Very Fast Decision Rules for Multi-class Problems

Petr Kosina
LIAAD - INESC Porto L.A., Portugal
and Fac. of Informatics, Masaryk University, Brno
petr.kosina@inescporto.pt

João Gama
LIAAD - INESC Porto L.A., Portugal
and Fac. of Economics, University of Porto, Portugal
jgama@fep.up.pt

ABSTRACT

Decision rules are one of the most interpretable and flexible models for data mining prediction tasks. Till now, few works presented online, any-time and one-pass algorithms for learning decision rules in the stream mining scenario. A quite recent algorithm, the Very Fast Decision Rules (VFDR), learns set of rules, where each rule discriminates one class from all the other. In this work we extend the VFDR algorithm by decomposing a multi-class problem into a set of two-class problems and inducing a set of discriminative rules for each binary problem. The proposed algorithm maintains all properties required when learning from stationary data streams: online and any-time classifiers, processing each example once. Moreover, it is able to learn ordered and unordered rule sets. The new approach is evaluated on various real and artificial datasets. The new algorithm improves the performance of the previous version and is competitive with the state-of-the-art decision tree learning method for data streams.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

Keywords

Data Streams, Classification, Rule Learning

1. MOTIVATION

In machine learning and particularly for classification tasks, decision trees are among the most popular tools: Decision trees exhibit high predictive capability, high degree of interpretability, and efficient algorithms are available. A decision tree can be converted to a set of IF-THEN decision rules. Each path from the root to a leaf generates a rule, and there are so many rules as leaves in the tree. The main advantage of decision rules is that each rule can be processed independently of the others. Although generating rules from decision trees is a common approach, several approaches learn such rule sets directly from data.

In stream mining scenarios, decision trees are also popular, but few works address the problem of rule learning. One of the pioneer works was presented in [13]. The authors introduced VFDR, an incremental classifier that is able to induce a set of rules (ordered or unordered) by scanning the data only once. In this work, we followed the concept and focused on the fact that many off-line learning approaches decompose a multi-class problem into two-class problems in order to achieve better results. We present Very Fast Decision Rules for Multi-Class problems (VFDR-MC) which is an extension to the rule set classifier for data streams VFDR. The contribution of this work is three-fold. First, VFDR-MC focus on the multi-class learning tasks and decomposes them into two-class problems. Second, it employs different criterion to find a new condition for rule expansion. Third, when expanding a rule it can create multiple rules from the original rule, each one for different class.

The paper is organized as follows. The next section discusses related work in rule learning and handling multi-class problems. The VFDR-MC system is presented in Section 3 and the experimental results are in Section 4. Finally, Section 5 concludes the paper and presents future work directions.

2. RELATED WORK

As it was mentioned before, a widely used strategy consists of building decision lists from decision trees, being C4.5rules [19] the most representative system in the literature. In any decision tree, the set of conditions in the path from the root to a leaf define the conditional part of a rule, whereas the class-label assigned to the leaf defines the conclusion of that rule. Each rule corresponds to the path from the root to a leaf, and there are as many rules as leaves. However, it has been shown that the antecedents of individual rules may contain irrelevant conditions. Empirical studies [19] have shown that the optimized set of rules is both simpler and more accurate than the initial tree. In [10] authors 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.

Several algorithms appear in the literature for building decision lists [20, 4, 5, 6, 23] with different capabilities and approaches to handle multi-class problems. The first proposed version of CN2 [4] evaluates every possible complex, i.e., conjunction of attribute tests (*if-condition*) of a rule, based on information-theoretic entropy measure. Namely, this classifier is able to learn multiple classes assigning the rule the most common class of examples covered by rule's complex. In its improved version [3], the entropy measure is replaced by Laplace accuracy estimate and the main loop of rule set algorithm searches the 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.

SAC'12 March 25-29, 2012, Riva del Garda, Italy.

Similarly, RIPPER [5] 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 minority class, covered examples are removed and the algorithm proceeds with the next class. Process stops when last single class remains and which is then assigned as default class. Another technique to reduce multi-class problem is pairwise class binarization [11]. The idea is to learn a classifier for each pair of classes thus transforming the c-class problem into $\frac{c(c-1)}{2}$ two-class problems. The prediction of 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, in which classifiers are learned for both problems obtaining in total $c(c-1)$ classifiers.

Fundamental incremental rule learners include STAGGER [21], the first system designed expressly for coping with concept drift, the FLORA family of algorithms [24] with FLORA3 being the first system able to deal with recurring contexts, and the AQ-PM family [17]. Since pure incremental rule learners take into account every training example, many of them have not still adapted to a data streams environment, especially those featuring numerical attributes. System Facil, presented in [8], is a classification system based on decision rules that may store up–to–date border examples to avoid unnecessary revisions when virtual drifts are present in data. Consistent rules classify new test examples by covering and inconsistent rules classify them by distance as the nearest neighbor algorithm. 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). A rule is said consistent when does not cover any negative (different label) example. The aim is to seize border examples up to a threshold, user defined ratio between the number of positive examples the rule covers and total number it covers, is reached. When the threshold is reached, the examples associated with the rule are used to generate new positive and negative consistent rules. This approach is similar to the AQ11-PM system [16, 17], 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).

3. VERY FAST DECISION RULES

In this section we present VFDR-MC algorithm. The following sections describe in detail the algorithms to learn ordered and unordered rule sets, and the algorithms to expand a rule.

3.1 Growing a Set of Rules

The VFDR-MC algorithm is designed for high-speed data streams, which needs only one scan of data and learns ordered and/or unordered rules.

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 $\mathcal{L}_r$. 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-MC 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 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 $p(c_k)$, 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 information maintained in $\mathcal{L}_r$ is similar to the sufficient statistics in VFDT [7] like algorithms. In [12] the authors present efficient algorithms to maintain $\mathcal{L}_r$.

The sample size to decide when to expand a rule is given 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 estimated mean after n independent observations by more than: $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$. In order to learn decision rule lists, Hoeffding bound determines the number of observations after which a rule can be expanded or new rule can be induced. 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 (denoted as VFDR-MC^o) or unordered (denoted as VFDR-MC^u) set of rules. In the former, every labeled example updates statistics of the first rule that covers it. This case learns decision lists. In the latter, every labeled example updates the statistics of all the rules that cover the example. If a labeled example is not covered by any rule, the default rule is updated.

Algorithm 1: VFDR: 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
                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)
```

We can further apply different strategies to functions that are responsible for creating new rules from *default rule* and expansion of other rules. There are as well differences between ordered and unordered strategies. They are presented in the following sections.

3.2 Expansion of a rule

Typically, rule learning systems for predictive problems with multiple classes divide it into binary class problems thus simplifying the learning task [11]. This is one of the major differences between VFDR-MC and VFDR, because VFDR-MC 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 of the classes $c \in C_r$, where C_r is the set of classes observed at rule r. The process to expand a rule by adding a new condition works as follows.

For each attribute X_i, the value of the gain function G adopted from FOIL [18] (described later) is computed for each attribute value v_j and class c_k. 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 the new condition $X_a = v_j \Rightarrow c_k$.

Algorithm 2 describes 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 minimum frequency among those that satisfy the Hoeffding bound condition. There is a difference between inducing new rule from *default rule* using ordered and unordered strategies. The ordered strategy stops after finding the first r'. In other words, ordered rules are dependent on each other, therefore we preferably induce new rule for only one (possibly more interesting) minority class. The unordered strategy continues with the search for new rules until the expansions for all possible classes are checked.

In case of expanding a rule that already contains some 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 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.

3.2.1 Gain measure

The gain measure used in the VFDR-MC is adopted from FOIL, where 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 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:

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

Statistics for class for which we compute the gain are considered as positives while all statistics for all other classes are considered as negatives.

Algorithm 2: ExpandEmptyRule: Expanding Empty Rule.

input : r: One Rule;
G: Gain evaluation function;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  $NR \leftarrow \{r\}$
  Compute $\epsilon = \sqrt{\frac{R^2 ln(2/\delta)}{2n}}$ (Hoeffding bound)
  foreach *class c_k in ascending order of class frequency* **do**
    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 $g^k_{2best} \leftarrow g_{ijk}$
    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

3.3 Classification strategies

The set of rules learned by VFDR-MC can be ordered or unordered. As they use different learning strategies, they must employ different classification strategies to achieve optimal prediction. In the former, only the first rule that covers an unlabeled example is used to classify the example. In the latter, 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 in each rule. Given the example $\vec{x} = (x_1, \ldots, x_j)$ and applying Bayes theorem, we obtain: $P(c_k|\vec{x}) \propto P(c_k) \prod P(x_j|c_k)$.

Using *NB* in VFDT like algorithms, is a well-known technique since it was introduced in [14]. 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

Algorithm 3: `ExpandRule`: Expanding Unordered Rule.

input : r: One Rule;
G: Gain evaluation function;
δ: Confidence
τ: Constant to solve ties
output: NR: New Rules Set;
begin
  Let $NR \leftarrow \{r\}$
  Compute $\epsilon = \sqrt{\frac{R^2 ln(2/\delta)}{2n}}$ (Hoeffding bound)
  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 of r
    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 $g^k_{2best} \leftarrow g_{ijk}$
  if $g^k_{best} - g^k_{2best} > \epsilon$ *or* $\epsilon < \tau$ **then**
    Extend r with a new condition based on the best attribute $X_a = v_j$
    foreach *class* $c_k \neq c$ *of rule* r **do**
      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 $g^k_{2best} \leftarrow g_{ijk}$
      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

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 `VFDR-MC` on datasets with different properties such that we can evaluate the behavior in various situations. We also provide comparison against `VFDR`, which considers all classes and expands only on one attribute, and against the state-of-the-art stream decision tree classifier. The evaluation method we employed uses the traditional train and test scenario. All algorithms and variants learn from the some training set and the error-rate is estimated from the some test sets. All algorithms were implemented in Java as an extension for KNIME [1]. Table 1 summarizes the parameters of the classifiers and the default values used in the experiments.

Algorithm 4: `ExpandRule`: Expanding Ordered Rule.

input : r: One Rule;
G: Gain evaluation function;
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 $\epsilon = \sqrt{\frac{R^2 ln(2/\delta)}{2n}}$ (Hoeffding bound)
  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
    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 $g^k_{2best} \leftarrow g_{ijk}$
  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

Table 1: Parameters of decision rules

Symbol	Short description	Default
N_{nb}	threshold to use *NB* in rule	50
w_s	minimal weight of new condition partition	0.1
δ	confidence level used in Hoeffding Bound	10^{-6}
N_{min}	# examples after which rule specialization is considered	400
τ	tie breaking constant	0.05

4.1 Datasets

The evaluation datasets include both artificial and real data, as well as sets with categorical and continuous attributes or their combination. In current settings we do not examine the reaction to a concept drift therefore in the synthetic datasets there is no specific concept drift added.

KDDCUP 99 is data set [9] of connections which are labeled either as normal or one of four categories of attack: DOS (denial-of-service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to local superuser privileges), and probing (surveillance and other probing). The test data is not from the same probability distribution as the training data and also there are new attack types that are not in the training data, yet in this case they have one of the four labels above. The set consist of 4,898,431 and 311,029 instances for training and test respectively.

Covtype is based on 54 cartographic attributes, continuous and categorical, the goal is to predict the forest cover type for given area [9]. The 581012 instances are divided into training and test set by stratified sampling with ration 80:20.

Hyperplane generated such that the class is given by rotating hyperplane [15]. 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.

SEA is an artificial dataset [22] commonly used in stream mining tasks that require time changing qualities of data. It is a two-class problem, defined by three attributes, where only two are relevant and 10% of noise. 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 $\beta = 7$ in this case since we were not interested in abilities to handle time changing data. Size of this dataset is 100,000 examples.

LED is formed by examples in this dataset [2] represent a digit on a seven-segment display. The $\{0, 1\}$ values of each attribute signals whether the LED is off or on respectively, and 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. The generated training set size is 1,000,000 instances and 100,000 as test set.

Random Tree is generated [2] such that it constructs a decision tree by choosing, at random, the splitting attributes, and assigning a random class label to each leaf. Once the tree is built, new examples are generated by assigning uniformly distributed random values to attributes which then determine the class label via the tree. The generated sets are specified by RT(#classes, #attributes, #values each, and max depth of the tree) with 10,000,000 training and 200,000 test instances.

4.2 Experimental Results

Tables 2 and 3 summarizes the error-rates of classifiers without and with using *NB* respectively. First, we focus on the performance on artificial datasets.

At the beginning we performed tests on a simplified version of the LED dataset, removing noise and irrelevant attributes. This is a 10 classes problem, and represents quite simple rule learning task. The aim was to confirm that the classifiers are able to learn basic problems. All variants of VFDR learned the target concepts, although using slightly different rule sets.

In the next experience, the data set was generated with additional 14 irrelevant attributes and 10% noise as we stated above. The results from these tests are in the Table 2. It shows that smaller rule sets (both VFDR's, VFDR−MC^o) do not achieve the accuracy of the larger rule set, using the majority class prediction within the rule. Using Bayesian predictions in the leaves (Table 3) all methods significantly improved their performance. The differences between error-rates are not significant.

Another experiment involves using random tree sets generated with different parameters and nominal attributes only. This is a simple task for a decision tree classifier which is able to perfectly learn the concept. Therefore, rules should evince same behavior, although it is expected that rule classifiers need more learning instances. In the first setting, (2,4,2,4), there are 2 classes, 4 attributes with 2 values each and max depth of the tree is 4. All classifiers are capable of learning such concept. If we add 2 more classes, to obtain *RT(4,5,2,5)*, VFDR−MC is still capable of learning the correct theory with a sufficient number of training examples. Whereas VFDR learns only partially the underlying concept, since there was no rule that would predict *class2*. This problem was easier for the ordered approach, which is built more in tree-like way. More complicated concept, for example *RT (4,15,5,4)*, requires much more instances to learn the concept, because there are many paths in such random tree that represent the concept and therefore even VFDR−MC was not able to induce a large enough rule set.

This set of experiments, using nominal attributes, clearly illustrates the behavior of VFDR like algorithms. Now we study the performance of VFDR in data sets with numerical attributes. This is a more challenging task but very important and relevant because many real problems are represented by continuous values. In the SEA dataset, without concept drift, where the two classes are linearly separable it is noticeable that using *NB* plays significant role in the rule set performance. For example, VFDR−MC^u without *NB* is the worst classifier, and it is the best when using *NB*.

For *Hyperplane*, the second numerical dataset though also only two-class problem, unordered VFDR−MC produces large set of rules that is comparable with decision trees in both situations, with and without *NB*, and outperforms the simple version of VFDR.

In both real dataset with nominal and continuous attributes, *KDDCup* intrusion detection task and *Covtype* the results similarly shows that unordered versions are generally better than the ordered ones with the exception of VFDR in *Covtype*. There is significant improvement using *NB*, which is mostly because these sets include rare classes and although VFDR−MC benefits minority classes, their frequency in the datasets is not always sufficient to produce a quality rule for such class.

The conclusion from these tests is that the unordered version are generally better than ordered ones and also the unordered VFDR−MC mostly outperforms VFDR in the multi-class sets, which supports our hypothesis that the rule learning system can benefit from decomposing it into two-class problems.

To analyze learning times, we report the relative learning times in comparison to VFDTc. The entries in Table 5 represent the ratio between the learning time of the algorithm referred in the first row

Table 2: Error rates of classifiers without using *NB*.

	Error rate % (variance)				
	VFDR^o	VFDR-MC^o	VFDR^u	VFDR-MC^u	VFDTc
LED	30.84 (6.62)	35.6 (28.53)	37.56 (8.43)	**26.9 (0.65)**	27.18(0.61)
RT(2,4,2,4)	0	0	0	0	0
RT(4,5,2,5)	0	0	28.13	0	0
RT(4,15,5,4)	32.97	38.42	59.95	19.64	0
SEA	15.61 (2.19)	17.83 (7.05)	17.81 (2.18)	18.52 (7.69)	**11.91 (0.06)**
Hyperplane	31.31 (28.42)	32.79 (36.09)	34.98 (36.43)	30.73 (19.56)	**30.24 (14.05)**
KDDCup	10.45	11.6	9.97	11.5	**8.55**
Covtype	64.72 (34.51)	56.75 (23.72)	66.16 (21.5)	43.43 (2.94)	**39.81 (6.39)**

Table 3: Error rates of classifiers using *NB*.

	Error rate % (variance)				
	VFDR$^o_{NB}$	VFDR-MC$^o_{NB}$	VFDR$^u_{NB}$	VFDR-MC$^u_{NB}$	VFDTc
LED	26.16 (0.04)	26.1 (0.05)	26.49 (0.59)	26.0(0)	26.0 (0.01)
RT(2,4,2,4)	0	0	0	0	0
RT(4,5,2,5)	0	0	15.59	0	0
RT(4,15,5,4)	26.14	20.69	42.26	11.1	0
SEA	13.24 (0.16)	12.86 (0.77)	14.71 (1.45)	**10.56 (0.11)**	11.12 (0.16)
Hyperplane	24.99 (11.53)	25.44 (12.14)	23.66 (9.82)	23.51 (15.09)	**23.12 (15.42)**
KDDCup	10.09	9.4	9.91	8.87	**8.3**
Covtype	58.71 (13.28)	49.49 (42.97)	60.65 (17.88)	**36.46 (7.38)**	38.15 (0.61)

Table 4: Size of classifiers counting number of rules or leaves in case of decision tree.

	Size				
	VFDR^o	VFDR-MC^o	VFDR^u	VFDR-MC^u	VFDTc
LED	22	21	47	1052	47
RT(2,4,2,4)	7	3	10	9	9
RT(4,5,2,5	21	18	33	128	23
RT(4,15,5,4)	259	263	85	5790	557
SEA	18	12	26	58	30
Hyperplane	136	55	186	700	208
KDDCup	23	24	33	212	616
Covtype	92	44	108	415	217

Table 5: Learning time compared to Hoeffding tree.

	Relative times			
	VFDR^o	VFDR-MC^o	VFDR^u	VFDR-MC^u
LED	0.8	0.9	2.3	18.8
RT(2,4,2,4)	0.8	0.9	1.4	1.3
RT(4,5,2,5)	1.1	1.2	3.2	10.7
RT(4,15,5,4)	2.6	2.6	2.5	126.9
SEA	0.8	0.9	1.3	2
Hyperplane	1.5	1.1	2.3	9.2
KDDCup	0.8	0.8	1	2.4
Covtype	1.1	0.9	1.4	3.8

and the learning time of VFDTc. We can conclude that the learning times of ordered rule sets are practically the same as in case of Hoeffding Tree. In case of the more interesting unordered rules the learning 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. In this setting the number of rules grows with more incoming examples.

5. CONCLUSIONS

Learning decision rules from data streams did not receive so much attention from stream mining community. In this work we present VFDR−MC, an on-line and any-time rule learning algorithm designed to process multi-class problems. VFDR−MC decomposes a n-class problem into n binary problems, and induces a rule set for each binary problem. Rules start from the most general case, and are specialized by adding new conditions. Depending on the strategy used to expand rules, we can obtain both ordered and unordered rule sets. The criterion used to select new specialization conditions is oriented towards two-class problems. The classifier considers in turn each class as positive against all the others as negatives, and allows a rule to be specialized with up to one condition for each class. The experimental results in various artificial and real datasets show the advantages of this approach, especially for the multi-class problems. The performance of decision rule learning classifiers for data streams are close to the predictive performance of state-of-the-art stream decision tree technique.

The gradual specialization of rules and induction of new rules implicitly adapt the learner to changes in data. The investigation of this property is out of the scope of this paper. The future work is intended to focus the rule learner in time-changing data with explicit drift detection.

6. 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.

7. REFERENCES

[1] M. R. Berthold, C. Borgelt, F. Hoeppner, and F. Klawonn. *Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data*, volume 42 of *Texts in Computer Science*. Springer-Verlag, 2010.

[2] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research (JMLR)*, 2010.

[3] P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. pages 151–163. Springer-Verlag, 1991.

[4] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[5] W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russel, editors, *Machine Learning, Proceedings of the 12th International Conference*. Morgan Kaufmann, 1995.

[6] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.

[7] P. Domingos and G. Hulten. Mining high-speed data streams. In *KDD*, pages 71–80, 2000.

[8] F. Ferrer, J. Aguilar, and J. Riquelme. Incremental rule learning and border examples selection from numerical data streams. *Journal of Universal Computer Science*, 11(8):1426–1439, 2005.

[9] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[10] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In J. Shavlik, editor, *Proceedings of the 15th International Conference - ICML'98*, pages 144–151. Morgan Kaufmann, 1998.

[11] J. Fürnkranz. Round robin rule learning. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01):146– 153*, pages 146–153. Morgan Kaufmann, 2001.

[12] J. Gama, R. Fernandes, and R. Rocha. Decision trees for mining data streams. *Intelligent Data Analysis*, 10:23–45, 2006.

[13] J. Gama and P. Kosina. Learning decision rules from data streams. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1255–1260. AAAI, Menlo Park, USA, 2011.

[14] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining*. ACM Press, New York, NY, 2003.

[15] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM, New York, NY, USA, 2001.

[16] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proceedings of the 3th International IEEE Conference on Data Mining*, pages 123–130. IEEE Computer Society, 2003.

[17] M. Maloof and R. Michalski. Incremental learning with partial instance memory. *Artificial Intelligence*, 154:95–126, 2004.

[18] J. R. Quinlan and R. M. Cameron-Jones. Induction of logic programs: Foil and related systems. *New Generation Comput.*, 13(3&4):287–312, 1995.

[19] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1993.

[20] R. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[21] J. C. Schlimmer and R. H. Granger. Incremental learning from noisy data. *Machine Learning*, 1:317–354, 1986.

[22] W. N. Street and Y. Kim. A streaming ensemble algorithm SEA for large-scale classification. pages 377–382. ACM Press, 2001.

[23] S. Weiss and N. Indurkhya. *Predictive Data Mining, a practical Guide*. Morgan Kaufmann Publishers, 1998.

[24] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996.