An Algorithm for Anticipating Future Decision Trees from Concept-Drifting Data

Mirko Böttcher and Martin Spott and Rudolf Kruse

Abstract Concept-Drift is an important topic in practical data mining, since it is reality in most business applications. Whenever a mining model is used in an application it is already outdated since the world has changed since the model induction. The solution is to predict the drift of a model and derive a future model based on such a prediction. One way would be to simulate future data and derive a model from it, but this is typically not feasible. Instead we suggest to predict the values of the measures that drive model induction. In particular, we propose to predict the future values of attribute selection measures and class label distribution for the induction of decision trees. We give an example of how concept drift is reflected in the trend of these measures and that the resulting decision trees perform considerably better than the ones produced by existing approaches.

1 Introduction

The induction of decision trees is a relatively mature and well-researched topic. Aiming at an increased classification accuracy many algorithms which emphasise on different aspects of decision tree learning have been proposed and proven to be successful in many industrial and business applications. Many of these algorithms have been developed assuming that the samples used for learning are randomly drawn

Mirko Böttcher
University of Magdeburg, Faculty of Computer Science, 39106 Magdeburg, Germany, e-mail: miboettc@iws.cs.uni-magdeburg.de

Martin Spott
Intelligent Systems Research Centre, BT Group plc, Adastral Park, Ipswich IP5 3RE, United Kingdom, e-mail: martin.spott@bt.com

Rudolf Kruse
University of Magdeburg, Faculty of Computer Science, 39106 Magdeburg, Germany, e-mail: kruse@iws.cs.uni-magdeburg.de

from a stationary distribution. This assumption does not hold when dealing with real world data, because they are almost always collected over long periods of time and the data generating processes are almost always very complex. Many real world data sets are exposed to changes in their data generating process and hence also show changes in their underlying hidden structure. This phenomenon is referred to as concept drift in the literature.

In the first place decision trees are used in many applications to learn a classification model for a certain target attribute and, secondly, to get a deeper understanding of a domain by interpreting the tree as a set of rules. However, the presence of concept drift imposes two problems on these tasks. With regard to the first task, historic data used for learning is usually drawn from a different distribution than the future samples to be classified. Consequently, the relations of the decriptive attributes to the target attribute described by the learned decision tree are different from the ones which are present in current and future data. As a result the classification accuracy will never be optimal. With regard to the second task, the gained knowledge about a domain is only valid for the past but it yields no knowledge about the present or future when concept drift is present. Although obtaining such prospective knowledge is crucial for many businesses, the problem has received significant less attention in publications.

Several methods have been published to learn decision trees in the presence of concept drift [19, 8, 10]. They all use some form of incremental learning which aims to efficiently learn or maintain models such that they are always based on the most recent samples. However, those methods only account for the first of the aforementioned problems. Moreover, their major drawback is that the learned models will only yield good results if concept drift happens very slowly. The faster the distribution changes, maybe only in some subspaces, the less accurate the decision trees get.

Our aim is to solve the two problems outlined above. In this paper we will introduce the *PreDeT* algorithm which anticipates future decision trees from concept drifting data. *PreDeT* does not track or model distribution shift as such. Instead, it models how distribution shift affects those measures which control the process of decision tree induction and predicts their future values. For this reason it is able to cope even with fast changing domains. The decision trees learned by *PreDeT* can be seen as a domain's projection into the future. Therefore they help a user to obtain insight into the data's likely hidden structure in the near future.

The paper is organized as follows. In Section 2 we present related work from the fields of concept drift. Some background on decision tree induction will be given in Section 3. The *PreDeT* algorithm will be explained in Section 4 and some preliminary experimental results shown in Section 5.

2 Related Work

As already pointed out in the previous section, several methods have been published for learning models in the presence of concept drift [19, 8, 10]. In the following we will give some more details about the methods itself and outline their drawbacks.

The two basic techniques employed are *moving temporal windows* [19, 8] and *age dependent weighting* [10]. The method of moving temporal windows learns the decision tree from samples that were gathered within a certain, recent time window. For instance, in [19] a framework is described which heuristically and dynamically adapts the window size during the learn process. A moving temporal window approach to learn decision trees – the CVFDT algorithm – which scales up to very-large databases was proposed in [8]. It maintains class counts in each tree node and when new data arrives decides whether or not a subtree needs to be re-learned.

For window-based approaches the choice of an appropriate window size is a crucial and difficult problem. In general, a compromise has to be found between small windows, which are required for a fast adaptation to concept drift, and large windows, which are required for a good generalization. In [12, 6] upper bounds on the speed of concept drift are determined which are acceptable to learn a concept with a fixed minimum accuracy. Hence a window with a certain minimal fixed size allows to learn concepts for which the speed of drift does not exceed a certain limit. Taking all of the aforesaid into account this means that window based approaches – independent from whether they use a fixed or adaptive window size – perform well in domains with slow concept drift but may result in models with a low accuracy in very dynamic domains.

Age dependent weighting simulates a kind of a data-ageing process by crediting more recent samples higher than old ones in the learning process. In [10] methods for age dependent weighting are shown and compared with temporal window approaches. The weights are chosen such that learning emphasises for slowly changing domains on a suitable large set of samples and fast changing domains on only the most recent samples. This, however, leads to the same problem as for window-based approaches. The speed of concept drift may change considerably amongst subspaces for which a global sample weighting scheme obviously does not account for.

3 Decision Trees

In the following discussion we will use some notations that will be introduced in this section. As already stated above, we assume that a dataset S of sample cases is described by a set of nominal input attributes $\{A^{(1)}, \ldots, A^{(m)}\}$ and a class attribute C. We assume that the domain of attribute A has n_A values, i.e. $\mathrm{dom}(A) = \{a_1, \ldots, a_{n_A}\}$, and that the domain of attribute C has n_C values, i.e. $\mathrm{dom}(C) = \{c_1, \ldots, c_{n_C}\}$.

A decision tree is a well-known type of classifier. As the name already indicates a decision tree is an acyclic graph having a tree-structure. Each inner node of the tree is labeled with an attribute A which is also called split attribute. For each attribute value $a \in \text{dom}(A)$ an edge to a child node exists and is labeled with a. Each leaf node has a class $c \in \text{dom}(C)$ assigned to it.

Given a new sample case for which the value of the class should be predicted, the tree is interpreted from the root. In each inner node the sample case is tested for the attribute stored within the node. According to the result of the test the corresponding edge is followed to a child node. When a leaf node is reached the class label assigned to it is taken as the class for the sample case.

A variety of algorithms to learn decision trees automatically from data have been published, for example the CART system [3] and C4.5 [13]. All algorithms for decision tree induction grow the three top-down using a greedy-strategy. Starting at the root node an attribute A is selected that yields the highest score regarding an attribute evaluation measure. The dataset is then split into n_A subsets each corresponding to one attribute value of A and a child node for each of them is created. If all its cases have the same class label, a subset is not split further and hence no children are created. The current node then becomes a leaf and is assigned the class label of its associated subset. Apart from minor variations all decision tree algorithms that we are aware of follow the schema above. In fact, one of the major differences between algorithms is the attribute evaluation measure used.

An attribute evaluation measure $I(C,A)$ rates the value of an attribute A for predicting the class attribute C. The most well-known measures are probably the Gini index [3], information gain [14] and information gain ratio [13]. Since we use the latter two in the experimental evaluation of our algorithm we will introduce them very briefly in the following. The information gain $I_{gain}(C,A)$ measures the information gained, on average, about the class attribute C when the value of the attribute A becomes known. A disadvantage of the information gain is its bias towards attributes with many values. To overcome this problem the information gain ratio $I_{gr}(C,A)$ was proposed which penalises many-valued attributes by dividing the information gain $I_{gain}(C,A)$ by the entropy of the attribute itself [14, 13].

4 Predicting Decision Trees

4.1 Basic Idea

Formally, concept drift can be described as the shift over time of the samples' probability distribution. We assume that such shift is usually not arbitrary but follows a certain pattern. It should be stressed that we do not make any assumptions about the speed of those distribution shift.

As already mentioned in the Introduction, the *PreDeT* algorithm does not track or model distribution shift as such. Instead it models the development of the attribute evaluation measure and the class label distribution over time. These models are then

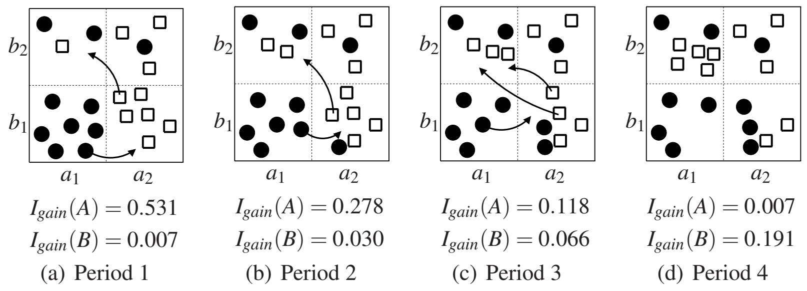


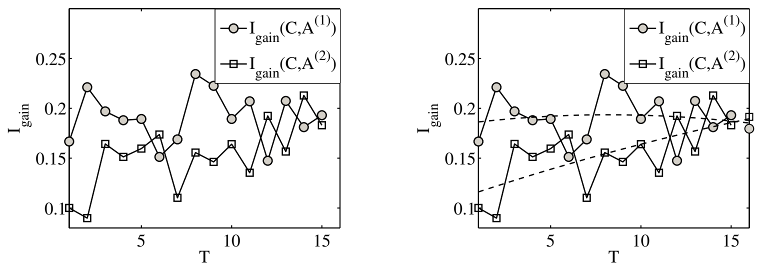
Fig. 1 Illustration of how concept drift can lead to trends in information gain

used to predict future values of the respective measure. The predictions, in turn, are used to control the decision tree induction.

Figure 1 illustrates the concept drift and the resulting change in information gain. It shows the distribution of samples over the attribute space at four consecutive time periods. Each sample belongs to one of two classes, squares and bullets, each described by two attributes A and B with domains $\{a_1, a_2\}$ and $\{b_1, b_2\}$, respectively. Lets assume that we learn a decision tree at the end of each period which predicts the samples in the next period. This is equivalent to a temporal window approach. In period 1, shown in Figure 1(a), the information gain of A is much higher than those of B and it therefore would have be chosen as the split attribute. However, the distribution of samples shifts over time which is indicated by arrows in Figure 1(a) to Figure 1(c). In period 3 the information gain of A is still higher than those of B and therefore A would be the split attribute. This would lead to an classification error of 8 using the samples from period 4 for testing. However, in period 4 attribute B would have been the superior split attribute. The choice solely based on the samples from period 3 was suboptimal. If we look at how the information gain developed between periods 1 and 3 we can see that it has a downward trend for A and an upward trend for B. Using an appropriate model for both time series it would have been possible to anticipate the change in the split attribute and to choose B. This choice leads to a much smaller classification error of 5.

Figure 2(a) shows an example obtained from the same real world dataset which we also use for our experimental evaluation in Section 5. The information gain history of the attribute $A^{(1)}$ is apart from noise stable whereas the information gain history of $A^{(2)}$ shows an upward trend. Furthermore, it can be seen that for the vast majority of time periods $T = 1, \dots, 15$ attribute $A^{(1)}$ has more predictive power and would therefore been chosen as the split attribute. However, due to the observed upward trend in the information gain of $A^{(2)}$ both histories will intersect and $A^{(2)}$ will become the split attribute in the near future.

Figure 2(b) shows the two histories from Figure 2(a) each modeled by a quadratic regression polynomial. In period 16 are the – at the time of modeling unknown – information gain values of both attributes marked. As it can be seen, the predictions



(a) The history of $A^{(1)}$ is apart from noise stable. The history of $A^{(2)}$ shows an upward trend.

(b) Both histories modeled by quadratic polynomials shown as dotted lines. In period 16 the values to be predicted are shown.

Fig. 2 Histories of information gain values for two different attributes

made by the regression models anticipate the change in the ranking of candidate split attributes which happens between period 15 and 16.

Summarising, the basic idea of *PreDeT* is to learn models which describe evaluation measure histories and class label distribution histories in each step of the decision tree induction. The models are then used to predict the value of the respective quantity for the next, future time period. Subsequently, the predictions are used to decide whether to grow a subtree and which class label to assign to a leaf node. As we already pointed out in Section 3 these two decisions are the main building blocks of the vast majority of decision tree learners. Because our algorithm leverages predictions for both it is finally capable to predict how a decision tree may look like in the future. In the context of concept drift this means that we are able to provide classifiers with a higher accuracy than those which are solely reflecting the characteristics of historic data.

4.2 Notation

Let S be a time-stamped data set and $[t_0,\ t_r]$ the minimum time span that covers all its samples. The interval $[t_0,\ t_r]$ is divided into $r > 1$ non-overlapping periods $[t_{i-1}, t_i[$, such that the corresponding subsets $S^i \subset S$ each have a size $|S^i| \gg 1$. Let, without loss of generality, $\hat{T} := \{1, \ldots, r, (r+1), \ldots\}$ be the set of all past $(i \leq r)$ and future $(i > r)$ period indexes.

Assume that we have a family of time-dependent data sets $(S^1, \ldots, S^r)$ each described by the same attributes $A^{(i)}, i = 1, \ldots, m$ having the same domains in each time period. Quantities crucial for decision tree induction like attribute selection measure and the distribution of class labels are now related to a specific data set S^i and thus to a certain time period T_i. Therefore they form sequences of values which we will denote by $\mathbf{I} := (I(S^1, A), \ldots, I(S^r, A))$ for attribute evaluation measures and $\mathbf{P} := (P^1, \ldots, P^r)$ for the sequence of class label distributions. Thereby $P^k := (p^k_{1\cdot}, \ldots, p^k_{n_C\cdot})$ is the distribution of class labels and $p^k_{i\cdot}$ is the relative frequency of class attribute value i in time period k. We will refer to these sequences as an attributes evaluation measure history and class label distribution history, respectively.

4.3 Predicting Attribute Evaluation Measures

A model φ for attribute evaluation measures is a function $\varphi : \hat{T} \longrightarrow \mathbb{R}$. In general, it will be determined based on a history $\mathbf{I} := (I(S^1, A), \ldots, I(S^r, A))$ of attribute evaluation measures which will be denoted by $\varphi[\mathbf{I}]$. A model φ is then used in each inner node to obtain a prediction $\varphi[\mathbf{I}](r+1)$ for attribute evaluation measure's value in the next time period T_{r+1}.

As the set of potential candidate models we chose the set of polynomials $\varphi(T) = \sum_{i=0}^{q} a_i T^i$ fitted to $\mathbf{I}$ using least squared regression. Linear regression in contrast to other possible model classes, like neural networks [7] or support vector regression [16], offers the advantage that no large sample sizes (long histories) are required and that the underlying algorithms are fast. The latter aspect is in particular important because models for a vast number of histories need to be learned. The advantage of polynomial linear regression is, specifically, that it offers a simple way to obtain a set of candidate models by varying the degree q of the polynomial.

Having a set of fitted regression polynomials the best polynomial needs to be selected. In this case 'best' means that polynomial which provides the best trade-off between goodness of fit and complexity and is, for this reason, less prone to overfit the data. This can be measured using the Akaike information criterion (AIC) [1]. Let r be the number of observations, i.e. the length of the history, $q+1$ the number of parameters of the polynomial and RSS the residual sum of squares of the fitted regression polynomial. Then AIC is defined as:

$$AIC = 2(q+1) + r \ln \frac{RSS}{r} \qquad (1)$$

Commonly, the number of time periods for which data is available can be rather small. For example, the data we use for our experiments in Section 5 consists of 25 data sets obtained weekly. The original Akaike information criterion, however, should only be applied to data sets with large sample sizes [4], i.e. if $r/(q+1) > 40$. To overcome this limitation a number of corrections of the Akaike criterion for small sample sizes have been developed. In our *PreDeT* algorithm we use the following known as AIC_C [9]:

$$AIC_C = AIC + \frac{2(q+1)(q+2)}{r-q-2} \qquad (2)$$

For large sample sizes $r\,AIC_C$ converges to AIC, therefore it is suggested that it is always used regardless of sample size [4].

4.4 Predicting the Majority Class in Leafs

A model ψ for histories of class label distributions is a function $\psi : \hat{T} \longrightarrow [0,1]^{n_C}$ It is learned from the history of class label distributions $\mathbf{P} := (P^1, \dots, P^r)$. The dependency of ψ from $\mathbf{P}$ will be denoted by $\psi[\mathbf{P}]$. Within our *PreDeT* algorithm a model ψ is used in each leaf node to predict the class label distribution at time point T_{r+1}.

The prediction model ψ is a vector of functions $\phi_i : \hat{T} \longrightarrow [0,1]$ each of which models a dependency between the time period and the relative frequency (estimated probability) of a class label. Because the relative frequencies must sum to one $\sum_{i=1}^{n_C} \phi_i(T) = 1$ must hold, i.e.

$$\psi(\hat{T}) = \begin{pmatrix} \phi_1(T) \\ \phi_2(T) \\ \vdots \\ \phi_{n_C}(T) \end{pmatrix} = \begin{pmatrix} \phi_1(T) \\ \phi_2(T) \\ \vdots \\ 1 - \sum_{i=1}^{n_C - 1} \phi_i(T) \end{pmatrix} \tag{3}$$

To model each ϕ we also use polyonomials of degree q, i.e. $\phi = \sum_{i=0}^{q} a_i T^i$. The degree of the polynomials is, similar to Section 4.3, determined using the Akaike information criterion.

Because values $\phi_i(T)$ are relative frequencies additional constraints have to be imposed on the choice of the function ϕ_i. In particular, $\forall T \in \{0, \dots, r+1\} : 0 \le \phi_i(T) \le 1$ should always hold. In our experience, however, this constraint can be too strict. For example, in the case $p_i^k = p_i^{k+1} = 1$ and $p_i^j \ne 1$ for $j \ne k$ and $j \ne k+1$ it is rather difficult to find a continuous model class for ϕ. For this reason and because we only aim to predict values for the period $r+1$ we use the weaker constraint $0 \le \phi_i(r+1) \le 1$. Applying this constraint the model ϕ_i cannot be derived using standard regression analysis anymore. Instead, we obtain the coefficients $\mathbf{a} := (a_0, \dots, a_q)^T$ of the polynomial $\phi_i = \sum_{i=0}^{q} a_i T^i$ by solving the constrained linear least-squares problem

$$\mathbf{a} = \underset{\mathbf{a}}{\operatorname{argmin}} \frac{1}{2} \|C\mathbf{a} - \mathbf{p}\|_2^2 \;\text{ with } C := \begin{pmatrix} 1^0 & \cdots & 1^q \\ \vdots & \ddots & \vdots \\ r^0 & \cdots & r^q \end{pmatrix} \text{ and } \mathbf{p} := \begin{pmatrix} p_i^1 \\ \vdots \\ p_i^r \end{pmatrix}$$

There exist several methods from the field of optimisation for solving constrained linear least-squares problems. They will not be discussed here in greater detail. For further reading see [5].

4.5 Putting the Parts Together

Having explained the main building blocks of our method in the previous two sections we will now go ahead and explain how they can be used in combination with a decision tree learner to predict future decision trees. This will lead us to the *PreDeT* algorithm.

Figure 3 shows the *PreDeT* algorithm. Similar to the vast majority of decision tree learners, like C4.5 and CART, it consists of two consecutive stages. In the first stage (lines 1–8) the split attribute for the current node is searched. In the second stage (lines 9–18) it is decided whether the current node is a leaf (line 9) or inner node (line 15). Respectively, either a class label is assigned to the leaf node based on the majority class in this node, or the data sets are split according to the split attribute and the *PreDeT* algorithm continues recursively (line 17). It should be clear that the basic ideas laid out in Section 4.3 and Section 4.4 can be used in connection with any decision tree learner that uses attribute evaluation measures to determine splits.

```
PREDET($(S^1,\dots,S^r)$)
 1  $I_{best} \leftarrow WORTHLESS$
 2  for all untested attributes A
 3  do $\mathbf{I} \leftarrow (I(S^1,A),\dots,I(S^r,A))$
 4     learn prediction model $\varphi[\mathbf{I}]$
 5     $\tilde{I} \leftarrow \varphi[\mathbf{I}](r+1)$
 6     if $\tilde{I} > I_{best}$
 7        then $I_{best} \leftarrow \tilde{I}$
 8             $A_{best} \leftarrow A$
 9  if $I_{best} = WORTHLESS$
10    then create leaf node $v$
11         $P^k \leftarrow (p^k_{1\cdot},\dots,p^k_{n_C\cdot}),\ k=1,\dots,r$
12         learn prediction model $\psi[(P^1,\dots,P^r)]$
13         $(\tilde{p}^{r+1}_{1\cdot},\dots,\tilde{p}^{r+1}_{n_c\cdot}) \leftarrow \psi[(P^1,\dots,P^r)](r+1)$
14         assign $c = \mathrm{argmax}_{c_i}(\tilde{p}^{r+1}_{1\cdot},\dots,\tilde{p}^{r+1}_{n_c\cdot})$ to $v$
15    else assign test on $A_{best}$ to $v$
16         for all $a \in dom(A_{best})$
17         do $v.child[a] \leftarrow$
18                PREDET($(S^1|_{A_{best}=a},\dots,S^r|_{A_{best}=a})$)
19  return $v$
```

Fig. 3 Outline of the *PreDeT* algorithm for predicting decision trees

In contrast to other decision tree learners *PreDeT* takes as input a sequence of data sets $(S^1,\dots,S^r)$ representing time periods $1,\dots,r$. It uses these data sets to estimate the value of the attribute evaluation measure in the next time period $r+1$ using a learned model φ (lines 4–5). The class label distribution within each data set is used to predict the likely class label distribution in time period $r+1$ using a learned model ψ (lines 11–13). Note that every decision about the structure of the tree – the choice of the split attribute in inner and of the class label in leaf nodes – is solely based on estimated future values of the used metrics but not directly on the historic or present data sets $(S^1,\ldots,S^r)$. For this reason the tree learned by *PreDeT* can be seen as a prediction of the decision tree in period $r+1$.

5 Experimental Evaluation

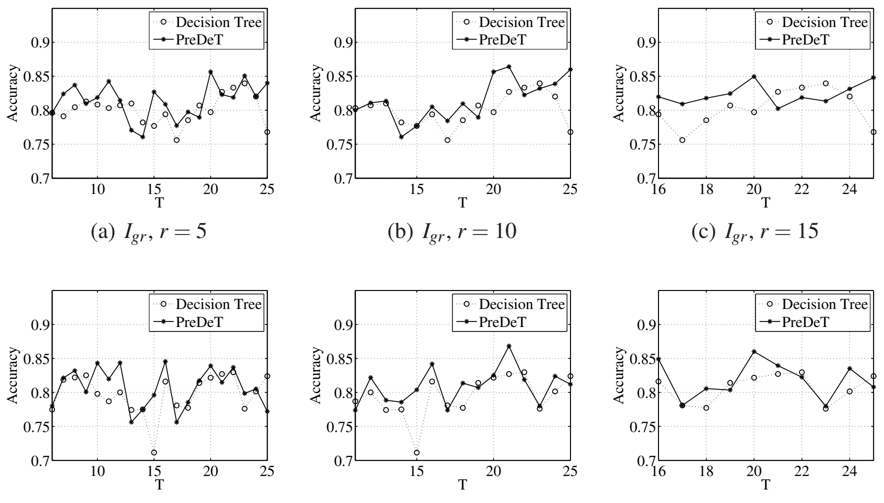
The *PreDeT* algorithm does depend on a number of factors: first of all, the length r of the sequence of data sets $(S^1,\ldots,S^r)$, secondly, the attribute evaluation measure I, and thirdly, the size of the individual data set S^i. In our experiments we evaluated how these factors influence the accuracy of the anticipated decision trees and how this accuracy compares to the one of decision trees obtained by a temporal moving window approach which is typically used for learning decision trees from concept-drifting data.

For our experiments we chose a representative real-life dataset from the domain of Customer Relationship Management (CRM). The dataset contains answers of customers to a survey conducted by a telecommunications company over a period of 25 weeks. Each sample is described by 13 nominal attributes with a domain size between 2 and 9. Goal of the classification task is to predict whether a customer will be satisfied or dissatisfied with a certain service using the remaining 12 attributes, i.e. the data set has two classes to predict.

First of all we analysed the influence of the length r of the sequence of data sets and the choice of the attribute evaluation measure I on the classification accuracy. We split the original data set into 25 subsets S^i, each corresponding to a time period of one week. The subsets contain between 243 and 399 samples. For each experiment we chose a sequence of r consecutive data sets $(S^i,\ldots,S^{i+r-1})$ within the available 25 ones. For each $i,\ i=0,\ldots,25-r$ we then learned a decision tree using the *PreDeT* algorithm and obtained classifications for the samples in the data set S^{i+r} that chronologically follows the sequence. For instance, for $r=5$ we have 20 sequences, learn 20 decision trees and thus obtain the classification accuracy for 20 data sets.

To learn a decision tree with which the performance of PreDeT can be compared it has to be considered that *PreDeT* implicitely learns a sequence of r decision trees each corresponding to a data set S^i and then anticipates the tree in the future period $r+1$ using a prediction model. As with any prediction model, the obtained result cannot have a better quality than its inputs used for learning. Since the quality of a decision tree is (amongst other factors) determined by the size of the data set used for training, it is clear that the tree anticipated by PreDeT does have a similar quality to a tree that would have been learned directly on a data set S^{r+1} with a size similar to those of each $S^i, i=1,\ldots,r$. Since we assume S^{r+1} to be unknown at the time of learning we took the most recent data set S^r to learn a decision tree[1] for comparism because the characteristics of S^r are very likely best reflecting those of S^{r+1}.

Another advantage of comparing the accuracies obtained by PreDeT with those of a decision tree learned from only the most recent data is that the latter is basically a temporal moving window approach and thus common practise for learning decision trees in the presense of concept drift. Moreover, such a temporal moving window approach performs similar to age dependent weighting approaches – the alternative method to learn decision trees in the presence of concept drift – in case of smooth, non-abrupt concept drift [11, 10]. Such a type of concept drift is present in our data as we know from previous studies on change mining carried out on the same data set [2].



(a) I_{gr}, $r = 5$ (b) I_{gr}, $r = 10$ (c) I_{gr}, $r = 15$

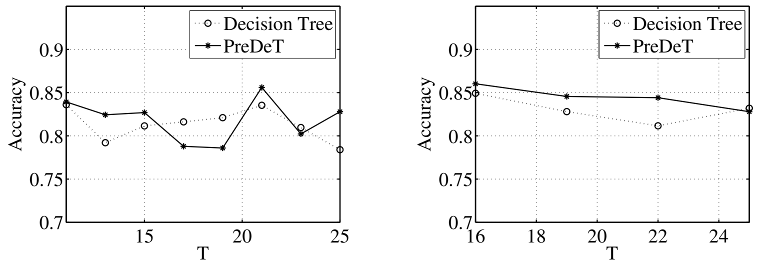
(d) I_{gain}, $r = 5$ (e) I_{gain}, $r = 10$ (f) I_{gain}, $r = 15$

Fig. 4 Classification accuracy for two different attribute evaluation measures in several consecutive time periods T. Different sequences of a length of r time periods were used to learn the least squares models used by *PreDeT*. For comparison, the performance of a traditional decision tree approach using only the most recent data of each sequence for inducing the decision tree is shown.

Using the above experimental setup we carried out experiments using the information gain ratio I_{gr} and information gain I_{gain} and varied in each case the length of the sequence by using $r = 5, 10, 15$. Figure 4 shows the results of our experiments. As we can see, the classification accuracy of *PreDeT* is on average superior to the one of the traditional decision tree approach and also independent of the choice of the attribute evaluation measure and the parameter r. By comparing Figure 4(a) with

[1] We used the decision tree implementation by Christian Borgelt that can be obtained from http://www.borgelt.net/dtree

Figure 4(c) (Figure 4(d) with Figure 4(f), respectively) it can be seen that the gain in classification accuracy increases when longer sequences are used. This again leaves space for further optimisations of the *PreDeT* algorithm with respect to the optimal choice of the parameter r.



(a) Using periods of two weeks for training. The accuracy median for the decision tree is 0.8138 and for *PreDeT* 0.8256.

(b) Using periods of three weeks for training. The accuracy median for the decision tree is 0.83 and for *PreDeT* 0.8449.

Fig. 5 Classification accuracy for information gain I_{gain} and $r = 5$ using data sets covering periods of two and, respectively, three weeks for training and of one week for testing. For comparison, the performance of an approach using only the most recent data set of each sequence for inducing the decision tree is shown.

We now evaluate the influence of the size of the individual data sets S^i on the performance of *PreDeT*. This aspect is in particular interesting because it is known that the performance of a moving temporal window approach, which we use in our experiments for comparison, does strongly depend on it [6]. We use a similar experimental setup as in our first experiments but instead of using a size of one week for each data set S^i we increase the size to two and three weeks, respectively. In particular, we split our initial data set into 12 non-overlapping subsets each covering a period of two weeks. Likewise, we obtained another split of 8 subsets each covering a period of three weeks. Similar to our first experiment we used sequences of $r = 5$ to learn decision trees using *PreDeT* and classified the samples of the week that immediately follows the respective sequence. Again, we compared the classification accuracy of *PreDeT* with the one of decision trees learned by a temporal moving window approach.

The results of this experiment are shown in Figure 5(a) for a period length of two weeks and in Figure 5(b) for a period length of three weeks, respectively. As we can see, in both cases the classification accuracy of *PreDeT* is on average higher than the one of the temporal moving window approach, i.e. a decision tree learned only on the most recent data of each sequence. We can also see by comparing Figure 5(a) with Figure 5(b) that the performance gain offered by *PreDeT* seems to increase with the size of individual data sets.

6 Conclusion and Future Work

We presented a novel approach to learn decision trees in the presence of concept drift. Our *PreDeT* algorithm aims to anticipate decision trees for future time periods by modelling how attribute evaluation measure and class label distribution evolve over time. Our experimental results show that our approach is able to learn decision trees with a higher classification accuracy than trees learned by a temporal window approach.

Currently we are working on several enhancements of our algorithm. In the first place, we investigate the advantages of using more sophisticated and more robust regression methods, e.g. support vector regression [16], instead of regression polynomials. Secondly, at the moment a new decision tree has to be predicted every time a new batch of data arrives. For this reason it would be advantageous w.r.t. computational costs to enhance *PreDeT* in order to support incremental learning. One starting point could be to leverage existing incremental algorithms for linear regression [15] and support vector regression [17, 18].

References

1. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control **19**(6), 716–723 (1974)
2. Boettcher, M., Nauck, D., Ruta, D., Spott, M.: Towards a framework for change detection in datasets. In: M. Bramer (ed.) Research and Development in Intelligent Systems, *Proceedings of AI-2006, the 26th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, vol. 23, pp. 115–128. BCS SGAI, Springer (2006)
3. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont (1984)
4. Burnham, K.P., Anderson, D.R.: Multimodel inference: understanding AIC and BIC in model selection. Sociological Methods & Research **33**, 261–304 (2004)
5. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press, London (1989)
6. Helmbold, D.P., Long, P.M.: Tracking drifting concepts by minimizing disagreements. Machine Learning **14**(1), 27–45 (1994)
7. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989).
8. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106. ACM Press, New York, NY, USA (2001).
9. Hurvich, C.M., Tsai, C.L.: Regression and time series model selection in small samples. Biometrika **76**, 297–307 (1989)
10. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis **8**(3), 281–300 (2004)
11. Klinkenberg, R., Rueping, S.: Concept drift and the importance of examples. In: J. Franke, G. Nakhaeizadeh, I. Renz (eds.) Text Mining – Theoretical Aspects and Applications, pp. 55–77. Physica-Verlag, Berlin, Germany (2003)
12. Kuh, A., Petsche, T., Rivest, R.L.: Learning time-varying concepts. In: Advances in Neural Information Processing Systems, pp. 183–189. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1990)
13. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1992)

14. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**(1), 81–106 (1996)
15. Scharf, L.: Statistical Signal Processing. Addison-Wesley (1991)
16. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing **14**(3), 199–222 (2004)
17. Syed, N.A., Liu, H., Sung, K.K.: Handling concept drifts in incremental learning with support vector machines. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 317–321. ACM Press, New York, NY, USA (1999).
18. Wang, W.: An incremental learning strategy for support vector regression. Neural Processing Letters **21**(3), 175–188 (2005).
19. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning **23**(1), 69–101 (1996).