Data Min Knowl Disc
DOI 10.1007/s10618-013-0340-z

Very fast decision rules for classification in data streams

Petr Kosina · João Gama

Received: 31 July 2012 / Accepted: 11 November 2013

Abstract Data stream mining is the process of extracting knowledge structures from continuous, rapid data records. Many decision tasks can be formulated as stream mining problems and therefore many new algorithms for data streams are being proposed. Decision rules are one of the most interpretable and flexible models for predictive data mining. Nevertheless, few algorithms have been proposed in the literature to learn rule models for time-changing and high-speed flows of data. In this paper we present the very fast decision rules (VFDR) algorithm and discuss interesting extensions to the base version. All the proposed versions are one-pass and any-time algorithms. They work on-line and learn ordered or unordered rule sets. Algorithms designed to work with data streams should be able to detect changes and quickly adapt the decision model. In order to manage these situations we also present the adaptive extension (AVFDR) to detect changes in the process generating data and adapt the decision model. Detecting local drifts takes advantage of the modularity of the rule sets. In AVFDR, each individual rule monitors the evolution of performance metrics to detect concept drift. AVFDR prunes rules whenever a drift is signaled. This explicit change detection mechanism provides useful information about the dynamics of the process generating data, faster adaptation to changes and generates more compact rule sets. The experimental evaluation demonstrates that algorithms achieve competitive results in comparison to

Responsible editor: Johannes Fürnkranz.

P. Kosina · J. Gama (✉)
LIAAD - INESC TEC, Porto, Portugal
e-mail: jgama@fep.up.pt

P. Kosina
Faculty of Informatics, Masaryk University, Brno, Czech Republic

J. Gama
Faculty of Economics, University of Porto, Porto, Portugal

alternative methods and the adaptive methods are able to learn fast and compact rule sets from evolving streams.

Keywords Data streams · Classification · Rule learning · Concept drift

1 Introduction

With the increasing role of computers in our everyday life we obtain more and more potentially useful information that can aid us in tasks such as supporting key decisions, finding interesting patterns, and discovering anomalies. The human brain can hardly match the capacities of computers in processing all the available information, which brings new challenges for machine learning. Therefore, computers are trained for tasks such as recognizing Spam, helping with medical diagnosis, predicting electricity consumption, and discovering frauds.

Recently, machine learning has undergone a change in the focus and application of its techniques. More and more attention shifts from off-line mining of static data towards ubiquitous mining and real-time applications dealing with data represented by streams. Data streams refer to a process where instances arrive continuously and possibly infinitely over time. Therefore, we might not have all the data available at once and consequently the data require sequential access rather that random access. New approaches and stream extensions to off-line methods emerge in order to improve performance under constraints given by the stream environment. The approaches for streams need to possess special characteristics such as to be 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 constraints (Gama 2010).

A common behavior in streaming data is that it evolves over time with unknown dynamics. This evolving nature of data might change the functional mapping between attributes and classes, a phenomenon known as concept drift. This is an issue that up-to-date streaming learning system must take into account. Almost everything in this world changes and so can the concepts in data, whenever they are being observed for long enough time. The influence of seasonal change, economic change, health conditions or wear of machinery can be the cause of decreasing quality of the models built based on previous observations. Therefore, fast model adaptation is an advantage for decision learning in most real world problems.

In machine learning, and especially in the classification tasks, various types of decision trees are among the most popular tools that are widely used. They are hierarchical structures with decisions in nodes and labels in leaves. Not only do they provide very good predictive capabilities, but also evince high degree of interpretability due to their comprehensible visualization. Consequently, the paths from root of the tree to the leaves can be rewritten into set of unordered IF-THEN rules. Those rules capture the main characteristics of the decision problem and its relevant features. Moreover, each unordered rule can be handled independently of the others in the set. A common and straightforward approach is to generate decision rules from decision trees. Another way to obtain the rule set is to induce them directly from data, similarly as when learning the tree.

Indeed, the popularity, quality and interpretability of decision trees are preserved in the stream mining scenarios. Among the best known and most used models for data stream classification are the algorithms based on Hoeffding trees (Domingos and Hulten 2000) (HT). These incremental algorithms automatically adapt to concept drift just by expanding the tree. The adaptation of HT, however, is rather slow. Faster adaptation might be achieved by employing explicit drift detection methods (Gama et al. 2004; Baena-Garcia et al. 2006; Hinkley 1970). Nevertheless, using explicit detection usually requires rebuilding the current tree, which could be computationally expensive. The aforementioned rule sets did not attract much attention from the community in the context of stream mining as opposed to the trees. Yet they have the advantage of having individual rules that can be managed independently. Therefore, in decision rules the implicit adaptation feature of the trees remains. Moreover, the set of rules can be altered more easily. Instead of rebuilding the classifier from scratch or executing a complicated change of the structure in the tree, individual rules which are considered outdated can be simply removed. The explicit change detection of individual rules can provide much faster adaptation mechanisms to changes and can also serve as rule set pruning mechanism.

This article is based on three conference papers. The first two introduced the algorithm for learning decision rules from data streams (Gama and Kosina 2011) and proposed a modification to the algorithm (Kosina and Gama 2012b). The work presented a classifier with different evaluation criteria and it was redesigned to be able to learn rules for each class. These algorithms were not specifically designed for time changing data and were not tested under such conditions. The adaptation of the rule learning algorithm to a time changing data was the aim of the third paper (Kosina and Gama 2012a). This work summarizes previous versions, proposes another modification of the algorithm, and provides more extensive and in-depth evaluation.

The paper is organized as follows. Section 2 discusses the related work in rule learning and handling time changing data. The `VFDR` algorithm and its extensions are presented in Sect. 3. The proposed algorithms are evaluated and compared to other stream classification algorithms in Sect. 4. The lessons learned and future work are discussed in Sect. 5.

2 Related work

This section presents the related works in the stream classification, the rule learning algorithms and their different approaches to multi-class problems, and the last part focuses on drift detection in time-changing data.

2.1 Stream classification algorithms

One of the most influential algorithms for classification in data streams was proposed by Domingos and Hulten (2000). His work presented what is known as very fast decision tree (`VFDT`), an on-line version of a decision tree classifier appropriate for data stream processing. `VFDT` is learned by recursively replacing leaves with decision nodes. Each leaf stores the sufficient statistics about attribute-values. The sufficient statistics are required by a heuristic function that evaluates the merit of split-tests based on the attribute-values. When an example is available it traverses the tree from the root to a leaf evaluating the corresponding attribute at each node, and following the branch according to the attribute's value of the example. When the example reaches a leaf the sufficient statistics are updated. The tree replaces leaves with test nodes when the Hoeffding bound condition is satisfied. The popularity of VFDT attracted a lot of attention and different extensions and improvements were proposed. Hulten et al. (2001) presented CVFDT, a decision tree learner for mining data streams with non-stationary distributions. CVFDT learns a model consistent with a sliding window of recent examples. When the concept is changing and a split that was previously selected would no longer be the best, the CVFDT algorithm starts learning an alternate subtree with a new best attribute as its root. The subtree replaces the original one when it becomes more accurate. The algorithm keeps the model up-to-date when there are large and frequent changes in a concept. Gama et al. (2003) introduced VFDTc algorithm that extended the base incremental tree learning algorithm in two ways. The first extension was to equip leaves with binary search trees in order to handle numerical attributes. The second improvement introduced the use, in the leaves of the tree, of Naive Bayes (NB) classifiers trained on the examples that fall into the leaf. This functionality significantly improves the predictive accuracy of the tree. The possible weakness that the statistics kept within the node using the binary tree can be relatively large when the examples have many unique values for the numerical attribute.

A recent system called IBLStreams proposed by Shaker and Hüllermeier (2012) introduces instance based learning classifier for data streams. Instance based learning is inherently incremental. This is a memory based (or lazy) approach to learning that consists of adding or removing instances from the model. IBLStream selects the case base (instances that form the classifier) based on temporal relevance, spatial relevance, and consistency. The idea is that the most recent examples are the most relevant; it aims to have more or less uniform coverage of the instance space, and to remove data that seem to be inconsistent with the current concept. New examples are added to the case base and the redundant (neighboring) examples are checked for removal. The neighborhood is given by k_{cand} nearest neighbors. The most recent are excluded from the candidates for removal, because it is difficult to distinguish between noise and examples of potentially new concept. The classifier used a drift detection technique (statistical process control, SPC, described later in this paper) computed over last 100 training instances to detect abrupt changes. If a change is detected IBLStream removes larger number of instances from the base. The extent of the change (%) is estimated by the difference of minimum classification error over last 100 training instances and the error over last 20 training instances. The examples to be removed are then chosen at random according to a distribution which is spatially uniform but temporally skewed. The advantage of IBLStream are the flexibility of the learner and the adaptation capabilities with very good accuracy. The weakness of the classifier can be the classification time and interpretability of the model.

2.2 Rule learning

A widely used strategy for learning rules consists of deriving rules from decision trees, as it is done in Quinlan (1993). Any decision tree can be easily transformed into a collection of rules. Each rule corresponds to the path from the root to a leaf, and there are as many rules as leaves. This process generates a set of rules with the same complexity as the decision tree. However, it has been shown that the antecedents of individual rules may contain irrelevant conditions. C4.5rules (Quinlan 1993) uses an optimization procedure to simplify conditions. Previous empirical studies (Quinlan 1993) have demonstrated that the set of rules is both simpler and more accurate than the initial tree. Frank and Witten (1998) present a method for generating rules from decision trees without using global optimization. The basic idea is to generate a decision tree in a breadth-first order, select the best rule, remove the examples covered by the rule and iteratively induce further rules for the remaining instances.

There are several algorithms in the literature for learning decision lists[1] (Rivest 1987; Clark and Niblett 1989; Cohen 1995; Domingos 1996; Weiss and Indurkhya 1998) with different capabilities and approaches to handle multi-class problems. `CN2` (Clark and Niblett 1989) is one of the first rule learning systems. It uses a top-down, general to specific approach. `CN2` evaluates every possible complex, i.e., conjunction of attribute tests (*if-conditions*) of a rule, based on information-theoretic entropy measure. This classifier is able to learn multiple class-problems. Each rule predicts the most common class of the examples covered by the rule. In its improved version, presented in Clark and Boswell (1991), the main loop of rule set algorithm learn a model (a rule set) for each class separately. In each iteration, the examples of one class are chosen as positive and all the others are labeled as negative examples. Only positive examples that satisfy a learned rule are removed from the training set for next iteration of the rule search. Similarly, `RIPPER` (Cohen 1995) decomposes the problem in one vs. all fashion. The classes are ordered in increasing order of their frequency in the training set. After learning rules that separate the minority class, covered examples are removed and the algorithm proceeds with the next class. This process stops when a last single class remains which is then assigned as default class. Another technique to reduce multi-class problem is pairwise classification or round-robin (Fürnkranz 2001). The idea is to learn a classifier for each pair of classes thus transforming the original c-class problem into $\frac{c(c-1)}{2}$ two-class problems. All binary classifiers are used to classify test examples. The predictions are aggregated using uniform voting. The predictions of the classifiers learned on the true class of an example are expected to outweigh those that were trained to recognize examples of different classes. If rule learning algorithm is not *class-symmetric* (i.e., problem of discriminating class i from class j is different than discriminating j from i), the authors suggest double round robin approach. The classifiers are learned for both problems thus obtaining in total $c(c-1)$ classifiers. The recent book (Fürnkranz et al. 2012) presents a comprehensive description of the most relevant issues and works in rule learning system.

[1] Note that decision lists are ordered rule sets.

Former incremental rule learners include STAGGER (Schlimmer and Granger 1986), the first system designed expressly for coping with concept drift, the FLORA family of algorithms (Widmer and Kubat 1996) with FLORA3 being the first system able to deal with recurring contexts, and the AQ-PM family (Maloof and Michalski 2004). All these systems are able to incorporate new information, but they need to maintain in memory previous examples and cannot deal with high-speed data streams. The first rule learner designed for processing data streams is the system Facil (Ferrer et al. 2005). Facil uses a bottom-up, specific to general, search strategy, similar to AQ-PM. The decision model consists of a set of rules and, for each rule, a set of positive and negative examples that define the borders of the rule. The core of this approach is that rules may be inconsistent by storing positive and negative examples which are very near one another (border examples). This approach is similar to the AQ11-PM system (Kolter and Maloof 2003; Maloof and Michalski 2004), which selects positive examples from the boundaries of its rules (hyper-rectangles) and stores them in memory. When new examples arrive, AQ11-PM combines them with those held in memory, applies the AQ11 algorithm to modify the current set of rules, and selects new positive examples from the corners, edges, or surfaces of such hyper-rectangles (*extreme* examples). The idea of creating new rules based on border examples is similar to the double induction approach introduced by Lindgren and Boström (2004). Instead of using frequency based or NB based decision, when more rules predict different labels for an example, they proposed to induce a new set of rules from the examples covered by the conflicting rules. Then the new rules are used to classify the examples that are covered by the conflicting rules. Facil uses a forgetting mechanism that can be either explicit or implicit. Explicit forgetting takes place when the examples are older than a user defined threshold. Implicit forgetting is performed by removing examples that are no longer relevant as they do not enforce any concept description boundary. Facil

2.3 Adaptive methods

The stream mining community has already introduced many different approaches to deal with the phenomenon of concept drift. Approaches such as sliding windows and example weights (Klinkenberg 2004) are widely used to maintain a classifier consistent to the most recent data. Bifet and Gavalda (2009) proposed the ADWIN algorithm, a detector and estimator which automatically adapts to current rate of change by keeping the window of recent examples of variable length. It employs 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 method, e.g., based on error rate (Gama et al. 2004) or distance between classification errors (Baena-Garcia et al. 2006), and forgetting mechanism.

As pointed out by Wang et al. (2003), 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. Bifet et al. (2009) presented two new decision tree ensemble methods: ADWIN bagging and adaptive-size Hoeffding tree bagging. The former extends on-line bagging (Oza and Russell 2001) 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 HT of different maximum sizes since smaller trees adapt faster to changes and larger work better for long periods with little or no change.

3 Very fast decision rules algorithm

In this section we present the classification rule learning system for data streams—very fast decision rules VFDR and variants. The rule learning algorithm strives to provide a flexible classifier for data streams that would produce output easy to interpret and that could adapt quickly to changes in the underlying concept.

As in many other systems a rule in VFDR is an implication of the form $A \Rightarrow C$. The A part of a rule is a conjunction of literals, that is, conditions based on attribute values. 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 VFDR produces literals of the form $X_i = v_j$ where v_j is a value in the domain of X_i. The C part of a rule r is not a constant as in most of rule based systems, but the class is given by a function—either majority class or NB. This is the most distinctive feature of VFDR.

In the first part of this section we describe the base algorithm VFDR-Base. The following subsections describe modifications for multi-class problems and time evolving data streams.

3.1 The basic algorithm

The VFDR-Base algorithm is designed for high-speed data streams. It learns ordered or unordered rule sets. It needs only one scan of data and is able to provide any-time classifications.

3.1.1 Growing a set of rules

The algorithm begins with an empty rule set (RS) and a $default\ rule$ $\{\} \rightarrow \mathcal{L}$, where $\mathcal{L}$ is initialized to $NULL$. $\mathcal{L}$ is a data structure that contains information used to classify the test instances, and the sufficient statistics needed to expand the rule.

As already said, each learned rule (r) is a conjunction of literals, that are conditions based on attribute values, and an $\mathcal{L}_r$. 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 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 binary search tree 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 information maintained in $\mathcal{L}_r$ is similar to the sufficient statistics used by Gama et al. (2006).

The number of observations, after which a rule can be expanded or new rule can be induced, is determined by the Hoeffding bound. It guarantees that, with probability at least $1 - \delta$, the true mean of a random variable x with a range R will not differ from the sample mean of size N 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 this is done only after every N_{min} observations.

The set of rules (RS) is learned in parallel as described in Algorithm 4. We consider two cases: learning ordered or unordered set of rules. 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.

The expansion of a rule is done using Algorithm 2 that employs the aforementioned Hoeffding bound. For each attribute X_i the value of split evaluation function G is computed for each attribute value v_j. The best and the second best literal merit named as g_{best} and g_{2best} respectively are used for the Hoeffding bound condition. If the best merit is better the second best with given confidence, i.e. satisfies condition $g_{best} - g_{2best} > \epsilon$, the rule is expanded with condition $X_a = v_j$ and the class of the rule is assigned according to the majority class of observations of $X_a = v_j$.

Algorithm 1: VFDR: Rule Learning Algorithm.

```
input  : $S$: Stream of examples
         $ordered\_set$: boolean flag
         $\delta$: user defined confidence level
output: $RS$: Set of Decision Rules
begin
    Let $RS \leftarrow \{\}$
    Let $defaultrule\ \mathcal{L} \leftarrow \emptyset$
    foreach example $(\mathbf{x}, y_k) \in S$ do
        foreach Rule $r \in RS$ do
            if $r$ covers the example then
                Update sufficient statistics of $r$
                $RS \leftarrow RS - \{r\}$
                $RS \leftarrow RS \cup ExpandRule(r, \delta)$
                if $ordered\_set$ then
                    BREAK
        if none of the rules in $RS$ covered example then
            Update sufficient statistics of the $default\ rule$
            $RS \leftarrow RS \cup ExpandRule(default\ rule, \delta)$
```

Algorithm 2: `ExpandRule-BASE`: Rule Expansion.

```
input  : $r$: One Rule
         $\delta$: user defined confidence level
output: $r$: Expanded Rule
begin
    Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
    Find the best $g_{best}$ and second best $g_{2best}$ attribute merit
    if ($g_{best} - g_{2best} > \epsilon$) then
        Extend $r$ with a new condition based on the best attribute $X_a = v_j$
        Reinitialize sufficient statistics of $\mathcal{L}_r$
        $r \leftarrow r \cup \{X_a = v_j\}$
    return $r$
```

3.1.2 Classification strategies

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 with maximum estimated $p(c_k)$. 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. `NB` takes into account not only the prior distribution of the classes, but also the conditional probabilities of the attribute-values given the class. The information available in each rule is therefore better utilized. Given the example $\mathbf{x} = (x_1, \ldots, x_j)$ and applying Bayes theorem, we obtain:

$$P(c_k|\mathbf{x}) \propto P(c_k) \prod P(x_j|c_k).$$

Using `NB` in `VFDT` like classification algorithms is a well-known technique since it was introduced in Gama et al. (2003). 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 of highly informed predictions based on `NB` classification.

There are different ways to determine and rank the predictions and their associated confidences from the rule set in case multiple rules fire. The set of rules learned by `VFDR-Base` can employ three different classification strategies as defined in PMML (Data Mining Group 2011): *First Hit*, *Weighted Sum*, and *Weighted Max*.

– *First Hit* uses the first firing rule to determine the predicted class, and the confidence is the weight of that rule.

- *Weighted Sum* calculates the total weight for each class by summing the weights for each firing rule. The prediction with the highest total weight is then selected. The confidence is the total weight of the winning class divided by the number of firing rules.
- *Weighted Max* selects the firing rule with the highest weight. The confidence returned is the confidence of the selected rule.

If two firing rules have the same weight (Weighted Max), or two or more classes are assigned the same weight (Weighted Sum) the winner is chosen alphabetically. As in Clark and Boswell (1991), the ordered rules use the *First Hit* strategy, while the unordered rules use the *Weighted Sum* strategy[2].

3.2 One versus all rule learning

This modified version of the base `VFDR-Base` employs one vs. all strategy for multi-class learning (`VFDR-OA`). In this strategy, the 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. This version is able to produce more rules in one call of `ExpandRule`. The number of rules induced from one rule r in RS in such a call is at most $|C_r|$.

The process to select new conditions for a rule works as follows. For each attribute X_i the value of gain function $G = Gain(r', r)$ adopted from `FOIL` (Quinlan 1991) 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 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 zero 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:

$$GainNorm(r', r) = \frac{Gain(r', r)}{N_+ \times \left(-\log_2 \frac{N_+}{N}\right)}.$$

In the case of expanding a non-default rule, that is a rule which already contains conditions, the algorithm works as follows. The rule was induced for a certain class that was considered as positive. To keep this class of interest in the rule set, the class is maintained as positive for the next computation of the merit. More specifically the

[2] *Weighted Max* generally did not produce results much different from the *Weighted Sum* therefore we opted for not including this setting in the results.

difference is that an ordered set, in Algorithm 7, considers only the positive class. The unordered rule set in Algorithm 8 additionally computes the merits of other classes considered as the positive and can expand the rule also with the best condition for such classes. The additional expansions are allowed only when the rule has already been expanded with the original class at that call of `ExpandRule`. The *default rule* does not have any previous class considered as positive thus the search for new rules proceeds until the merit for all possible classes are checked.

In other words, the process creates multiple rules for different classes marked as positive, but not necessarily for all the available classes in one call. The advantage is that the algorithm is able to learn more rules and more specialized (complex) rules from fewer examples. As a result it can achieve higher accuracy at the possible cost of producing larger sets.

The search for the best and second best values considers the value of *GainNorm* **for a given class**. If g^{k}_{best} is the true best gain measure, i.e., satisfies condition $g^{c_k}_{best} - g^{c_k}_{2best} > \epsilon$ **for a given class** c_k, the rule is expanded with condition $X_a = v_j \Rightarrow c_k$.

Algorithm 3: `ExpandRule-OA-OR`: Expanding Ordered Rule.

```
input  : r: One Rule;
         δ: user defined confidence level
         τ: Constant to solve ties
output: NR: New Rules Set;
begin
    Let NR ← {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
    Find the best $g_{best}$ and second best $g_{2best}$ attribute merit for $c_r$
    if $g^{c_r}_{2best} - g^{c_r}_{best} > \epsilon$ or $\epsilon < \tau$ then
        Extend r with a new condition based on the best attribute $X_a = v_j$
        Reinitialize sufficient statistics of r
    return NR
```

3.2.1 Classifier with multiple sets

Another modification further develops the one vs. all approach in rule stream learning. While the previously described version computed the merit for each class in one vs. all fashion, the following learner creates a separate rule set for each of the classes in data. Consequently the learning and prediction phase reflect certain differences. This modification is denoted as `VFDR-MS` (multiple sets). The aim of these modifications is to have rules for all classes with less exhaustive expansion search.

Learning phase The classifier contains multiple rule sets, i.e., rule *subsets* and one *default rule*. For each class in data there is a *subset* characterized by it (*subset class*) The rule *subset* distinguishes between its *subset class*, which is considered as a positive class, and all the other classes that are considered as a *negative class*, i.e., each $\mathcal{L}_r$ of a rule r in a *subset* contains information about a two-class problem. When a new

Algorithm 4: `ExpandRule-OA-UN`: Expanding Unordered Rule.

```
input  : r: One Rule;
         δ: user defined confidence level
         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 $r$
    Find the best $g_{best}$ and second best $g_{2best}$ attribute merit for $c_r$
    if $g_{best}^{c_r} - g_{2best}^{c_r} > \epsilon$ 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
            Find the best $g_{best}$ and second best $g_{2best}$ attribute merit for $c_k$
            if $g_{best}^{c_k} - g_{2best}^{c_k} > \epsilon$ or $\epsilon < \tau$ then
                create new $r'$ by extending $r$ with a new condition $X_a = v_j$ and class $c_k$
                $NR \leftarrow NR \cup \{r'\}$
        Reinitialize sufficient statistics of $r$
    return NR
```

example arrives, it updates statistics of rules that cover the example in all the *subsets*. If the class of the example is the same as the *subset class* the example is used as is. If the classes differ, the class of the example for learning such a rule is changed to *negative class* label. Therefore, a rule grows only when the statistics for the *subset class* indicate the expansion.

The *default rule* of the classifier keeps sufficient statistics covering all classes, i.e., $\{\} \rightarrow \mathcal{L}$ stores summarized information about a multi-class problem. A new rule can be induced for multiple *subsets* at one time. More specifically, every time *default rule* observes N_{min} examples, the classifier computes the `FOIL` gain for each class as positive and the others as negative so that each *subset* can possibly obtain a new rule.

This approach represents a balanced strategy. In general, the rule set grows larger with fewer examples than the basic algorithm, but not as large as the `VFDR-OA` version. *Prediction phase* The rule learner with multiple sets uses slightly modified approach to obtain predictions. Similarly to previous versions the individual rules from each of the rule *subsets* can use either majority class (*MC*) prediction or `NB` prediction. When the class predicted by (*MC*) or (*NB*) does not correspond to the *subset class* the weight of such firing rule is not considered that is it is treated as if the rule did not cover the example. In other words, if the class predicted by the individual rule is the *negative class*, it indicates that the rule is not appropriate to predict the class of a given example. The rule *subset* itself then provides prediction of its *subset class* supported by the weight given by the rules that covered the example. *Subset* weights are determined based on the three strategies for rule learning: First Hit, Weighted Max, or Weighted Sum. Then the final prediction of the classifier is selected by using the *subset class* with maximum weight. If none of the rules from any of the rule *subsets* provide weight, i.e., there was no rule that would cover the example and predicted the class of its *subset*, then *default rule* provides the prediction based on *MC* or *NB*.

3.3 Adaptive very fast decision rules

In this section we present an extension to VFDR that can be applied to any of the previously described versions of the classifier. The aim of the AVFDR (adaptive VFDR) extension is the adaptation of the rule learner to the phenomenon of concept drift.

3.3.1 Rule adaptation in the presence of drift

Data streams are characterized by their evolving nature. The ability to detect and react to concept drift is crucial when modeling continuous flows of data generated by processes with unknown dynamics.

Adaptation in rule models is facilitated due to their modularity. As opposed to decision trees the set of rules offer the possibility to remove individual rules without the need for rebuilding the entire model. This characteristic is very important for incremental learning from evolving data, because it allows faster adaptation of the model.

The previous algorithms were adapting only implicitly by inferring new rules and specializing the existing ones. AVFDR extension embeds an explicit drift detection into the learning process as presented in Algorithm 9. In addition to $\mathcal{L}_r$, a drift detection mechanism tracks performance of each rule r during learning. The method employed in AVFDR is the SPC (Gama et al. 2004) (Statistical Process Control) described in Algorithm 10. 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 SD. Every time a new example i is covered by the rule those values are updated when

$$p_i + s_i < p_{min} + s_{min}.$$

The learning process of a given rule can be in one of the following 3 states: *In-Control*, *Out-of-Control*, or in *Warning*. We follow the 3-sigma rule (Grant and Leavenworth 1996):

$$p_i + s_i \geq p_{min} + \alpha \times s_{min},$$

where $\alpha = 2$ changes the state to *Warning* and $\alpha = 3$ signals that the *Out-of-Control* state is reached.

In the warning state, the rule stops learning until the status of the rule becomes again *In-Control*. The *default rule* stores the examples during warning in a short-term memory. 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 rule set. This control enables to keep the rule set up-to-date and prevent the rule set from excessive growth. The default rule's statistic is re-initialized with the examples from short-term memory if it reaches the *Out-of-Control* state.

Algorithm 5: AVFDR: Rule Learning Algorithm.

```
input  : $S$: Stream of examples
         $\delta$: user defined confidence level
         $ordered\_set$: boolean flag
output: $RS$: Set of Decision Rules
begin
    Let $RS \leftarrow \{\}$
    Let default rule $\mathcal{L} \leftarrow \emptyset$
    foreach example $(\mathbf{x}, y) \in S$ do
        foreach Rule $r \in RS$ do
            if r covers the example then
                /* Estimate Rule Status                                   */
                Status $\leftarrow SPC(r, \mathbf{x}, y)$
                if $Status == Out - of - Control$ then
                    $RS \leftarrow RS - \{r\}$
                else
                    if $Status == In - Control$ then
                        Update sufficient statistics of $r$
                        $RS \leftarrow RS - \{r\}$
                        $RS \leftarrow RS \cup ExpandRule(r, \delta)$
                        if $ordered\_set$ then
                            BREAK
        if none of the rules in $RS$ covered example then
            Update sufficient statistics of the default rule
            $RS \leftarrow RS \cup ExpandRule(default\ rule, \delta)$
```

3.4 Illustrative example

In this section we present the functionalities of the VFDR on a simple example for better understanding of the whole process. The illustrative problem is an artificial dataset with class given by the logical rule $(A \wedge B) \vee (C \wedge D)$, where the attributes $A, B, C, D \in \{true, false\}$ are randomly generated. The number of examples is 1,500.

The learned rule sets are presented in Tables 1 and 2 for unordered and ordered rule sets respectively. The different VFDR variants produce different rule sets due to the different search strategies for the best conditions.

From Table 1 we conclude that all the rules are consistent with the target concept. In this simple example, the basic approach differs only in one rule from VFDR-MS, although the rule was induced in slightly different way. VFDR-OA requires fewer examples to produce rule sets with larger coverage. Therefore, VFDR-OA on the same dataset generated larger rule set with more specialized rules, which is an advantage in more difficult tasks.

Table 2 presents ordered rule sets learned on the illustrative dataset. Note that when interpreting the ordered sets, special care needs to be taken because each rule implicitly includes the negation of the previous rules. All the sets are again consistent with the concept of learning data. VFDR-OA creates only two rules that describe the class True (T). The two rules are sufficient because the rest of the cases are treated by default rule

Algorithm 6: The SPC Algorithm

```
Input  : $r^j$: rule
         /* $i^{th}$ example covered by rule $r^j$                                  */
         Current example: $\mathbf{x}$, $y$
Output: Status $\in$ {In-Control, Warning, Out-of-Control}
begin
    Let $\hat{y} \leftarrow r^j(\mathbf{x})$
    Let $error_i^j \leftarrow L(\hat{y}, y)$
    Compute the mean of the error $p_i^j$ and variance $s_i^j$
    if $p_i^j + s_i^j < p_{min}^j + s_{min}^j$ then
        $p_{min}^j \leftarrow p_i^j$
        $s_{min}^j \leftarrow s_i^j$
    if $p_i^j + s_i^j < p_{min}^j + 2 \times s_{min}^j$ then
        /* In-Control                                                   */
        Status $\leftarrow$ 'In-Control'
    else
        if $p_i^j + s_i^j < p_{min}^j + 3 \times s_{min}^j$ then
            /* Warning Zone                                             */
            Status $\leftarrow$ 'Warning'
        else
            /* Out-of-Control                                           */
            Status $\leftarrow$ 'Out-of-Control'
    Return: Status
```

Table 1 Unordered rule sets generated by different versions of VFDR

VFDR-Base	VFDR-OA	VFDR-MS
$A = T \wedge B = T \rightarrow T$	$A = T \wedge B = T \rightarrow T$	$A = T \wedge B = T \rightarrow T$
$C = T \wedge D = T \rightarrow T$	$B = F \wedge D = F \rightarrow F$	$C = T \wedge D = T \rightarrow T$
$B = F \wedge C = F \rightarrow F$	$B = F \wedge C = T \wedge D = T \rightarrow T$	$B = F \wedge D = F \rightarrow F$
$A = F \wedge C = F \rightarrow F$	$B = F \wedge C = F \wedge D = T \rightarrow F$	$A = F \wedge C = F \rightarrow F$
	$A = F \wedge B = T \wedge C = F \rightarrow F$	
	$A = F \wedge B = T \wedge C = T \wedge D = T \rightarrow T$	
	$A = F \wedge B = T \wedge C = T \wedge D = F \rightarrow F$	

as F. Nevertheless, VFDR-MS searches for rules explaining all classes and that is why there are the two rules for T, but also two rules for F, leaving nothing to be resolved by the default rule.

In order to provide an illustration of VFDR learning process, we examine the growth of a rule set over time. The example shown in Table 3 presents the unordered rules learned by VFDR-MS. The upper part of the table corresponds to the stationary dataset. Each column represents a state of the rule set during the learning process. The rows labeled with numbers represent a time stamp specified by the number of training examples processed by the learner at the point when the rule set evolved—it either

Table 2 Ordered rule sets generated by different versions of VFDR

VFDR-Base	VFDR-OA	VFDR-MS
1.$B = F \wedge C = F \rightarrow F$	1.$A = T \wedge B = T \rightarrow T$	1.$A = T \wedge B = T \rightarrow T$
2.$A = T \wedge B = T \rightarrow T$	2.$C = T \wedge D = T \rightarrow T$	2.$C = T \wedge D = T \rightarrow T$
3.$C = F \rightarrow F$		1.$B = F \wedge D = F \rightarrow F$
4.$D = T \rightarrow T$		2.$A = F \wedge C = F \rightarrow F$

added a new rule, specialized existing one or removed a rule. At each presented point, new rules or rules that added a literal at that point are highlighted in bold. Note that the learner creates one rule for each class. Since the data set represents a two class problem and the attributes have only two values, new rules are added to both rule *subsets* and have the opposite values.

In our illustrative example we can proceed to a non-stationary data test. In order to illustrate the learning process on a non-stationary dataset we extend the learning dataset used above by adding another 1,500 examples for which the rule defining the target concept is $(A \wedge \neg B) \vee (C \wedge D)$. The evolution of the rule set after the first stationary 1,500 examples corresponds to the bottom part of the Table 3. When the drift appears, we can observe the process of reaction to the drift by removing incorrect rules and inducing new ones. Note that the rules describing the part of the space unaffected by the drift are kept in the rule set. Therefore, the rule set learner reveals the local changes. The comparison of prequential error-rate in Fig. 1 illustrates the faster adaptation of AVFDR even in this simple example, where the change in data is easily handled also by further growth of the rule set.

4 Experimental evaluation

In this section we analyze the performance of the proposed rule learning algorithms for data streams. We would like to answer following research questions:

- What is the behavior of the VFDR variants?
- How does the rule algorithm perform compared to stream classification algorithms on stationary datasets?
- What is the performance on non-stationary datasets compared to the data stream classifiers?

Since there are many combinations of the various versions, we first analyze the behavior of the different versions of VFDR. We chose one version as the best representative of the group and compare it to the streaming classifiers in further experiments.

Next, we compare the selected version of decision rules from Sect. 3, with and without explicit drift detection, against standard stream classifiers: a decision tree classifier VFDTc (Gama et al. 2003), a NB, and an instance based classifier IBLStream (IBLS), recently proposed by Shaker and Hüllermeier (2012). These experiments answer the research questions.

The following subsections provide information about experimental methodology and results from the experiments.

Table 3 Illustrative example of the rule set growth using `VFDR-MS`

Evolution of the rule set in the first part (stationary) of the stream					
200	244	314	318	393	731
$\mathbf{B = T \rightarrow T}$	$B = T \rightarrow T$	$\mathbf{A = T \wedge B = T \rightarrow T}$	$A = T \wedge B = T \rightarrow T$	$A = T \wedge B = T \rightarrow T$	$A = T \wedge B = T \rightarrow T$
			$\mathbf{C = T \rightarrow T}$	$\mathbf{C = T \wedge D = T \rightarrow T}$	$C = T \wedge D = T \rightarrow T$
$\mathbf{B = F \rightarrow F}$	$\mathbf{B = F \wedge D = F \rightarrow F}$	$B = F \wedge D = F \rightarrow F$	$B = F \wedge D = F \rightarrow F$	$B = F \wedge D = F \rightarrow F$	$B = F \wedge D = F \rightarrow F$
			$\mathbf{C = F \rightarrow F}$	$C = F \rightarrow F$	$\mathbf{A = F \wedge C = F \rightarrow F}$
Evolution of the rule set after the change					
1,511	1,566	1,647	1,771	2,036	
$A = T \wedge B = T \rightarrow T$		$\mathbf{B = F \rightarrow T}$	$\mathbf{A = T \wedge B = F \rightarrow T}$	$A = T \wedge B = T \rightarrow T$	
$C = T \wedge D = T \rightarrow T$	$C = T \wedge D = T \rightarrow T$	$C = T \wedge D = T \rightarrow T$	$C = T \wedge D = T \rightarrow T$	$C = T \wedge D = T \rightarrow T$	
		$\mathbf{B = T \rightarrow F}$	$B = T \rightarrow F$	$\mathbf{B = T \wedge C = F \rightarrow F}$	
$A = F \wedge C = F \rightarrow F$	$A = F \wedge C = F \rightarrow F$	$A = F \wedge C = F \rightarrow F$	$A = F \wedge C = F \rightarrow F$	$A = F \wedge C = F \rightarrow F$	

New rules or expanded rules are in bold

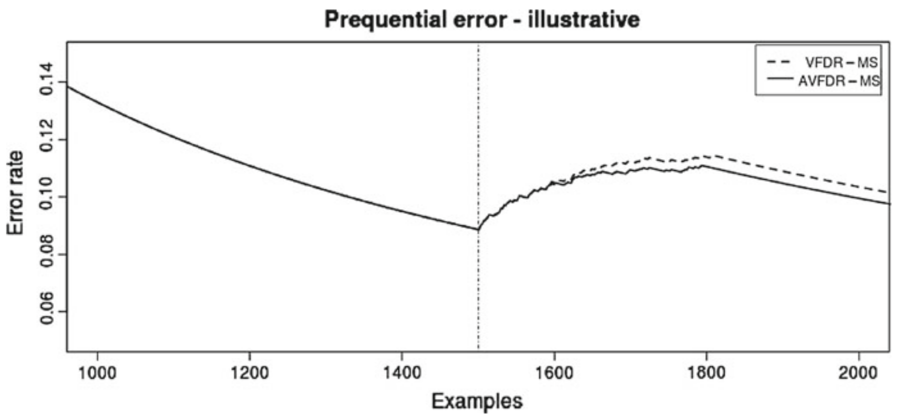

Fig. 1 Prequential error of `VFDR-MS` and `AVDFR-MS` classifiers in illustrative dataset. Adaptive version reacts faster to a change

4.1 Experimental methodology

We will present two sets of experiments. The first set of experiments aim to select the best representative from the `VFDR` algorithms. The main focus is on the second set of experiments, where the selected rule learning algorithm is compared to standard streaming algorithms.

4.1.1 Datasets

In order to compare the classifiers we use large scale artificial and real world datasets. The real world datasets were previously used in other works when testing on-line learning algorithms as they represent large datasets and may contain drifts. The main characteristics of the datasets are summarized in Table 4. A more detailed description of the datasets appears in the Appendix 1. The artificial datasets can be generated with or without a concept drift; therefore they can be used for evaluation on stationary and non-stationary data.

4.1.2 Algorithms

We use the notation `VFDR` and `AVFDR` for the version without drift detection and adaptive version respectively. The suffixes `-Base`, `-OA`, and `-MS` refer to the base version, the one vs. all and multiple class versions, and suffixes `-UN` and `-OR` refer to unordered and ordered respectively. `VFDTc+SPC` and `NB+SPC` stand for the decision tree and `NB` classifiers with `SPC` drift detection method respectively. All algorithms were implemented in Java as an extension for KNIME (Berthold et al. 2009) except for `IBLS`, which is implemented in MOA framework. The common parameters for `VFDR` and `VFDTc` are confidence $\delta = 0.000001$, tie breaking constant $\tau = 0.05$, $N_{min} = 200$ and the functional leaves use `NB` after observing at least one example. These parameters are selected based on default values in MOA. `IBLS` is used with default parameters except for using AdaptK parameter.

Table 4 Overview of the datasets

Dataset	# Examples	# Attributes	Attribute types	Noise	# Classes
Hyperplane	100,000	10	Continuous	Yes (5 %)	2
LED	200,000	24	Categorical	Yes (10 %)	10
SEA	60,000	3	Continuous	Yes (10 %)	2
RBF	100,000	10	Continuous	No	2
Waveform	100,000	21	Continuous	Yes	3
Airlines	539,383	7	Mixed	NA	2
Bank	45,211	16	Mixed	NA	2
Connect-4	67,557	42	Categorical	No	3
Elec	45,312	8	Mixed	NA	2
ForestCovtype	581,012	54	Mixed	NA	7
Intrusion	4,898,431	41	Mixed	NA	5
Pokerhand	829,201	10	Categorical	No	10
Spam	9,324	500	Categorical	NA	2

4.1.3 Performance metrics

The evaluation method in the experiments is the predictive sequential (prequential) method (Gama et al. 2009). In prequential evaluation whenever a training example is available, the classifier first makes a prediction. After, the prediction is compared to the class of the example and the error-rate is updated. Finally, the classifier is trained with the example. We report the error and SD in the tables. The best results are presented in bold.

The significance of the observed differences is tested with Friedman (1937, 1940) test to compare multiple classifiers on multiple datasets based on average ranks as suggested by Demšar (2006). When the null hypothesis is rejected, we use the post-hoc Nemenyi test (Nemenyi 1963). We also use two-tailed paired *t* test to demonstrate differences between classifiers with and without drift detection.

The size of the classifiers is presented in a following way. Each rule is closely related to a leaf in the decision tree, because they contain almost the same information and the same functionality. Thus the size of the classifiers can be compared by considering the number of rules in a set and number of leaves in a tree.

We compare the learning times of the classifiers in prequential evaluation in CPU time seconds running on Intel i5-3210M 3.1GHz DC, 8GB DDR3, Ubuntu 13.04.

4.2 Comparison of VFDR algorithms

4.2.1 Stationary data

First, we focus on comparing the different modifications of `VFDR` classifiers as well as their respective ordered and unordered versions. The goal is to select the most promising classifier so that it is used in further comparisons. Table 5 presents the prequential error rates of the classifiers tested on the artificial stationary datasets.

Table 5 The prequential error of the ordered and unordered VFDR classifiers on stationary data

Dataset	VFDR-Base-UN	VFDR-Base-OR	VFDR-MS-UN	VFDR-MS-OR	VFDR-OA-UN	VFDR-OA-OR
Hyperplane	14.72 (0.07)	14.95 (0.12)	15.65 (0.54)	14.62 (0.19)	**13.25** (0.59)	16.06 (0.63)
LED	41.67 (3.11)	26.46 (0.05)	26.60 (0.14)	26.61 (0.19)	**26.22** (0.09)	27.43 (0.09)
SEA	15.33 (0.04)	15.54 (0.09)	14.75 (0.33)	13.77 (0.19)	**12.10** (0.17)	13.48 (0.40)
RBF	22.23 (1.88)	24.42 (0.23)	19.56 (3.29)	22.82 (4.86)	**18.98** (3.52)	22.20 (4.28)
Waveform	19.09 (0.31)	19.84 (0.09)	16.49 (0.19)	17.35 (0.20)	**16.35** (0.25)	20.63 (0.79)
Average rank	4.4	4.6	3.2	3.4	1	4.4

The best results are presented in bold

Table 6 Number of rules of the ordered and unordered VFDR classifiers on stationary data

Dataset	VFDR-Base-UN	VFDR-Base-OR	VFDR-MS-UN	VFDR-MS-OR	VFDR-OA-UN	VFDR-OA-OR
Hyperplane	34	26	19	7	53	12
LED	18	12	90	31	4,910	32
SEA	29	22	19	7	44	13
RBF	51	27	17	4	71	13
Waveform	13	12	17	3	218	14
Average rank	4	3	3.8	1.4	6	2.8

Table 7 The learning times in seconds in prequential evaluation of the ordered and unordered VFDR classifiers on stationary data

Dataset	VFDR-Base-UN	VFDR-Base-OR	VFDR-MS-UN	VFDR-MS-OR	VFDR-OA-UN	VFDR-OA-OR
Hyperplane	8	6	15	11	12	7
LED	39	25	53	34	105	22
SEA	2	1	4	3	3	2
RBF	12	6	12	9	14	7
Waveform	30	20	34	20	86	15
Average rank	3.6	1.5	5.3	3.4	5.5	1.7

Comparing the respective variants of ordered and unordered classifiers using paired t test we obtain that the unordered is significantly better on three of the five datasets in the base version of the algorithm (VFDR-Base-UN) and significantly worse in one case. The VFDR-MS-UN version is also better in three cases, but worse in the remaining two. The unordered VFDR-OA-UN is significantly better in all five datasets and also achieves the best average results from all the variants on all the datasets. We can conclude that the unordered rule sets generally achieve better results and furthermore, they are considered more flexible and more interpretable. Since they are not dependent on other rules in the set, they are better suited for the adaptive extension of the rule learning algorithm. Comparing the average ranks with Friedman test, we obtain $\chi_F^2 = 13.11$ and $F_F = 4.41$ with critical value $F(5, 20) = 2.711$ and so we reject the null hypothesis. The result of post-hoc Nemenyi test with critical distance $CD = 3.372$ is that VFDR-OA-UN is significantly better than VFDR-OA-OR, VFDR-Base-UN, and VFDR-Base-OR.

The size of the ordered classifiers is smaller than the size of unordered sets in Table 6. The differences between VFDR-Base-UN and VFDR-Base-OR are smaller than in the case of VFDR-MS, VFDR-OA, which is expected since both create multiple rules in their respective unordered variants.

The learning times, Table 7, in the prequential evaluation of ordered rule set classifiers are smaller because when learning or predicting, the classifiers generally execute less tests whether a rule covers an example. Firstly they usually have smaller sets and secondly once a rule that covers the example is found, the search stops.

Table 8 The prequential error of the AVFDR classifiers on non-stationary data

Dataset	AVFDR-Base-UN	AVFDR-MS-UN	AVFDR-OA-UN	VFDR-Base-UN	VFDR-MS-UN	VFDR-OA-UN
Hyperplane	14.27 (0.38)	15.27 (1.39)	**13.75** (1.49)	14.51 (0.68)	16.05 (2.07)	15.05 (2.27)
LED	27.48 (0.02)	30.62 (0.70)	**27.43** (0.17)	28.41 (0.14)	30.60 (0.61)	28.00 (0.35)
SEA	15.96 (0.12)	16.46 (0.44)	**13.86** (0.27)	15.87 (0.27)	16.08 (0.43)	14.93 (0.44)
RBF	27.24 (0.92)	28.82 (1.72)	24.61 (1.75)	25.14 (0.67)	28.45 (2.15)	**22.29** (2.09)
Waveform	19.55 (0.14)	21.10 (1.22)	17.14 (0.28)	20.17 (0.23)	20.88 (1.10)	**17.04** (0.25)
Average rank	3	5.8	1.4	3.4	5.2	2.2

The best results are presented in bold

Table 9 Number of rules of the AVFDR classifiers on non-stationary data

Dataset	AVFDR-Base-UN	AVFDR-MS-UN	AVFDR-OA-UN	VFDR-Base-UN	VFDR-MS-UN	VFDR-OA-UN
Hyperplane	16	4	50	43	32	124
LED	9	41	5,435	15	85	15,378
SEA	16	8	26	29	21	54
RBF	9	2	13	49	22	113
Waveform	7	4	146	14	18	316
Average rank	1.8	1.4	4.4	3.8	3.6	6

4.2.2 Non-stationary data

The main interest when using streaming algorithm is in the situation when data is non-stationary. We focus on the unordered rule sets and their adaptive variants. Table 8 presents the prequential error of the classifiers. VFDR-OA-UN and AVFDR-OA-UN variants achieve the best average results. The average rank comparison demonstrates that for $\alpha = 0.05$, the $\chi_F^2 = 20.77$ and $F_F = 19.64$ with critical value $F(5, 20) = 2.711$, AVFDR-OA-UN is better than AVFDR-MS-UN and VFDR-MS-UN. With the critical distance $CD = 3.372$ for the post-hoc test VFDR-OA-UN is also better than AVFDR-MS-UN.

The adaptive versions remove rules thus it is expected that the sizes of the adaptive classifiers are smaller than the sizes of the classifiers without drift detection as can be observed in Table 9. Consequently the classifiers with drift detection are faster in the prequential evaluation (Table 10).

Based on the results from this and the previous section, we select AVFDR-OA-UN and VFDR-OA-UN classifiers as the most promising classifiers and compare them to other streaming classifiers.

4.3 Comparison between streaming classification algorithms

In this section, we compare the VFDR-OA-UN and AVFDR-OA-UN classifiers with standard streaming classification algorithms. In the first part we evaluate the per-

Table 10 The learning times in seconds in prequential evaluation of the AVFDR classifiers on non-stationary data

Dataset	AVFDR-Base-UN	AVFDR-MS-UN	AVFDR-OA-UN	VFDR-Base-UN	VFDR-MS-UN	VFDR-OA-UN
Hyperplane	8	10	16	8	18	21
LED	50	42	1,687	38	45	9,873
SEA	2	3	3	2	4	3
RBF	6	9	25	8	12	38
Waveform	25	23	90	23	31	192
Average rank	2.2	2.7	4.6	1.5	4.4	5.6

formance on stationary datasets. The main focus of the proposed classifiers is non-stationary data, which is evaluated in the second part of this section.

4.3.1 Stationary data

The Table 11 presents results from five runs on the artificial data sets generated without drift. Among the classifiers which do not employ any explicit drift detection method, VFDR-OA-UN ranks as the top classifier. Moreover, comparing the VFDTc and VFDR-OA-UN algorithms, which have similar learning approach, VFDR-OA-UN exhibits better performance. We notice that NB can learn the hyperplane concept much better than the other classifiers.

In the next experiment we compare the algorithms equipped with a drift detection method. Because the datasets do not incorporate drifts, the results of the classifiers with drift detection method are very similar to those without it. The small differences are caused mostly by incorrect drift detections in some cases. Note that *RBF* is devised so that it is not easy to capture the concept with a decision tree model. Therefore, neither VFDTc nor VFDR-OA-UN achieves the performance of IBLS. On average VFDR-OA-UN ranked the best followed by IBLS, however they did not rank significantly better than the other algorithms.

We anticipate that the real world data contain concept drift therefore the order of examples is important. For illustration, we can shuffle the order of examples using random seeds. It is expected that the possible concept drift is removed and the performance of classifiers with and without drift detection is not different. The tables with results can be found in Appendix 2. The p values of the respective paired t tests of classifiers with and without drift detection demonstrates that for $\alpha = 0.05$, the only significant difference is on *bank* data and *intrusion*. The performance of AVFDR-OA-UN compared to VFDR-OA-UN generally does not differ on stationary datasets and real world datasets. The main difference is in the number of rules the two algorithms produce. VFDR-OA-UN produces larger models than VFDR-OA-UN. While the number of rules in AVFDR-OA-UN still exceeds the number of leaves in VFDTc in many cases, the size of the model is reduced by removing the rules with decreasing performance.

Table 11 Prequential error of classifiers without drift detection on stationary data

Dataset	VFDR-OA-UN	VFDTc	NB	AVFDR-OA-UN	VFDT+SPC	NB+SPC	IBLS
Hyperplane	11.67 (0.18)	11.44 (0.16)	**7.22 (0.23)**	11.82 (0.2)	11.56 (0.32)	**7.22 (0.48)**	14.42 (0.46)
LED	26.23 (0.01)	26.23 (0.01)	**26.07 (0.01)**	26.28 (0.1)	26.47 (0.33)	**26.07 (0.01)**	31.38 (2.51)
SEA	11.95 (0.04)	12.29 (0.07)	11.99 (0.04)	11.94 (0.24)	13.07 (0.76)	12.01 (0.2)	**11.80** (0.1)
RBF	14.55 (9.11)	15.55 (7.35)	26.50 (47.47)	14.97 (2.94)	15.63 (2.81)	26.71 (7.27)	**5.55** (1.30)
Waveform	**18.48** (0.24)	19.24 (0.09)	19.54 (0.01)	18.16 (0.66)	19.31 (0.45)	19.54 (0.1)	**17.50** (0.3)
Average rank	3.3	4.1	3.9	3.6	5.4	4.3	3.4

The best results are presented in bold

4.3.2 Evaluation on non-stationary and real world data

Most of the current problems concerning mining data streams are treated as data with a non-stationary distribution, because if a phenomenon is observed for long enough time, it is highly probable that some change will appear. Therefore it is important to analyze the behavior under such conditions. Apart from the artificial data generated with drifts, this section includes large real world data, in which the presence of concept drift is not known but can be expected. We separately analyze the results obtained from the experiments on artificial datasets and on real world data as they represent more challenging problems.

Examining the results on the artificial data from the first part of Table 12 we can observe that `VFDR-OA-UN` performs very well since it adapts very fast even without the drift detection. Also in the real world datasets, the `VFDR-OA-UN` outperforms `NB`, and achieves slightly better results than `VFDTc`.

The right part of Table 12 presents results of classifiers with drift detection. The upper part shows that `VFDR-OA-UN` has the best average rank, but the `IBLS` algorithm also works well on the artificial datasets. Again the most noticeable difference is in *RBF* datasets. Comparing the average ranks of classifiers with drift detection on artificial datasets we obtain $\chi_F^2 = 1.32$ and $F_F = 1.32$, while the critical value for $\alpha = 0.05$ is 3.49. Therefore the null-hypothesis is not rejected.

The variability of real world datasets shows that each of the classifiers for time changing data has different strengths and weaknesses. Overall the `AVFDR-OA-UN` achieves the best average rank. The `IBLS` achieves best result in *forestCovtype*. The `NB+SPC` and `VDFTc+SPC` perform similarly since often the tree does not grow very much before the drift is detected thus the `NB` in the the root (functional leaf) is responsible for the prediction for many examples. The statistical tests for comparison of classifiers with drift detection on real world data achieve value of $\chi_F^2 = 10.05$ and $F_F = 5.04$ with critical value $F(3, 21) = 3.072$ and so we reject the null hypothesis. The post-hoc Nemenyi test with critical distance $CD = 1.658$ suggests that `AVFDR-OA-UN` is significantly better than `IBLS`.

We can analyse the differences between `AVFDR-OA-UN` and `VFDR-OA-UN`. The artificial datasets can be randomly generated to contain drift and therefore provide a suitable data for the analysis. Using t test we get the p value $= 0.048$ for *hyperplane* datasets, which signals that there is a statistical evidence for rejecting the null-hypothesis that they have the same accuracy. The p value for *SEA* datasets is 0.0002 and we can reject the null-hypothesis, and also in *LED*, where the p value $= 0.0099$ The null-hypothesis is not rejected in *waveform* datasets because the p value $= 0.2573$. Finally, the p value in *RBF* is 0.0002 and we can reject the null-hypothesis. The conclusion is that `AVFDR-OA-UN` mostly performs better or the same in the presence of concept drift with the benefit of producing less rules and learning faster.

Testing the differences between the two algorithms on the real datasets, we get mean difference of 0.9363 with SD of 4.2445. The p value for the t test is 0.5525, therefore we cannot reject that the two classifiers achieve the same error. The advantages of `AVFDR-OA-UN` are the smaller classifier models (in term s of number of rules) and faster learning times.

Table 12 Prequential error rates of classifiers on non-stationary and real-world datasets

Dataset	VFDR-OA-UN	VFDTc	NB	AVFDR-OA-UN	VFDTc+SPC	NB+SPC	IBLS
Artificial							
Hyperplane	**15.05 (2.27)**	17.22 (4.07)	22.79 (7.98)	13.75 (1.49)	14.19 (1.68)	**13.42 (0.96)**	16.19 (1.21)
LED	28.00 (0.35)	**27.83** (0.56)	47.51 (0.01)	27.43 (0.17)	26.91 (0.16)	**26.67** (0.11)	44.06 (1.24)
SEA	**14.93** (0.44)	15.67 (0.25)	17.56 (0.22)	13.86 (0.27)	14.39 (0.33)	14.78 (0.33)	**13.30** (0.28)
RBF	**22.29** (2.09)	26.60 (1.86)	40.43 (3.31)	24.61 (1.75)	27.08 (1.36)	37.15 (5.39)	**7.22** (1.37)
Waveform	**17.04** (0.25)	18.92 (0.63)	19.19 (0.57)	**17.14** (0.28)	18.48 (0.37)	19.06 (0.96)	17.78 (0.18)
Average rank	1.2	1.8	3	2	2.8	2.8	2.4
Real world							
Airlines	**32.78**	33.68	35.44	**33.20**	33.76	34.02	36.14
Bank	**10.31**	10.96	10.90	**10.20**	10.35	10.32	11.11
Connect-4	**24.68**	26.44	30.78	**24.91**	26.03	26.02	27.10
Elec	19.90	**19.13**	26.65	**15.13**	16.05	15.98	17.96
ForestCovtype	**11.39**	15.77	39.47	11.78	13.00	12.15	**7.80**
Intrusion	0.02	0.03	7.61	0.02	0.02	0.02	0.05
Pokerhand	**10.12**	24.26	40.44	**20.57**	27.78	25.00	24.72
Spam	8.67	**6.26**	8.85	9.55	**8.70**	9.58	20.86
Average rank	1.25	1.875	2.875	1.35	2.75	2.5	3.375

The best results are presented in bold

Table 13 Hyperplane varying the noise level

	Error rate		No. of rules	
	VFDR-OA-UN	AVFDR-OA-UN	VFDR-OA-UN	AVFDR-OA-UN
Noise 5 %	11.67 (0.18)	11.82 (0.04)	56	48
Noise 10 %	15.61 (0.03)	15.97 (0.13)	56	55
Noise 15 %	20.22 (0.11)	20.12 (0.11)	46	39
Noise 20 %	24.86 (0.09)	24.93 (0.08)	42	41

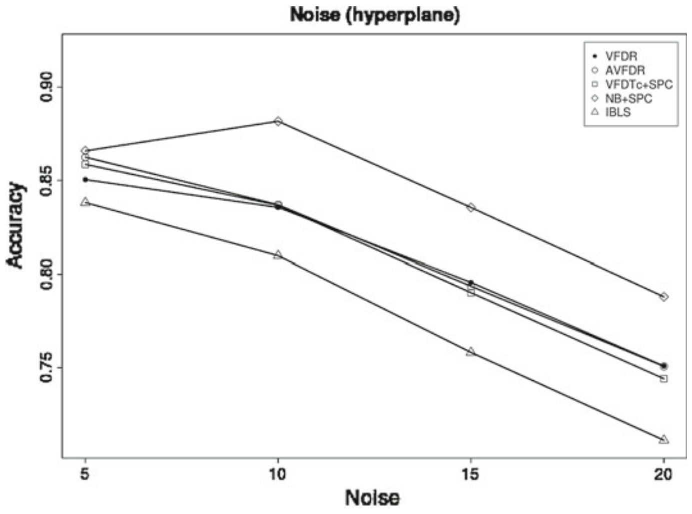

Fig. 2 Varying noise in hyperplane dataset

In order to examine the robustness to noise we introduced various levels of artificial noise in the *hyperplane* dataset. The *hyperplane* is chosen for being numerical and not being as 'simple' as *SEA* dataset, and the generator itself provides the option to choose arbitrary noise level. In the Table 13 we observe that increasing the noise level does not have a strong influence on the number of the rules neither in VFDR-OA-UN nor in AVFDR-OA-UN. The noise also did not deteriorate the performance of the AVFDR-OA-UN compared to VFDR-OA-UN and the number of removed rules does not increase with the increasing noise. Figure 2 shows the influence of varying noise in hyperplane dataset on the accuracy of the classifiers. The robustness to noise is very similar for VFDTc+SPC and AVFDR-OA-UN algorithms. It is confirmed by the comparison of the two algorithms on three artificial datasets with varying noise plotted in Fig. 3.

Since AVFDRs adapt to changes within their rules the detections are more frequent than in classifiers tracking the changes in the performance of the whole classifier. Thus the change in data can affect many rules. All those rules that model the feature space under change. In order to have the idea of the process, we plot bars for each drift detected by AVFDRs and standard stream classifiers with SPC.

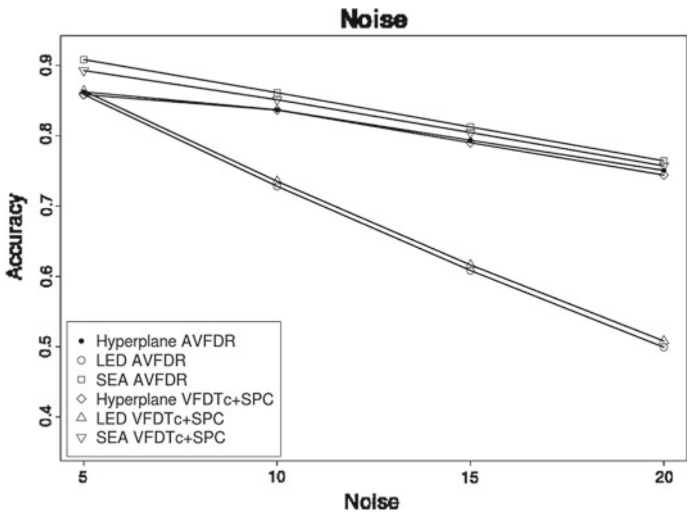

Fig. 3 Comparison of noise robustness of `AVFDT-OA-UN` and `VFDTc+SPC`

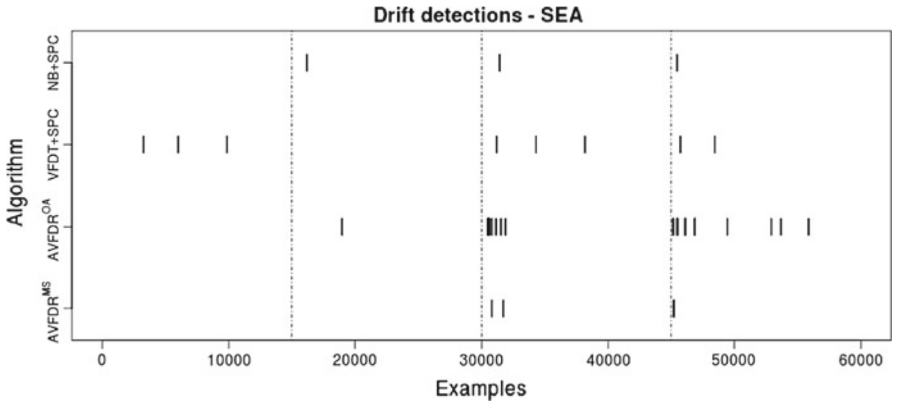

Fig. 4 Drift detections in *SEA* dataset

In the *SEA* dataset it is known that the change occurs exactly after every 15,000 examples and therefore we can examine the behavior of classifiers with drift detection mechanisms on Fig. 4. `NB+SPC` reacts exactly to each of the drifts with detection delay depending on the severity of the drift. `AVFDR-MS-UN` did not detect any change when first drift occurred, but it reacted even faster than `NB+SPC` for the next two changes. `AVFDR-OA-UN` induces more rules and thus it is expected there would be more changes detected (more rules influenced by the change). This is confirmed by the experiment. The closer to the change point the more frequently the drift detections in the rules occur. Some detections appear also later (some rules cover smaller space and are updated infrequently), nevertheless the occurrences are sparser.

Table 14 presents the number of rules of unordered sets and the number of leaves of the tree (the average number in case of the artificial datasets). As previously mentioned,

Table 14 Number of rules of rule classifiers and leaves of VFDTc on non-stationary data

Dataset	VFDR-OA-UN	AVFDR-OA-UN	VFDTc
Artificial			
Hyperplane	124	50	48
LED	15,378	5,435	7
SEA	54	26	31
RBF	113	13	55
Waveform	316	146	12
Average rank	3	1.6	1.4
Real world			
Airlines	543	52	1,055
Bank	55	12	49
Connect-4	72	13	31
Elec	55	7	30
ForestCovtype	1,834	49	402
Intrusion	249	47	624
Pokerhand	6,106	225	314
Spam	12	3	14
Average rank	2.6	1	2.4

VFDR-OA-UN generally produces large number of rules. In most of the cases the number is higher than number of leaves of VFDTc. We can observe that the highest number of rules appears in large sets with many classes since the algorithm allows the rules to expand for each class. The obvious benefit of using AVFDR-OA-UN is in the noticeable reduction of the size. With the exception of artificial *LED* dataset, the size of the AVFDR-OA-UN is not much different from the tree consequently the memory consumption is very similar. Moreover, it is possible to set the maximum number of rules in the set and choose an appropriate strategy for the rule replacement in order to limit the requirements for computational resources or limit the maximum number of classes for which one rule can expand.

Another dimension of analyses is the learning times. Table 15 presents the learning times (in CPU time seconds) of prequential evaluation of the classifiers. The IBLS algorithm has the highest learning times with prequential evaluation, but note that it is implemented in a different framework.

We can conclude that the times of VFDR-OA-UN are higher than the learning times of VFDTc and NB, but still sufficient for processing thousands of examples. The time complexity in comparison to trees increases with the number of rules which need to be checked if they cover the train/test example. The highest running times are on datasets with many classes, consequently the classifier generates many rules for each of the classes. It is desired to keep the rule set reasonably small so that the increase is negligible. In our case it is done by the use of the adaptive version. We can clearly observe that the advantage of AVFDR-OA-UN is that it removes potentially incorrect rules thus keeps the set smaller and consequently the learning and prediction times are reduced. Other possibility could be to have the limit the maximum number of rules in the set.

Table 15 Learning times of classifiers in seconds

Dataset	VFDR-OA-UN	VFDTc	NB	AVFDR-OA-UN	VFDTc+SPC	NB+SPC	IBLS
Artificial							
Hyperplane	21	8	4	16	9	4	870
LED	9,873	25	20	1,687	28	34	3,271
SEA	3	6	1	3	6	1	24
RBF	38	23	8	25	22	12	1,113
Waveform	192	36	7	90	34	11	1,210
Average rank	5.7	3.7	1.2	4.7	3.7	2.2	6.8
Real world							
Airlines	120	53	11	34	40	14	2,574
Bank	14	7	4	13	8	8	161
Connect-4	11	6	4	9	6	7	95
Elec	3	2	1	2	3	1	42
ForestCovtype	819	76	83	209	60	79	5,364
Intrusion	5,383	1,438	1,412	2,095	877	785	17,037
Pokerhand	6,548	108	100	337	71	90	1,978
Spam	14	12	18	17	13	18	59
Average rank	5.7	3	2.6	4.4	2.9	2.7	6.9

5 Conclusions

In this paper we present the `VFDR` system, an on-line, any-time and one-pass algorithms for learning decision rules in data streams. We present a base version, and two modifications to the basic algorithm, which focus on learning specialized rules for each class. The extended algorithms exhibit better results compared to baseline algorithm. Moreover, we describe an adaptive extension for the rule learning classifier `AVFDR`. In this extension each rule is equipped with explicit drift detection method, which is able to react to changes in data. The detection technique serves not only as a faster adaptation method, but also as a rule pruning mechanism. It allows keeping high performance when learning from evolving data and reduces the size by reducing the number of rules of the classifier. The detection method can provide relevant information about the structural changes in the process generating data. The rule learning algorithms we present generate highly flexible models with comparable performance against the standard stream mining techniques. The decision rules presented in this paper offer an interesting, interpretable and modular alternative to well-known and widely used techniques, like decision trees.

Acknowledgments The authors would like to express their gratitude to the reviewers of previous versions of the paper. This work is partially funded by FCT - Fundao para a Ciłncia e a Tecnologia/MEC - Ministrio da Educao e Ciłncia through National Funds (PIDDAC) and the ERDF - European Regional Development Fund through ON2 North Portugal Regional Operational Programme within the projects Knowledge Discovery from Ubiquitous Data Streams FCT-KDUS(PTDC/EIA/098355/2008), NORTE-07-0124-FEDER-000059. Authors also acknowledge the support of the European Commission through the project MAESTRA (Grant

Number ICT-2013-612944). Petr Kosina also acknowledges the support of Faculty of Informatics, MU, Brno.

Appendix 1: Datasets

In this section we describe the datasets that are used in the experiments. We have used large scale artificial and real world datasets. The real world datasets were previously used in other works when testing on-line learning algorithms as they represent large datasets and it is likely that they contain drifts, but their presence and nature is not known.

Artificial datasets

The artificial datasets are obtained using generators proposed by Bifet et al. (2010), each generator was used to produce five datasets with different random seeds. The *hyperplane* dataset is generated such that the class is given by rotating hyperplane (Hulten et al. 2001). A hyperplane in d-dimensional space is set of points $\mathbf{x}$ that satisfy $\sum_{i=1}^{d} w_i x_i = w_0$ where x_i is the ith coordinate of $\mathbf{x}$. $\sum_{i=1}^{d} w_i x_i \geq w_0$ then represents the positive and $\sum_{i=1}^{d} w_i x_i < w_0$ the negative concept. This set with 100,000 examples has two classes, ten attributes and five of them changing at speed 0.01 with 5 % noise (probability for each instance to have its class inverted).

Another artificial dataset is *SEA* concepts (Street and Kim 2001) 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). The domain of the attributes is: $x_i \in [0, 10]$, where $i = 1, 2, 3$. The target concept is $x_1 + x_2 \leq \beta$, where here $\beta \in \{7, 8, 9, 9.5\}$. There are four concepts; the size of each is 15,000 examples, with a total 60,000 for the whole dataset.

LED is formed by examples (Breiman et al. 1984) with $\{0, 1\}$ values of each attribute signaling whether given LED is off or on. Only seven out of 24 are relevant. Class label reflects the number (0–9) displayed by the diodes. There is 10 % of noise added to this dataset (probability for each attribute that it would have its value inverted). The generated set size is 200,000 instances. The drift in this dataset is caused by changing relevant attributes with irrelevant.

The goal in the *Waveform* dataset is to recognize three different classes of waveform. The waveforms are generated from a combination of two or three base waves. The optimal Bayes classification rate is known to be 86 %. The dataset has 21 numeric attributes, all of which include noise, and consists of 100,000 examples. The drift switches the positions (attributes) of the generated attribute-values.

The radial basis function (*RBF*) generates a fixed number of random centroids. Each center has a random position, a single SD, class label and weight. A new example is generated by a randomly selected center. The weights are considered and centers with higher weight are more likely to be chosen. Then a random direction is chosen to offset the attribute values from the central point. The displacement length is randomly drawn from a Gaussian distribution with SD determined by the chosen centroid. The chosen centroid also determines the class label of the example. The generated *RBF* datasets have ten numerical attributes and 50 centers with two classes. The number of examples is 100,000, the speed of change of centroids is 0.0001, and the number of centroids with drift is 50.

Real-world datasets

The real-world datasets are large datasets, which are used with the ordering of examples the way they were collected as it is likely that they contain drift. The *intrusion* detection from KDDCUP 99 obtained from the UCI repository (Frank and Asuncion 2010), is a data set describing connections which are labeled either as normal or one of four categories of attack. The dataset consists of 4,898,431 instances.

The next dataset is *forestCovtype* also from the UCI repository (Frank and Asuncion 2010), 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 (Harries 1999) contains data collected from electricity market of New South Wales, Australia. It has 45,312 instances.

The task of *Airlines* dataset based on data from Data Expo (2009) is to predict whether a flight will be delayed given the information of the scheduled departure in seven attributes. It consists of 539,383 instances.

The *connect-4* dataset from the UCI repository (Frank and Asuncion 2010) consists of 42 categorical attributes and contains 67,557 examples.

The *pokerhand* (Frank and Asuncion 2010) consists of 829,201 instances and ten predictive attributes. Each example represents a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank). The class describes the poker hand. This dataset was modified so that the cards are sorted by rank and suit and the duplicates were removed.

The *bank* dataset (Moro et al. 2011) is related with direct marketing campaigns of a Portuguese bank institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required in order to access if the product (bank term deposit) would be (or not) subscribed. The classification task is to predict if the client will subscribe a term deposit. The full dataset has 45,211 examples with 16 attributes.

Katakis et al. (2009) presented the *spam* dataset, a real world text data stream that is chronologically ordered to represent the evolution of Spam messages over time. There are two classes, legitimate and Spam messages, and 9,324 examples with 500 attributes.

Appendix 2: Results from tests on shuffled real world datasets

See Appendix Tables 16, 17 and 18.

Table 16 Prequential error rates of the classifiers on shuffled real world data

Dataset	VFDR-OA-UN	AVFDR-OA-UN	VFDTc	VFDTc+SPC	NB	NB+SPC
Airlines	37.53 (0.13)	37.68 (0.13)	35.35 (0.24)	35.25 (0.10)	36.68 (0.03)	36.69 (0.03)
Bank	13.06 (0.48)	13.49 (0.40)	12.01 (0.18)	12.70 (2.06)	15.24 (0.38)	15.13 (0.77)
Connect-4	27.25 (0.23)	27.43 (0.12)	28.71 (0.32)	31.30 (3.78)	27.98 (0.17)	29.24 (3.00)
Elec	23.83 (0.58)	24.03 (0.42)	23.81 (0.58)	30.38 (0.22)	30.37 (0.23)	23.58 (0.57)
ForestCovtype	23.38 (0.17)	23.23 (0.32)	29.77 (0.26)	29.92 (0.58)	33.01 (0.09)	33.03 (0.06)
Intrusion	0.06 (0.01)	0.08 (0.02)	0.11 (0.01)	0.11 (0.02)	7.24 (0.07)	6.01 (0.22)
Pokerhand	17.92 (0.73)	17.93 (0.73)	32.51 (0.85)	32.51 (0.85)	49.93 (0.91)	49.93 (0.91)
Spam	10.84 (0.24)	10.98 (0.50)	11.43 (1.41)	11.43 (1.41)	12.96 (0.11)	12.96 (0.11)
Average rank	2	2.7	2.7	3.7	5.1	4.8

Table 17 The *p* values of paired *t* tests on shuffled real world datasets

Dataset	VFDR-OA-UN vs. AVFDR-OA-UN	VFDTc vs. VFDTc+SPC	NB vs. NB +SPC
Airlines	0.229	0.412	0.183
Bank	0.024	0.811	0.507
Connect-4	0.235	0.385	0.176
Elec	0.551	0.388	0.477
ForestCovtype	0.595	0.613	0.388
Intrusion	0.182	0.783	0.000
Pokerhand	0.380	0.391	0.182
Spam	0.229	0.412	0.183

Table 18 Number of rules of rule classifiers and leaves of VFDTc on stationary data and shuffled real world datasets

Dataset	VFDR-OA-UN	AVFDR-OA-UN	VFDTc
Hyperplane	56	48	28
LED	938	510	3
SEA	36	34	27
RBF	60	56	34
Waveform	197	137	12
Average rank	3	2	1
Airlines	761	391	6,169
Bank	32	19	34
Connect-4	59	48	24
Elec	30	19	14
ForestCovtype	3,426	3,128	71
Intrusion	1,011	684	707
Pokerhand	11,832	11,329	64
Spam	4	3	3
Average rank	2.75	1.56	1.69

References

Baena-Garcia M, Campo-Avila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams. ECML-PKDD, Berlin, pp 77–86

Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Thiel K, Wiswedel B (2009) KNIME: the konstanz information miner: version 2.0 and beyond. SIGKDD Explor Newsl 11:26–31

Bifet A, Gavalda R (2009) Adaptive learning from evolving data streams. In: Advances in intelligent data analysis VIII. Lecture notes in computer science, vol 5772. Springer, Berlin/Heidelberg, pp 249–260

Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res (JMLR) 11:1601–1604

Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavaldà R (2009) New ensemble methods for evolving data streams. In Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '09. ACM Press, New York, pp 139–148

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees, 1st edn. Chapman and Hall/CRC, Boca Raton

Clark P, Boswell R (1991) Rule induction with CN2: some recent improvements. In: Proceedings of the European working session on machine learning, EWSL '91. Springer, London, pp 151–163

Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3:261–283

Cohen W (1995) Fast effective rule induction. In: Proceedings of the 12th international conference on machine learning, ICML'95. Morgan Kaufmann, San Francisco, pp 115–123

Data Expo (2009) ASA sections on statistical computing statistical graphics. http://stat-computing.org/dataexpo/2009/. Accessed 1 Feb 2013

Data Mining Group (2011) Predictive model markup language (pmml 4.1). http://www.dmg.org/v4-0-1/RuleSet.html. Accessed 1 Feb 2013

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Domingos P (1996) Unifying instance-based and rule-based induction. Mach Learn 24:141–168

Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '00. ACM Press, New York, pp 71–80

Ferrer F, Aguilar J, Riquelme J (2005) Incremental rule learning and border examples selection from numerical data streams. J Univ Comput Sci 11(8):1426–1439

Frank A, Asuncion A (2010) UCI machine learning repository. University of California, Irvine

Frank E, Witten IH (1998) Generating accurate rule sets without global optimization. In: Proceedings of the 15th international conference on machine learning, ICML'98. Morgan Kaufmann, San Mateo, pp 144–151

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701

Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11(1):86–92

Fürnkranz J (2001) Round robin rule learning. In: Proceedings of the 18th international conference on machine learning, ICML'01. Morgan Kaufmann, San Mateo, pp 146–153

Fürnkranz J, Gamberger D, Lavrač N (2012) Foundations of rule learning. Springer, New York

Gama J (2010) Knowledge discovery from data streams. Chapman and Hall/CRC, Baco Raton

Gama J, Kosina P (2011) Learning decision rules from data streams. In: Proceedings of the 22nd international joint conference on artificial intelligence. AAAI, Menlo Park, pp 1255–1260

Gama J, Rocha R, Medas P (2003) Accurate decision trees for mining high-speed data streams. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'03. ACM Press, New York, pp 523–528

Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: SBIA Brazilian symposium on artificial intelligence, LNCS 3171. Springer, Heidelberg, pp 286–295

Gama J, Fernandes R, Rocha R (2006) Decision trees for mining data streams. Intell Data Anal 10:23–45

Gama J, Sebastiao R, Rodrigues PP (2009) Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '09. ACM Press, New York, pp 329–338

Grant E, Leavenworth R (1996) Statistical quality control. McGraw-Hill, New York

Harries M (1999) Splice-2 comparative evaluation: electricity pricing. Technical report, The University of New South Wales, Sydney

Hinkley D (1970) Inference about the change point from cumulative sum-tests. Biometrika 58:509–523

Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 97–106

Katakis I, Tsoumakas G, Banos E, Bassiliades N, Vlahavas I (2009) An adaptive personalized news dissemination system. J Intell Inf Syst 32:191–212

Klinkenberg R (2004) Learning drifting concepts: example selection vs. example weighting. Intell Data Anal 8(3):281–300

Kolter JZ, Maloof MA (2003) Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Proceedings of the 3th international IEEE conference on data mining. IEEE Computer Society, New York, pp 123–130

Kosina P, Gama J (2012a) Handling time changing data with adaptive very fast decision rules. In: Proceedings of the 2012 European conference on machine learning and knowledge discovery in databases, ECML PKDD'12, vol I. Springer, Berlin, Heidelberg, pp 827–842

Kosina P, Gama J (2012b) Very fast decision rules for multi-class problems. In: Proceedings of the 2012 ACM symposium on applied computing. ACM Press, New York, pp 795–800

Lindgren T, Boström H (2004) Resolving rule conflicts with double induction. Intell Data Anal 8(5):457–468

Maloof M, Michalski R (2004) Incremental learning with partial instance memory. Artif Intell 154:95–126

Moro S, Laureano R, Cortez P (2011) Using data mining for bank direct marketing: an application of the crisp-dm methodology. In: Proceedings of the European simulation and modelling conference, ESM'2011. EUROSIS, Guimaraes, pp 117–121

Nemenyi P (1963) Distribution-free multiple comparisons. PhD thesis, Princeton University

Oza NC, Russell S (2001) Online bagging and boosting. In: Artificial intelligence and statistics 2001. Morgan Kaufmann, San Mateo, pp 105–112

Quinlan JR (1991) Determinate literals in inductive logic programming. In: Proceedings of the 12th international joint conference on artificial intelligence, IJCAI'91, vol 2. Morgan Kaufmann Publishers Inc, San Francisco, pp 746–750

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Mateo

Rivest R (1987) Learning decision lists. Mach Learn 2:229–246

Schlimmer JC, Granger RH (1986) Incremental learning from noisy data. Mach Learn 1:317–354

Shaker A, Hüllermeier E (2012) IBLStreams: a system for instance-based classification and regression on data streams. Evol Syst 3:235–249

Street WN, Kim Y (2001) A streaming ensemble algorithm SEA for large-scale classification. In: Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '01. ACM Press, New York, pp 377–382

Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '03. ACM Press, New York, pp 226–235

Weiss SM, Indurkhya N (1998) Predictive data mining: a practical guide. Morgan Kaufmann Publishers, San Francisco

Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23:69–101