	**1.64e+9 ± 0.19e+9**	76.50
Mv Delve	3.66 ± 0.49	111.30	**2.31 ± 0.47**	505.89	**2.76 ± 0.27**	132.80
Pol	224.17 ± 105.72	27.40	**123.81 ± 19.24**	926.29	**140.08 ± 15.20**	38.09
Wind	45.44 ± 1.75	18.7	**44.51 ± 1.53**	104.59	45.02 ± 1.44	21.10
Winequality	0.59 ± 0.07	11.70	**0.57 ± 0.06**	147.6	0.59 ± 0.06	16.4

Table 2. Results from 10-fold cross-validation on stationary real-world datasets for ORTO-BT^5, and ORTO-BT^3. The final MSE, number of models and size (number of leaves/rules) of models are given.

	ORTO-BT^5			ORTO-BT^3		
Dataset	Models	MSE	Size	Models	MSE	Size
Abalone	63.20	6.49 ± 0.89	12.5	20.88	6.58 ± 0.97	11.7
Cal Housing	8.1	4.88e+9 ± 0.34e+9	57.29	3.2	4.89e+9 ± 0.37e+9	66.19
Elevators	94.7	19.9e-6 ± 1.29e-6	35.90	94.7	19.9e-6 ± 1.29e-6	35.90
House 8L	26.50	1.27e+9 ± 0.13e+9	59.20	14.4	1.27e+9 ± 0.14e+9	54.6
House 16H	36.50	1.64e+9 ± 0.16e+9	52.79	12.70	1.66e+9 ± 0.19e+9	47.70
Mv Delve	2.90	3.55 ± 0.39	112.80	2	3.68 ± 0.54	111.00
Pol	8.90	148.99 ± 18.72	35.50	5.2	158.38 ± 28.19	33.00
Wind	1.6	45.19 ± 1.57	19.00	1.6	45.43 ± 1.91	19.3
Winequality	15.00	0.60 ± 0.06	16.79	8.3	0.58 ± 0.06	14.2

the option nodes enable faster growing and efficient resolution of the tie-situations. As a result the learned trees are larger in size and more accurate.

Although an option tree can have a significantly larger number of leaves, if we transcribe it into a single ordinary tree by considering only the best options, such a tree will be of a comparable size as the ordinary tree.

A statistically significant reduction of error is also obtained by averaging the predictions. In this sense, ORTO-A smooths the crisp boundaries between the leaves and reduces the variance. To investigate and confirm this hypothesis we performed a bias-variance analysis of the error for all of the algorithms.

5.2. Bias-variance profile

In Figure 3 we show a separate comparison of the bias and the variance components of the error per dataset. The datasets are sorted by the size of the option trees in a decreasing order. To be able to make a comparison across all the datasets we normalized the error components with corresponding error components of the ordinary regression trees (FIMT-DD).

The results suggest that in comparison to regular regression trees, option trees with averaging mainly reduce the variance component of the error. However, on the left plot we can see that ORTO-A and especially ORTO-BT trees have smaller bias component of the error as well. Furthermore, the ORTO-BT trees (obtained by transcribing an option tree into a single ordinary tree by considering only the best options) have an increased variance component of the error for most of the datasets. This can be explained having in mind that the best options are evaluated incrementally using estimates of the squared error obtained over the training data observed so far. However, further analysis is required to examine the situation in more detail.

5.3. Analysis of memory and time requirements

A possible drawback of option trees is their size. Inspired by the previous work on option trees (Kohavi & Kunz, 1997) we tried to examine whether option nodes are most useful near the root of the tree. If that is true, by limiting the maximal level at which option nodes can be introduced, we can significantly reduce the size of the trees, the memory allocation, and at the

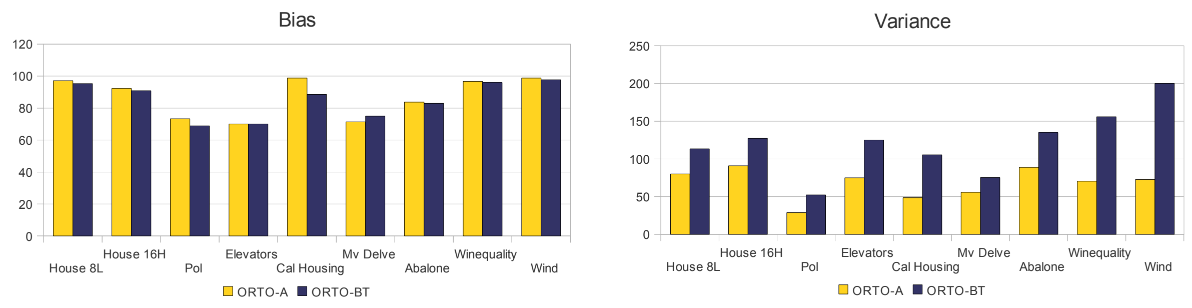

Figure 3. Bias-variance decomposition of the mean squared error of ORTO-BT, and ORTO-A relative to their corresponding counterparts for FIMT-DD.

same time not sacrifice much of their accuracy.

To evaluate this hypothesis we introduced a decaying option factor ($o = k \cdot y^{level}$) and a restriction on the level of the tree where an option node can be introduced, first to maximum 3 levels (ORTO-BT^3) and later to maximum 5 levels (ORTO-BT^5). The results from the second part of the evaluation show that while the size is substantially reduced, error increased only slightly. By constraining the maximum level to 3, the option trees were too restricted and rarely ever managed to induce more than 5 models on average (number of different trees compressed in the option tree). For that reason we increased the maximum level to 5.

In Table 2 we present the results from the 10-fold cross-validation on stationary real-world datasets for ORTO-BT^5 and ORTO-BT^3. The results confirmed that, by decaying the number of options with the tree depth and using a restriction on the maximum level at which option trees can be introduced, we can significantly reduce the memory allocated while not suffering a significant increase of the error.

We also performed the same type of bias-variance analysis of the error for the constrained versions of the option trees. The bias component of the error does not increase or increases only slightly, but remains smaller than for the FIMT-DD trees. On the other hand, the best tree transcribed from the constrained option tree has smaller variance than the best tree from the non-constrained option tree for some of the datasets.

In Table 3 we present the results on memory allocation (in MB) for all analyzed algorithms. The required memory depends on the number of leaves where the sufficient statistics are being maintained, thus with a decaying option factor the savings in memory are significant, for some datasets in an order of magnitude.

The option tree has an increased processing time per example due to the fact that each record from the stream is used to update all the leaves in which it may end up, thus all the options need to be visited in a sequential manner. Therefore, the relative increase of processing time depends on the number of options nodes and their distribution in the tree structure. Maintaining smaller trees with options only at the higher levels reduces the update time and saves memory. On average, the processing time per example of ORTO is 10 times higher than of FIRT-DD.

Table 3. Comparison of FIMT-DD, ORTO (basic version), ORTO with a decaying factor and restricted to maximum 3 levels (ORTO^3), and 5 levels (ORTO^5) on memory (MB).

Data	FIMT-	ORTO	ORTO^3	ORTO^5
Abalone	4.66	49.08	20.81	29.72
CalHous.	25.92	296.64	45.02	54.77
Elevat.	8.70	369.31	138.62	138.62
House8L	25.74	451.27	197.17	249.65
House16H	55.49	684.90	287.86	477.14
MvDelve	59.77	219.84	99.93	106.71
Pol	7.52	375.36	44.76	50.39
Wind	13.88	60.59	22.31	22.62
Wineq.	3.75	38.43	15.38	20.53

5.4. Linear models in the leaves

The above presented results correspond to the trees in which no linear models were allowed in the leaves. We placed the focus on regression trees instead of model trees in order to obtain a clear evaluation of the advantages and disadvantages from introducing options, and without the effect from the incremental process of computing linear models in the leaves of the tree. Although the criterion for introducing options does not depend in any way on the presence of linear models, we performed the same type of analyses described above also for model trees. The results and conclusions are in general very similar to the ones presented above on regression trees.

6. Conclusions

In this paper, we propose a novel method for on-line learning regression trees with option nodes from data streams. To the best of our knowledge, option trees have not been studied in the context of regression, neither in the batch nor in the on-line setting.

Our method for learning Hoeffding-based option trees for regression addresses the problem of instability of tree learning, commonly seen in the case of highly correlated or equally discriminative attributes, i.e., in tie situations. Hoeffding trees can suffer from a delay in the learning process in such tie situations, because they assume that the data is in abundance and will never stop to stream in: Decisions on split selection are postponed resulting in lower learning rates.

We show that option nodes are a natural and effective solution to the problem of dealing with multiple equally discriminative attributes (the tie problem). The additional structure of the option trees provides interesting and useful information on the ambiguity of the splits and thus on the existence of several equally relevant attributes.

In future work, we would like to perform a more thorough analysis of the overall performance of the proposed system, including a comparison with ensemble methods, such as on-line bagging and random forests methods for regression. The latter would be of special interest since the option trees with averaging can be viewed also as an ensemble method. Finally, some of the ideas explored in this paper would be also applicable to Hoeffding trees for classification: Since they have not been applied in that context, they deserve further investigation.

Acknowledgments

This research work is supported by the grant Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA/098355/2008), the grant PHAGOSYS by the European Commission, and the Slovenian Research Agency.

References

Bouckaert, R. R. Practical bias variance decomposition. In *Proc. 21st Australasian Joint Conf. on Artificial Intelligence*, pp. 247–257. Springer, 2008.

Buntine, W. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.

Dawid, P. Statistical theory: The prequential approach (with discussion). *Journal of Royal Statistical Society A*, 147:278–292, 1984.

Domingos, P. A unified bias-variance decomposition for zero-one and squared loss. In *Proc. 17th National Conf. on Artificial Intelligence and 12th Conf. on Innovative Applications of Artificial Intelligence*, pp. 564–569. AAAI Press/The MIT Press, 2000.

Domingos, P. and Hulten, G. Mining high-speed data streams. In *Proc. 6th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 71–80, 2000.

Frank, A. and Asuncion, A. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

Gama, J., Rocha, R., and Medas, P. Accurate decision trees for mining high-speed data streams. In *Proc. of 9th Int. Conf. on Knowledge Discovery and Data Mining*, pp. 523–528. ACM, 2003.

Gama, J., Medas, P., and Rocha, R. Forest trees for on-line data. In *Proc. 2004 ACM Symposium on Applied Computing*, pp. 632–636. ACM, 2004.

Gama, J., Sebastião, R., and Rodrigues, P. P. Issues in evaluation of stream learning algorithms. In *Proc. 15th Int. Conf. on Knowledge Discovery and Data Mining*, pp. 329–338. ACM, 2009.

Hulten, G., Spencer, L., and Domingos, P. Mining time-changing data streams. In *Proc. 7th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 97–106. ACM, 2001.

Ikonomovska, E., Gama, J., and Džeroski, S. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, pp. 1–41, 2010.

Kohavi, R. and Kunz, C. Option decision trees with majority votes. In *Proc. 14th Intl. Conf. on Machine Learning*, pp. 161–169. Morgan Kaufmann Publishers Inc., 1997.

Liu, J., Li, X., and Zhong, W. Ambiguous decision trees for mining concept-drifting data streams. *Pattern Recognition Letters*, 30(15):1347–1355, 2009.

Pfahringer, B., Holmes, G., and Kirkby, R. New options for hoeffding trees. In *Proc. 20th Australian Joint Conf. on Artificial Intelligence*, pp. 90–99. Springer, 2007.

Potts, D. and Sammut, C. Incremental learning of linear model trees. *Machine Learning*, 61(1-3):5–48, 2005.