Performance Evaluation of Feature Selection Algorithms Applied to Online Learning in Concept Drift Environments

Matheus B. de Moraes[1], **André L. S. Gradvohl**[1]

[1]School of Technology – University of Campinas (UNICAMP)
R. Paschoal Marmo, 1888. CEP 13484-332. Limeira-SP – Brazil

matheuzmoraes@gmail.com, gradvohl@ft.unicamp.br

Abstract. *Data streams are transmitted at high speeds with huge volume and may contain critical information need processing in real-time. Hence, to reduce computational cost and time, the system may apply a feature selection algorithm. However, this is not a trivial task due to the concept drift. In this work, we show that two feature selection algorithms, Information Gain and Online Feature Selection, present lower performance when compared to classification tasks without feature selection. Both algorithms presented more relevant results in one distinct scenario each, showing final accuracies up to 14% higher. The experiments using both real and artificial datasets present a potential for using these methods due to their better adaptability in some concept drift situations.*

1. Introduction

According to [Gradvohl 2016], Complex Event Processing (CEP) systems are stream processing distributed systems, which take one or more linearly ordered sequence of events (known as Data Streams) as an input and produces another ordered sequence of events as output. Another common concept is the Stream Paradigm (SP), which [Andrade et al. 2011] state as a distributed computational model, supporting the continuous, heterogeneous, real-time collection and analysis of data streams.

[Ramírez-Gallego et al. 2017] define data stream as a potentially unbounded, ordered sequence of data, generally transmitted at high volume and velocity, generated from different applications and devices, such as wireless sensors, RFID readers, GPS and social media, among others. Most of the time, it contains critical information and must be processed and analyzed in real time, as in network attack or anomaly detection, disaster management, trend analysis and financial market, among others [Gradvohl et al. 2014]. We will call this process of online processing and analysis as Online Learning.

Besides its volume and velocity, one of the main challenges in learning from data streams is the dynamically changing, or non-stationary environment, which means data distribution can change over time. This phenomenon, known as Concept Drift [Jankowski et al. 2016], imposes difficulties for building useful solutions since every model, application or algorithm needs to adapt to different changes in data distribution.

CEP systems can handle different types of streams, including ones containing concept drifts. According to [Gama et al. 2014], there are four types of drifts: sudden (or abrupt), incremental, gradual and recurring (or reoccurring).

The sudden drift occurs when data distribution rapidly switches from one concept to another (e.g., replacement of a sensor with a different calibration in a plant). Incremental drift happens when there are many intermediate concepts between two different concepts in data distribution (e.g., when a sensor slowly wears off and becomes less accurate). In turn, the Gradual drift occurs when the concept keeps changing from one concept to another for some time (e.g., sales of a football team increases and decreases every time it loses or wins in a season). Finally, the recurring drift, when concepts may reoccur after some time (e.g., sales of a product may increase every winter and decrease every summer).

As a result, many researchers tend to apply techniques to reduce the size or dimensionality of the streams, aiming to optimize learning accuracy, reduce computational costs and improve the time needed for processing and analyzing the data. Feature selection (FS) is a technique that intends to reduce the number of features (or attributes) in a dataset by removing irrelevant or redundant features [Han and Kamber 2006]. The FS, widely used in stationary environments, is not a trivial task in dynamically changing environments, such as in data streams processing, since features dependency also may change over time.

There is already some feature selection algorithms for online learning in literature. However, as pointed by [Ramírez-Gallego et al. 2017], most of these algorithms are adaptations from the ones used in stationary environments. Therefore, they were not built to deal specifically with data streams and, consequently, with the concept drift phenomenon, and may not present suitable results in those situations.

Hence, there is a need to evaluate these algorithms when applied to online learning in concept drift environments. This paper presents a performance evaluation of two feature selections methods, the Katakis Method [Katakis and Tsoumakas 2005] and Online Feature Selection [Wang et al. 2014], applied to online learning in different situations, using both real and artificial datasets containing all types of drifts.

We organized the remaining of this paper as follows. Section 2 introduces the Katakis Method. Section 3 presents the Online Feature Selection Algorithm. Section 4 describes the Experimental Setup for the experiments. Section 5 presents the results and discussion. Finally, Section 6 presents the conclusions.

2. The Katakis Method

The Katakis Method, proposed by [Katakis and Tsoumakas 2005] is one of the most used feature selection methods, initially used for text classification. This method proposes two components in conjunction: a) an incremental feature ranking method and b) an incremental learning algorithm that can consider a subset of features during prediction.

A feature ranking method is the one who can evaluate the predictive power of all features of a data streams, selecting the N best ones. These methods evaluate each feature based on a cumulative statistic considering the number of times it appears in each different class in the streams. This approach implies these methods are, mainly, incremental.

When new data streams arrive, the algorithm updates the statistics and immediately calculates the evaluation, without re-processing past data. The authors inform they can achieve the first part of their approach using many different methods, such as information gain, χ^2, and mutual information. In this paper we selected, the Information Gain (IG), proposed by [Quinlan 1986], as an incremental feature rank algorithm.

The incremental re-evaluation of features and addition of new features will inevitably result in specific features being promoted to or demoted from the top N features. That re-evaluation raises a problem solved with the second part of this approach: a learning algorithm that can classify a new tuple, taking into account different features over time. For this part, the authors propose the Naïve Bayes algorithm, due to its simplicity and flexibility.

3. Online Feature Selection algorithm

The Online Feature Selection (OFS) algorithm, proposed by [Wang et al. 2014], is an ϵ-greedy wrapper algorithm for online learning with partial inputs. This algorithm is used in situations where the full domain of attributes is unknown. The authors set the problem as this: Let $\{(x_t, y_t) \mid t = 1, \ldots, T\}$ be a sequence of input streams received, where each $x_t \in \mathbb{R}^d$ is a vector of d dimension and $y_t \in \{-1, +1\}$ is the class of each tuple. They assume d is a large number and, for computational efficiency, there is a need to select a small number of features for linear classification. That means, for each trial t, the algorithm will use the learner $w_t \in \mathbb{R}^d$ to classify a tuple x_t.

Rather than using all features for classification, the classifier $w + t$ will use, at most, B non-zero features, i.e., $||w_t||_0 \leq B$, where $B \geq 0$ is a predefined threshold. The authors propose the utilization of this algorithm for feature selection in situations where the full domain of attributes is unknown, using an exploration-exploitation trade-off technique, first defined by [March 1991].

They assume that the algorithm knows only a part of feature domain and tuples. In this approach, the algorithm will spend ϵ trials in $exploration$, choosing randomly B features of the total d and $1 - \epsilon$ trials in $exploitation$ choosing B features in which the classifier w_t has non-zero values. The Algorithm 1 presents the pseudo-code for OFS.

The Algorithm 1 initiates its execution by initializing the classifier w_t as zero. In the next step, for each tuple presented in the stream, it extracts a sample Z_t from a Bernoulli distribution with probability ϵ. If the sample $Z_t = 1$, then the algorithm will randomly select B from the total $[d]$, storing it in the C_t variable. If sample $Z_t \neq 1$, then the features stored in C_t will be the ones who have non-zero values at the classifier w_t.

After that, the algorithm will receive a tuple $\tilde{x_t}$ by only requiring the features presented in C_t. Then, the algorithm makes a prediction of the possible class with the linear function $sgn(w_t^T \tilde{x_t})$. The algorithm receives the true class y_t. If the prediction is incorrect ($y_t w_t^T \tilde{x_t} \leq 1$), the classifier must be updated. In this case, it computes a new variable $\hat{x_t}$, where each feature $[\hat{x_t}]_i$ will receive as a value the division between the feature in the same position at $\tilde{x_{ti}}$ and the operation $\frac{B}{d}\epsilon + I([w_t]_i \neq 0)(1 - \epsilon)$ for each feature i in the full domain d.

4. Experimental setup

We elected the Massive Online Analysis (MOA) framework as a benchmark tool for running the experiments presented in this paper. MOA is an open-source framework developed by the University of Waikato, which allows the manipulation and simulation of data streams. It is integrated with the Waikato Environment for Knowledge Analysis (WEKA) and provides a group of native tools to evaluate different algorithms in streaming environments.

```
Algorithm 1: Online Feature Selection
    Input : $R$ : maximum L2 norm
            $\eta$ : step size
            B: the number of selected features
            $\epsilon$: the exploration-exploitation trade off
 1  $w_1 = 0$
 2  for $t = 1, 2, \ldots, T$ do
 3      $w_t = 1$
 4      Sample $Z_t$ from a Bernoulli Distribution with probability $\epsilon$
 5      if $Z_t = 1$ then
 6          Randomly Choose B attributes $C_t$ from [d]
 7      end
 8      else
 9          Choose the attributes that have nonzero values in $w_t$, i.e.,
              $C_t = \{i : [W_t]_i \neq 0\}$
10      end
11      Receive $\tilde{x_t}$ by only requiring the attributes in $C_t$
12      Make prediction sgn($w_t^T \tilde{x_t}$)
13      Receive $y_t$
14      if $y_t w_t^T \tilde{x_t} \leq 1$ then
15          Compute $\hat{x_t}$ as
16          $[\hat{x_t}]_i = \frac{[\tilde{x_t}_i]}{\frac{B}{d}\epsilon + I([w_t]_i \neq 0)(1-\epsilon)}, i = 1, \ldots, d$
17          $\widetilde{w}_{t+1} = w_t + y_t \eta \hat{x}_t$
18          $\hat{w}_{t+1} = min\left\{1, \frac{R}{||\widetilde{w}_{t+1}||_2}\right\} \widetilde{w}_{t+1}$
19          $w_{t+1} = \text{TRUNCATE}(\hat{w}_{t+1}, B)$
20      end
21      else
22          $w_{t+1} = w_t$
23      end
24  end
```

We selected this framework because of its widespread use in the literature – [Bifet et al. 2010], [Turkov et al. 2016] and [Ramírez-Gallego et al. 2017] –, its user-friendly interface, for being open-source, and, at last, for its built-in tools to evaluate algorithms and procedures of machine learning when applied to data streams, such as classifiers, feature selectors, predictors, and regressors. We choose the following versions: MOA 2016.04 and WEKA 3.8.

For being open-source, MOA allows the development and incorporation of algorithms in its structure. We choose the Java programming language to implement the algorithms, the same used by MOA and WEKA, to facilitate the integration between components, avoiding any communication conflicts with the rest of the components of MOA and WEKA.

We executed the whole experimental environment in a single standard desktop machine, with the following structure: Intel Core i7-3770 processor (4 cores/ 8 threads, 3.40 GHz, 8M cache), 16 GB DD3 of RAM, 1 TB SATA HDD, wireless Internet connection and Ubuntu 16.04 LTS (Linux).

4.1. Algorithms and Parameters

We selected an incremental version of the Naïve Bayes (NB) algorithm as a base classifier. We elected this algorithm due to its simplicity and flexibility, for allowing the use of different subsets of features along the classification phase and by the previous utilization in literature as a base classifier for evaluating feature selection algorithms in data streams environments [Ramírez-Gallego et al. 2017].

We also implemented IG and OFS algorithms to verify their performance in online learning. Therefore, we evaluated three different situations in this work: NB-only, NB + IG, and NB + OFS. We will refer to these situations as NB, IG, and OFS, respectively. We developed and packed them into a library to work together with MOA. The source code of the project is available in our repository [de Moraes 2018].

Except for the NB algorithm, we can configure both IG and OFS algorithms to execute with two parameters: `numFeatures` and `winSize`. We used the first parameter to configure how much attributes the selected feature selection method must use for the learning process. Therefore, the method will select the `numFeatures` best attributes. We used the `winSize` parameter to configure how the algorithm will perform the learning process. When dealing with data streams, a CEP system can analyze and process data streams tuple by tuple or in windows of size N. Therefore, `winSize` is the number of tuples that the algorithm will process at a time.

4.2. Metrics

As pointed out by [Ramírez-Gallego et al. 2017], when evaluating online learning algorithms, one must use the right metrics. Therefore, we evaluate the algorithms under the following metrics:

- **Accuracy**: In data stream mining, due to its evolving nature, there is a fact that the relevance of instances diminishes over time. Therefore, using average measures does not reflect how a learning algorithm was able to adapt and react to the changes in the data distribution. Thus, it is imperative to use methods which are calculated using only the most recent instances. In this paper, aside from final accuracy, we also evaluated Prequential Accuracy. This metric provides the accuracy of only the last processed instances.
- **Memory Consumption**: a learning algorithm must not overload the machine since many CEP systems run on limited computational resources shared with many other applications. Therefore, the learning algorithm must execute at a low computational cost. The metric considered in this paper is RAM per hours (RAM/hours).
- **Response time**: Adding a feature selection method must not negatively and significantly impact the learning algorithm response time since sources transmit data streams at high velocities and huge volumes. This impact may cause an overload in the operators' queue. This situation would impact the real-time processing, a core characteristic of CEP systems.

4.3. Datasets

To evaluate the presented algorithms, we used several datasets in the experiments, listed in Table 1. Spam_data [Katakis et al. 2009] uses part of the SpamAssassin collection, an anti-spam tool maintained by the Apache Foundation, and contains several instances, each one is a different e-mail. It has a high dimensional feature domain, where each attribute is binary and defines whether a specific word exists or not in the e-mail. Two classes – legitimate and spam – compose the dataset, which we can use for e-mail classification. In this dataset, 20% of instances are spam emails.

Table 1. Selected datasets

Dataset	#Instances	#Features	#Classes	Type of drift	Artificial
spam_data	9.324	40.000	2	Gradual	No
mailing_list	6.000	28.000	2	Sudden	No
incremental_drift	100.000	6	2	Incremental	Yes
gradual_drift	100.000	6	2	Gradual	Yes
sudden_drift	100.000	6	2	Sudden	Yes
recurring_drift	100.000	12	2	Recurring	Yes

The mailing_list [Katakis et al. 2009] dataset is a part of 20 newsgroups collection and involves a user that subscribes to and removes from different general mailing lists. It was created to simulate concept drift. The artificial datasets have numeric attributes and they were all generated using MOA, which has a built-in tool to generate datasets simulating concept drifts.

4.4. Experiments

To asses the quality of the methods, we must use an online evaluation approach. This approach, known as interleaved test-then-train, proposed by [Bifet and Kirkby 2011], defines a model which evaluates each example/batch, arriving at a time t, against a $t-1$ model, and then it serves as an input to update that model. Therefore, the algorithm uses each tuple to evaluate the model. This approach makes the model work incrementally. This evaluation is available for use in MOA.

Using this approach, we perform the following experiments:

- **Experiment 1**: classification using only NB, without feature selection. In this case, the default window is 1 (tuple by tuple).
- **Experiment 2**: use of IG and OFS algorithms in conjunction with NB. For real datasets, the parameters were `winSize` = 1 and `numFeatures` = 4, 10 and 100. For artificial datasets, the parameters were : `winSize` = 1 and `numFeatures` = 4.
- **Experiment 3**: Window variation (`winSize`) between 10, 100 and 1000.

We repeated each experiment 10 times. The results presented in this paper refer to the mean of the obtained values in each execution. For the OFS algorithm, we considered $\lambda = 0.01$ and $\eta = 0.02$, according to the authors' criteria [Wang et al. 2014]. For the B selected features, it is inputted by the `numFeatures` parameter already described.

5. Results and Discussion

Results showed that IG presents the worst response time and memory consumption in all evaluated scenarios. Considering artificial datasets and `window` = 1, it consumed a time 500.000 higher (about 7 hours against 5 seconds for NB and OFS). For larger windows, their performance was superior.

In the *incremental_drift*, where IG showed the worst result for `window` = 1, considering `winSize` = 10, the algorithm reduced the response time by approximately 700%, reaching 1 hour execution. With `winSize` = 100, it classified all tuples in 5 minutes and, at last, with `winSize` = 1000, it took only 25 seconds. In the latter, the accuracy was the best of all windows for this algorithm. However, still far below that obtained by NB. For the memory consumption, in the worst case, IG needed 2.64×10^{-5} RAM/hours and, in the best case, it took 3.21×10^{-8} RAM/hours.

Concerning the OFS, it showed competitive speeds in comparison with NB. In the worst case, considering `winSize` = 1 in the *incremental_drift* dataset, it processed and classified all tuples in 5 seconds. With `winSize` = 10,100 and 1000 it took, respectively, 1.5 seconds, 0.9 seconds and 0.5 seconds. In the latter, it was only 0.07 seconds behind NB. As for the memory consumption, in the worst case, OFS only needed 5×10^{-9} RAM/hours and, in the best case, 5.2×10^{-10} RAM/hours. The latter was the same obtained by using only NB.

Tables 2 and 3 presents the final accuracies. Figure 1 shows the obtained prequential accuracies in the experiments for both real and artificial datasets.

Table 2. Final accuracy (%) by method, window and number of selected features for real datasets. We highlighted the best results in bold. We did no selection for the Naïve Bayes.

	NB (%)	Window	IG (%)			OFS (%)		
			4	10	100	4	10	100
spam_data	74.57	1	85.81	**88.73**	88.56	76.16	77.73	76.56
		10	81.85	**83.10**	82.92	75.81	77.97	76.29
		100	85.67	85.23	**86.16**	76.11	77.58	73.73
		1000	**85.73**	81.50	83.22	76.03	77.37	76.31
mailing_list	**84.60**	1	60.23	66.31	80.79	53.30	53.76	56.92
		10	60.29	70.62	78.65	58.55	59.77	64.93
		100	50.29	69.46	78.21	54.13	54.82	57.99
		1000	60.92	67.67	75.46	53.66	54.63	57.22

The non-parametric Friedman test was used to validate if the results were concise, according to the method presented by [Demšar 2006]. This test proposes a null hypothesis where all evaluated algorithms are equivalent and, therefore, are statistically equal. To verify this hypothesis, the test evaluates each algorithm by setting an individual rank, where the best algorithm gets the rank 1, the second best the rank 2, and so forth. If the test rejects the null hypothesis, it means that the values are concise.

For a confidence interval of $\alpha = 0.05$, the test rejects the null hypothesis. Then, we also applied a *post-hoc* test, known as Bonferroni-Dunn [Demšar 2006], to verify if

Table 3. Final accurracy (%) by method and window artificial datasets. We highlighted the best results in bold. We did no selection for the Naïve Bayes.

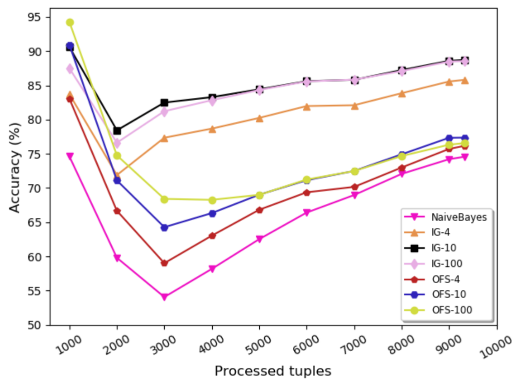
	NB (%)	Window	IG (%)	OFS (%)
incremental_drift	**77.60**	1	71.60	76.93
		10	71.63	77.04
		100	71.83	77.04
		1000	71.90	76.92
gradual_drift	**92.91**	1	92.72	92.87
		10	92.76	92.81
		100	92.59	92.86
		1000	92.60	92.87
sudden_drift	73.38	1	77.89	**78.11**
		10	77.90	**78.31**
		100	77.80	**78.02**
		1000	71.00	**78.10**
recurrent_drift	**73.53**	1	72.18	63.69
		10	72.10	65.34
		100	72.18	64.12
		1000	67.65	63.70

the results of the base classifier NB were significantly higher than IG and OFS. This test uses the previous rank generated by Friedman test and, if the difference between the algorithms is greater or equal than a predefined critical value, then it indicates that there is a statistical difference between the results. Figure 2 shows the results for both real and artificial datasets, considering all windows.

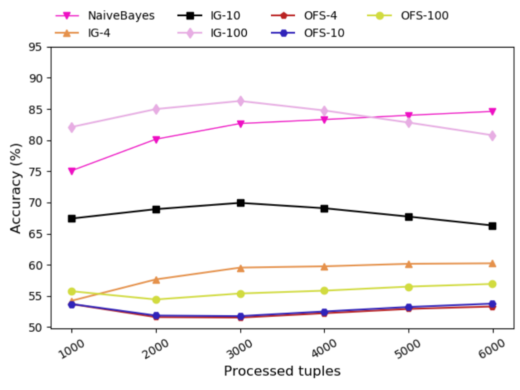
For both real and artificial datasets, considering a confidence interval of $\alpha = 0.05$, the critical values were 2.638 and 2.343, respectively. Analyzing the results, there is evidence that, although NB obtains higher final accuracies in most scenarios (4 of 6), this superiority is not significant since in both types of datasets the results of NB are lower than the critical value in comparison of IG and OFS.

[Katakis and Tsoumakas 2005] present the accuracy of the proposed method using an incremental version of NB as a base classifier and χ^2 as the feature selection method. They used the *spam_data* dataset to simulate data streams. The results obtained in our work shows similar accuracies. However, the authors did not evaluate any other metric. Therefore, the experiments performed in this work show that, although the final accuracy is reasonable, the high memory consumption and response time make this method using IG unfeasible.

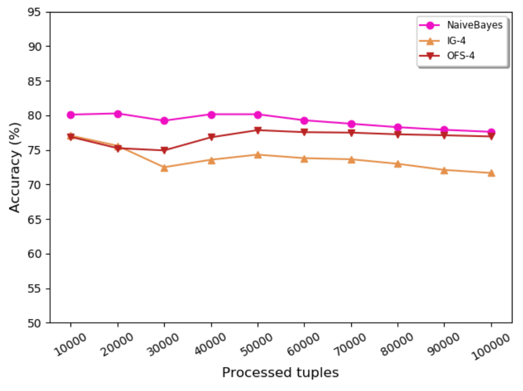
However, considering prequential accuracy, IG shows the better adaptability for gradual and sudden drifts in high dimensionality data streams, and in recurrent drifts in both high and low dimensionality. This fact demonstrates that there is a potential for using this method in concept drift environments, especially in high dimensional data streams, if we reduced the response time and memory consumption.



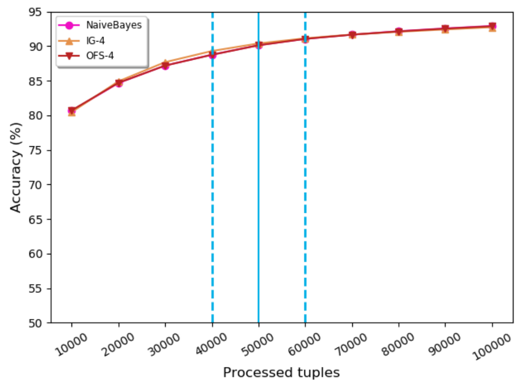
(a) spam_data



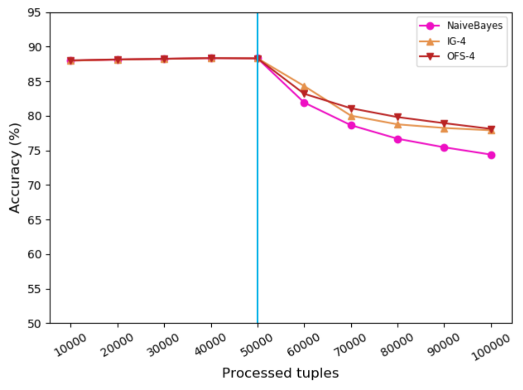
(b) mailing_list



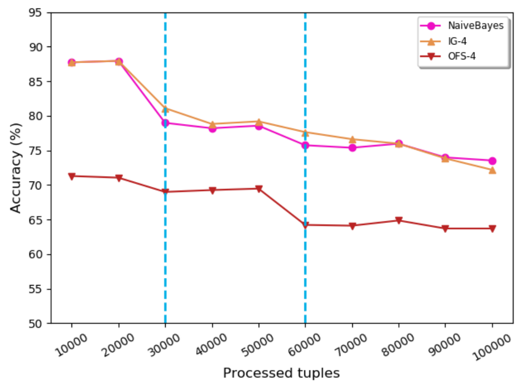
(c) incremental_drift



(d) gradual_drift



(e) sudden_drift



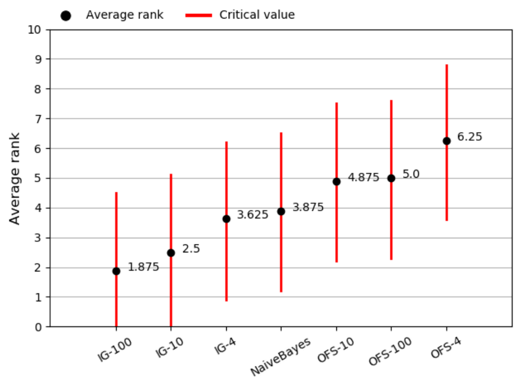
(f) recurrent_drift

Figure 1. Prequential accuracy (%) for each dataset. Solid and dashed vertical blue lines indicates drifts and drift window start/end, respectively. In the incremental drift, the drift occurs in all dataset. For the spam_data and mailing_list these informations are unknown.

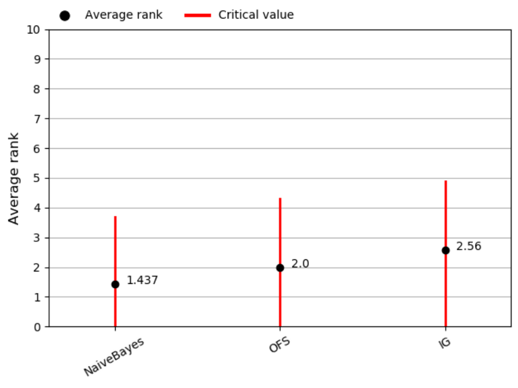
One possible solution for this is the use of high-performance computing libraries, which permits the parallelization of the most expensive parts of the algorithm. Another alternative is the use of Graphics Processing Units (GPU) for the feature rank calculation. In this way, the processor would be responsible only for the classification phase, which would decrease the workload and reduce the response time and memory consumption.

For the OFS algorithm, [Wang et al. 2014] present the results of a systematic evaluation of OFS performance in different situations, including dealing with huge amount of data. In these scenarios, OFS demonstrates a high capacity to deal with large volumes of data, both in high dimensional tuples and attributes, when there is no change in data distribution.

However, our experiments showed that OFS is sensitive to concept drift, presenting lower accuracies in comparison with NB in 5 out of 6 evaluated scenarios. Besides, it showed lower accuracy compared to IG for high dimensional datasets, which may make it unfeasible in those situations. Concerning prequential accuracy, OFS showed the best adaptability to sudden drift when the data dimensionality is small.



(a) Real datasets



(b) Artificial datasets

Figure 2. Average ranks and critical values of the Bonferroni-Dunn *post-hoc* test.

6. Conclusions and Future Works

Feature selection algorithms are a useful tool to reduce data streams dimensionality in real-time. Thus, data streams classification in a complex event processing system can be less costly, faster and efficient. However, these algorithms must demonstrate adaptability to the concept drift phenomenon, which affects data streams forecasting.

The results of our work show that Information Gain and Online Feature Selection demands a higher memory consumption and a higher response time for online learning in comparison to the use of the Naïve Bayes algorithm only. As for accuracy, the Bonferroni-Dunn *post-hoc* test shows evidence that, although Naïve Bayes presents higher final results in 4 out of 6 situations, this superiority is not significant, since its results were below the critical value in all evaluated scenarios.

The Information Gain algorithm obtained the worst response time and memory consumption in all situations. However, it presented competitive accuracies in when compared to the classification without feature selection, overcoming the later in real datasets with high dimensionality. Besides, its accuracy was higher than OFS in 3 out of 6 situations. Therefore, there is a potential for using this method if its performance is improved. Some possible solutions include the use of high-performance computing libraries to parallelize the code or the use of graphics processing units for the feature rank calculation, decreasing the processors' workload.

On the other hand, the OFS algorithm showed memory consumption and response time relatively close to the values obtained without feature selection. Besides, it presented the best adaptability to the sudden drift when the dimensionality of the dataset is small. However, its accuracy in high dimensional datasets was inferior when compared to the IG. This result indicates that, although fast and with low computational cost, its use in high dimensional data streams may be impracticable.

For future works, we intend to extend our research and verify the performance of other feature selection algorithms for online learning in concept drift environments, such as Fast-Correlation Based Filter, Extreme Feature Selection, Online Extreme Feature Selection, and Online Group Feature Selection.

References

Andrade, H., Gedik, B., and Turaga, D. (2011). Fundamentals of stream processing: Application design, systems, and analytics. *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*, pages 1–529.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). MOA: massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604.

Bifet, A. and Kirkby, R. (2011). Data Stream Mining. *Methodology*, 8(May):127–141.

de Moraes, M. B. (2018). Data Streams With Feature Selection. Disponível em: `https://github.com/mbdemoraes/data-streams-feature-selection/`. Last accessed in May 11th, 2018.

Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37.

Gradvohl, A. L. S. (2016). Investigating metrics to build a benchmark tool for complex event processing systems. *Proceedings - 2016 4th International Conference on Future Internet of Things and Cloud Workshops, W-FiCloud 2016*, pages 143–147.

Gradvohl, A. L. S., Senger, H., Arantes, L., and Sens, P. (2014). Comparing distributed online stream processing systems considering fault tolerance issues. *Journal of Emerging Technologies in Web Intelligence*, 6(2):174–179.

Han, J. and Kamber, M. (2006). *Data Mining Concepts and Techniques*, volume 2. Elsevier, Burlington.

Jankowski, D., Jackowski, K., and Cyganek, B. (2016). Learning Decision Trees from Data Streams with Concept Drift. *Procedia Computer Science*, 80:1682–1691.

Katakis, I., Tsoumakas, G., Banos, E., Bassiliades, N., and Vlahavas, I. (2009). An adaptive personalized news dissemination system. *Journal of Intelligent Information Systems*, 32(2):191–212.

Katakis, I. and Tsoumakas, I. V. (2005). On the utility of incremental feature selection for the classification of textual data streams. In Fagerberg, J., Mowery, D. C., and Nelson, R. R., editors, *Advances in Informatics*, pages 338–348. SpringerLink, Volos,Greece.

March, J. G. (1991). Exploracion Y Explotacion En Aprendizaje Organizacional.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1(1):81–106.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., and Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57.

Turkov, P., Krasotkina, O., Mottl, V., and Sychugov, A. (2016). Feature Selection for Handling Concept Drift in the Data Stream Classification. In *12th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 614–629. SpringerLink, New York, USA.

Wang, J., Zhao, P., Hoi, S. C., and Jin, R. (2014). Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710.