Interdisciplinary Journal of Information, Knowledge, and Management

An Official Publication of the Informing Science Institute
InformingScience.org

IJIKM.org

Volume 19, 2024

Technique Analysis for Multilayer Perceptrons to Deal with Concept Drift in Data Streams

Paulo Mauricio Gonçalves, Júnior*	Federal Institute of Pernambuco, Recife, Brazil	paulogoncalves@recife.ifpe.edubr
Sylvain Chartier	University of Ottawa, Ottawa, Canada	sylvain.chartier@uottawa.ca

* Corresponding author

Abstract

Aim/Purpose	This paper describes how to use a multilayer perceptron to improve concept drift recovery in streaming environments.
Background	Classifying instances in a data stream environment with concept drift is a challenging topic. The base learner must be adapted online to the current data. Several data mining algorithms have been adapted/used to this type of environment. In this study, two techniques are used to speed up the adaptation of an artificial neural network to the current data, increasing its predictive accuracy while detecting the concept drift sooner.
Methodology	Experiments were performed to analyze how some techniques behave in different scenarios and compare them with other classifiers built to deal with data streams and concept drifts.
Contribution	This study suggests two techniques to improve the classification results: an embedded concept drift detection method to identify when a change has occurred and setting the learning rate to a higher level whenever a new concept is being learned to give more weight to recent instances, with its value decreased over time.
Findings	Results indicate that gradually reducing the learning rate with an embedded concept drift detector has better statistical results than other single classifiers built to deal with data streams and concept drifts.
Recommendations for Practitioners	Based on the empirical results, this study provides recommendations on how to improve the multilayer perceptron in data stream environments suffering from concept drifts.

Accepting Editor Dirk Frosch-Wilke │ Received: September 21, 2024 │ Revised: November 11, 2024 │ Accepted: November 25, 2024.
Cite as: Gonçalves, P. M., Jr., & Chartier, S. (2024). Technique analysis for multilayer perceptrons to deal with concept drift in data streams. *Interdisciplinary Journal of Information, Knowledge, and Management, 19,* Article 34. https://doi.org/10.28945/5405

Recommendations for Researchers	Researchers should conduct investigations to increase the number of base classifiers used in data stream environments and in situations where concept drifts occur.
Impact on Society	The objective of this study is to increase the use of multilayer perceptrons in data stream environments suffering from concept drifts, as nowadays, Hoeffding Trees and Naive Bayes are the base classifiers mostly used.
Future Research	Additional research includes adapting the online learning rate by increasing/decreasing it based on the performance of the Multilayer Perceptron. This scheme would allow the removal of parameters that must be set by the user, like learning rate upper bound and number of instances to return to the stable value.
Keywords	data streams, concept drift, multilayer perceptron, online learning

INTRODUCTION

Classification in streaming environments is gaining more and more attention in the research areas of pattern recognition and machine learning. This is due to the number of applications that generate data online and must be processed as they arrive, like the stock market, spam filtering (Abad et al., 2014), sensor networks (Branch et al., 2013; Diallo et al., 2012), fraud detection (Pozzolo et al., 2014), among others.

These streaming environments may not be stationary, i.e., they may continually change and evolve. Thus, past data becomes increasingly irrelevant over time, diminishing the accuracy of prediction models trained on it. This phenomenon is described as concept drift (Gonçalves et al., 2024). It can be categorized in terms of speed of change: gradual, when a new concept gradually replaces an old one over a period of time; incremental, when there is a smooth transition between two different concepts; abrupt, when this transition occurs suddenly; and recurring, when an old concept may reoccur after some time (Agrahari & Singh, 2022).

Concept drift can also be categorized in terms of the type of change that is occurring in the data (Agrahari & Singh, 2022). Real concept drift occurs when there is a change in the function generating the instances. Virtual concept drifts occur when there is a change in the data distribution but not necessarily in the classification function. More than one type of change may also be occurring at the same time.

There are three main groups of drift adaptation methods (Lu et al., 2019). The first group includes adapting batch classifiers to deal with data streams and concept drifts. This is usually done by changing its internal structure. The second group is to use a concept drift detection method to identify when a change occurs and reset or create a new classifier. The third and last group is to use an ensemble of classifiers.

Concerning the first group, several batch learners have been adapted to deal with data streams and concept drifts and have freely available implementations. One example is the decision tree classifier (Agu et al., 2015; Jin et al., 2009). It has been adapted to work with data streams (Domingos & Hulten, 2000) and concept drifts (Bifet & Gavaldà, 2009; Bifet et al., 2009; Hulten et al., 2001). Another example is the decision rule classifier (Gani et al., 2004; Liu et al., 2000). It was adapted to work with data streams (Gama & Kosina, 2011) and concept drifts (Kosina & Gama, 2012).

Perceptrons (Pavlidis et al., 2011) were also adapted to deal with data streams. They take multiple input values, multiplies each input by a corresponding weight, sums the weighted inputs, and applies an activation function to produce an output. The activation function introduces non-linearity, allowing the perceptron to learn complex patterns. While single perceptrons are limited in their capabilities, they form the building blocks of more complex neural networks, enabling them to learn and make decisions on intricate data. For example, multilayer perceptrons are neural networks composed of multiple layers of interconnected perceptrons, enabling them to learn and represent more complex patterns.

Multilayer perceptrons (MLPs) (Awodele & Jegede, 2009; Kruse et al., 2022) are often designed for static data and struggle to adapt to these dynamic changes. As a result, their performance degrades, leading to inaccurate predictions and suboptimal decision-making. To address this issue, developing effective techniques for adapting MLPs to concept drift is crucial. By enabling MLPs to learn and evolve continuously, we can improve their ability to handle non-stationary data streams, leading to more robust and reliable models in a wide range of applications.

In this paper, some techniques are evaluated to increase the speed at which an MLP adapts to a stable concept and to a concept drift in streaming environments. The techniques used include a temporary increase of the learning rate at the beginning of a dataset and when a concept drift is identified and the use of an embedded concept drift detection method based on the quadratic error (squared difference between predicted and actual values) of the forward phase. Experiments were performed in artificial and real-world datasets with and without concept drifts. Several single classifiers were compared, and the results indicate that temporarily increasing the learning rate and detecting concept drifts based on the cumulative sum of the quadratic errors presents superior results compared to other single classifiers previously published.

The remainder of this paper is organized as follows: Background presents several classifiers built to deal with data streams and concept drift. The proposal describes techniques to improve an MLP that may be used to increase the learning speed in data streams with and without concept drift. System Implementation shows the hardware, software, setup, and parameter settings of the experiments. Evaluation Experiments present the datasets used in the experiments, the evaluation methodology, and the statistical tests used to compare the proposal and several classifiers. Data Analysis & Results presents tables and graphics with the results of the experiments. The Findings describe some of the major results of the current work. The Discussion compares the current work with others in the literature. Finally, the Conclusion presents the conclusions of this paper.

BACKGROUND

Batch classifiers work in a dataset with a fixed size. To learn, they iterate over the instances of the dataset several times until convergence, which usually consumes much time. After this training phase, the base learner is able to test incoming instances. In streaming environments, instances arrive online, so it is not possible to store all the instances for posterior analysis. Thus, a base learner must satisfy some requirements that are not existent for batch learners: (1) instances must be processed one at a time; (2) use a limited amount of memory; (3) work in a limited amount of time; and (4) be ready to classify arriving instances at any point. Thus, batch classifiers cannot be used directly in streaming environments, which means they must be adapted to work correctly in this kind of scenario.

Very Fast Decision Tree (VFDT) (Domingos & Hulten, 2000) is a proposal that adapts a decision tree to deal with data streams. It can deal with huge amounts of data using few computational resources, presenting similar performance to a batch decision tree, given enough instances. A decision tree is learned by recursively replacing leaves with test nodes, starting at the root. In order to identify the best attribute to test at a given node, only a small subset of the streaming data is considered. The number of instances to be considered at each node is identified by using a Hoeffding bound (Hoeffding, 1963), which ensures that with high probability, the same attribute would be chosen using infinite examples.

Concept-adapting Very Fast Decision Tree (CVFDT) (Hulten et al., 2001) is an extension of VFDT. It maintains its speed and accuracy but adds the ability to detect a concept drift and respond to that change. It uses a sliding window with the most recent instances. At every node, it tries to detect a change in data distribution. If positive, alternate decision subtrees are created and replaced with the old tree. Besides the window size, the user must set three other parameters. The first is the frequency of checking at all nodes if the splitting attribute is still the best; if not, an alternate subtree is created. The second is the number of instances used to create the alternate subtree. The last attribute indicates the number of instances to test the alternate subtree compared to the current one. If the alternate subtree is more accurate, the current one is substituted.

Hoeffding Adaptive Tree (HAT) improves over CVFDT by using Adaptive Windowing (ADWIN) (Bifet & Gavaldà, 2007, 2009) at each node to monitor the performance of branches on the tree. ADWIN is a change detector, and when it detects a drift, a new branch is created with newer instances. If the new branches are more accurate than the current ones they are replaced. Thus, there is no need to set a window size parameter because each node can decide which previous instances are relevant.

Another decision tree proposal is named Adaptive Size Hoeffding Tree (ASHT) (Bifet et al., 2009). The difference to VFDT is that there is a maximum number of split nodes (sizes); thus, after a node splits, nodes are deleted to reduce their size if the number of split nodes surpasses the maximum number. It considers that smaller trees adapt faster to changes while larger trees have better performance in stable concepts because they were built on more data. So, several trees are created with different sizes to deal with stable and drifting concepts, making it an ensemble classifier.

Very Fast Decision Rule (VFDR) (Gama & Kosina, 2011) is a proposal designed for high-speed data streams. It is able to deal with multiple classes, and categorical and numerical attributes. It stores information about each rule as the number of covered examples, the probability of observing examples of each class and per attribute, and the probability of observing values greater than a threshold of continuous attributes per class. Hoeffding bounds are used to compute the minimum number of instances to be processed, after which a rule can be expanded, or a new rule can be induced. VFDR deals with concept drift implicitly by inferring new rules and specializing in existing ones.

Adaptive Very Fast Decision Rules (AVFDR) (Kosina & Gama, 2012) is an extension of VFDR in the context of time-changing data. It can learn ordered and unordered rule sets, adapting the decision model via incremental induction and specialization of rules. Different than VFDR, AVFDR contains explicit drift detection. Each individual rule monitors its performance metrics by the use of a change detector based on the Drift Detection Method (DDM) (Gama et al., 2004). If a drift is identified, the performance of the rule is considered to negatively influence the predictions of the classifier, being thus removed from the rule set. If a warning is raised, the rule stops learning until it returns to a no-drift stage.

Domeniconi and Gunopulos (2001) describe incremental learning techniques to make Support Vector Machines (SVMs) suitable for learning with data streams. It addresses the limitations of traditional batch-mode SVM training on large datasets. It introduces and compares various incremental learning techniques for constructing SVMs, demonstrating their ability to achieve performance similar to batch algorithms while significantly improving training time. The paper also explores the application of incremental schemes to stream data, aiming to maintain an updated representation of recent data batches. Experimental results confirm the effectiveness of incremental SVMs in handling large datasets and adapting to dynamic data streams.

Weighted Incremental–Decremental SVM (WIDSVM) (Gâlmeanu & Andonie, 2022) is an adaptation of an SVM to deal with data streams and concept drift using the weighted shifting window technique. The central proposal of the work is a method that uses weighted incremental-decremental SVMs, allowing the model to dynamically adjust to these changes. The method employs a shifting window that adjusts its size based on the relevance of the most recent data, ensuring that the model remains up-to-date and effective. This approach is particularly useful in scenarios where data is continuously generated, and the ability to quickly adapt is crucial for maintaining predictive accuracy. Experiments demonstrated the effectiveness of the proposed method compared to traditional techniques, highlighting its responsiveness to both abrupt and gradual changes in the data.

MLP is one of the most used classification methods to learn from data. Several adaptations have been proposed to deal with streaming data. Ghazikhani et al. (2013b, 2014) propose an online neural network model for non-stationary and imbalanced data stream classification composed of two layers. An objective function composed of two parts is proposed. The first part is a forgetting function used to handle concept drift by weighting recent instances to change from the old to the new concept. The second part is a specific error function to deal with two-class imbalance, assigning different importance to errors in separate classes.

Another proposal (Martínez-Rego et al., 2011) describes a single-layer neural network model based on a forgetting function that gives a monotonically increasing importance to new data. It makes the network rapidly forget in the presence of changes while maintaining a stable behavior when there is a stationary context. Thus, it does not need to explicitly detect changes or other information like past batch data or old models.

Two different single-layer online classifiers based on Martínez-Rego et al. (2011) were proposed (Ghazikhani et al., 2013a) to add the ability to deal with imbalanced data in two class datasets. Two cost-sensitive strategies were proposed. The first one uses a fixed misclassification cost matrix. The second uses an adaptive one, which also may handle concept drift. Its values are adjusted based on the true positive and true negative rates at specific time steps.

Selective ensemble-based online adaptive deep neural networks (SEOA) (Guo et al., 2021) propose a novel approach to address concept drift in streaming data by leveraging deep neural networks. The method involves creating an ensemble of online adaptive deep neural networks, each trained on a specific window of data. A selective ensemble strategy is then employed to dynamically select the most suitable models from the ensemble to make predictions on incoming data. This approach effectively adapts to concept drift by continuously updating the ensemble members and their selection criteria, leading to improved performance and robustness in dynamic environments.

Bilevel Online Deep Learning (BODL) (Han et al., 2021) explores a novel approach to deep learning that addresses the challenges posed by non-stationary environments, where data distributions change over time. The authors propose a bilevel optimization framework that operates online, meaning it updates continuously as new data arrives. This framework consists of two levels: the upper level adjusts the model's hyperparameters to optimize performance, while the lower level focuses on updating the model parameters themselves. By doing so, the method effectively adapts to concept drift and maintains high predictive accuracy in dynamic settings. The paper includes experimental results that demonstrate the superiority of this bi-level approach over traditional methods, showcasing its ability to quickly and efficiently respond to evolving data patterns.

NADINE (Pratama et al., 2019) explores a method for dynamically building MLP networks in response to streaming data. This approach addresses the challenge of learning from data that arrives continuously, requiring the model to adapt without prior knowledge of the data's structure or volume. The proposed method incrementally constructs the MLP by adding neurons and layers as needed based on the complexity and characteristics of the incoming data. This allows the network to grow and adjust its architecture in real time, optimizing its performance as new information becomes available. The paper demonstrates the effectiveness of this adaptive strategy through experiments that show improved learning efficiency and accuracy compared to static network structures, making it particularly suitable for environments where data is constantly evolving by using a concept drift detection method.

CIDD-ADODNN (Priya & Uthra, 2023) manages the challenges of concept drift and class imbalance in streaming data environments by the use of a deep learning framework designed to adapt to the evolving nature of data distributions over time together with a drift detection algorithm. This framework incorporates mechanisms to detect and respond to concept drift, ensuring that the model remains accurate and relevant despite changes in the underlying data patterns. Additionally, it addresses class imbalance, a frequent problem where certain classes are underrepresented, by employing techniques that enhance the model's ability to make balanced decisions across all classes. Experimental results show the framework's effectiveness in maintaining high performance in dynamic and imbalanced data scenarios, showcasing its potential for complex decision-making tasks in real-world applications.

As described above, several approaches have been published, adapting the internal structure of batch data classifiers to handle data streams, such as decision trees, decision rules, support vector machines, and artificial neural networks. In addition, more studies have been published on adjusting these classifiers to handle concept changes, mainly by using an embedded concept drift detection method.

In this study, we use an embedded concept drift detection method, similar to NADINE and CIDD-ADODNN, to identify when concept drift occurs. When a drift is identified, we dynamically adjust the learning rate parameter to speed up the training of the MLP. SEOA and BODL also adjust the parameters of MLPs to handle concept drift.

PROPOSAL

In this paper, two techniques are investigated to analyze how they influence the performance of an MLP in a data stream environment with and without concept drift. Considering a stable concept, the first technique initially sets the learning rate to a higher value and decreases it until it reaches a stable value. Considering concept drifts, the second technique monitors the quadratic error obtained from the comparison of the actual result to the expected one obtained by an MLP to identify if a concept drift has occurred.

```
1: procedure PROPOSAL((s) Stream)
2:     d ← false
3:     for all i in s do
4:         d ← UpdateLearningRate(d)
5:         r ← Learn(i)
6:         c ← Class(i)
7:         e ← Error(r, c)
8:         l ← CUSUM(e)
9:         if l = true then
10:            d ← true
11:        end if
12:    end for
13: end procedure
```

Figure 1 presents the base algorithm where the techniques are used. Each arriving instance (line 3) is learned by an MLP (line 5). The quadratic error is computed (line 7) by comparing the output of the MLP and the desired output obtained from the current instance (line 6). The computed quadratic error is passed to a drift detection method (line 8), as described in the algorithm presented in Figure 2.

When a change is detected (line 9), the learning rate is set to a higher value and reduced over time (line 4). This gives more weight to arriving instances, making the MLP learn faster the new concept. This occurs for the next n instances, a parameterized value set by the user.

Figure 1. Proposed algorithm

The drift detection method used here is the cumulative sum of errors and is presented in Figure 2. The quadratic error returned by the MLP is the input value of the CUSUM function. The current average quadratic error is computed (line 3). Following, the cumulative sum of the quadratic errors is computed (line 4), with zero being the lowest possible value. To avoid raising a drift too early when the statistics are still being computed, drifts can only be raised after processing a minimum specified number of values (line 6). When the current sum is above a threshold, a drift is raised. If it occurs, the internal statistics are then reset to start a new monitoring. In summary, every time the MLP does not correctly classify an instance, the average error will increase, while if it correctly classifies an instance, the average error will decrease. If the sum of errors surpasses a specified threshold, a concept drift is raised.

```
1: $m_n$= 1, $\bar{e}$= sum = 0, δ = 0.005, λ = 20, min = 30
2: **function** CUSUM((v) Input value)
3:         $\bar{e} \leftarrow \bar{e} + (v - \bar{e})/m_n$
4:         sum ← MAX(0, sum + v − $\bar{e}$ − δ)
5:         $m_n \leftarrow m_n + 1$
6:         **if** $m_n \geq$ min **then**
7:                 **if** sum > λ **then**
8:                         **return** true
9:                 **end if**
10:        **end if**
11:        **return** false
12: **end function**
```

Figure 2. Drift detection method algorithm

Concerning the function used to decrease the learning rate, infinite options can be used, where *l* is the current learning rate, $l_s$the starting value,$l_e$the end value of the learning rate, *n* the number of instances to change from$l_s$tol_e, and *i* the i*th* instance. The first one is a simple linear decreasing function, as described in (1).

$$l = l_s - \frac{l_s - l_e}{n} \quad (1)$$

The second and third functions proposed here are a family of functions that allow to decrease from$l_s$to$l_e$based on a parameter *s,* which indicates the speed of change. They use a hyperbolic tangent to simulate slower changes, as described in (2), and faster changes, as described in (3). Higher values of *s* indicate a slower decrease in the learning rate in (2) and a faster decrease in (3).

$$l = l_e + tanh\left((n - i) \times \left(\frac{s}{n}\right)\right) \times (l_s - l_e) \quad (2)$$

$$l = l_s - tanh\left(i \times \left(\frac{s}{n}\right)\right) \times (l_s - l_e) \quad (3)$$

A graph with an example of these functions, decreasing from 1.2 to 0.5 in 100 instances, is described in Figure 3, with different options for the *s* value.

The procedure that updates the learning rate, called from line 4 in Figure 1, is described in Figure 4. Initially, a vector with the learning rates is created based on the upper bound and lower bound of the learning rate, the speed, and how many instances need to be processed to change from the upper to the lower learning rate (line 2). If a change is detected (line 4), a new value of the learning rate is obtained to be used by the MLP (line 9). When the maximum number of instances to change from the upper to the lower bound is achieved (line 5), no more updates in the learning rate are required.

After several experiments, the parameter values of the proposed method were set as learning rate as 0.7, momentum as 0.3, sigmoid as activation function, and learning rate upper bound set as 1.5 when a drift is identified. The number of perceptrons included in the initial layer is equal to the number of attributes, and in the last layer, the number is equal to the number of classes. Concerning the decreasing function, the slower one (with *s* = 4) presented better results during 1500 instances.

Preprocessing the attributes is fundamental for MLP. Unlike batch data, where preprocessing can be done on the entire dataset before using the classifier, in an online environment, preprocessing must be executed as data arrives. Two tasks can be performed: how to treat missing values and how to process categorical and numerical values. Concerning data streams, the approach here used substitutes missing values as follows: categorical attributes by the most frequent value (mode) and numerical attributes by its mean. Besides this, nominal to binary operations are also used in categorical attributes, while numerical attributes are normalized to the [0,1] interval.

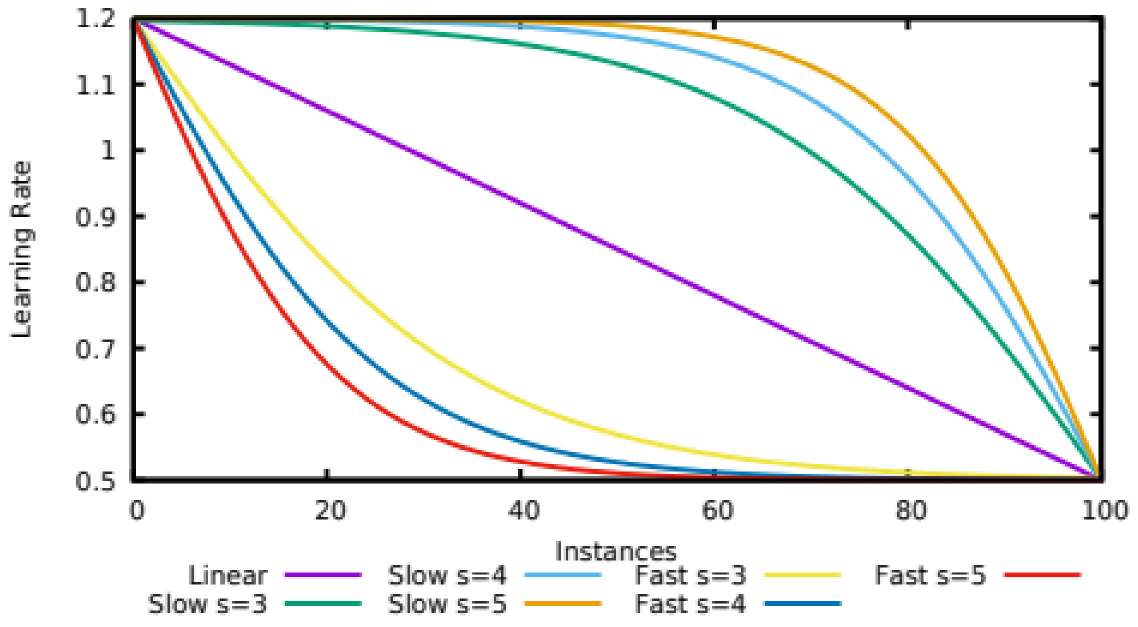


Figure 3. Functions used to decrease the learning rate

```
1: $m_n$= 0, size = 1500, $l_u$ = 1.5, $l_l$ = 0.7, s = 4
2: CreateLearningRates(size, lu, ll, s)
3: **function** UPDATELEARNINGRATE((d) Input value)
4:     **if** d = true **then**
5:         **if** mn = size **then**
6:             mn = −1
7:             d ← false
8:         **else**
9:             SetLearningRate(mn)
10:        **end if**
11:        mn ← mn + 1
12:    **end if**
13:    **return** d
14: **end function**
```

Figure 4. Update learning rate algorithm

System Implementation

The experiments were performed on a Windows 11 Intel Core i7-8565U with 16GB of main memory, using Visual Studio Code (https://code.visualstudio.com/) as the programming tool. The Massive Online Analysis (MOA) (https://moa.cms.waikato.ac.nz/) framework was used to execute the tests using the Java Development Kit (JDK) version 21. It is a free and open-source framework containing evaluation methodologies, classifiers, graphs, and datasets out of the box to use.

The proposed approach – along with the two datasets – was implemented and integrated in the MOA framework. This allowed the use of all functionalities of the framework, speeding the execution of the experiments and data gathering. Freely available implementations of the current proposal can be obtained at https://sites.google.com/site/moaextensions/ as a MOA extension.

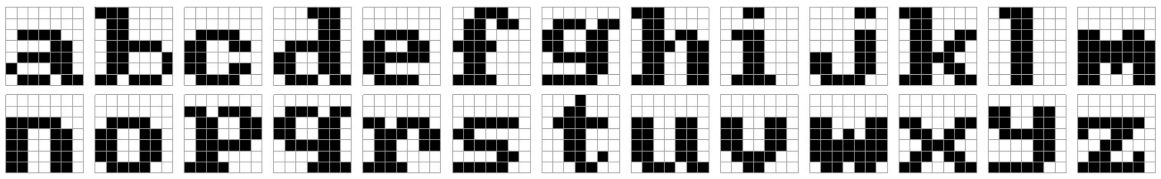
Evaluation Experiments

Six artificial datasets were used in the experiments and are described below. Each dataset uses ARFF (Attribute-Relation File Format). It offers a structured way to represent datasets, making them easily interpretable by various machine learning algorithms. While similar to CSV in that it stores data in a tabular format, ARFF provides additional features that make it more suitable for machine-learning tasks. ARFF files have a distinct header section that explicitly defines the attributes (features) of the data, including their names and data types (e.g., numeric, nominal, string). Moreover, ARFF supports the designation of a class attribute, which is crucial for supervised learning tasks as it identifies the target variable that the model aims to predict.

The LED data set (Bifet et al., 2009; Breiman et al., 1984) is composed of 24 categorical attributes, where 17 are irrelevant, and one categorical class with ten possible values. It represents the problem of predicting the digit shown by a seven-segment LED display, where each attribute has a 10% probability of being inverted (noise).

The Letters dataset, as shown in Figure 5, represents lowercase and uppercase letters in a7×7grid, where a value of +1 indicates a black dot and -1 is a white dot. Two versions were used concerning how noise is included: (a) no noise and (b) reversing the value of each attribute with a 10% probability.

The Mixed dataset (Baena-García et al., 2006; Gama et al., 2004) is composed of two Boolean (*v* and *w*) and two numerical (*x* and *y*) attributes. Three conditions are compared using these attributes: *v*, *w*, and$y < 0.5 + 0.3 \times sin(3\pi x)$. If at least two of these conditions are satisfied, the example is set to class positive. This is a noise-free data set.


(a) Lowercase letters

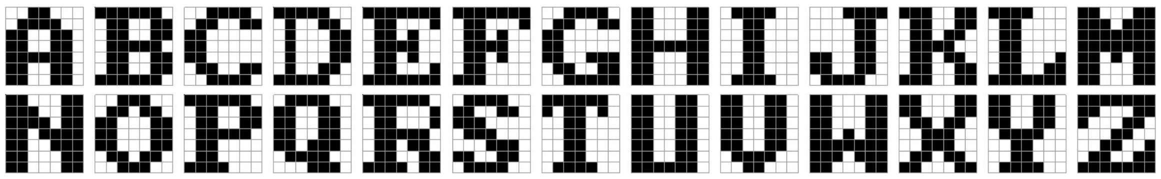

(b) Uppercase letters

Figure 5. Letters dataset

The Operators dataset is composed of four Boolean attributes, with the last two being irrelevant and two classes. Three versions were created, applying the Boolean operators `AND`, `OR`, and `XOR` to the first two attributes. This is a noise-free data set.

Radial Basis Function (RBF) (Bifet et al., 2009) creates complex concept drifts that are not straight-forward to approximate with a decision tree model. It works as follows by creating a fixed number of random centroids where each center has a random position, a single standard deviation, a class label, and a weight. New examples are generated by selecting a centroid and taking weights into consideration so that centers with higher weights are more likely to be chosen. Every centroid is moved towards a random direction by changing the attribute values from the central point by a specified offset. The length of the displacement is randomly drawn from a Gaussian distribution with a standard deviation determined by the chosen centroid. This effectively creates a normally distributed hypersphere of examples surrounding each central point with varying densities. This dataset is solely composed of numeric attributes. In the experiments, the number of centroids was set to 20.

The Sine dataset (Baena-García et al., 2006; Gama et al., 2004; Ross et al., 2012) is composed of two numerical attributes representing the coordinates of a point and two irrelevant attributes. All attributes are values uniformly distributed in the [0,1] interval. Points above a specified curve are classified as positive. After each concept drift, the classification is reversed. Two different curves are used:

Sine1,$y = sin(x)$; Sine2,$y < 0.5 + 0.3sin(3\pi x)$

Initially, experiments were performed in the datasets without concept drifts. Then, experiments were performed in the same datasets but with concept drifts included. In the LED dataset, drifts were included by changing the position of four and seven attributes (similar results occur with different numbers of drifting attributes). In the Letter dataset, a concept drift is created by changing from lowercase to uppercase letters. In the Mixed dataset, when a change occurs, the classification is reversed. The Operator dataset has three versions: changing from `AND` to `OR`, changing from `AND` to `XOR`, and changing from `OR` to `XOR`. In the Random RBF dataset, changes are created by changing the position of 5, 10, 15, and 20 centroids by a 1.5 speed. Finally, in the Sine dataset, similarly to the Mixed dataset, drifts are created by reversing the classification. In these datasets, the concept of drifts was introduced in the middle of the dataset.

For each artificial dataset, 30 different versions were created, changing the seed for random generation of instances (parameter *i* in the MOA framework) from 1 to 30. Each version is composed of 10,000 instances.

The evaluation methodology used was the Interleaved Test-Then-Train. Every arriving instance is initially used to test the classifier. In this phase, the evaluator verifies if the classifier correctly classified the incoming instance, keeping track of the percentage of correct classifications at every moment (for every incoming instance). Then, the instance is used for training. This methodology allows the full use of the instances for both training and testing, avoiding over-fitting.

After all instances were tested and trained, the percentage of correct classifications over the whole dataset was obtained, and the average results over the 30 datasets were then computed. This process is repeated for all tested classifiers. After the average results were obtained for each classifier in each dataset, it was time to apply a statistical test.

Besides the percentage of correct classifications, four other metrics were used in some experiments in datasets with concept drifts. The first one is the distance to the drift point. The smaller this value, the closer the classifier identified the correct change point. By pinpointing the exact moment of concept drift, classifiers can quickly adapt to evolving data distributions, ensuring that models remain relevant and effective in dynamic environments. Also, accurate change point detection prevents the propagation of errors arising from outdated models, thereby enhancing the overall reliability and trustworthiness of classification results.

The second and third metrics are the true positive (sensitivity) and true negative (specificity) rates. As it is almost impossible for a classifier to identify a drift in the exact instance that it occurs, we consider a correct detection if the detection is made up to 50 instances after the correct change point. They provide insights into the classifier's ability to correctly identify if a concept drift occurred or not. They both range from 0 to 1. High sensitivity values indicate that the classifier correctly identifies a concept drift when one occurs. High specificity values indicate that the classifier does not identify a concept drift when one did not occur. Similarly to Ghazikhani et al. (2014), we computed the geometric mean (GM) metric for evaluating the algorithms. This metric offers a:

$$GM = \sqrt{sensitivity \times specificity} \quad (4)$$

The Friedman non-parametric statistical test was chosen, as described in Demšar (2006), to identify if there is a statistical difference among the classifiers over multiple datasets. The result of this test only indicates if there are statistical differences in performance among the alternatives. If positive, the Holm post-hoc test is then performed to compare the proposal to the other classifiers and identify if it is statistically different, i.e., it yielded better results. In both tests, the significance level was set to 0.05.

DATA ANALYSIS & RESULTS

Initially, experiments are performed to analyze how the temporary increase in the learning rate influences the results in datasets without concept drift. Thus, three MLPs are compared: two ordinary MLPs, one with the learning rate set to the lower bound (MLP_l= 0.7) and another with the upper bound (MLP_u= 1.5), and an MLP with a decreasing learning rate (from 1.5 to 0.7,MLP_d). Table 1 presents the results of the comparison, including the mean value and the mean rank.

Table 1. Accuracies of MLP versions with different learning rates in datasets without concept drift

Datasets	MLP_l	MLP_u	MLP_d
led	66.285	67.915	68.710
letter-lower-a	86.125	85.195	85.010
letter-lower-b	95.435	94.875	95.710
letter-upper-a	82.275	80.120	83.235
letter-upper-b	93.870	92.310	93.690
mixed	89.700	89.620	90.005
op-and	99.370	99.620	99.665
op-or	99.290	99.575	99.565
op-xor	91.580	95.690	95.050
rbf	89.780	90.660	90.500
sine1	94.400	94.180	95.195
sine2	81.260	83.095	81.990
Mean	**89.11**	**89.40**	**89.86**
Rank	**2.33**	**2.08**	**1.58**

The results indicate that slowly decreasing the learning rate from an upper to a lower-level results in better mean and rank. As an example of the results in Table 1, Figure 6 presents the accuracy curve in the LED dataset. We can notice that the decreasing learning technique starts to have better accuracy, around 2,000 instances, and stays above the version with a higher learning rate.

Now, theMLP_dclassifier is compared with other single classifiers suited to deal with streaming data without concept drift. Not many published proposals have publicly available implementations to facilitate comparisons. Thus, the experiments used classifiers that have freely available implementations in the MOA framework. The first classifier is a single perceptron classifier performing multi-class learning. The second is VFDT, and the third is VFDR. Table 2 presents a comparison of the accuracies of these single classifiers in datasets without concept drift.

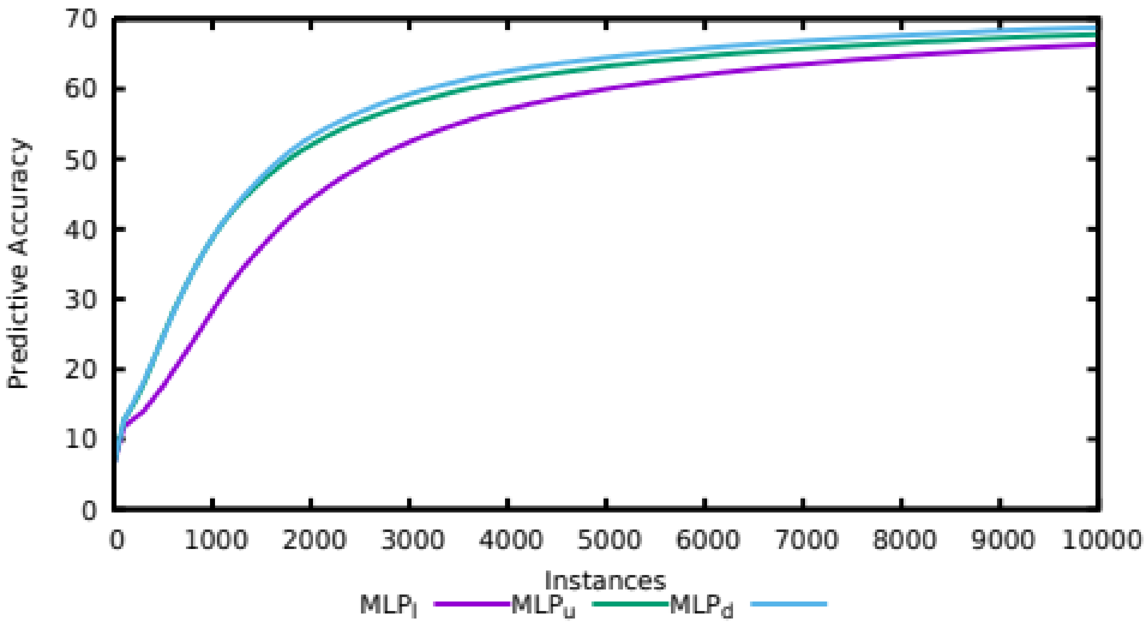


Figure 6. Accuracy in the LED dataset

Table 2. Accuracies of single classifiers in datasets without concept drift

Datasets	MLP_d	Perceptron	VFDT	VFDR
led	68.820	71.670	73.860	10.030
letter-lower-a	86.320	80.550	85.085	01.890
letter-lower-b	95.675	89.700	92.725	03.780
letter-upper-a	83.180	75.985	85.490	01.910
letter-upper-b	93.745	87.130	92.705	03.820
mixed	89.925	90.545	92.290	74.680
op-and	99.660	99.830	99.920	92.060
op-or	99.555	99.750	99.930	91.555
op-xor	95.335	51.930	82.765	65.360
rbf	90.370	88.110	90.475	78.395
sine1	95.215	96.600	96.600	86.920
sine2	81.745	81.645	88.315	84.600
Mean	**89.96**	**84.45**	**90.01**	**49.58**
Rank	**2.17**	**2.71**	**1.38**	**3.75**

Table 2 shows that VFDT has higher mean accuracies and better rank but is closely followed by MLP. A Friedman test was conducted to determine whether single classifiers had differential rank-ordered accuracies for the tested datasets. The results of that analysis indicated that there was a differential rank-ordered preference for the four single classifiers ($\chi^2 = 21.48$, $p < 0.05$). A post hoc comparison of the rank-ordered preferences for the four single classifiers was conducted using the Holm procedure. Results of this analysis indicated that there were significantly more favorable rankings of VFDT over the other classifiers and MLP_dover VFDR ($p < 0.05$). There was, however, no significant difference in how the datasets were classified by $MLP_d$and Perceptron.

Now that we have found thatMLP_dhave a competitive accuracy when compared to other classifiers in streaming datasets without concept drift, it is time to analyze the influence of using a drift detection method to handle concept drifts. The previous versions are again used, and two more alternatives are tested with a drift detection method included. One resets the MLP when a drift is identified (MLP_r) and another sets the learning rate to an upper level and decreases it on time (MLP_s). Thus, both versions use the decreasing learning rate approach, but the first one starts with a new MLP, while the second one uses the current MLP. Table 3 presents the accuracy results of the experiments.

Table 3. Accuracies of MLP versions in datasets with concept drift

Datasets	MLP_l	MLP_u	MLP_d	MLP_r	MLP_s
led4	61.730	59.785	62.370	65.005	63.930
led7	49.915	45.315	49.165	65.125	53.505
letter-a	92.210	88.845	90.400	94.740	90.780
letter-b	79.690	71.800	77.305	89.390	75.715
mixed	87.005	88.330	87.785	89.140	88.445
op-and-or	99.220	99.540	99.510	99.090	99.500
op-and-xor	89.205	93.515	87.335	95.010	92.205
op-or-xor	87.340	91.005	87.715	94.565	91.925
rbf5	86.145	86.990	86.905	85.735	86.895
rbf10	82.940	83.630	83.835	83.030	83.720
rbf15	82.130	82.465	82.580	82.820	82.765
rbf20	82.335	82.715	82.845	82.885	82.845
sine1	69.670	72.510	72.700	93.480	73.305
sine2	77.340	76.860	76.030	80.000	77.480
Mean	**80.49**	**80.24**	**80.46**	**85.72**	**81.64**
Rank	**4.00**	**3.50**	**3.25**	**1.79**	**2.46**

The results indicate that using the drift detection method improves the predictive accuracy in almost all tested datasets, presenting higher mean values and lower ranks. Figure 7 presents how the tested versions deal with concept drift in the LED dataset. It is possible to notice that after the concept drift, the versions with a drift detector recover faster than the other versions. Considering the versions with the concept drift detection method, the version that resets the MLP presents better results.

A Friedman test was conducted to determine if the MLP versions with and without support to handle concept drifts had a differential rank-ordered preference for the tested datasets. The results of that analysis indicated that there was a differential rank-ordered preference for the five versions of MLPs ($\chi^2 = 17.21$, $p < 0.05$). A post hoc comparison of the rank-ordered accuracies for the five versions of tested MLPs was conducted using the Holm procedure. The results of this analysis indicated that there were significantly more favorable rankings ofMLP_roverMLP_l,MLP_u, andMLP_d($p < 0.05$). There was, however, no significant difference in the ranking results ofMLP_randMLP_s.

Following, $MLP_r$is compared with single classifiers suited to deal with concept drifts in data streams. The following classifiers were used: ASHT, HAT, and AVFDR. Table 4 presents the results of the classifiers in datasets with concept drifts. MLP_rhad better average accuracies and rank values. In the

LED dataset, when the amount of drift increases, MLP and AVFDR remain stable while the accuracy of decision trees is substantially reduced. A Friedman test was conducted to determine whether the four classifiers had a differential rank-ordered preference for the tested datasets. The results of that analysis indicated that there was a differential rank-ordered preference for the four tested classifiers ($\chi^2 = 25.11$, $p < 0.05$). A post-hoc comparison of the rank-ordered accuracies for the four classifiers was conducted using the Holm procedure test. The results of this analysis indicated that there were significantly more favorable rankings ofMLP_rover AVFDR ($p < 0.05$). There was, however, no significant difference in the ranking results of ASHT and HAT.

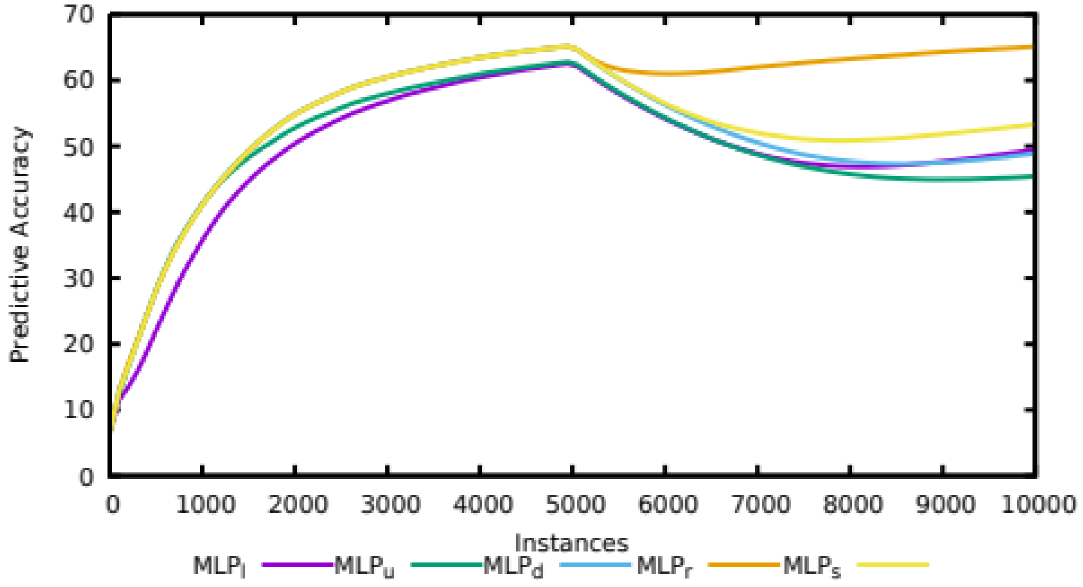


Figure 7. Accuracy in the LED dataset with concept drift

Table 4. Accuracies of single classifiers in datasets with concept drift

Datasets	MLP_r	ASHT	HAT	AVFDR
led4	65.005	56.560	56.570	10.455
led7	65.125	48.165	48.285	10.380
letter-a	94.740	99.120	99.060	05.705
letter-b	89.390	83.655	83.115	04.110
mixed	89.140	80.145	88.315	65.015
op-and-or	99.090	94.960	98.585	91.025
op-and-xor	95.010	90.950	97.455	77.855
op-or-xor	94.565	98.925	99.605	85.275
rbf5	85.735	86.970	86.885	77.845
rbf10	83.030	83.805	83.870	77.725
rbf15	82.820	81.970	82.620	77.755
rbf20	82.885	81.935	82.810	77.770
sine1	93.480	59.060	92.165	67.750
sine2	80.000	59.905	85.815	69.055
Mean	**85.72**	**79.01**	**84.65**	**56.98**
Rank	**1.71**	**2.64**	**1.79**	**3.86**

None of the previous single classifiers has an explicit global drift detection method. For example, AVFDR has a drift detection method attached to each rule and removes it when its performance is too low. HAT is similar, with a drift detection method attached to each node of the tree. Thus, a similar drift detection method, as used byMLP_r,is plugged into these classifiers to compute the metrics just described. The results are shown in Table 5.

Table 5. Drift detection of classifiers in datasets with concept drift

	Distance				**Sensitivity**			
Datasets	MLP_r	ASHT	HAT	AVFDR	MLP_r	ASHT	HAT	AVFDR
led4	32.00	35.50	36.00	-	0.967	0.900	0.900	0.000
led7	26.50	28.50	29.00	-	1.000	1.000	1.000	0.000
letter-a	25.00	22.00	22.00	-	1.000	1.000	1.000	0.000
letter-b	25.50	36.00	34.50	-	1.000	0.967	0.967	0.000
mixed	12.50	21.50	22.00	51.00	1.000	0.967	0.967	0.433
op-and-or	22.00	41.00	41.00	51.50	1.000	0.933	0.867	0.433
op-and-xor	13.50	26.00	26.00	36.50	1.000	1.000	1.000	0.667
op-or-xor	42.00	86.50	91.00	177.50	0.800	0.000	0.000	0.167
rbf5	210.00	273.00	248.00	2200.00	0.000	0.033	0.000	0.000
rbf10	105.00	138.50	126.50	1890.50	0.000	0.033	0.100	0.000
rbf15	73.50	98.50	111.00	1739.00	0.000	0.067	0.133	0.000
rbf20	66.00	83.50	85.50	1864.50	0.067	0.133	0.133	0.000
sine1	10.50	21.00	21.00	26.00	1.000	1.000	1.000	1.000
sine2	20.50	23.00	23.00	27.00	1.000	1.000	1.000	1.000
	Specificity				**Geometric mean**			
Datasets	MLP_r	ASHT	HAT	AVFDR	MLP_r	ASHT	HAT	AVFDR
led4	1.000	1.000	1.000	1.000	0.983	0.949	0.949	0.000
led7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000
letter-a	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000
letter-b	1.000	1.000	1.000	1.000	1.000	0.983	0.983	0.000
mixed	1.000	1.000	1.000	1.000	1.000	0.983	0.983	0.658
op-and-or	1.000	1.000	1.000	1.000	1.000	0.966	0.931	0.658
op-and-xor	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.816
op-or-xor	1.000	1.000	1.000	1.000	0.894	0.000	0.000	0.408
rbf5	1.000	1.000	1.000	1.000	0.000	0.183	0.000	0.000
rbf10	1.000	1.000	1.000	1.000	0.000	0.183	0.316	0.000
rbf15	1.000	1.000	1.000	1.000	0.000	0.258	0.365	0.000
rbf20	1.000	1.000	1.000	1.000	0.258	0.365	0.365	0.000
sine1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
sine2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Analyzing the distance to the drift point, results clearly show thatMLP_ridentifies a concept drift closer to the true change point. The only dataset that did not present the best result was the Letter dataset, which did not include any noise and detected a drift three instances later than the decision trees tested. In the LED and Letter datasets, the AVFDR classifier did not identify any changes. The Friedman test indicated differences in performance, and the Holm post-test informed that MLP was statistically superior to all the other classifiers.

Considering the sensitivity metric, MLP_rpresents similar or better results compared to the other classifiers, except in the RBF versions up to changes in 15 centroids. This result is related to the distance to the drift point. As the classifiers, on average, identify the drift point close to the change point, the sensitivity scores are usually high. The exceptions are the RBF dataset, where the distance to the drift point is higher than 50, and the AVFDR in the LED and Letter datasets because it does not identify changes. Again, the Friedman test indicated differences in performance, and the Holm post-test informed that MLP was statistically superior to AVFDR and similar to the decision tree classifiers.

Considering the specificity metric, as the number of false positives is small when compared to the size of the datasets, all the values are very close to 1.0, showing 1.0 in the table due to rounding. In this metric, the Friedman test indicated that all classifiers have similar performances.

The last metric computed, the geometric mean, summarizes how the methods deal with concept drifts. MLP_r presents better results in all datasets, except for the RBF dataset versions up to changes in 15 centroids. This is associated with the value set to discriminate between true and false positives (50). For example, if the distance were increased to 75, MLP_r would present similar or better results to all other tested classifiers. Here, the Friedman test indicated differences in performance, and the Holm post-test informed that MLP was statistically superior over AVFDR and similar to the decision tree classifiers.

Finally, Table 6 presents the results of running the classifiers on the collection of real-world datasets obtained from the UCI Machine Learning Repository (https://archive.ics.uci.edu) and the MOA website (http://moa.cms.waikato.ac.nz/datasets/).

Table 6. Accuracies of classifiers in real-world datasets

Datasets	MLP_r	ASHT	HAT	AVFDR
covertype	85.99	80.31	81.89	61.07
poker hand	73.79	76.07	66.87	62.54
balance-scale	82.88	76.48	77.60	45.92
car	86.57	82.00	82.76	70.02
credit-a	92.75	80.15	80.87	55.22
credit-g	99.20	99.00	98.90	99.20
heart-c	57.76	52.48	55.45	54.13
ionosphere	80.63	77.21	81.20	50.14
kr-vs-kp	99.12	96.15	93.55	64.55
mfeat-morph	94.80	73.25	73.60	10.00
nursery	93.74	89.82	90.44	81.41
optdigits	89.59	88.13	88.31	18.10
page-blocks	91.38	92.67	92.27	89.77
tic-tac-toe	98.75	73.07	73.49	81.63
vote	91.95	91.49	89.89	60.69
yeast	44.41	46.70	46.63	30.66
Mean	**85.21**	**79.69**	**79.61**	**58.44**
Rank	**1.41**	**2.63**	**2.31**	**3.66**

MLP_r presented the best mean and rank values. Similarly to the results in the datasets with concept drifts, the Friedman test indicated differences in performance among the classifiers. The Holm post hoc test showed that the proposal is statistically superior to all the other tested classifiers.

FINDINGS

As presented in the Data Analysis and Results section, starting MLP training with a higher learning rate and gradually reducing it yielded better results than keeping the learning rate constant over time on datasets without concept drift, regardless of whether the rate initially set was higher or lower.

Another finding was that restarting the MLP after identifying the occurrence of a concept change, rather than maintaining it and trying to adjust its internal state, allows for faster adaptation to the concept change. We also found that if the goal is to identify a concept drift as close as possible to the drift point, MLP is an approach that performed very well compared to other classifiers tested, with better results in 13 out of 14 datasets.

Overall, MLP presented competitive results with other classifiers that were adapted to deal with data streams with and without concept drifts.

DISCUSSION

The main goal of this study was to analyze how to improve the predictive accuracy of an MLP in streaming environments with and without concept drifts. The focus was on MLP because several classifiers developed for batch data had already been adapted to deal with data streams. There are several parameters that can affect the execution of an MLP: the number and size of hidden layers, learning rate, momentum, and activation function, among others.

Concerning streaming data without concept drift, we focused on adapting the learning rate. At the beginning of the training process, a higher learning rate was set, which gradually decreased over time. This produced higher mean values and lower ranks compared to keeping the learning rate constant, even though statistically, the results were similar.

At this point, we obtained other classifiers that were adapted from batch learners and that have open-source implementations to test how MLP would behave. Here, MLP obtained statistical results similar to or superior to those of the other classifiers tested.

We now analyze how the MLP would behave in a streaming environment with concept drifts. We implemented an MLP with a built-in concept drift detection method and two variations: one resetting the classifier when a change occurs and another reusing the classifier but adjusting the learning rate. The experiments indicated that the MLP with the embedded drift detection method and resetting it when a drift is identified obtained the higher mean and lower ranks, with similar or superior statistical results to the other classifiers tested.

This improved MLP version was compared against three other single classifiers, all able to deal with streaming data with concept drifts and broadly used in this research field (Mi et al., 2023; Yin et al., 2022). Statistical results indicated that this improved MLP version obtained the same or better results than the other classifiers, with higher mean and lower rank.

Four other metrics were used to allow a broader evaluation of the classifiers. MLP had similar results in the specificity metric and similar or better results in almost all datasets in the distance to the drift point, sensitivity, and geometric mean.

Finally, a last comparison was performed in sixteen real-world datasets where MLP yielded higher mean accuracy, lower rank value, and statistically superior results than all other classifiers.

Resetting the classifier when a concept drift is identified to increase its accuracy was already discussed in various previous papers (Chen et al., 2023; Maciel et al., 2015; Wang et al., 2013) and confirmed in this study. The same is true for changing the inner structure of an MLP (Han et al., 2021; Pratama et al., 2019), where the number of neurons or the number of layers is changed to adapt to concept drifts. A deeper analysis of how each characteristic influences the learning process would give more insights into how to adapt them together to improve the results.

CONCLUSION AND FUTURE WORK

Dealing with concept drift is an important challenge for supervised classifiers. Several single classifiers have been adapted to handle changes in context, especially decision trees and decision rules.

This paper proposed some techniques that can improve the performance of an MLP. One is to set the learning rate to an upper level and slowly reduce it over time. Another is to compute the quadratic error of arriving instances and use it to compute if a concept drift occurred. These techniques help us understand how changing the learning rate can influence the adaptation of an MLP to a new concept and also confirm that the use of a drift detection method can improve the performance of an MLP in an environment with changing concepts, as described in other research in the machine learning and data stream processing areas. Experiments were performed in six artificial datasets with and without concept drifts as well as 16 real-world datasets.

Results comparing an ordinary MLP to one with a decreasing learning rate presented higher accuracies and lower ranks to the latter. Proposals that reset the MLP after a concept drift or use a decreasing learning rate were also compared, with better results than the former one.

This version was compared with other single classifiers suited to deal with streaming data and concept drifts. Results indicate that setting the learning rate to a higher value and gradually decreasing it at the beginning of the dataset and after the detection of concept drifts presents competitive results, with similar or superior statistical results concerning predictive accuracy, distance to the drift point, sensitivity, specificity scores, and the geometric mean.

The results of this study demonstrate the effectiveness of setting the learning rate to a higher value and gradually decreasing it to improve the accuracy of an MLP. Future work includes adapting the learning rate online by increasing/decreasing it based on the performance of the MLP. This scheme would allow the removal of the parameters that must be set by the user (learning rate upper bound, number of instances to return to the stable value) in the present proposal. Another possibility would be to analyze the impact of dynamically adjusting the momentum parameter, in addition to including and/or removing hidden layers from the neural network when a concept drift occurs.

REFERENCES

Abad, M. A., Gomes, J. B., & Menasalvas, E. (2014, July). Recurring concept detection for spam filtering. Proceedings of the *International Conference on Information Fusion, Salamanca, Spain,* 1-7. https://ieeexplore.ieee.org/abstract/document/6916145

Agrahari, S., & Singh, A. K. (2022). Concept drift detection in data stream mining: A literature review. *Journal of King Saud University-Computer and Information Sciences*, *34*(10), 9523-9540. https://doi.org/10.1016/j.jksuci.2021.11.006

Agu, M. N., Nabareseh, S., & Osakwe, C. N. (2015). Investigating web based marketing in the context of micro and small-scale enterprises (MSEs): A decision tree classification technique. *Proceedings of Informing Science & IT Education Conference, Tampa, Florida*, 13-28. https://doi.org/10.28945/2201

Awodele, O., & Jegede, O. (2009). Neural networks and its application in engineering. *Proceedings of Informing Science & IT Education Conference*, *Macon, USA,* 83-95. https://doi.org/10.28945/3317

Baena-García, M., Del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. *Proceedings of the International Workshop on Knowledge Discovery from Data Streams*, 77–86.

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. *Proceedings of the Seventh SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA*, 443-448. https://doi.org/10.1137/1.9781611972771.42

Bifet, A., & Gavaldà, R. (2009). Adaptive learning from evolving data streams. In N. M. Adams, C. Robardet, A. Siebes, & J. F. Boulicaut (Eds.), *Advances in intelligent data analysis* (pp. 249-260). Springer. https://doi.org/10.1007/978-3-642-03915-7_22

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 139–148). Association for Computing Machinery. https://doi.org/10.1145/1557019.1557041

Branch, J. W., Giannella, C., Szymanski, B., Wolff, R., & Kargupta, H. (2013). In-network outlier detection in wireless sensor networks. *Knowledge and Information Systems*, *34*(1), 23–54. https://doi.org/10.1007/s10115-011-0474-5

Breiman, L., Friedman, J. H., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Taylor & Francis.

Chen, Z., Han, M., Wu, H., Li, M., & Zhang, X. (2023). A multi-level weighted concept drift detection method. *The Journal of Supercomputing*, *79*, 5154–5180. https://doi.org/10.1007/s11227-022-04864-y

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30. http://dl.acm.org/citation.cfm?id=1248547.1248548

Diallo, O., Rodrigues, J. J., & Sene, M. (2012). Real-time data management on wireless sensor networks: A survey. *Journal of Network and Computer Applications*, *35*(3), 1013–1021. https://doi.org/10.1016/j.jnca.2011.12.006

Domeniconi, C., & Gunopulos, D. (2001, November). Incremental support vector machine construction. *Proceedings IEEE International Conference on Data Mining, San Jose, CA, USA,* 589-592. https://doi.org/10.1109/ICDM.2001.989572

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 71-80). Association for Computing Machinery. https://doi.org/10.1145/347090.347107

Gâlmeanu, H., & Andonie, R. (2022). Weighted incremental-decremental Support Vector Machines for concept drift with shifting window. *Neural Networks*, *152*, 528-541. https://doi.org/10.1016/j.neunet.2022.05.018

Gama, J., & Kosina, P. (2011). Learning decision rules from data streams. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 1255-1260. https://www.ijcai.org/Proceedings/11/Papers/213.pdf

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In A. L. C. Bazzan, & S. Labidi (Eds.), *Advances in artificial intelligence – SBIA 2004* (pp. 286-295). Springer. https://doi.org/10.1007/978-3-540-28645-5_29

Gani, A., Zakaria, O., & Jumaat, N. B. A. (2004). A Markov Decision Process model for traffic prioritisation provisioning. *Issues in Informing Science and Information Technology*, *4*, 905-913. https://doi.org/10.28945/2750

Ghazikhani, A., Monsefi, R., & Sadoghi Yazdi, H. (2013a). Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams. *Neural Computing and Applications*, *23*, 1283-1295. https://doi.org/10.1007/s00521-012-1071-6

Ghazikhani, A., Monsefi, R., & Sadoghi Yazdi, H. (2013b). Ensemble of online neural networks for non-stationary and imbalanced data streams. *Neurocomputing*, *122*, 535–544. https://doi.org/10.1016/j.neucom.2013.05.003

Ghazikhani, A., Monsefi, R., & Sadoghi Yazdi, H. (2014). Online neural network model for non-stationary and imbalanced data stream classification. *International Journal of Machine Learning and Cybernetics*, *5*, 51-62. https://doi.org/10.1007/s13042-013-0180-6

Gonçalves, V. P. M., Silva, L. P., Nunes, F. L. S., Ferreira, J. E., & Araújo, L. V. (2024). Concept drift adaptation in video surveillance: A systematic review. *Multimedia Tools Application, 83*, 9997-10037. https://doi.org/10.1007/s11042-023-15855-3

Guo, H., Zhang, S., & Wang, W. (2021). Selective ensemble-based online adaptive deep neural networks for streaming data with concept drift. *Neural Networks*, *142*, 437-456. https://doi.org/10.1016/j.neunet.2021.06.027

Han, Y.-N., Liu, J.-W., Xiao, B.-B., Wang, X.-T., & Luo, X.-L. (2021). Bilevel online deep learning in non-stationary environment. In I. Farkaš, P. Masulli, D. Otte, & S. Wermter (Eds.), *Artificial neural networks and machine learning* (pp. 14-17). Springer. https://doi.org/10.1007/978-3-030-86340-1_28

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13-30. https://doi.org/10.1080/01621459.1963.10500830

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 97-106). Association for Computing Machinery. https://doi.org/10.1145/502512.502529

Jin, C., De-lin, L., & Fen-xiang, M. (2009, July). An improved ID3 decision tree algorithm. *Proceedings of the 4th International Conference on Computer Science & Education, Hanning, China,* 127-130. https://doi.org/10.1109/ICCSE.2009.5228509

Kosina, P., & Gama, J. (2012). Handling time changing data with adaptive very fast decision rules. In P. A. Flach, T. De Bie, & N. Cristianini (Eds.), *Machine learning and knowledge discovery in databases* (pp. 827-842). Springer. https://doi.org/10.1007/978-3-642-33460-3_58

Kruse, R., Mostaghim, S., Borgelt, C., Braune, C., & Steinbrecher, M. (2022). Multilayer perceptrons. In R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, & M. Steinbrecher (Eds.), *Computational intelligence: A methodological introduction* (3rd ed., pp. 53-124). Springer. https://doi.org/10.1007/978-3-030-42227-1_5

Liu, B., Ma, Y., & Wong, C. K. (2000). Improving an association rule based classifier. In D. A. Zighed, J. Komorowski, & J. Żytkow (Eds.), *Principles of data mining and knowledge discovery* (pp. 504-509). Springer. https://doi.org/10.1007/3-540-45372-5_58

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, *31*(12), 2346-2363. https://doi.org/10.1109/TKDE.2018.2876857

Maciel, B. I. F., Santos, S. G. T. C., & Barros, R. S. M. (2015, November). A lightweight concept drift detection ensemble. *Proceedings of the IEEE 27th International Conference on Tools with Artificial Intelligence, Vietri sul Mare, Italy*, 1061-1068. https://doi.org/10.1109/ICTAI.2015.151

Martínez-Rego, D., Pérez-Sánchez, B., Fontenla-Romero, O., & Alonso-Betanzos, A. (2011). A robust incremental learning method for non-stationary environments. *Neurocomputing*, *74*(11), 1800–1808. https://doi.org/10.1016/j.neucom.2010.06.037

Mi, Y., Wang, Z., Liu, H., Qu, Y., Yu, G., & Shi, Y. (2023). Divide and conquer: A granular concept-cognitive computing system for dynamic classification decision making. *European Journal of Operational Research*, *308*(1), 255-273. https://doi.org/10.1016/j.ejor.2022.12.018

Pavlidis, N. G., Tasoulis, D. K., Adams, N. M., & Hand, D. J. (2011). λ-Perceptron: An adaptive classifier for data streams. *Pattern Recognition*, *44*(1), 78-96. https://doi.org/10.1016/j.patcog.2010.07.026

Pozzolo, A. D., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., & Bontempi, G. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, *41*(10), 4915-4928. https://doi.org/10.1016/j.eswa.2014.02.026

Pratama, M., Za'In, C., Ashfahani, A., Ong, Y. S., & Ding, W. (2019). Automatic construction of multilayer perceptron network from streaming examples. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (pp. 1171–1180). Association for Computing Machinery. https://doi.org/10.1145/3357384.3357946

Priya, S., & Uthra, R. A. (2023). Deep learning framework for handling concept drift and class imbalanced complex decision-making on streaming data. *Complex & Intelligent Systems, 9*, 3499-3515. https://doi.org/10.1007/s40747-021-00456-0

Ross, G. J., Adams, N. M., Tasoulis, D. K., & Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, *33*(2), 191-198. https://doi.org/10.1016/j.patrec.2011.08.019

Wang, S., Minku, L. L., Ghezzi, D., Caltabiano, D., Tino, P., & Yao, X. (2013, August). Concept drift detection for online class imbalance learning, *Proceedings of the International Joint Conference on Neural Networks*, *Dallas, TX, USA,* 1-10. https://doi.org/10.1109/IJCNN.2013.6706768

Yin, J., Tang, M., Cao, J., Wang, H., You, M., & Lin, Y. (2022). Vulnerability exploitation time prediction: An integrated framework for dynamic imbalanced learning. *World Wide Web*, *25*, 401-423. https://doi.org/10.1007/s11280-021-00909-z

AUTHORS

Paulo M. Gonçalves Jr. received B.Sc. (2002), M.Sc. (2005), and Ph.D. (2013) degrees in Computer Science from the Universidade Federal de Pernambuco, Brazil, and a Postdoctoral from the University of Ottawa, Ottawa, ON, Canada, in 2017. He is currently a Professor at the Instituto Federal de Pernambuco. He is the author or coauthor of over 20 journal and conference papers in machine learning. His research interests include machine learning, data streams, and concept drift.

Sylvain Chartier received his B.A. degree from the University of Ottawa, Ottawa, ON, Canada, in 1993 and his B.Sc. and Ph.D. degrees from the Université du Québec à Montréal, Montréal, QC, Canada, in 1996 and 2004, respectively, all in Psychology. He has been a Professor at the University of Ottawa since 2007 and is currently the Director of the Laboratory for Computational Neurodynamics and Cognition (CONEC). He is the author or coauthor of over 100 journal and conference papers on neural networks and quantitative methods. His research interests are in developing unsupervised and supervised recurrent neural associative memories.