# Application of On-line Regression Tree Induction to Forecast Traffic Flows

Marlies Vanhulsel, Davy Janssens, Koen Vanhoof, and Geert Wets

Transportation Research Institute, Hasselt University,
Wetenschapspark 5 bus 6, B-3590 Diepenbeek, Belgium
{marlies.vanhulsel, davy.janssens, koen.vanhoof, geert.wets}@uhasselt.be

**Abstract.** Modelling efforts aiming at predicting traffic flows accurately, should be capable of handling a continuous stream of data while being able to account for structural changes. To this end, the current research proposes three regression tree induction algorithms, including a batch, incremental and hybrid batch/incremental technique. These algorithms are discussed, implemented and evaluated based on their predictive power and computational requirements. The incremental and batch/incremental algorithms prove to be particularly suited as on-line learning approaches, whereas the batch algorithm needs to be adapted in order to handle a continuous stream of data. The outcomes of the batch/incremental and batch algorithms are comparable and these algorithms also prove to forecast traffic flows better than a selected baseline model. Additionally, the incremental and batch/incremental approaches show to consume considerably less memory capacity and computational time compared to the adapted batch algorithm.
**Key words:** regression tree, on-line learning, incremental regression tree induction, traffic flow forecasting

## 1 Introduction

Travel information systems, such as intelligent transportation systems (ITS) and advanced travellers information systems (ATIS), are used to supply information to road users. To this end, these systems need to provide a forecast of the travel conditions at the moment the user plans to access a particular road segment. For that purpose, travel times can be derived from the future demand on the network and the current and future state of this network. [1] The reliability of travel information systems is thus to a large extent determined by the accuracy of the underlying predictions. In this area of research, a number of modelling efforts has already been proposed. For instance, Smith and Demetsky [2] have compared four techniques, covering historical average, time-series, neural network and nonparametric regression. The current research focuses on estimating the future number of vehicles on a selected stretch of road based on historical data recorded in traffic counts by means of regression tree induction.

In order to accurately predict these future traffic counts per minute, the selected modelling technique needs to satisfy a number of conditions. First, because traffic sensors continuously record the number of vehicles on a specific stretch of road, the technique considered here has to be able to *handle a continuous stream of data*. In addition, the technique has to be *responsive to structural changes*, such as a long-run decrease in the number of vehicles on a particular road segment as a result of a policy, for instance the introduction of a road toll to use a particular stretch of road or the adoption of high occupancy vehicle lanes.

Within this context, the current research aims at exploring the applicability of a decision tree induction algorithm. For this purpose, three regression tree induction algorithms are presented, evaluated and compared based on their compliance to the above defined conditions, predictive performance and computational requirements. The first algorithm is a batch regression tree algorithm and the second is an incremental learning algorithm; both algorithms are introduced by Potts and Sammut[3]. The third algorithm consists of a combination of the batch and the incremental regression tree induction technique which meets the shortcomings of both founding algorithms. Additionally, the batch regression tree algorithm is adapted so as to enable handling an on-line data stream.

Section 2 will introduce the reader into the regression tree induction algorithms considered here. Section 3 will specify the criteria used to evaluate these approaches and discuss the data used. This section will also present the implementation and results of a case study to prove the applicability, suitability and performance of the proposed techniques within the current research problem. Section 4 will summarize the major findings of this study.

## 2 Methodology

### 2.1 Batch Regression Tree Algorithm

Potts and Sammut [3] have founded the regression tree induction algorithms applied here. The present paper will only provide a brief overview of this algorithm. A more detailed description of this algorithm can be found in [3].

The batch regression tree initially consists of a root node containing all available training instances. The regression tree is then refined by recursively adding binary splits according to a predefined splitting criterion. In the current research, this criterion is based on the distribution of the residuals (RA) of the observations assigned to the node into consideration. These residuals are calculated by taking the difference between the actual value and the predicted value, which equals the average value of the dependent variable of all examples included in the node. According to these residuals, the algorithm divides the $N$ instances of this node in two subsets: the $N^+$ examples with non-negative residuals belong to $S^+$ and the $N^-$ examples with negative residuals belong to $S^-$ $(N = N^+ + N^-)$. For each dependent variable $j$ following statistic is calculated:

$$T_j^{(1)} = \frac{\overline{x}_j^+ - \overline{x}_j^-}{s_j\sqrt{\frac{1}{N^+} + \frac{1}{N^-}}}$$

Where $\overline{x}_j^+$ and $\overline{x}_j^-$ are the mean of variable $j$ in subset $S^+$ and subset $S^-$ respectively and $s_j$ is the pooled variance of this variable over both subsets. $T_j^{(1)}$ tests for difference in means.

Subsequently, the absolute differences $z_{ij}^+ = \left|x_{ij} - \overline{x}_j^+\right|$ for $i \in S^+$ and $z_{ij}^- = \left|x_{ij} - \overline{x}_j^-\right|$ for $i \in S^-$, are determined. Based on these $z$-values $T_j^{(2)}$ is defined, which tests for a difference in variances:

$$T_j^{(2)} = \frac{\overline{z}_j^+ - \overline{z}_j^-}{w_j \sqrt{\frac{1}{N^+} + \frac{1}{N^-}}}$$

Where $\overline{z}_j^+$ and $\overline{z}_j^-$ are the mean of variable $j$ in subset $S^+$ and subset $S^-$ respectively and $w_j$ the pooled variance of $z$-vales over both subsets. Having calculated both $T$-values for all attributes, the split attribute is defined to be the variable corresponding to $T = \max_{j,n} \left|T_j^{(n)}\right|$. The corresponding attribute test value is determined as the average of the means $\overline{x}_j^+$ and $\overline{x}_j^-$.

Additionally, the probability $\alpha$ where $|t| > T$ is retrieved from the Student's $t$ distribution and compared to a predefined threshold to check whether the proposed split is meaningful. Moreover, the growth of the tree is limited by estimating the contribution of the split to the overall tree accuracy by calculating:

$$\delta = \frac{1}{s_{root}^2} \left( \frac{RSS}{N - d} - \frac{RSS_L + RSS_R}{N_L + N_R - d} \right)$$

Where $RSS$ is the residual sum of squares and $d$ is the number of dimensions. This value is also put against a predefined threshold.

Yet, the current configuration of the algorithm is fitted off-line, which implies that the algorithm is not able of handling a continuous stream of data. Therefore, the regression tree induced by means of the batch regression tree algorithm has to be re-estimated from time to time, based on the available data. However, the number of instances rises rapidly, causing both the amount of memory required to store the training instances and the time needed to estimate the regression tree to increase. Consequently, aiming at reducing memory requirements and speeding up the regression tree induction, only a fixed number of the most recently encountered instances will be used to fit the regression tree. This process is illustrated in Figure 1. For instance, the regression tree is re-estimated every week based on the data collected during the 100 preceding days. The most recently fitted regression tree is then applied for prediction until the next re-estimation of the regression tree.

Both parameters (frequency of re-estimation and number of training instances used) need to be considered very carefully. First, re-fitting the regression tree more frequently increases the ability to adjust for changes, but increases the burden of updating the regression tree. Next, using more data to train the regression tree increases the accuracy and enables the regression tree to level out temporary fluctuations. However, including training instances generated a long time ago restrict the responsiveness of the algorithm to structural changes

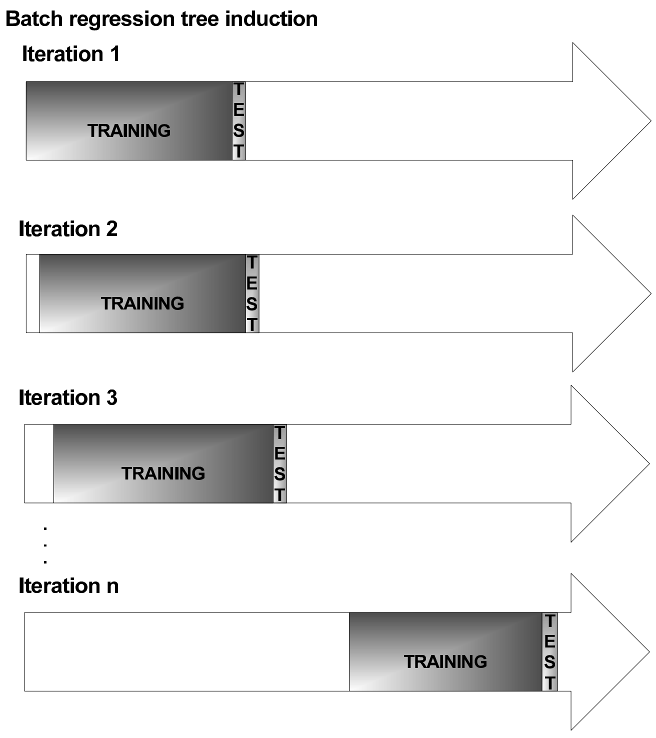

**Fig. 1.** Adjustment of the batch regression tree algorithm

because the information recorded by these 'old' instances may be outdated and could thus interfere with the prediction of recent trends. This issue can also be tackled by decreasing the importance of older instances by assigning weights to the instances based on their age.

However, both the memory capacity needed to store a large amount of instances and the computational time needed to estimate a regression tree grow as the number of instances increases. In the current research, two combinations of these parameters will be evaluated. To obtain the on-line batch regression tree induction algorithm described above, the regression tree is re-fitted every week based on data of the 100 preceding days (on-line 100d, every 7d) and based on data of the 365 preceding days (on-line 365d, every 7d).

The adjustment of the batch regression tree induction algorithm includes one major disadvantage: the computational time required to fit the batch regression tree is needed each time the regression tree is re-estimated forcing the total computational time to rise considerably as will be illustrated in section 3.4.

## 2.2 Incremental Regression Tree Algorithm

The drawbacks of the batch induction algorithms described in the previous paragraph can also be tackled by applying either an on-line regression tree induction algorithm [3] or a hybrid form combining the batch regression tree algorithm with an incremental one.

The main idea behind the incremental learning process is similar to the batch regression tree approach discussed in the previous paragraph. In essence, these algorithms differ in the fact that the incremental algorithm starts with an empty tree and updates the regression tree on each encounter with a new instance. To this end, the incremental approach first updates the statistics within each node of the regression tree that is affected by the training instance into consideration. If necessary, the algorithm can alter the existing tree by either creating an additional split or pruning a number of existing leaves. [3] The latter action allows the algorithm to correct previously defined splits which are no longer significant based on the most recently observed instances. Subsequently, the training instance can be discarded. As a result, it is not required to store the training instances encountered during the learning process. Consequently, the incremental approach allows saving memory space.

In addition to this advantage, an intrinsic feature of the incremental algorithm consists of the ability of handling a continuous stream of data without having to re-train the regression tree from scratch when a new instance becomes available. However, the gradual updating of the regression tree implies that the sequence in which the training instances are processed, influences the structure of the regression tree to a large extent. This causes the incremental regression tree to reveal a different structure compared to the batch regression tree based on the same dataset, but drawn up examining a large number of training instances at once.

## 2.3 Batch/Incremental Regression Tree Algorithm

The authors also propose a hybrid batch/incremental regression tree induction method combining both algorithms, which allows merging the advantages of both approaches. To start with, the batch regression tree induction algorithm is applied to fit a regression tree based on $n$ available instances at a certain moment in time. In the present research, all instances collected during the first 365 days are used to estimate the initial regression tree. Thereafter, the resulting regression tree is updated by applying the incremental induction algorithm when a new instance enters the system. This approach allows formulating an initial regression tree, which already captures the most significant splits deduced from the data available for the batch algorithm. The existence of such structure facilitates the updating process within the incremental learning process. Because of this, it is assumed that the batch/incremental algorithm will perform better than the incremental approach.

As is the case for the incremental approach, the batch/incremental algorithm discards the training instances after the initial batch training and the subsequent updating of the regression tree. The major advantage of this true on-line approach with regard to the adjusted batch algorithm includes the reduced requirements for memory capacity, next to a decrease in the computational time needed to update the trees compared to the time needed to fit the regression tree from scratch.

## 3 Case Study

### 3.1 Evaluation Criteria

First, to prove the applicability of these algorithms within the research area, the three regression tree induction approaches are evaluated against a number of requirements. As already mentioned in the introduction, the techniques proposed here need to be able to handle a continuous stream of data and be responsive to structural changes within the data.

Next, the performance of the algorithms is assessed and compared. To this end, the regression trees are used to obtain traffic count forecasts, which are compared to the corresponding observed values. In case of the batch regression tree re-fitted every $d$ days based on the $n$ preceding days, the forecasts are generated as follows: to start with, the regression tree is fitted based on data of the first $n$ days. Then, the regression tree is applied to estimate traffic counts per minute of the following $d$ days, until the regression tree needs to be re-fitted. Subsequently, this new regression tree is used to predict traffic counts per minute of the next $d$ days, until the regression tree is re-fitted again.

In case of the incremental regression tree algorithm, the regression tree is updated every minute, immediately after the traffic count becomes available within the system. The continuously updated regression tree can thus be applied every minute to generate the predicted traffic count for the following minute. However, in the current case study is assumed that the forecast is needed for a stretch of road which will be reached ten minutes in the future. This implies that the traffic count predictions per minute are based on all data available up to ten minutes before the time slot into consideration.

Furthermore, in case of the batch/incremental regression tree algorithm, the regression tree is fitted off-line first and updated on-line thereafter, as is the case for the incremental algorithm. From that time, the batch/incremental regression tree provides traffic count estimates per minute in the same way as the incremental algorithm, also taking into consideration a prediction horizon of ten minutes in the future.

In order to be able to assess the performance of the algorithms, a baseline model is defined. This model also takes into account a prediction horizon of ten minutes: at time $t$ the baseline model assigns a predicted value for time $t + 10$, which equals the most recently observed value up to that time. The predictive power of the algorithms is put against the predictions of the baseline model based on the sum of the squared errors of the predicted value compared to the actual value (SSE) and the average relative error (ARE), which is calculated by dividing the absolute difference of the observed count and the estimate by the observed value.

Moreover, because the amount of time required to estimate or update the regression tree determines the applicability of the algorithm within this domain, the computational time of the algorithms are also recorded and compared. To this end, the algorithms are implemented on a computer equipped with an Intel Core Dual processor (1.80 G Hz and 1.79 G Hz) and 2.0 GB of RAM.

## 3.2 Data

The three algorithms described above are applied to estimate future travel counts on one particular road segment. The data used for this case study include 1,429,933 instances and stem from traffic counts gathered by the Flemish Traffic Control Centre (Vlaams Verkeercentrum) in the course of 2003, 2004 and 2005. Traffic cameras and loop detectors record traffic counts on approximately 600 locations scattered on an area of 13,500 km$^2$. The road segment analysed here compromises of two lanes, each of them monitored by means of a separate loop detector. The dependent variable is the traffic count per time slot, which is composed of the sum of the measurements on both lanes. The time series contains a number of cycles: the data is subject to a daily cycle, a monthly cycle and a (rather weak) yearly cycle. For the purpose of accounting for structural changes, the authors expect that the proposed techniques are the most appropriate ones for this type of problem.

In order to fit a regression tree to forecast traffic counts per minute, the regression algorithm needs a number of independent variables. In this case study, next to the time of the day expressed in minutes elapsed since midnight and the weekday, three dummy variables are included, which indicate whether or not the traffic counts were recorded on a day within a school holiday, a public holiday or a weekday (vs. weekend day). These variables are assumed to be able to capture the cyclic effects within the data. Furthermore, for the purpose of illustrating the applicability of the techniques, only these variables are considered, but it is obvious that more variables, such traffic announcements, can be taken into consideration as well.

## 3.3 Structural Changes

In order to be able to test the responsiveness of the algorithm to structural changes, the data are transformed artificially to replicate a structural change. Suppose for instance that a policy has been implemented from January 1st, 2005, imposing a congestion charge between 6am and 10 am and 4 pm and 7 pm. It is assumed here that this policy impacts the number of vehicles on the road segment into consideration as follows: the traffic counts per minute drop on average by 25 percent during above defined time slots and by ten percent during the gap in between these peak moments (10 am to 4 pm). No changes are assumed to occur before 6 am and after 7 pm. To replicate these changes, the observed values are multiplied by factors randomly drawn from a normal distribution with an average of 0.75 (i.e. decrease of 25 percent) and 0.90 (i.e. decrease of 10 percent) respectively and a standard deviation of 0.025. The responsiveness of the algorithms to structural will be examined by taking a closer look at the structure of the resulting regression trees fitted on the data without and with the proposed changes.

### 3.4 Results

**Predictive Power** The highlights of the results based on the data including changes due to the policy are summarized in Table 1. First, both evaluation criteria, average SSE and average relative error, indicate that the accuracy of the batch algorithms and the batch/incremental algorithm lie within the same order of magnitude, and is superior to the accuracy of the incremental approach. The weak results of the incremental approach are due to the fact that the sequence in which the instances are processed influence the structure of the regression tree to a large extent. Especially in the current research, in which the data exhibits multiple cycles, the incremental algorithm produces a different tree than the batch and batch/incremental algorithms. After all, as the data enter the system minute by minute, the incremental algorithm processes the daily cycle and the weekly cycle differently than the batch component which considers all available data at once.

Moreover, when comparing the results of the regression tree approaches to the baseline model defined previously, the batch and the batch/incremental algorithms predict better than the baseline model. Furthermore, when observing the accuracy of the batch algorithms, one will note that the amount of training data influences the accuracy only to a limited degree.

Next, Figure 2 shows the evolution of the cumulative relative error for the regression trees based on the data with the changes due to introduction of the policy. All regression tree induction algorithms start with a high average error, but prove to converge within a considerable amount of time. Table 2 compares the proposed algorithms based on the speed of convergence of the relative error. In this table, the time needed to converge represents the amount of time that the average cumulative relative error fluctuates for more than 1%. As the table shows, the time to convergence is the highest in the case of the incremental algorithm, which reflects the initial setup time of the incremental regression tree algorithm. The batch algorithm estimating the regression tree on data of 100 preceding days requires the least time to converge. With respect to the height of the average of the cumulative relative error, the batch/incremental approach and the batch algorithm based on 365 preceding days produce the highest maximum, whereas the incremental approach produces the smallest maximum cumulative relative error.

Figure 3 reflects the evolution of the cumulative relative error at the time of the introduction of a structural change into the data at January 1st, 2005. This figure reveals that all algorithms suffer only a minor increase in the average of the cumulative error.

**Computational Requirements** Bearing in mind that the analysis in the current research provides forecasts for only one road segment, and that applications

|  | Batch: on-line 100d, every 7d | Batch: on-line 365d, every 7d | Batch/ Incremental | Incremental |
|---|---|---|---|---|
| Number of times the model is re-estimated/ updated | 143 | 105 | 914,930 | 1,429,933 |
| Learning time (s) |  |  |  |  |
| Average | 7.541 | 36.808 | 40.844 (B) +0.000 (I) | 0.000 |
| Minimum | 5.484 | 31.672 | 40.844 (B) +0.000 (I) | 0.000 |
| Maximum | 11.438 | 43.765 | 40.844 (B) +0.016 (I) | 0.046 |
| Average SSE* | 49.676 | 50.568 | 42.005 | 90.919 |
| ARE† | 32.655% | 33.314% | 30.040% | 40.604% |

*Average SSE of baseline model is 70.301.
†Average Relative Error of baseline model is 39.082%.

**Table 1.** Prediction results of regression trees

using such forecasts, such as ITS and ATIS, require predictions for each road segment, the authors would like to draw special attention to the algorithm's computational requirements. Firstly, with respect to the memory requirements, for the batch algorithm the instances need to be stored and need to be processed at the same time. To give an impression of the amount of memory that can be saved: in case of the fitting the batch regression tree, the size of the input file containing all instances for the road segment into consideration is more than 58MB. In case of the fitting or updating the incremental and batch/incremental trees, each record is handled separately and can be discarded afterward. The file containing only one record consumes less than 1kB.

Secondly, the computational times in Table 1 attract the attention. While the times needed to update the incremental and batch/incremental regression trees are almost negligible, the average computational time of the batch approach is rather high. Furthermore, as these computational times are required every time the regression tree is re-estimated, the total time needed by the batch algorithm

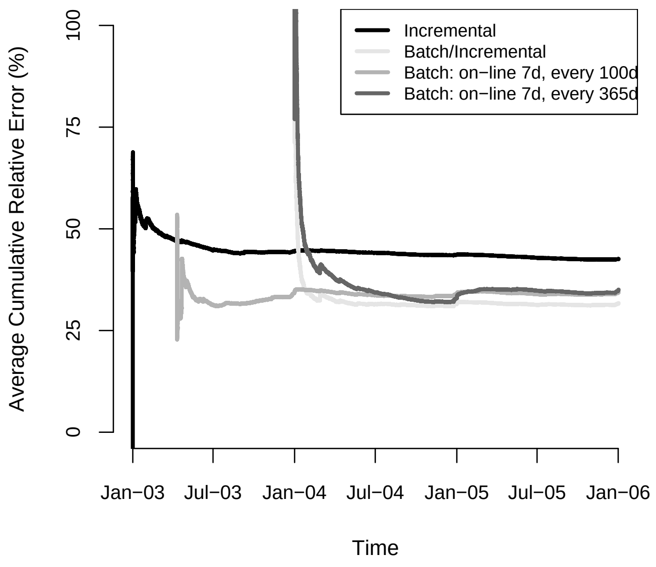

**Fig. 2.** Evolution of the average cumulative relative error

increases rapidly. For instance, the algorithm requires approximately 1,078.363 s (= 143 x 7.541 s) in total in the case the regression tree is re-fitted every week based on data of 100 preceding days or 3,864.840 s (= 105 x 36.808 s) in the case the regression tree is re-estimated every week based on data of 365 preceding days. On the contrary, the incremental approach only takes maximum 142.933 s (= 1,429,933 x 0.0001 s) and the batch/incremental approach 132.337 s (= 40.844 s + 914,930 x 0.0001 s). Both true on-line algorithms are thus considerably faster compared to the batch approach. In addition, the batch/incremental approach does not lose on accuracy.

| | Batch: on-line 100d, every 7d | Batch: on-line 365d, every 7d | Batch/ Incremental | Incremental |
|---|---|---|---|---|
| Maximum height | 53.490% | 420.979% | 420.979% | 68.829% |
| Time needed to converge | 106,035 | 356,229 | 178,711 | 303,597 |

**Table 2.** Convergence of cumulative relative error

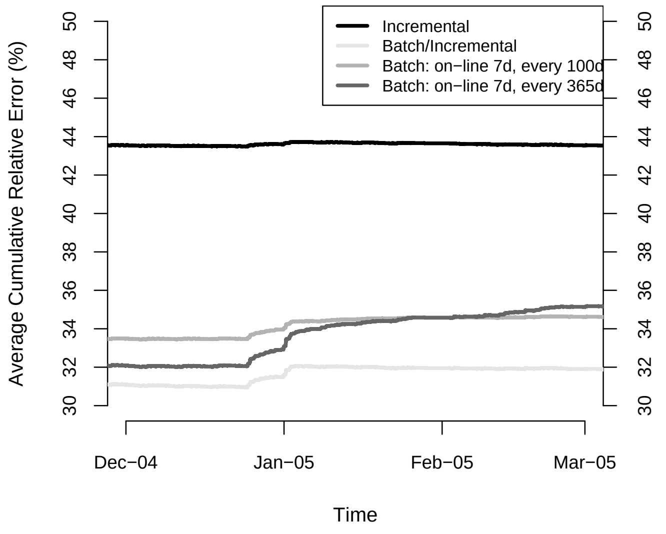

**Fig. 3.** Evolution of the average cumulative relative error between December 1st, 2004 and March 1st, 2005

## 4 Conclusions

The goal of the current research consisted of testing the applicability of a selected regression tree induction algorithm to predict traffic counts per minute on a particular road segment based on a number of explanatory variables (time of the day, weekday and three dummy variables to mark school holidays, public holidays and weekdays vs. weekend days). Three versions of the regression tree induction algorithm were proposed and revised within this application: a batch, incremental and hybrid (batch/incremental) algorithm were compared based on their characteristics, predictive power and required computational time. In order to enable processing an uninterrupted stream of data, the batch induction approach was adapted. The incremental and batch/incremental algorithms did not need such alterations, as the way they fit the regression tree model implies that these algorithms are able to handle a continuous stream of data.

The results have shown that the batch/incremental and batch algorithms performed equally well, whereas the predictive performance of the incremental approach limps behind. Additionally, when considering the computational and memory requirements, the batch/incremental approach has proven to be superior due to its lower total computational time and due to its reduced memory requirements as the training instances do not need to be stored.

Moreover, the outcome of this research has also indicated that the approaches are able to respond to structural changes which were simulated artificially within the available data. However, further research is required to define and quantify the ability of the algorithms to respond to structural changes.

## References

1. Nikovski, D., Nishiuma, N., Goto, Y., Kumazawa, H.: Univariate short-term prediction of road travel times. In: 2005 IEEE Intelligent Transportation Systems Conference, Vienna, Austria (2005) 1074–1079
2. Smith, B., Demetsky, M.: Traffic flow forecasting: comparison of modeling approaches. Journal of Transportation Engineering **123**(4) (1997) 261–266
3. Potts, D., Sammut, C.: Incremental learning of linear model trees. Machine Learning **61**(1–3) (2005) 5–48