Handling Time Changing Data with Adaptive Very Fast Decision Rules

Petr Kosina[1] and João Gama[2]

[1] LIAAD-INESC Porto, FI Masaryk University, Czech Republic
petr.kosina@inescporto.pt
[2] LIAAD-INESC Porto, FEP-University of Porto
jgama@fep.up.pt

Abstract. Data streams are usually characterized by changes in the underlying distribution generating data. Therefore algorithms designed to work with data streams should be able to detect changes and quickly adapt the decision model. Rules are one of the most interpretable and flexible models for data mining prediction tasks. In this paper we present the Adaptive Very Fast Decision Rules (AVFDR), an on-line, any-time and one-pass algorithm for learning decision rules in the context of time changing data. AVFDR can learn ordered and unordered rule sets. It is able to adapt the decision model via incremental induction and specialization of rules. Detecting local drifts takes advantage of the modularity of rule sets. In AVFDR, each individual rule monitors the evolution of performance metrics to detect concept drift. AVFDR prunes rules that detect drift. This explicit change detection mechanism provides useful information about the dynamics of the process generating data, faster adaption to changes and generates compact rule sets. The experimental evaluation shows this method is able to learn fast and compact rule sets from evolving streams in comparison to alternative methods.

1 Motivation

Machine learning in recent years has undergone a change in the focus and application of its techniques. More and more attention is shifted towards ubiquitous mining and real-time applications as more and more data is represented by possibly infinite streams. New approaches and stream extensions to off-line methods emerge in order to improve performance under constrains given by the stream environment. It includes characteristics such as being able to produce a model while scanning the data only once, the model must be available at any point of time, must be up-to-date, and all must be able to run under computational and memory constrains [19].

One of the common phenomena in stream mining is a change in data that might occur over time. This phenomenon addresses mostly the up-to-date requirement of the stream approaches. Almost everything in this world changes and so can the concepts in data if they are being observed for long enough time. The influence of seasonal change, economical change, health conditions or wear of machinery can cause decrease of quality of the models built on previous observations. Therefore, fast adaptation of models has an advantage for most of the real world problems.

Among the best known and most used models for data stream classification are algorithms based on Hoeffding Trees [12] (HT). The adaptation in HT algorithms is present

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 827–842, 2012.
© Springer-Verlag Berlin Heidelberg 2012

in an inexplicit form via incremental growth of the tree. The adaptation in this case is slow. Faster adaptation might be achieved by employing explicit drift detection methods, e.g. [20,1,22]. This approach, however, require rebuilding the current tree which might be inefficient. Decision rules are a classification model similar to decision trees, which has an advantage of having individual rules that can be managed independently. Therefore, in decision rules the implicit adaptation feature that is present in the trees remains but in addition to it, the set of rules can be altered more easily. Instead of rebuilding the classifier from scratch or executing a complicated change of the structure of a tree, individual rules which are considered outdated can be simply removed. This paper presents Adaptive Very Fast Decision Rules (AVFDR) classifier as an extension to VFDR [16]. The main contribution is that this system focuses on time changing data and it incorporates a drift detection mechanism for each individual rule. It is capable of faster adaptation to a change and the mechanism also serves as rule set pruning.

The paper is organized as follows. The Section 2 discusses the related work in rule learning and handling time changing data. The AVFDR system is presented in Section 3 and the experimental results are in Section 4. Finally, Section 5 concludes the paper and mentions future work directions.

2 Related Work

Decision rules are well known classification method in off-line learning with a strong connection to decision tree learning. There are approaches that build decision lists from the tree [28], where each rule then corresponds to the path from the root to a leaf and there are as many rules as leaves. The set can be then optimized to obtain simpler and more accurate classifier.

There exist many algorithms for building decision lists [29,7,9,11,34]. CN2, presented in [7], evaluates every possible conjunction of attribute tests (*if-conditions*) of a rule based on information-theoretic entropy measure. Later in [8], the entropy measure is replaced by Laplace accuracy estimate. The algorithm searches for new rules for each class in turn each time separating one (positive) from all the others (negative) as two-class problem. Only positive examples that satisfy learned rule are removed from the training set for next iteration of the rule search. RIPPER [9] orders classes in increasing order of their frequency in the training set. After learning rules that separate minority class, the covered examples are removed and the algorithm proceeds with the next class. The process stops when the last single class remains that represents the default class.

Incremental rule learners include STAGGER [30], the first system designed expressly for coping with concept drift, the FLORA family of algorithms [35] with FLORA3 being the first system able to deal with recurring contexts, and the AQ-PM family [27]. The first rule learning system designed for streaming numerical data is system Facil, presented in [13]. Facil learns decision rules that might overlap. Expansion of a rule is controlled by border examples that are stored together with rules when they are learned. Adaptation to drift is blind, by deleting older rules.

The original VFDR system, which incrementally learns new rules and specialises existing ones, was presented by [16]. It is capable of learning ordered or unordered sets and employs Bayesian leaves. The system is further extended in [26] to focus the

attention on learning rules for various classes of labeled data and it can induce multiple rules (one for each such class) at one point when specialization is evaluated.

The stream mining community has already introduced many different approaches to deal with the phenomenon of concept drift. Sliding windows and example weights [25] are widely used approaches to maintain a classifier consistent to the most recent data. In [4] the authors proposed the ADWIN algorithm, a detector and estimator which automatically adapts to the current rate of change by keeping the window of recent examples of variable length. It employs the Hoeffding bound to guarantee that the window has maximal length without a change inside the window. Other methods can explicitly detect change-points or small time-windows where the concept to learn has changed. A classifier can be equipped with such drift detection methods associated with a forgetting mechanism. Drift detection methods might monitor the evolution of the error rate, like the SPC algorithm [20], monitor the distance between classification errors, like in [1], etc.

In [23], the authors presented system CVFDT, a decision tree learner for mining data streams with non-stationary distribution. CVFDT learns model consistent with a sliding window of recent examples. When concept is changing and split that was previously executed would no longer be the best, it starts learning an alternate subtree with new best attribute as its root. The subtree replaces the original one when it becomes more accurate.

As pointed out in [33], a drawback of decision trees is that even a slight drift of the target function may trigger several changes in the model and severely compromise learning efficiency. On the other hand, ensemble methods avoid expensive revisions by weighting the members, but may run the risk of building unnecessary learners when virtual drifts are present in data. In [5] two new decision tree ensemble methods were presented: ADWIN Bagging and Adaptive-Size Hoeffding Tree bagging. The former extends on-line bagging with ADWIN change detector, which works as an estimator for the weights of the boosting method. The worst performing classifier is removed from the ensemble when change is detected and it is replaced by a new one. The latter uses Hoeffding trees of different maximum sizes since smaller trees adapt faster to changes and larger work better for long periods with little or no change.

3 Adaptive Very Fast Decision Rules

In this section we present AVFDR algorithm which extends VFDR classifier for data streams by integrating drift detection to each rule.

3.1 Growing a Set of Rules

The AVFDR algorithm is designed for high-speed data streams which learns unordered set rules and needs only one scan of data.

The algorithm begins with a empty rule set (RS) and a default rule $\{\} \rightarrow \mathcal{L}$, where $\mathcal{L}$ is initialized to $\emptyset$. $\mathcal{L}$ is a data structure that contains information used to classify test instances, and the sufficient statistics needed to expand the rule. Each learned rule (r) is a conjunction of literals, that are conditions based on attribute values, and a $\mathcal{L}_r$.

Algorithm 1: AVFDR: Rule Learning Algorithm.

```
input : S: Stream of examples
        ordered_set: boolean flag
output: RS: Set of Decision Rules
begin
    Let RS ← {}
    Let default rule ℒ ← ∅
    foreach example (x, y_k) ∈ S do
        foreach Rule r ∈ RS do
            if r covers the example then
                /* Estimate Rule Status                    */
                Status ← SPC(r, x_t, y_t)
                if Status == OutControl then
                    RS ← RS − {r}
                else
                    if Status == InControl then
                        Update sufficient statistics of r
                        RS ← RS − {r}
                        RS ← RS ∪ ExpandRule(r)
                        if ordered_set then
                            BREAK
        if none of the rules in RS covered example then
            Update sufficient statistics of the default rule
            RS ← RS ∪ ExpandEmptyRule(default rule)
```

For numerical attributes, each literal is of the form $X_i > v$, or $X_i \leq v$ for some feature X_i and some constant v. For categorical attributes AVFDR produce literals of the form $X_i = v_j$ where v_j is a value in the domain of X_i. Please note that for simplicity we use the $X_i = v_j$ notation for both cases, numerical and categorical, in the context of adding a new condition.

If all the literals are true for a given example, then the example is said to be *covered* by the rule. The labeled examples covered by a rule r are used to update $\mathcal{L}_r$. A rule is expanded with the literal that has the highest gain measure of the examples covered by the rule. $\mathcal{L}_r$ accumulates the sufficient statistics, is similar to statistics in [15], to compute the gain measure of all possible literals. $\mathcal{L}_r$ is a data structure that contains: an integer that stores the number of examples covered by the rule; a vector to compute $p(c_k)$, i.e., the probability of observing examples of class c_k; a matrix $p(X_i = v_j|c_k)$ to compute the probability of observing value v_j of a nominal attribute X_i per class; and a `btree` to compute the probability of observing values greater than v_j of continuous attribute X_i, $p(X_i > v_j|c_k)$, per class.

The number of observations after which a rule can be expanded or new rule can be induced is determined by Hoeffding bound. It guarantees that with the probability $1 - \delta$ the true mean of a random variable x with a range R will not differ from the

Algorithm 2: The SPC Algorithm

Input : r: rule ;
/* n^{th} observation of rule r */
Current example: $\boldsymbol{x}_n, y_n$;
Output: Status $\in$ {InControl, OutControl, Warning}
begin
  Let $\hat{y}_n \leftarrow r(\boldsymbol{x}_n)$;
  Let $error_n \leftarrow L(\hat{y}_n, y_n)$;
  Compute error's mean p_n and variance s_n;
  if $p_n + s_n < p_{min} + s_{min}$ **then**
    $p_{min} \leftarrow p_n$;
    $s_{min} \leftarrow s_n$;
  if $p_n + s_n < p_{min} + \beta \times s_{min}$ **then**
    Status $\leftarrow$ 'InControl' ;
  else
    if $p_n + s_n < p_{min} + \alpha \times s_{min}$ **then**
      Status $\leftarrow$ 'Warning';
    else
      Status $\leftarrow$ 'OutControl';
  Return: Status;

estimated mean after N independent observations by more than: $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$. It is not efficient to check for the sufficient number of examples with every incoming example, therefore it is done after only N_{min} observations.

The set of rules (RS) is learned in parallel as described in Algorithm 1. We consider two cases: learning ordered or unordered set of rules (AVFDR^o and AVFDR^u). In the former, every labeled example updates statistics of the first rule that covers it. In the latter, every labeled example updates statistics of all the rules that cover it. If a labeled example is not covered by any rule, the default rule is updated.

We can further apply different strategies to functions that are responsible for creating new rules from *default rule* and expansion of other rules.

3.2 Expansion of a rule

The AVFDR classifier applies one vs. all strategy in which examples of class $c_k \in C$ are positive and $\forall c_l \in C, c_l \neq c_k$ are negative. It considers a rule expansion for each class $c \in C_r$, where C_r is the set of classes observed at rule r. The process to select new condition for a rule works as follows.

For each attribute X_i the value of gain function G adopted from FOIL is computed. The change in gain between rule r and a candidate rule after adding a new condition r' is defined as $Gain(r', r) = s \times \left(\log_2 \frac{N'_+}{N'} - \log_2 \frac{N_+}{N} \right)$, where N is the number of examples covered by r and N_+ is the number of positive examples in them, N'_+ and

Algorithm 3: `ExpandRule`: Expanding Unordered Rule.

input : r: One Rule;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  Let $NR \leftarrow \{r\}$
  Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
  `/* Expand rule for the original class */`
  Let c_r be class of r
  $EvaluateLiterals(c_r)$
  if $g^k_{best} - g^k_{2best} > \epsilon$ *or* $\epsilon < \tau$ **then**
    Extend r with a new condition based on the best attribute $X_a = v_j$
    `/* Expand rule for other classes */`
    foreach *class* $c_k \neq c_r$ **do**
      $EvaluateLiterals(c_k)$
      if $g^k_{best} - g^k_{2best} > \epsilon$ *or* $\epsilon < \tau$ **then**
        create new r' by extending r with a new condition $X_a = v_j$ and class c_k
        $NR \leftarrow NR \cup \{r'\}$
    Release sufficient statistics of r
  return NR

N' represent the same for r', and s is the number of true positives in r that are still true positives in r', which in this case corresponds to N'_+.

We are interested only in positive gain, therefore we consider the minimum of the gain function as 0 and the maximum for a given rule is $N_+ \times \left(-\log_2 \frac{N_+}{N}\right)$. We can then normalize the positive gain as: $GainNorm(r', r) = \frac{Gain(r',r)}{N_+ \times \left(-\log_2 \frac{N_+}{N}\right)}$.

G is computed for each attribute value v_j, which was observed in more than certain fraction of examples (e.g., 10% of examples) and class c_k. Procedure in Algorithm 6 searches for the best and second best value of $G()$ for given class. If g^k_{best} is the true best gain measure, i.e., satisfies condition $g^k_{best} - g^k_{2best} > \epsilon$ for given class value c_k, the rule is expanded with condition $X_a = v_j \Rightarrow c_k$.

Algorithm 5 describes the expansion of *default rule* to a new rule of the rule set. The new literal of the new rule has the best attribute-value evaluation and its positive class is the one with the minimum frequency among those that satisfy the Hoeffding bound condition. The search for new rules continues until the expansions for all possible classes are checked.

In case of expanding some rule that already contains conditions the procedure works as follows. The rule was induced for a certain class that was set as positive. To keep this class of interest in the set, it is maintained as positive for the next computations of the measure criterion (Algorithm 4). The unordered rule set in Algorithm 3 considers computations of other classes set as the positive one and expanding the rule with the

Algorithm 4: `ExpandRule`: Expanding Ordered Rule.

input : r: One Rule;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  Let $NR \leftarrow \{r\}$
  Let c be the class of rule r
  Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
  Let c_r be class of r
  $EvaluateLiterals(c_r)$
  if $g^k_{2best} - g^k_{best} > \epsilon$ *or* $\epsilon < \tau$ **then**
    Extend r with a new condition based on the best attribute $X_a = v_j$
    Release sufficient statistics of r
  return NR

best condition for such class. Nevertheless, it is allowed only when the rule has already been expanded with the original class at that call of `ExpandRule`. This setting is able to produce more rules in one call of `ExpandRule` and the number of rules induced from one rule r in RS in such call is at most $|C_r|$. Should the same rule already exist in the set, duplicate rule is not allowed to be expanded with given condition $X_a = v_j$. This process creates multiple rules for different classes marked as positive, but not necessarily for all the available classes in one call.

3.3 Rule Reaction to a Drift

As opposed to decision trees the set of rules offers the possibility to remove individual rules, which do not perform well, without the need of rebuilding the model. This characteristic is very important for incremental learning where change might occur, because it allows faster adaptation of the model.

The previous algorithm `VFDR` was adapting only implicitly by inferring new rules and specializing the existing ones. `AVFDR` extends the algorithm with explicit drift detection. In addition to $\mathcal{L}_r$, each rule r in `AVFDR` contains a drift detection method which tracks the performance of the rule r during learning. The method employed in `AVFDR` is the SPC [20] described in Algorithm 2. With every labeled training example covered by a rule, the rule makes prediction and updates its error rate. `SPC` monitors the error rate and manages two registers during training: p_{min} and s_{min}, where p is error rate and s is standard deviation. Every time a new example $(\boldsymbol{x}_n, y_n)$ is covered by the rule those values are updated when $p_n + s_n$ is lower than $p_{min} + s_{min}$.

The learning process of given rule can be in one of the following 3 stages: *in-control*, *out-of-control*, or in *warning*. We follow the 3-sigma rule [21]: the *warning* stage is reached if $p_n + s_n \geq p_{min} + \beta \times s_{min}$ and the *out-of-control* stage is reached if $p_n + s_n \geq p_{min} + \alpha \times s_{min}$, where $\alpha = 3$ and $\beta = 2$.

Algorithm 5: `ExpandEmptyRule`: Expanding Empty Rule.

input : r: One Rule;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  $NR \leftarrow \{r\}$
  Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
  foreach *class* c_k *in ascending order of class frequency* **do**
    $EvaluateLiterals(c_k)$
    if $g^k_{best} - g^k_{2best} > \epsilon$ *or* $\epsilon < \tau$ **then**
      create new r' as $X_a = v_j \Rightarrow c_k$
      $NR \leftarrow NR \cup \{r'\}$
  if NR *is not empty* **then**
    Release sufficient statistics of r
  return NR

In warning stage the rule stops learning until either the status if the rule becomes again *in-control*. If the rule reaches *out-of-control* it implies that the performance of the particular rule has degraded significantly and can negatively influence the quality of predictions of the classifier therefore it is removed from the set. This control enables to keep the set up-to-date and prevent the set from excessive growth.

3.4 Classification Strategy

The set of rules learned by `AVFDR` can employ different classification strategies: First Hit, Weighted Max, and Weighted Sum. The ordered rules use First Hit strategy as the most appropriate. For the unordered rules we decided to apply Weighted Sum in the rest of the paper. In that case all rules covering the example are used for classification and the final class is decided by using weighted vote.

More specifically, assume that a rule r covers a test example. The example will be classified using the information in $\mathcal{L}_r$ of that rule. The simplest strategy uses the distribution of the classes stored in $\mathcal{L}_r$, and classify the example in the class that maximizes $p(c_k)$. This strategy only uses the information about class distributions and does not look for the attribute-values; therefore it uses only a small part of the available information. In a more informed strategy, a test example is classified with the class that maximizes the posteriori probability given by Bayes rule assuming the independence of the attributes given the class. There is a simple motivation for this option. $\mathcal{L}$ stores information about the distribution of the attributes given the class usually for hundreds or even thousands of examples, before expanding the rule and re-initializing the counters. Naive Bayes (`NB`) takes into account not only the prior distribution of the classes, but also the conditional probabilities of the attribute-values given the class. This way, there is a much better exploitation of the available information

Algorithm 6: `EvaluateLiterals`

input : G: Gain evaluation function;
c_k: Class
begin
  foreach *attribute* X_i **do**
    Let g_{ijk} be the $G()$ of the best literal based on attribute X_i and value v_j for class c_k
    if $g_{ijk} > g^k_{best}$ **then**
      Let $g^k_{2best} \leftarrow g^k_{best}$
      Let $g^k_{best} \leftarrow g_{ijk}$
    else
      if $g_{ijk} > g^k_{2best}$ **then**
        Let $g^k_{2best} \leftarrow g_{ijk}$

in each rule. Given the example $\boldsymbol{x} = (x_1, \ldots, x_j)$ and applying Bayes theorem, we obtain: $P(c_k|\boldsymbol{x}) \propto P(c_k) \prod P(x_j|c_k)$.

Using `NB`[1] in `VFDT` like algorithms, is a well-known technique since it was introduced by [17]. One of its greatest advantages is the boost in any-time learning property because even though the learned rule set might not be robust enough or the individual rules might not provide sufficient information for expert interpretation (not being specialized enough, i.e., having only one or few conditions), it may already be able highly informed predictions based on `NB` classification.

4 Experimental Evaluation

In this section we test `AVFDR` on datasets with different properties such that we can evaluate the behavior in various situations. We also provide comparison against decision rules which does not have explicit drift detection, and against stream decision tree classifiers[2] - `VFDTc` and Hoeffding Tree with `ADWIN`.

As an evaluation method we employed prequential classification scenario [18], where a classifier first makes a prediction and consequently the class label is observed, the error-rate is updated, and classifier is trained with said example. All algorithms were implemented in Java as an extension for `KNIME` [2] except `ADWIN`, which is implemented in MOA [3]. The evaluation datasets include both artificial and real data, as well as sets with categorical and continuous attributes or their combination.

[1] Other Bayesian techniques might be used. We use `NB` due to simplicity and being incremental.

[2] Default parameters for tests: N_{nb} (threshold to use `NB`) = 50, w_s (minimal weight of new condition partition - in rules only) = 0.1, δ (confidence level) in HB = 10^{-6}, N_{min} (examples after which HB is computed) = 200, τ (tie breaking constant) = 0.05. Due to space limitation we do not provide thorough discussion about the influence of different parameter settings.

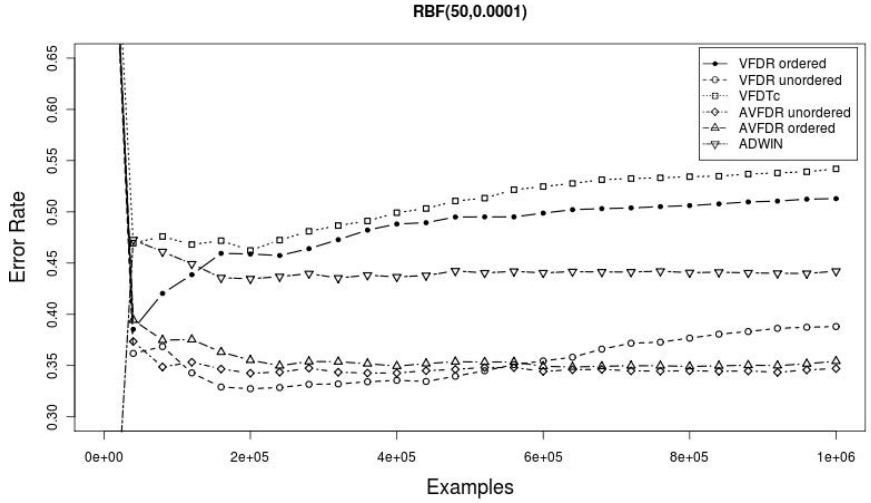

Fig. 1. The prequential error during the learning process in RBF(50,0.0001) dataset with gradual drift. Non-adaptive methods are slowly increasing their error rate.

4.1 Artificial Datasets

The artificial datasets were obtained using generators in [3].

The **Hyperplane** is generated such that the class is given by rotating hyperplane [23]. This set with 1,000,000 examples has 2 classes, 10 attributes changing at speed 0.001 with 5% noise (probability for each instance to have its class inverted). Another artificial dataset is **SEA** Concepts [32] and is commonly used in stream mining tasks that require time changing qualities of data. It is a two-class problem, defined by three attributes (two relevant) and 10 % of noise (the same as previous). There are four concepts, each containing 15,000 examples. The total number of examples is 60,000. **LED** is formed by examples [6] with $\{0, 1\}$ values of each attribute signalling whether given LED is off or on. Only seven out of 24 are relevant. Class label reflects the number (0 to 9) displayed by the relevant diodes. There is 10 % of noise added to this dataset (probability for each attribute that it would have its value inverted). The generated training set size is 1,000,000 instances. The **RBF**'s are generated datasets with drift introduced by moving centroids with constant speed. The sets have 1,000,000 examples and are described as RBF(x, y) where x is the number of centroids moving at the speed y. In **Waveform** dataset [6] there are 3 classes of waves each described with 40 attributes. This dataset had 100,000 instances and it is generated with 10 drift attributes occurring at 50,000.

4.2 Real Datasets

These datasets are mostly obtained from [14] unless cited otherwise.

The **Intrusion** detection from KDDCUP 99, is a data set describing connections which are labeled either as normal or one of four categories of attack. The dataset

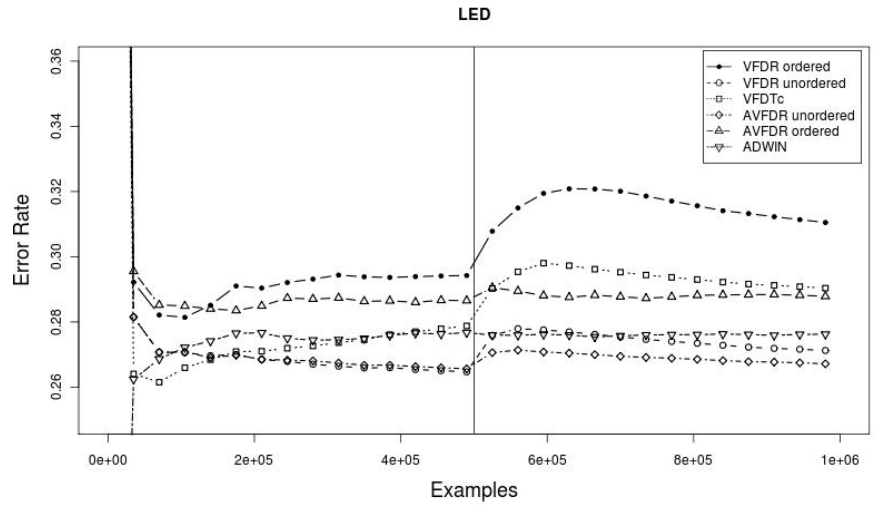

Fig. 2. The prequential error evolution throughout the learning process in LED dataset with abrupt drift. The increase of error rate after abrupt drift occurring at 500,000 is visibly larger in non-adaptive methods.

consists of 4,898,431 instances. The next set is **Covtype**, which has 54 cartographic attributes, continuous and categorical. The goal is to predict the forest cover type for given area. The dataset contains 581,012 instances. The **Elec** dataset contains data collected from electricity market of New South Wales, Australia. It has 45,312 instances. Inspired by a regression dataset used in [24], the task of **Airlines** dataset [3] is to predict whether a flight will be delayed given the information of the scheduled departure in 7 attributes. It consists of 539,383 instances. The **Connect-4** dataset from UCI repository consists of 42 categorical attributes and contains 67,557 examples. The **Activity** dataset is the Localization Data for Person Activity. The set has 164,860 instances and 8 attributes, but we removed the time information (timestamp and date).

4.3 Results

We conducted 5 runs over the synthetic datasets using different random seeds. These datasets are designed to contain concept drift therefore provide evidence of the benefit of adaptation. Figure 1 and Fig. 2 depict examples of the performance of the classifiers in selected artificial datasets. The first one, Figure 1, shows RBF(50,0.0001) dataset, which contains gradual concept drift. It can be observed that both AVFDRs cope best with the drift. ADWIN contains adaptation technique to adapt to drift and sustains its accuracy as opposed to VFDT and VFDRs which over time decrease their respective accuracies. In the second example, LED dataset on Figure 2, the sudden concept drift is highlighted by vertical line at 500,000 where all classifiers except ADWIN have noticeably increased error-rate. The increase is lower in the case of adaptive methods. AVFDR^u has the best reaction and ADWIN keeps its performance on very stable level.

Table 1. Prequential error rates of rule based classifiers using NB. For comparative purposes, we report the prequential error of VFDTc using NB classifiers at the leaves.

	Error rate % (variance)					
	VFDR^o	VFDR^u	VFDTc	AVFDR^u	AVFDR^o	ADWIN
LED	30.67 (0.53)	27.17 (0.06)	29.35 (0.11)	**26.65 (0.01)**	28.79 (0.04)	27.63 (0.00)
RBF(0,0)	20.43 (2.62)	**11.90 (2.48)**	18.6 (3.74)	12.07 (2.1)	21.99 (0.63)	19.05 (4.70)
RBF(50,0.001)	68.74 (6.48)	62.15 (0.83)	69.37 (4.93)	57.63 (19.05)	60.27 (2.34)	**52.50 (13.1)**
RBF(50,0.0001)	49.74 (1.19)	37.62 (2.82)	54.54 (2.10)	**33.60 (0.86)**	34.93 (2.18)	43.81 (1.15)
SEA	15.64 (0.34)	14.68 (0.31)	15.51 (0.13)	14.14 (0.41)	15.6 (0.03)	**13.48 (0.08)**
Hyperplane	14.63 (0.47)	13.24 (0.53)	15.01 (0.76)	**12.64 (0.31)**	14.72 (0.43)	12.76 (0.18)
Waveform	24.43 (0.38)	19.07 (0.35)	20.27 (1.07)	**18.90 (0.61)**	23.40 (013)	19.96 (0.02)
Intrusion	7.0E-4	**2.9E-4**	5.4E-4	4.3E-4	5.0E-4	0.15
Covtype	18.88	**13.65**	15.21	15.34	15.35	19.53
Elec	28.75	25.53	27.58	25.37	24.38	**19.56**
Airlines	33.46	**32.46**	33.67	32.94	33.41	37.33
Connect-4	27.67	27.73	**26.41**	26.86	27.38	27.96
Activity	34.47	27.16	38.54	**17.03**	18.91	42.48
Average rank	5	2.54	4.31	1.78	3.62	3.77

Using artificial datasets with known position of drifts is especially beneficial for closer examination of the drift detection. In the case of SEA datasets we observed for AVFDR^u that first drifts appeared on average after 570, 502 and 285 examples of the second, third and fourth concept respectively. This reflects the increasing difference between the concepts. There were various numbers of other drifts following with various delays, which is expected since a rule reflects only local change of the feature space and might react much later depending on the examples arriving. This is a good result. A standard NB with SPC, a very good classifier for this task, needed on average 1328, 1024 and 461 examples to detect the (global) change. AVFDR^o was not that successful. In some cases the some drifts were not detected at all or very late. LED datasets were also generated with known sudden concept drift. The results are not that straightforward. The noise, irrelevant attributes and relatively high number of classes in combination with higher sensitivity in local detection caused that many rules, that were generated, were pruned before they were precise enough. Besides these false drifts many of the actual local drifts were detected mostly between 30-100 examples.

The second group of datasets is real world data. The presence of concept drift is unknown but can be expected. In these datasets, VFDR^u mostly performs well since change, if there is any, is probably very slow. Nevertheless, the size of the decision model grows one order of magnitude larger than the other rule learning systems. AVFDR^u still presents very competitive performance in most of the real world sets with the advantage of smaller rule set size. The example of prequential error evolution throughout the learning process in real world dataset is plotted on Fig. 3. The noticeably worst performing classifier in Airlines dataset is ADWIN. The other classifiers perform relatively similarly with the unordered rules sets being slightly better than the rest.

The results from all the tests are collected in Tab. 1 with averaged score ranks. In order to compare multiple classifiers on multiple datasets we applied the Friedman test [31] and post-hoc Nemenyi test [10], based on average ranks achieved on all the datasets, with p-value of 0.05. At this level AVFDR^u was significantly better than VFDR^o

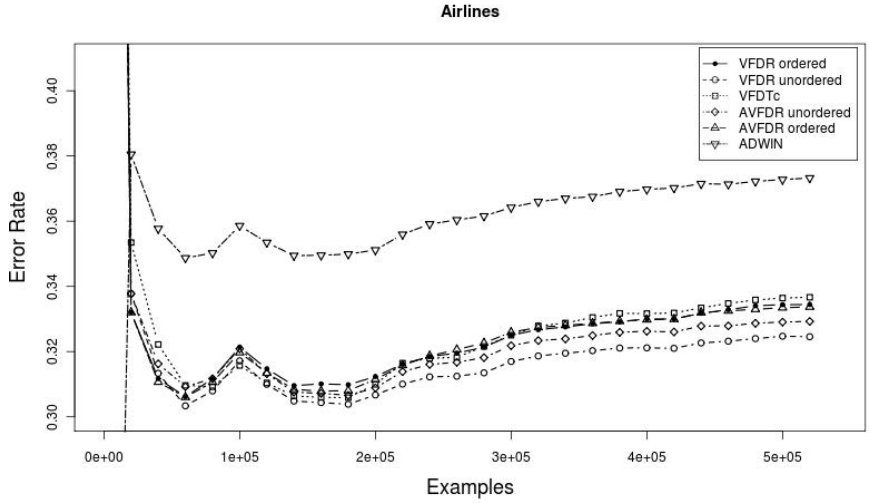

Fig. 3. The prequential error evolution throughout the learning process in real-world Airlines dataset where the presence of drift is not known

and VFDTc. VFDR^u was significantly better than VFDR^o. With p-value of 0.1 AVFDR^u was also significantly better than ADWIN.

Conceptually, each rule is closely related to a path in a decision tree. The consequent of a rule plays similar role as a leaf. They contain practically the same information and possess the same functionality. Therefore, the size of the classifiers can be compared by considering the number of rules in a set and number of leaves in a tree. The Table 2 shows the number of rules/leaves (the average number in case of the artificial datasets) that were available at the end of the prequential process. Note that among these classifiers only AVFDRs are able to decrease the number over the learning process. VFDR^u produces the largest classifier, which is expected due to its design. But it can be observed that by using the drift detecting technique AVFDR is able to effectively reduce the set while having good accuracy. Only in Covtype, complex real dataset, the much larger set of VFDR^u achieved notably better result.

In order to analyze the time we provide the prediction and learning times in prequential evaluation process of the classifiers in relation with the learning time of VFDTc in the right part of Tab. 2. Since ADWIN was running on different framework it is not included in this evaluation. We can conclude that the times of ordered rule sets are practically the same as in case of VFDTc. For the unordered rules of VFDR^u the time is dependent on the number of rules, because as opposed to tree search the whole rule set is scanned for rules that cover an example. The advantage of AVFDR is that it removes potentially incorrect rules thus keeps the set smaller and learning and prediction times are reduced.

Table 2. The number of rules in decision rule learners at the end of the learning process. For comparative purposes we present also the number of leaves generated by VFDTc. Relative times refer to the ratio of the learning time of rule learners with respect to the learning time of VFDTc.

	Size of Decision Models					Relative times			
	VFDR^o	VFDR^u	VFDTc	AVFDR^u	AVFDR^o	VFDR^o	VFDR^u	AVFDR^u	AVFDR^o
LED	26	3698	50	972	16	1.1	14.5	11.9	1.5
RBF(0,0)	33	2012	81	1255	22	0.8	14.2	11.7	1.1
RBF(50,0.001)	21	6029	52	1	2	1.3	52.3	1.7	1.3
RBF(50,0.0001)	56	745	80	6	1	0.9	9.8	0.9	0.9
SEA	13	68	28	35	12	1.1	1.5	1.1	1
Hyperplane	79	1348	353	514	62	1.2	12.2	5.4	1.2
Waveform	13	637	12	129	4	0.8	9.5	2.7	0.9
Intrusion	28	424	642	5	5	1	6.6	0.9	1.1
Covtype	79	844	393	23	16	1.4	7.8	1.3	0.9
Elec	22	71	38	4	6	1.1	2.7	0.9	0.5
Airlines	59	501	1096	157	34	1.1	2.7	0.9	0.5
Connect-4	14	21	31	7	4	1	1.4	0.9	1.2
Activity	61	4278	85	31	7	1	36.9	0.9	0.8

5 Conclusions

This work presented Adaptive Very Fast Decision Rule classifier, which incrementally induces rules for each class in a decision problem. It requires only one scan of data and provides any-time classification model that is capable of fast adaptation to changes in data. The adaptation is achieved by exploiting the modularity and independence of single rules within the rule set and assigns an error based on a drift detection method to each rule. Whenever the quality of a rule decreases significantly, the rule is removed from the set. This approach works not only as fast adaptation tool but also as rule set pruning method.

The proposed AVFDR method was tested on multiple artificial and real-world datasets and achieved very promising results. Especially the unordered version, AVFDR^u, was very successful in fast adaptation in artificial sets with known drift. In all the datasets AVFDR^u achieved very good ranks and was significantly better than state-of-the-art Hoeffding Tree algorithm VFDTc. The extension effectively reduced the rule set size for time changing data streams which also reduced the learning and prediction in prequential evaluation process.

Acknowledgments. The authors acknowledge the financial support given by the project Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA/098355/2008) funded by FCT. Petr Kosina also acknowledges the support of Fac. of Informatics, MU, Brno.

References

1. Baena-Garcia, M., Campo-Avila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early drift detection method. In: 4th International Workshop on Knowledge Discovery from Data Streams, ECML-PKDD, Berlin, Germany (2006)

2. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., Wiswedel, B.: Knime - the konstanz information miner: version 2.0 and beyond. SIGKDD Explor. Newsl. 11, 26–31 (2009)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research, JMLR (2010)
4. Bifet, A., Gavaldà, R.: Adaptive Learning from Evolving Data Streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009)
5. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 139–148. ACM, New York (2009)
6. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees, 1st edn. Chapman and Hall/CRC (1984)
7. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning 3, 261–283 (1989)
8. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. pp. 151–163. Springer (1991)
9. Cohen, W.: Fast effective rule induction. In: Prieditis, A., Russel, S. (eds.) Machine Learning, Proc. of the 12th International Conference. Morgan Kaufmann (1995)
10. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
11. Domingos, P.: Unifying instance-based and rule-based induction. Machine Learning 24, 141–168 (1996)
12. Domingos, P.: Mining high-speed data streams. pp. 71–80. ACM Press (2000)
13. Ferrer, F., Aguilar, J., Riquelme, J.: Incremental rule learning and border examples selection from numerical data streams. Journal of Universal Computer Science 11(8), 1426–1439 (2005)
14. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
15. Gama, J., Fernandes, R., Rocha, R.: Decision trees for mining data streams. Intelligent Data Analysis 10, 23–45 (2006)
16. Gama, J., Kosina, P.: Learning decision rules from data streams. In: Proc. of the 22nd International Joint Conference on Artificial Intelligence, pp. 1255–1260. AAAI, Menlo Park (2011)
17. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Proc. of the 9th International Conference on Knowledge Discovery and Data Mining, ACM Press, New York (2003)
18. Gama, J., Sebastiäo, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 329–338. ACM, New York (2009)
19. Gama, J.: Knowledge Discovery from Data Streams. Chapman and Hall/CRC (2010)
20. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with Drift Detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
21. E., Grant, Leavenworth, R.: Statistical Quality Control. McGraw-Hill (1996)
22. Hinkley, D.: Inference about the change point from cumulative sum-tests. Biometrika 58, 509–523 (1970)
23. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106. ACM, New York (2001)
24. Ikonomovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. Data Min. Knowl. Discov. 23, 128–168 (2011)

25. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis 8(3), 281–300 (2004)
26. Kosina, P., Gama, J.: Very fast decision rules for multi-class problems. In: Proc. of the 2012 ACM Symposium on Applied Computing, pp. 795–800. ACM, New York (2012)
27. Maloof, M., Michalski, R.: Incremental learning with partial instance memory. Artificial Intelligence 154, 95–126 (2004)
28. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
29. Rivest, R.: Learning decision lists. Machine Learning 2, 229–246 (1987)
30. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine Learning 1, 317–354 (1986)
31. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 4th edn. Chapman & Hall/CRC (2007)
32. Street, W.N., Kim, Y.: A streaming ensemble algorithm SEA for large-scale classification, pp. 377–382. ACM Press (2001)
33. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235. ACM Press (2003)
34. Weiss, S., Indurkhya, N.: Predictive Data Mining, a practical Guide. Morgan Kaufmann Publishers (1998)
35. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23, 69–101 (1996)