Feature Ranking in Hoeffding Algorithms for Regression

João Duarte
LIAAD, Inesc Tec, University of Porto
Porto, Portugal
jduarte@inesctec.pt

João Gama
LIAAD, Inesc Tec, University of Porto
Porto, Portugal
jgama@fep.up.pt

ABSTRACT

Feature selection and feature ranking are two aspects of the same learning task. They are well studied in batch scenarios, but not in the streaming setting. This paper presents a study on feature ranking from data streams in online learning regression models. The main challenge here is the relevance of features might change over time: features relevant in the past might be irrelevant now and vice-versa. We propose three new online feature ranking algorithms designed for Hoeffding algorithms. We have implemented the three methods in AMRules, a streaming regression algorithm to learn model rules. We compare their behaviour experimentally and present the pros and cons of each method.

Categories and Subject Descriptors

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

Keywords

Data Streams, Howffding Algorithms, Feature ranking

1. INTRODUCTION

Currently, data dimensionality in learning tasks is increasing exponentially. High-dimensional data present serious problems to existing learning methods. One of them is the curse of dimensionality, a well-known problem for distance based methods. Due to the presence of a large number of features, the instance space becomes sparse and a learning model tends to underfit, resulting in performance degradation. To address the problem of high dimensionality several feature selection techniques have been studied. All methods aim to select a small subset of features from the original ones according to a certain relevance evaluation criterion.

SAC'17, April 3-7, 2017, Marrakesh, Morocco

http://dx.doi.org/10.1145/0000000.0000000

These methods usually lead to an improved learning performance (e.g., higher learning accuracy for classification), a lower computational cost, and a better model interpretability [15]. Most of the methods presented so far, assume that the relevance of a feature does not change over time. Nevertheless, in many real-world applications, where data is collected over time, the relevance of a feature might also evolve. A feature that is relevant during a certain period might become less important. Also, an irrelevant feature might become relevant, reducing the predictiveness of the decision model. These variations can occur due to concept drift, changes in hidden variables or in key properties of the observed variables [8]. Detecting irrelevant features can be used to simplify the decision model, and to save space when storing the statistics about that feature.

In this paper we focus on online regression learning from data streams when the relevance of features evolve over time. The online learning community has studied this problem for a long time. A major paper in this area is [13], that presents one of the first incremental linear models that identify irrelevant attributes using a threshold over the weights of the linear model. It is one of the methods we use, and is described later in Section 2. In [12], the authors reviewed filter and wrapper approaches for feature selection and concluded the filter approach to be more adequate for on-line processing due to its lower computational costs. Filters are based on cumulative statistics (i.e., contingency tables) of the number of times a feature appears in each distinct class. Updating such statistics is incremental by nature, which makes the method suited for data stream processing. The predictive score of each feature can be computed using popular methods such as, the information gain, χ^2 or mutual information [12].

Feature selection and feature weighting are the basic strategies for quantifying feature relevance. While feature selection is used to reduce the size of the kept models, feature weighting is used to sort features by relevance and often used for providing useful information about the process generating the data. In this paper, we present an experimental study comparing three feature-ranking methods from data streams. The ranking dynamically evolves over time showing which features are relevant at each time instant. The paper is organized as follows. The next section presents the related work on feature ranking and describes AMRules, the Hoeffding algorithm used to implement the proposed ranking methods. Section 3 describes the proposed streaming feature ranking methods and how they are embedded in AM-

Rules. Section 4, describes the experimental results using benchmark, both artificial and real world, datasets. Section 5 presents the main lessons learned.

2. RELATED WORK

Depending on how class information is used, feature ranking algorithms can be categorized into supervised, unsupervised and semi-supervised. Furthermore, supervised feature ranking methods can be broadly categorized into filter models, wrapper models and embedded models [15]. The filter model separates the task of feature selection from the task of learning a classifier so that the bias of the learning algorithm does not interact with the bias of a feature selection algorithm. It relies on measuring generic characteristics of the training data such as distance, consistency, dependency, information, and correlation. Relief [17], Fisher score [15] and Information Gain based methods [15] are among the most representative algorithms of the filter model. The wrapper model uses the predictive accuracy of a predetermined learning algorithm to determine the quality of the selected features. These methods are prohibitively expensive to run for data with a large number of features.

The roots of online learning ensembles can be found in the `WinNow` algorithm [13]. `WinNow` is an online algorithm that combines the predictions of several *experts* by majority weighted voting. Each expert, which can be an attribute, is associated with a weight. When the weighted vote misclassifies an example, the weight of the experts in error is updated. `WinNow` uses a multiplicative weight-update scheme, that is, the weight is multiplied by a constant $\beta < 1$. It is restricted to binary decision problems, and exhibits very good performance when many dimensions are irrelevant. Later on, the same author presented the Weighted-Majority Algorithm [14] to combine predictions from a set of base classifiers.

Random forests are among the most popular machine learning methods thanks to their relatively good accuracy, robustness and ease of use. A *random forest* consists of a number of decision trees. Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set. The base measure on which the (locally) optimal condition is chosen is called impurity. For classification, it is typically either Gini impurity or information gain/entropy and for regression trees, it is variance. Thus when training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features ranked according to this measure. They also provide two straightforward methods for feature selection: mean decrease impurity and mean decrease accuracy.

Few algorithms address the problem of feature selection or feature ranking from data streams. An online algorithm was proposed in [19], but it requires multiple passes over the data. In [10], the authors present an unsupervised feature selection approach on data streams that selects important features by making only one pass over the data using limited storage. The proposed algorithm uses ideas from matrix sketching to efficiently maintain a low-rank approximation of the observed data and applies regularized regression on this approximation to identify the important features. A recent study, for classification, appears in [1].

Since the seminal work on Very Fast Decision Trees [4], the idea of using the Hoeffding bound as a heuristic to define the sample size used to select the splitting-test were used elsewhere. For example, learning regression and model trees appears in [11], learning decision and regression rules in [6], multiple models in [2, 5], etc. In this Section, and without loss of generality, we focus on regression rules, because the Hoeffding method is used to implement the feature ranking algorithms described here.

AMRules is a rule-based algorithm that learns from time evolving data streams. It incrementally grows a set of rules which are used to make classification [7] and regression [6] predictions. In this work, we will focus on AMRules for regression. However, the same concepts could be directly applied to classification rules and Hoeffding trees. A rule R is an implication in the form $\mathcal{A} \Rightarrow \mathcal{C}$. The antecedent $\mathcal{A}$ is a conjunction of literals L based on the values of the input attributes, and the consequent $\mathcal{C}$ consists of a predicting function.

A rule collects summaries of past data and, from time to time, evaluates a merit-function responsible for measuring the merit of splitting the data according to a given value v for each attribute X_j. For regression, AMRules uses the Variance Ratio (VR) as the merit-function. VR is defined as

$$VR(X_j, v) = 1 - \frac{|E_L|}{|E|}\frac{\text{var}(E_L)}{\text{var}(E)} - \frac{|E_R|}{|E|}\frac{\text{var}(E_R)}{\text{var}(E)}, \qquad (1)$$

where E is the set of examples seen by the rule since its last expansion and $\overline{y} = \frac{1}{|E|}\sum_{i=1}^{|E|} y_i$. If X_j is a numeric attribute, E_L is the set of examples $\{\mathbf{x}_i \in E : x_{i,j} \leq v\}$ and E_R is the set of examples $\{\mathbf{x}_i \in E : x_{i,j} > v\}$. If X_j is a nominal attribute, E_L is the set of examples $\{\mathbf{x}_i \in E : x_{i,j} = v\}$ and E_R the set of the remaining examples. A rule is expanded if there is sufficient confidence in choosing the best split, i.e. enough examples have been seen by the rule to make the splitting decision. For this purpose the Hoeffding bound [9] is used. It states that the true mean of a random variable r, with range P, will not differ from the sample mean more than ϵ with probability $1 - \delta$. It is defined as $\epsilon = \sqrt{\frac{P^2 \ln(1/\delta)}{2n}}$ where n is the number of observations. Let VR_{Best} and $VR_{2ndBest}$ correspond to the best and second best split options according to the Variance Ratio criterion. If $VR_{Best} - VR_{2ndBest} > \epsilon$, we can state that VR_{Best} correspond to the best split with probability $1 - \delta$ and the rule is expanded.

AMRules starts with an empty rule set $\mathcal{R}$ plus a default rule D. When a new training example $(\mathbf{x}, y)$ is available, the rules covering $\mathbf{x}$ are updated: for each rule $R_l \in S(\mathbf{x})$, if $\mathbf{x}$ is not an anomaly a change detection test is performed; if change is detected R_l is removed from the rule set since it is no longer valid; otherwise, the sufficient statistics of the rule $\mathcal{L}_l$ are updated; the rule is expanded if it satisfies the Hoeffding bound test. To detect changes, the evolution of the error of each individual rule is monitored using the Page-Hinkley (PH) test [16]. If no rule covers $\mathbf{x}$, the sufficient statistics of

the default rule $\mathcal{L}_D$ are updated and the expansion of the default rule is attempted. If the default rule expanded successfully it is added to the rule set $\mathcal{R}$ and a new default rule is created. Note that the split evaluation and corresponding rule expansion is only attempted from time to time to save computation time. To classify a test example, all the rules that cover the example make a prediction. These predictions are aggregated by weighted vote. The weights are inversely proportional to the prequential error of the corresponding rules.

The Random AMRules algorithm [5] consists of learning an ensemble $\mathcal{F} = \{\hat{f}_1 \cdots, \hat{f}_k\}$ of k models $\hat{f}_m$ using the AMRules algorithm as the base learner. Similarly to the Random Forests [3] algorithm, a different version of the data set is used to learn each individual model $\hat{f}_m$ following an online Bagging approach. For each example $(\mathbf{x}, y)$ and learning model $\hat{f}_m$ a different weight p_m is sampled from the Poisson distribution $p_m \sim Poisson(1)$ which is used by AMRules to update $\hat{f}_m$. If $p_m = 0$ the model $\hat{f}_m$ stays unchanged, otherwise AMRules learns from $(\mathbf{x}, y)$ using weight p_m. Also, when a rule R_l is created or expanded, a subset of the data attributes with size d' $(1 \leq d' \leq d)$ is randomly chosen. In order to prevent the models from being correlated, only this subset of attributes are considered for the next splitting decision of R_l. The prediction $\hat{y}_*$ of Random AMRules is computed as a linear combination of the estimations produced by the models $\hat{f}_m \in \mathcal{F}$: $\hat{y}_* = \hat{f}_*(\mathbf{x}) = \sum_{m=1}^{k} \theta_m \hat{f}_m(\mathbf{x})$. In this paper, the weights θ_m are computed using a uniform weighting function, such that all the predictors have the same importance, $\theta_m = \frac{1}{k}$.

3. ONLINE FEATURE RANKING

In this section, we propose three incremental and adaptive feature-ranking methods. All methods maintain a vector of weights of the same size as the number of features. The weights are updated during the process of learning rule sets: *1)* When evaluating the merit of the splits; *2)* When expanding a rule; *3)* When removing a rule from the rule set.

The features used in the the antecedents of the rules are clear indicators of the importance of the input attributes. Another indicator is the impact in variance reduction of a split. The methods we propose explore these indicators. It is intended to keep the importance of the attributes available and up-to-date at all times, using low computational cost. Therefore, the importance of each feature is computed in an on-line fashion. Of course, these methods may be applied to one or several rule sets. Therefore, the ranking methods can be used to track the relevance of the input features both in the AMRules and Random AMRules algorithms.

3.1 The Frequency-based Method.

Let $\mathbf{a}^* = [a_1^*, \cdots, a_d^*]$ be the vector that maintains the overall attribute importance, and $\mathbf{a}^l = \left[a_1^l, \cdots, a_d^l\right]$ be the contribution of the rule R_l to the overall attribute importance $\mathbf{a}^*$.

The first proposed method consists of simply counting the number of times that each input attribute appears in the literals of the rules. It starts by initializing $\mathbf{a}^*$ as a null vector. Every time a default rule is created, the corresponding contribution vector $\mathbf{a}^D$ is also initialized as a null vector. Then, these data structures are updated when a rule expansion or a change detection event occurs:

- **Rule expansion event** - Let `op` be a comparison operator. When R_l is extended by adding a literal $L = (X_j \texttt{ op } v)$ involving input attribute X_j, then the corresponding accumulated values are updated as $a_j^* \leftarrow a_j^* + 1$ and $a_j^l \leftarrow a_j^l + 1$.
- **Rule drop event** - When a change is detected in a rule R_l, resulting in dropping the rule from the rule set, the overall attribute importance is updated as $\mathbf{a}^* \leftarrow \mathbf{a}^* - \mathbf{a}^l$.

3.2 The Merit-based Method.

The merit-based method updates the feature importance every time the splits are evaluated. It ranks the importance of the attributes considering the merit-function used to evaluate the splits. For each attribute X_j, a weight w_j^l is computed in accordance to the merit of the best split of X_j and to the number of literals n_l in the rule R_l. In regression problems w_j is computed considering the variance reduction (defined in Equation 1) of the split, and is computed as:

$$w_j^l = \frac{1}{1 + n_l} VR(X_j, v). \tag{2}$$

As n_l increases the influence of $VR(X_j, v)$ diminishes so that higher relevance is given to the first-added literals when compared to the latter ones.

The merit-based method initialises $\mathbf{a}^*$ and $\mathbf{a}^D$ as null vectors, and n_l as 0. The method proceeds by reacting to the following three events.

- **Split evaluation event** - Let $\mathbf{w} = [w_1, \cdots, w_d]$ be a vector containing the weights for all d input attributes. After evaluating the merit of the best splits for a given rule R_l, the overall relevance of the attributes are updated as $\mathbf{a}^* \leftarrow \mathbf{a}^* - \mathbf{w}_{old}^l + \mathbf{w}^l$, where $\mathbf{w}_{old}^l$ is the weight vector resulting from the previous split evaluation of R_l. If the split evaluation results in a rule expansion then $\mathbf{w}_{old}^l \leftarrow \mathbf{0}$. Similarly, the local relevance is updated as $\mathbf{a}^l \leftarrow \mathbf{a}^l - \mathbf{w}_{old}^l + \mathbf{w}^l$.
- **Rule expansion event** - If the event above results in a rule extension, then the number of literal in the rule should also be updated: $n_l \leftarrow n_l + 1$.
- **Rule drop event** - When a rule R_l is pruned from the rule set the local relevance contribution is deducted from the overall attribute importance: $\mathbf{a}^* \leftarrow \mathbf{a}^* - \mathbf{a}^l$.

In contrast to the frequency-based method, this method is able to express feature relevance even before any rule (besides the default rule) is added to the rule set. In addition, it can express feature importance variations without the need for rule expansion.

3.3 The Weighted Majority Method.

The last feature ranking method is based in the Weighted Majority algorithm [14]. Each rule's importance vector $\mathbf{a}^l$ is initialized as an all-ones vector. Every time the split evaluation event occurs, if the best merit of a given attribute

X_j is less than a threshold λ and X_j does not belong to the antecedent of R_l then the corresponding importance is multiplied by a factor θ. The local importance of the attributes that are not relevant for a particular rule decreases as time goes by. Also, when a rule R_l expands by an attribute X_j the corresponding local attribute importance is reset to $\mathbf{a}_j^l = 1$. The idea is that an attribute that is part of the antecedent of a rule is always relevant. The factor θ depends on the number of times m_l that R_l has expanded, and is defined as $\theta = \frac{m_l+1}{m_l+2}$. If the rule as seen no expansions then $\theta = \frac{1}{2}$, and as the number of expansions increases the value of θ gets closer to 1. The motivation is to be gentler in dropping the importance of the attributes as the number of antecedents of the rule increases.

The weighted majority method maintains a global attribute importance structure $\mathbf{a}^*$ which corresponds to the mean vector of the local importance vectors $\mathbf{a}^l$:

$$a_j^* = \frac{1}{n}\sum_{l=1}^{n} a_j^l \tag{3}$$

where n is the number of rules in the rule set. However, Equation 3 is not computed every time an event occurs. In order to reduce computational costs, $\mathbf{a}^*$ is computed on-line as described below.

- **Split evaluation event** - Let $\mathbf{a}^{l'}$ and $\mathbf{a}^l$ be the local importance vectors before and after the split evaluation event. Each component of the overall attribute importance vector is updated as $a_j^* \leftarrow \frac{a_j^* n - a_j^{l'} + a_j^l}{n}$, where n is the number of rules before the expansion.
- **Rule expansion event** - If the rule expanded is a default rule, then a new default rule will be created resulting in a new local importance vector defined as an all-ones vector, $\mathbf{a}^D = \mathbf{1}$. Therefore, the overall attribute importance vector will be updated as $a_j^* \leftarrow \frac{a_j^* n + 1}{n+1}$.
- **Rule drop event** - When change is detected resulting in dropping a rule R_l, the overall attribute importance is updated by: $a_j^* \leftarrow \frac{a_j^* n - a_j^l}{n-1}$.

3.4 Discussion.

For FB and MB methods, an attribute i is considered irrelevant if its relevance a_i^* is below $\frac{1}{2d}\sum_{j=1}^{d} a_j^*$. For WM, an attribute has no relevance if $a_i^* < 0.5$. All the methods are efficient, with low computational complexity and easy to implement. They can be applied both for classification and regression problems. The first two methods, the *Frequency-based* and *Merit-based* methods, use the feature selection criterion to quantify the relevance of features. In these methods, the feature relevance starts from 0 and increases when the feature is selected by the feature selection criteria. The *Frequency-based* method, only applies when a rule is expanded. In that case, we add 1 to the relevance of the selected attribute. The *Merit-based* method, updates the feature score whenever the set of features are evaluated. In that case, the relevance of all attributes is updated with the value of the merit-function for that attribute. The relevance of these two methods has no upper limit, the score varies from $(0, inf[$. The relevance in *Frequency-based method* grows by steps, while in the *Merit-based method* it grows smoothly. The starting point for the *Weighted-majority feature ranking* method is 1, that is, in the absence of any information, all the features are relevant. The relevance of features decreases over time. The relevance in the weighted majority feature ranking is bounded, taking values in the interval $(0, 1]$.

4. EXPERIMENTAL EVALUATION

We implemented the three proposed methods for ranking feature relevance in AMRules and Random AMRules. Both algorithms run using their default parameter settings.

In most of the real-world datasets, the relevance of attributes is unknown. In order to study the ability of the proposed method to identify the relevant attributes we need to know the ground truth. For this propose, we use artificial data. Artificial data allow us to design controlled experiments where we know which are the relevant attributes; when they become irrelevant, which ones become relevant, and so forth. In this section, we present results using two artificial data sets. In these datasets, the relevance of the attributes evolve over time. It is known which attributes are relevant and irrelevant in each period.

In the set of experiments reported here, the impact of the feature selection methods in the performance of AMRULES is reduced and without statistical significance. Due to the lack of space, we prefer focusing the analysis in the set of features select by each method.

4.1 Experiences with the SEA dataset.

This is a benchmark dataset [18] for drift detection. It is a two-class problem, described by 3 attributes and 60k examples. Attributes are numeric, taking values between 0 and 10. Only two of them are relevant. There are three drifts, dividing the dataset into 4 regions. In the first 2 regions, the relevant attributes are the first and second. For the last 2 regions, the relevant attributes are the first and third.

Using artificial datasets, we can perform controlled experiences, quantifying reactions and delays. Figure 1 plots the relevance of the 3 features over time. The first column of plots report the results of AMRules, and the second column refer to RandomRules. Each row refers to one of the methods: frequency-based, merit-based and the weighted majority feature ranking method. While in the first two methods, the relevance of features starts from 0, in the weighted majority feature ranking method the relevance starts from 1. The relevance of the first two methods has no upper limit, varies from $(0, inf[$. The relevance in the weighted majority feature ranking is bounded, taking values in the interval $(0, 1]$.

From these plots, we can conclude that all the three methods capture the relevant and irrelevant features. Moreover, the three methods are able to track the changes in relevance of the features in the presence of drift. RandomRules, that aggregates the information from 10 trees, better captures the relevance of features. The weighted majority feature ranking method (WM) with RandomRules is the method that first captures the change in relevance of the features.

Table 1: Summary of results using benchmark data sets.

Dataset	Examples	Attributes	Irrelevant	AMRules FB	AMRules MB	AMRules WM	RAMRules FB	RAMRules MB	RAMRules WM
2D planes	40768	10	3	3(0+0)	3(0+0)	3(0+0)	3(0+0)	3(0+0)	3(0+0)
bank8FM	8192	8	0	6(6+0)	1(1+0)	1(1+0)	4(4+0)	1(1+0)	0(0+0)
Fried	40768	10	5	7(2+0)	5(0+0)	5(0+0)	5(0+0)	5(0+0)	5(0+0)
FriedDrift	50000	10	5	7(2+0)	5(0+0)	5(0+0)	6(1+0)	5(0+0)	5(0+0)
Californian Housing	20460	8	–	4	0	0	3	0	0
Ailerons	13750	40	–	37	20	18	31	20	15
Elevators	16559	18	–	16	8	9	12	9	5
House8L	22784	8	–	4	1	0	4	1	0
House16H	22784	16	–	12	0	3	8	0	0

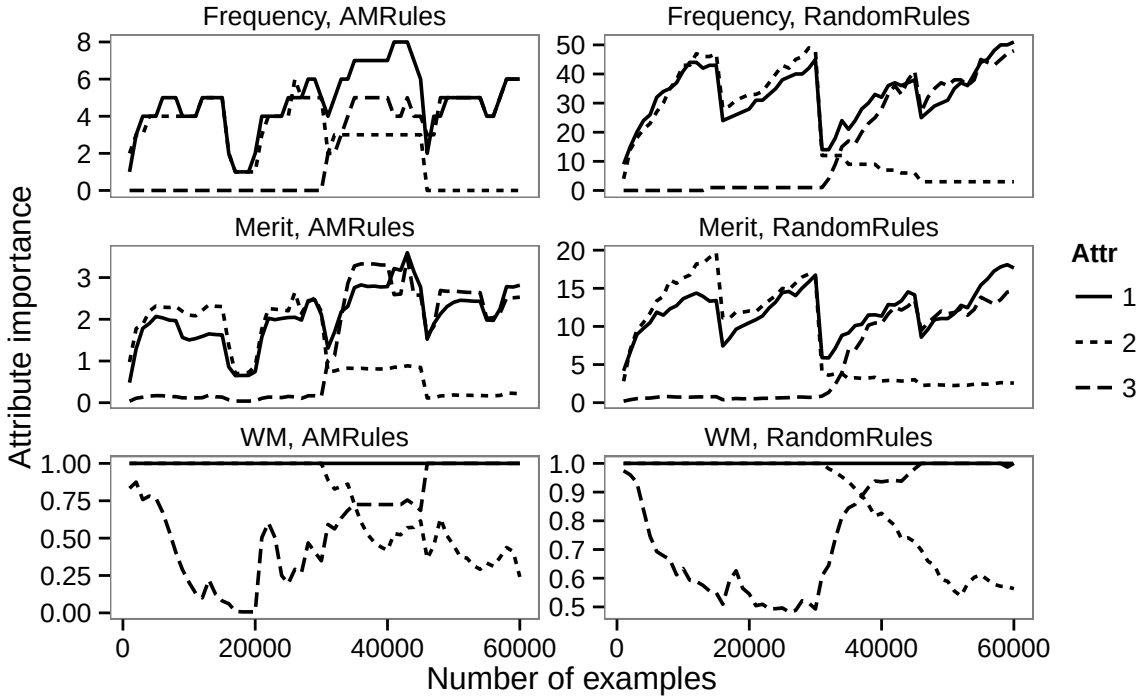

Figure 1: Study on the Relevance of Features using the SEA Dataset

4.2 Experiences with the FRIED dataset.

This is a well-known regression dataset with 40768 cases, described by 10 continuous attributes. The only relevant attributes are the first five ones; the last five attributes are irrelevant.

Figure 2 plots the relevance of the 10 features over time. Again, the first column of plots report the results of AMRules, and the second column refer to RandomRules. Each row refers to one of the methods: frequency-based, merit-based and the weighted majority feature ranking method. From these plots, we can conclude that all the three methods capture the relevant and irrelevant features.

Again, the best method is the weighted majority feature ranking method (WM) with RandomRules. In this set of experiments, RandomRules with WM correctly detect all the irrelevant features. No relevant feature was signalled as irrelevant.

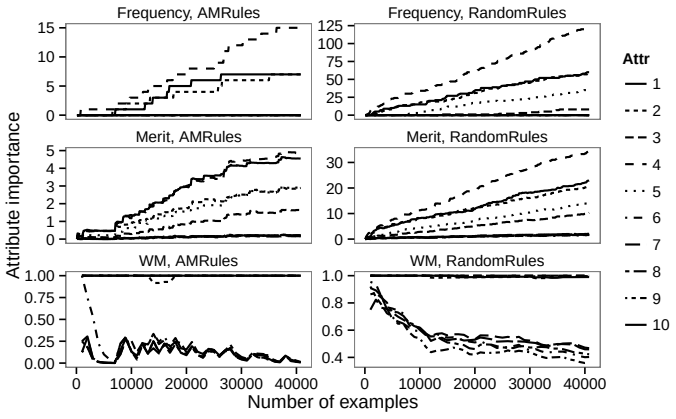

Figure 2: Study on the Relevance of Features using the Fried dataset.

4.3 Summary of results in benchmark datasets.

Table 1 presents a summary of results using benchmark datasets. We use artificial data (first 4 datasets), where the irrelevant attributes are known, and real-data with unknown categorization of feature relevance. The last six columns show the number of irrelevant attributes found by the frequency-based (FB), merit-based (MB) and weighted majority (WM) feature ranking methods when used embed with AMRules and Random AMRules (RAMRules). For each experience, we report a string: **'NR(T1+T2)'**, where **NR** is the number of irrelevant attributes reported by the algorithm, **T1** the number of relevant attributes reported as irrelevant, and **T2** the number of irrelevant attributes reported as relevant.

All methods correctly identified all irrelevant features for 2D planes. However, only WM detected the correct number of non-relevant features for bank8FM (0) and only with RAMRules. MB detected 1 irrelevant feature both with AMRules and RAMRules and FB was the worst performing method. The low number of expanded rules in the rule sets explains the poor performances of MB and (especially) FB methods. This resulted from the small size of the dataset. For Fried and FriedDrift, MB and WV always correctly identified all irrelevant features. FB incorrectly classified two features as irrelevant with AMRules on both datasets and one with RAMRules in FriedDrift. From the analysis of the synthetic datasets we conclude that WM is the best method followed by MB. These methods correctly identified the irrelevant features in most of the cases. Also, just counting the frequencies of attributes on the antecedents of rules is not an effective method for detecting unimportant features. Analysing the results for the real data sets, we observe that FB usually detects more irrelevant features than the other methods. FB only takes into account the presence of the attributes in the antecedents of the rules and not its merits. The MB and WM methods overcome this problem and consequently identify much less features as irrelevant.

5. CONCLUSIONS

In this paper, we discuss three online methods for continuously quantifying the relevance of features: the *Frequency-based* method, the *Merit-based* method, and the *Weighted-majority feature ranking* method. The first two methods are online versions of the feature ranking in Random Forests. The third one, embed in rule learning the weight update rule of the WinNow algorithm. The three methods exploit characteristic of the Hoeffding algorithms for learning trees and rules. Our experimental evaluation, WM with RandomRules is the most effective and robust method to detect relevant features. Moreover, the WM is the fastest to adapt to changes in relevance of the features in presence of drift.

Acknowledgements: Authors acknowledged the support given by European Commission through MAESTRA (ICT-2013-612944) and the Project TEC4Growth financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement.

6. REFERENCES

[1] J. P. Barddal, H. M. Gomes, F. Enembreck, B. Pfahringer, and A. Bifet. On dynamic feature weighting for feature drifting data streams. In *ECML PKDD 2016*, pages 129–144, 2016.

[2] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. Moa: Massive online analysis. *Journal of Machine learning Research (JMLR)*, pages 1601–1604, 2010.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] P. Domingos and G. Hulten. Mining High-Speed Data Streams. In *Proc. of the ACM Sixth International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Boston, USA, 2000. ACM Press.

[5] J. Duarte and J. Gama. Ensembles of adaptive model rules from high-speed data streams. In *Proc. of the 3rd Int. Workshop BigMine*, pages 198–213, 2014.

[6] J. Duarte, J. Gama, and A. Bifet. Adaptive model rules from high-speed data streams. *TKDD*, 10(3):30, 2016.

[7] J. Gama and P. Kosina. Learning decision rules from data streams. In *IJCAI*, pages 1255–1260. AAAI, Menlo Park, USA, 2011.

[8] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[9] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[10] H. Huang, S. Yoo, and S. P. Kasiviswanathan. Unsupervised feature selection on data streams. In *Proc. of CIKM*, pages 1031–1040, 2015.

[11] E. Ikonomovska, J. Gama, and S. Dzeroski. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.*, 23(1):128–168, 2011.

[12] I. Katakis, G. Tsoumakas, and I. P. Vlahavas. On the utility of incremental feature selection for the classification of textual data streams. In *Panhellenic Conference on Informatics'05*, pages 338–348, 2005.

[13] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, pages 285–318, 1988.

[14] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, Feb. 1994.

[15] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[16] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[17] M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53(1-2):23–69, 2003.

[18] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proc. of the 7th ACM SIGKDD*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM.

[19] H. Yang, M. R. Lyu, and I. King. Efficient online learning for multitask feature selection. *ACM Trans. Knowl. Discov. Data*, 7(2):6:1–6:27, Aug. 2013.