An Error-Based Measure for Concept Drift Detection and Characterization

Antoine Bugnicourt[1,2][0000−0002−5035−1904], Riad Mokadem[1], Franck Morvan[1], and Nadia Bebeshina[2]

[1] Institut de Recherche en Informatique de Toulouse (IRIT)
[2] MeetDeal https://meetdeal.fr/

Abstract. Continual learning is an increasingly studied field, aiming at regulating catastrophic forgetting for online machine learning tasks. In this article, we propose a prediction error measure for continual learning, to detect concept drift induced from learned data input before the learning step. In addition, we check this measure's ability for characterization of the drift. For these purposes, we propose an algorithm to compute the proposed measure on a data stream while also estimating concept drift. Then, we calculate the correlation coefficients between this estimate and our measurement, using time series analysis. To validate our proposal, we base our experiments on simulated streams of metadata collected from an industrial dataset corresponding to real conversation data. The results show that the proposed measure constitutes a reliable criterion for concept drift detection. They also show that a characterization of the drift relative to components of the stream is possible thanks to the proposed measure.

Keywords: Online learning · Continual learning · Concept drift · Change detection

1 Introduction

In the last years, successive advancements in machine learning allowed its use in multiple fields. The most common approach is batch learning, with multiple learning steps over a fixed dataset. Some tasks may require to learn from a continuous stream of data. Among those, there are applications requiring *plasticity*, i.e. the learning model being able to take into account concept drift: changes in the distribution of data [1,11]. For batch learning, this creates an over-cost on both storage (the dataset must be updated with every new input) and computation time (the model must be re-trained regularly on the dataset) [17].

In this context, online learning comes as an alternative to batch learning; the model learns iteratively on a data stream, without the need to store data. This new approach allows for change detection and integration in real time. However, the learning model will tend to favoritize newly learned knowledge, and to forget

past knowledge. In the literature, this phenomenon is named catastrophic forgetting [20,24]. Catastrophic forgetting infringes on model *stability*, the model's ability to retain knowledge.

To counter catastrophic forgetting, a wide array of works consider a kind of online learning named continual learning [26]. It aims to improve online learning models to be more resistant to outliers, and to provide them with a better stability of acquired knowledge.

As an example, let's suppose we want to analyze user interactions on a business website, to build a question answering (QA) system. This site undergoes regular changes to catalogue items, their availability, prices, etc. Changes can also occur in users' interactions on the website. Every one of these changes must be integrated into the QA process as quickly as possible, in order to always produce relevant answers for the users. A continual learning approach trained on a stream of interactions on the website would allow the model to acquire new information while staying accurate for regular questions. Let's then consider that one key product undergoes a sale, preparing for the release of a new version of the same product. The sale needs to be taken into account in the QA system, but we also wish to notice when the sale ends, in order to "forget" its effect on the system, i.e. to come back to the state of the model prior to the sale integration.

In the literature, few works have considered solutions where a model could need to come back to a previous state of knowledge. Continual learning as a field consists of various methods to counter catastrophic forgetting, which can be classified in two main categories derived from Biesialska et al. [6]: **model-based methods**, that use dynamic adaptations over learning parameters [12,15,30] or the model structure [16,19] to consolidate knowledge and avoid forgetting; and **memory-based methods** [2,14,25,27] that use an external memory and a rehearsal or replay mechanism to have the model remember previously learned knowledge.
Model-based methods are mostly used in multi-task systems. In those systems, returning back to a previous state doesn't make sense, as the goal is to manage an ever-increasing set of skills. On the other hand, memory-based methods, used in this work, have a distinct advantage: using the rehearsal mechanism specific to these methods and the notion of catastrophic forgetting, it is indeed possible to have the model come back to a previous state.

Considering we induce forgetting through the aforementioned rehearsal, one remaining issue is determining when to trigger that rehearsal mechanism; we need to preserve the balance between stability and plasticity. For this reason, we wish to be able to detect concept drift and to decide on a sequence of actions based on some characteristic traits of that drift. To do so, a characterization of the current *context* (i.e. the set of all concepts and the associated probabilities

of occurrence) available at all times is required. The decision process itself is two-fold:

1. Detecting concept drift: determining whether the new input integrated to the model creates a drift.

2. Determining the direction of the trend shift: a model under concept drift could be either evolving towards a new model (leading to an update of the sampled replay memory to memorize the older state of knowledge) or devolving towards an older state (leading to the use of the sampled replay memory for induced forgetting over the previous registered changes).

In this paper, we propose a new measure of the concept drift, which is used to determine the significance of the change in the data, and as such, to perform concept drift detection. Furthermore, this measure pave the way towards characterization of the drift. The measure is based on the predictive errors by the model. To measure this detection process, we use statistical measures (e.g. distance between labels) that allow to notice a significant correlation between forecast error and model shift. In order to validate the detection of concept drift, an algorithm is proposed to compute this correlation.
The next step would be to explore the characterization potential. This would then allow us to decide which approach to follow: classical continual learning if the model is evolving or returning back if the model is devolving. We defer this last step to a future work.

To validate our proposal, we use a proprietary dataset based on metadata from online chats provided by a corporation. Our experiments show that using the proposed algorithm, we compute correlation values that allow us to consider a characterization of concept drift. In summary, the main contribution of this paper is a correlation-based measure allowing both concept drift detection and characterization decision.

The paper follows as described: in section 2, we establish a state of the art for continual learning approaches, specifically on memory-based methods, and for concept drift detection. In section 3, we propose an approach allowing concept drift detection and the criteria to consider for concept drift characterization using this approach. In section 4, we present our experimental results. Finally, we conclude this paper with a summary of our work and some future considerations.

2 Related work

The core idea of catastrophic forgetting is a loss of knowledge stability in a machine learning model due to integrating a degree of plasticity towards new inputs. This phenomenon has long been an issue of concern [15,20], and a significant goal for both incremental and continual learning is to counter it [21].

Continual learning encompasses a broad range of approaches [6], all aiming to control catastrophic forgetting and to preserve the balance between stability and plasticity. We have compiled a classification of these approaches.

Model-based methods

Various approaches deal with adaptation to concept drift by enacting structural change of the machine learning model. Those model-based methods can be split into two main categories:

1. Regularization methods use weight manipulation and forgetting factors to influence the learning process and preserve stability. As examples: Kirkpatrick et al [15] propose an algorithm called *Elastic Weight Consolidation* (EWC) operating on neural networks, singling out essential neurons for memorization of specific knowledge and slowing the learning on those neurons. Gupta et al., on their STAFF tool [12] propose a generally weaker forgetting thanks to a stabilization coefficient. Finally, Yu and Webb [30] link their forgetting factor's value to the analysis of concept drift on the data stream over a period of time.

2. Architectural methods consist of a set of dynamic adaptation processes on the neuronal architecture (i.e. addition of layers or parameters) in order to integrate new behaviors. Examples include the work of Li et al. [16], in which a model learns to adapt weight values or to create new neurons in parallel with the learning process; or Masana et al. [19], who store in an external structure various masks and normalization parameters related to specific tasks.

These methods are particularly suitable for multitask applications, but are otherwise sub-optimal for repetitive tasks over long periods of time. They also don't include any flexibility over the plasticity mechanics: the applied transformations to the model are not revertible by default.

Memory-based methods

Memory-based methods use incremental learning plasticity itself to prevent catastrophic forgetting. These methods set up a replay or rehearsal of various informations (learning data) in the model input, mixed with the input stream, to create stability. There are multiple ways to proceed:

1. Rehearsal or replay methods, rely on maintaining a separate dataset of examples parallel to the learning process, that is re-integrated to the learning process on a regular basis. Works to mention include iCarl from Rebuffi et al. [25]. Their method, used for multi-class classification, relies on task-specific examples dataset, updated with each new class. Aljundi et al. [2] define a "Maximal Interference" criterion, applied post-learning over model

predictions, in order to identify data entries most susceptible to suffer from catastrophic forgetting, and to use them as a replay dataset.

2. Pseudo-rehearsal methods don't use an entry dataset, but store generalizations built from rehearsal data entries. Biesialska et al. mention two examples: DCR [27] and FearNet [14], using generative adversarial networks (GAN) and autoencoders respectively, to make this generalization.

For all of these methods, knowledge stability is consolidated by a memory module distinct from the model. This kind of approach is more suited to the notion of bringing back a previous context, since rehearsal allows the recreate a previous state of the model. The idea of reusing past concepts already exists in other works related to multi-task learning, such as Wang et al.'s SDR architecture [29]. SDR relies on checking for similarities in new data entries compared to previously learned tasks, in order to determine if a new entry is an instance of one of these tasks, or instead illustrates a new task to be integrated to the model.

Some works on concept drift [1,11] define two categories of drift, depending on the considered source of change:

- Virtual concept drift is a measurable change on distribution of the values of specific data fields;
- Real concept drift is a change in the relation between features and labels.

Sometimes a third kind of drift is considered: *label* or *prior-probability shift*, related to a change in the distribution of the prediction labels.

There are various existing tests to detect concept drift based on model performance. A number of them are mentioned in the Bayram et al. review of concept drift methods [5]:

- A first set of detectors called "Statistical Process Control" detectors, watch the evolution of the error rate of the learner as an indicator of concept drift. Drift Detection Method (DDM) [10] tracks the rate of prediction errors over the data stream, compared to specific thresholds for warning (announcing a potential drift) and drift itself, following the hypothesis that a rising rate means a drift is either coming or happening. Early Drift Detection Method (EDDM) EDDM [4] uses the same principle, but looks at the distance between consecutive errors rather than their rate, which improves detection in gradual drift.
- Another set of detectors compare statistical measures over sliding windows to detect change. Those include ADWIN (short for Adaptive Windows) [7], that tracks the change in the distribution of a variable in two sliding windows of varying sizes, dynamically adapting the size to optimize the detection.

Raab et al. propose another detector called KSWIN [23] (KS being short for Kolmogorov-Smirnov), using the Kolmogorov-Smirnov test [18] over two sliding windows to check for concept drift.

The base measure we propose in this article has elements from both categories. It is error-based, as it relies on prediction error for new data inputs — although we're not specifically interested in the rate of those errors, but rather on their amplitude. Furthermore, the drift detection decision is based on aggregates of those errors in sliding windows.

Our general approach differs from previously mentioned works by the attention given to the chronology of consecutive concepts, which requires an accurate characterization of those concepts and even more so, of the concept drift. We not only want to detect concept drift, but also to be able measure its amplitude and *direction* — in the sense of whether the drift is towards a new concept, or a previously occurring one. This measure must come before the learning step (like in Wang et al.'s SDR), in order to use it in further work for decision over the kind of rehearsal to enact.

In summary, we propose a new measure for concept drift detection and characterization in new learning data inputs, based on prediction error, and available before the learning step.

3 Proposal

Our process involves analyzing data from a single data stream: we first aim to detect a variation in this stream regarding the relation between a vector their corresponding labels, and for a second step, we will consider if characterization of this relation is possible. The variation of the relation between vector and label is measurable through the variation of the predictions performed by a learning model over a test dataset of labeled vectors.

In this paper, let's consider that a continual learning model M is set up over two distinct data streams: $\mathcal{S}_{predict}$, made of data vectors X for which we want to make a forecast; and $\mathcal{S}_{learn}$, made of labeled vectors (X, y) to be integrated to the model through learning.
In the example from section 1, the prediction stream $\mathcal{S}_{predict}$ is made on questions asked by the users while browsing on the website. The learning stream $\mathcal{S}_{learn}$ contains questions with relevant answers given by an outside source — for example a human expert. The model then learns on the questions/answers couples from $\mathcal{S}_{predict}$ so that it may improve answers given for the questions in $\mathcal{S}_{learn}$.

Our goal here is to confirm the existence of a correlation between the changes induced on the model by the learning of data entries from $\mathcal{S}_{learn}$, and the prediction error measured on those same data entries before learning them. When this

is confirmed, a significant change in the model — i.e. a concept drift — might be anticipated using the prediction made by the model itself. We describe in the following segments some tools and a method aiming to measure this correlation between predictions and model change.

3.1 General notations

To represent the learning stream over which our proposal operates, we consider a discrete time period $T \subseteq \mathbb{N}$. The labeled vectors of data (X_t, y_t) of this stream are indexed by time periods $t \in T$. It is then necessary to explicitly define the essential operations of our continual learning model M:

The operation of learning a labeled vector (X, y) to update a model M into a new model M' has the following notation:

$$M' = learn(M, X, y) \tag{1}$$

The operation of predicting a label $\hat{y}$ for a vector X, with a trust value $p \in [0, 1]$ associated to the prediction, has the following notation:

$$(\hat{y}, p) = predict(M, X) \tag{2}$$

To give a formal notation for the prediction error on a measure, we need a comparison criterion — a *distance* measure between predicted and actual labels. The distance between two labels y, y' is noted as follows:

$$d = dist(y, y') = \|y - y'\| \in \mathbb{R} \tag{3}$$

The meaning of the calculation for a norm $\|y - y'\|$ varies depending of the use case, since it represents a relation between each pair of labels. If those labels are already represented by numerical values in $\mathbb{R}$, we can simply use $|y - y'|$. In our example, the distance between two answers may simply be a binary value (good or bad answer), with the following:

$$dist(y, y') = \begin{cases} 0 \text{ if } y = y' \\ 1 \text{ otherwise} \end{cases} \tag{4}$$

If more than a pair of categories of answers are considered, the notion of gap can become more complex, and as such, may justify to use label values from $\mathbb{R}$, or even vectors with values in $\mathbb{R}$.

The formalism for time series is required to invoke the measures computed in our algorithm. Time series are sequences of values with a time index. They are useful to study the interdependency of these values through time. **Cross correlation** is a measure of this dependency; it compares two series shifted by a lag τ [9,22]. The cross correlation between variables A and B on a discrete

period of time T (with values in $\mathbb{C}$) is a function of τ defined as:

$$(B \star A)(\tau) = \sum_{t \in T} A(t+\tau)\overline{B(t)} \tag{5}$$

where $\overline{B(t)}$ is the complex conjugate of $B(t) : \forall z \in \mathbb{C}, z = a+ib \implies \overline{z} = a-ib$. We finally define a specific notation for the cross correlation measured at $\tau = 0$ (a significant value for our proposal, considering the relation we want to measure would happen without lag): $corr\,(A, B) = (B \star A)(0)$.

3.2 Algorithm

The proposed algorithm aims to determine the relation between the model prediction over a given data vector, and the evolution of the model induced by the integrated labeled vector. This algorithm 1 applied to a model M, operates on a stream of labeled vectors $(X_t, y_t)_{t \in T}$.

For each time period t of T, we retrieve the corresponding labeled vector (X_t, y_t), and first measure the prediction error over this vector (lines 3 and 4 of the algorithm). This error is defined as the distance between the prediction 2 and the true label. This error is weighted by the trust value given to the prediction, also obtained through the prediction operation. The error function for predictions from current model M_t is:

$$err(X, y, t) = p \cdot dist(y, \hat{y}) \text{ where } (\hat{y}, p) = predict(M_t, X) \tag{6}$$

This error value is then used in a computation over error values from previous data entries (in a sliding window). We call $Err(S, W)$ the linear combination of the error values in the sliding window $S = (e_t)_{t \in [0;n]}$, weighted by $W = (w_t)_{t \in [0;n]}$ such that:

$$Err(S, W) = \sum_{w \in W, e \in S} w \cdot e \tag{7}$$

The linear combination value is saved in an array $\widehat{D}$. Next step (line 6) is M_t learning the labeled vector, and following equation 1, returning a new model $M_{t+1} = learn(M_t, X_t, y_t)$. Both models M_t and M_{t+1} are compared to measure the model shift induced by learning the new entry.
To compare the models, we can use equation 6 to make a variation function, defined for a pair of models $M = M_t$ and $M' = M_{t+k}$, and for a test dataset $\langle \mathcal{X}, Y \rangle = (X'_i, y'_i)_{i \in [1,N]}$, as:

$$\Delta_M^{M'}(\mathcal{X}, Y) = \sum_{i=1}^{N} \Big(err(X_i, y_i, t+k) - err(X_i, y_i, t) \Big) \tag{8}$$

The variation value $\Delta_M^{M'}(\mathcal{X}, Y)$ is stored in an array D.

When the last value of T is reached (end of the stream), cross correlation values are computed for any value of lag $\tau \in [-N \dots N]$. The remaining lines of the algorithm (from line 9 to the end) correspond to the correlation checking. We consider the value $corr(\widehat{D}, D)$, i.e. the cross correlation value for lag 0 (cf equation 5) for values in arrays $\widehat{D}$ and D (line 9). If the value of $corr(\widehat{D}, D)$ measured through the algorithm is not 0, values stored in $\widehat{D}$ can be used — to a certain extent — to predict shift in value of D. This would account for the measure's ability to detect change, but would not immediately solve the characterization issue, which will require further work.

Algorithm 1: Concept drift estimation and computation of correlation coefficients

Inputs : time period $T \subseteq \mathbb{N}$, classification model M, test dataset $\langle \mathcal{X}, Y \rangle$
Data : labeled vectors $(X_t, y_t)_{t \in T}$
Output : correlation coefficient C

1 initialization of $\widehat{D}[\,]$ and $D[\,]$
2 **For all** $t \in T$ **do**
3 $\quad X, y \leftarrow X_t, y_t$
4 $\quad S \leftarrow S.update\big(err(X, y, t)\big)$ `/* measure of prediction error */`
5 $\quad \widehat{D}[t] \leftarrow Err(t, S, W)$ `/* computation with values from` S `*/`
6 $\quad M' \leftarrow learn(M, X, y)$
7 $\quad D[t] \leftarrow \Delta_M^{M'}(\mathcal{X}, Y)$ `/* measure of model shift over` (X, y) `*/`
8 $\quad M \leftarrow M'$
9 $C \leftarrow corr\left(\widehat{D}, D\right)$ `/* cross correlation with lag` $\tau = 0$ `*/`
10 **If** $C \neq 0$ **then**
11 $\quad$ **return** C `/* correlation is identified */`
`/* (possible characterization) */`
12 **else**
13 $\quad$ **return** $\perp$ `/* no correlation revealed */`

4 Evaluation

4.1 Protocol

To evaluate our proposal, we use a dataset from a proprietary source. The dataset is made of various metadata collected over a volume of real online conversations provided by a corporation, each associated through the result of the conversation, to a positive or negative label. Five categories of metadata are considered for just over 10,000 conversations, with all or a subset of these categories being used

at once. Online learning classification is implemented using the River library[3]; the model used is Hoeffding tree classifiers ([8,13]), as implemented in the same library. The classification model is used with default parameters.

Through our experiments, we first want to check if the prediction error measure can be used for model shift detection. To show this, we apply our algorithm 1 on a simulated stream of data, for the full set of features available, and measure the cross correlation around lag 0. Streaming simulation is also performed through the River library.

In order to qualify the relevancy of our prediction measure err_M as a concept drift detection tool, we apply our algorithm 1 over the KSWIN test [23] in parallel to the measure, with the same window size for both. KSWIN can only detect concept drift over a single feature stream at once. To compensate for that fact, we use an aggregate (specifically a logical disjunction or OR operation) of detection values from all possible streams: one for each feature, and one for the label. Cross correlation is also measured between this aggregate and the model shift. We first perform the experiment on a window of size 100, which is the default size for KSWIN. We then perform multiple runs with various window sizes to find an optimal size for cross correlation for our dataset.

We then consider the issue of characterization of the measured model shift. In order to use an already established detection measure as a characterization criterion, one way could be to look for a function mapping from the measure to the actual model shift. To evaluate this possibility, we compute the ratio between both values of the prediction measure and the model shift and look for an identifiable trend between the two quantities.

[3] https://riverml.xyz/0.14.0/

4.2 Results

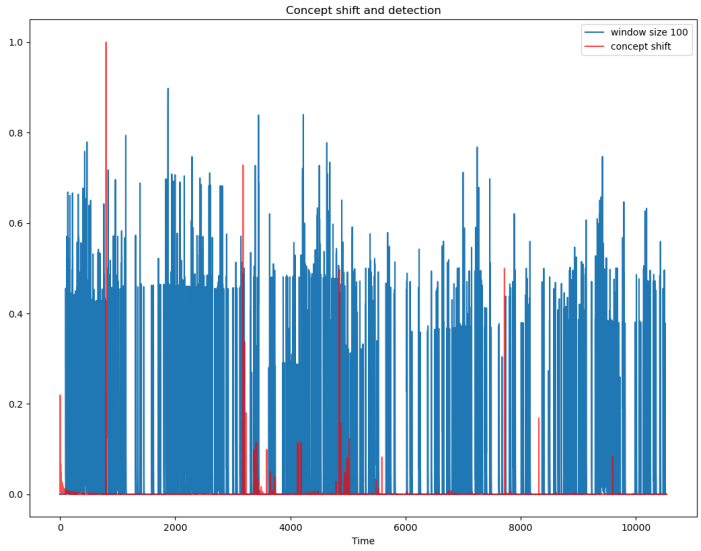


Fig. 1. Computed real concept drift (in red) against detected shift using the error-based measure, with a window size of 100 (in blue)

Concept drift detection During the first experiment, we measured values for the measure, identified the vectors for which KSWIN was detecting a model shift, and generated two figures: figure 1 shows compared values of concept drift computed on the model, and of the normalized measure, over time; figure 2 shows cross correlation curves for the measure and KSWIN, both in relation to concept drift.

Even though figure 1 presents a very noisy curve for the measure, figure 2 reveals that cross correlation for our method offers a similar curve shape to KSWIN's. Furthermore, when looking at the cross correlation values at lag 0 and under (i.e. when past values of the criteria align with future values of concept drift, giving a *forecast* of sorts), our method outperforms KSWIN.

By comparing the measure and KSWIN, we can say that the former seems efficient as a concept drift detection mechanism, or even as a concept drift predictor. However, the aggregate of KSWIN detection values we compared the measure to isn't a real concept drift detector, but merely an improved virtual drift detector, and the performance seen here is relative to various experiment criteria, among which the aggregation method itself.

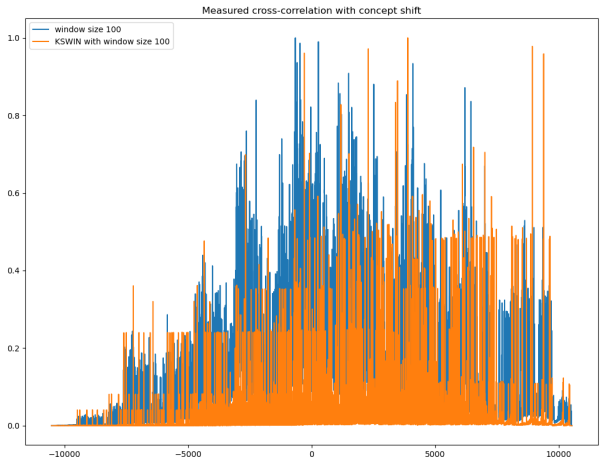
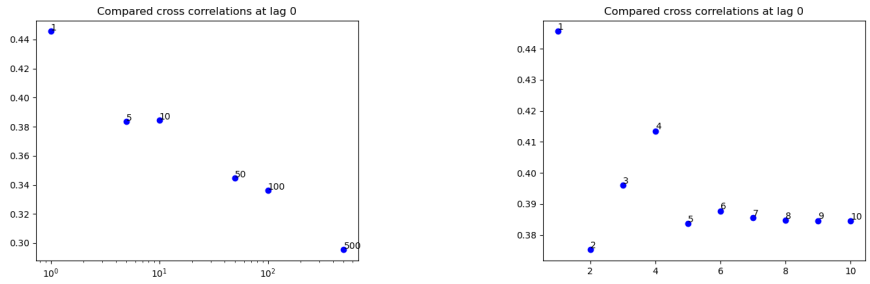


Fig. 2. Cross-correlation with computed model concept drift from the error-based measure err_M *(in blue)* and the KSWIN aggregate *(in orange)*

Setting the measure parameters We started with a low grain setting, in order to select a range of window sizes optimizing correlation at lag 0, shown in figure 3a. This first setting shows that lower window sizes give better cross correlation values. We then restrict our setting to a finer grain, low sizes range (figure 3b). Again, the lower sizes, specifically between 1 and 4 (2 being an outlier, probably due to the weight computation method) perform better on cross correlation.
Comparing various window sizes allows us to conclude that a window of size 1 is the optimal choice for both this data source and this weighting of the window. This conclusion is not an absolute, and both the data and the weighting approach influence the optimal outcome.

Characterization A detection criterion can be used as a characterization of concept drift criterion if there is a function mapping criterion values to the amplitude of the concept drift. In order to check for that potential characterization, we measured various ratios between the measure values and computed drift. The following measures are made with a window of size 1, following the conclusion from subsection 4.2. We consider various subsets of features $\{A, B, C, D\}$ to generate figures 4 (features A, B and C), 5a (features A and B) and 5b (features B, C and D).

In the resulting figures, we can see that multiple *trends* appear in various intervals of the data stream, since in those intervals, the ratio follows linear trajectories. This can be seen in figures 5a (with one such interval visible between time indices 0 and 3000, and another, sparser one with a steeper slope, between



(a) (b)

Fig. 3. Cross-correlation values (Y-axis) at lag 0, depending on the window size (X-axis and indices of points): (a) in low grain on a logarithmic scale; (b) in finer grain, from 1 to 10.

3000 and 9000) and 5b (two noticeable intervals respectively between 0 and 5000, and between before 5000 and after 7000). Looking back at model shift values on figure 1, we notice that those trends happen over intervals of low model shift, or between high model shift periods. We also note that those trends don't necessarily happen in succession to each other, but can overlap (e.g. between 3000 and 5000 on figure 4).
Even though none of these trends can describe a single, constant relation between the measure and real concept drift, they each take part in a set of relations between both of them. The aforementioned overlap between trends suggest that this variety of relations is not only due to model shift, but also (and most importantly) to specific values taken by some of the features in those intervals.

Mutiple remarks can be made about characterization. The experiments we performed here show that the measure highlights a number of trends during the learning process, seemingly related to the values taken by specific features. This allows us to hypothesize that a more particular analysis of these feature values, further developed in the conclusion of this article, would allow an even more thorough characterization than expected, even for other works.

5 Conclusion

In this paper, we proposed a concept drift detection and characterization measure related to a continual learning model. Our measure uses model prediction to determine if and to what degree a new data entry creates a concept drift. Our experiments show that real concept drift detection is possible using the measure, and that various parameters allow an adaptation to multiple kinds of data or models.

The experiments also show a set of distinct trends that can be observed in the relation between the measure and real drift, thus opening new perspectives for

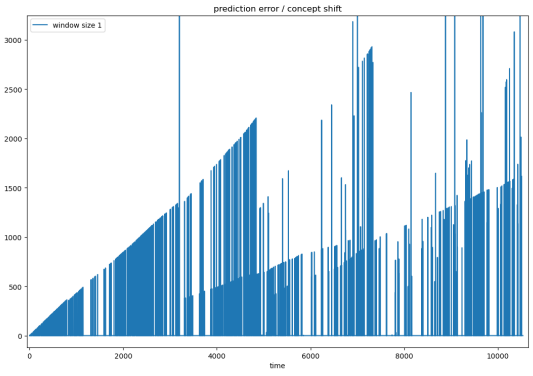
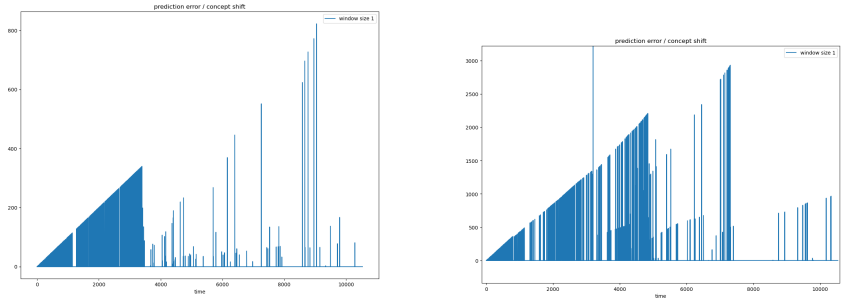


Fig. 4. Ratio of error-based measure over concept drift for a subset of features $\{A, B, C\}$



(a) (b)

Fig. 5. Ratio of error-based measure over concept drift for subsets of features: (a) $\{A, B\}$ and (b) $\{B, C, D\}$

characterization. No simple aggregate of those trends can be used as a general characterization of concept drift; however, those trends and their overlapping suggest that this behavior is related directly to distinct trends in the learning features themselves.

As a future work, explainability approaches, as studied in the Explainable AI (XAI) field of research [3,28], could be combined to our measure to produce a more accurate characterization. Furthermore, if we were to store the parameters related to each trend as representations of the state of the model, this would be a step towards building a continual learning system sensitive to concept temporality and recurrence. We could also take into account real use conditions of continual learning in terms of memory resource capacity and computation time. Finally, although the experiments were already lead on a real dataset, we project

to experiment over a real datastream instead of experimenting with a simulated stream.

References

1. Agrahari, S., Singh, A.K.: Concept Drift Detection in Data Stream Mining : A literature review. Journal of King Saud University - Computer and Information Sciences (Dec 2021). https://doi.org/10.1016/j.jksuci.2021.11.006
2. Aljundi, R., Caccia, L., Belilovsky, E., Caccia, M., Lin, M., Charlin, L., Tuytelaars, T.: Online Continual Learning with Maximally Interfered Retrieval. arXiv:1908.04742 [cs, stat] (Oct 2019)
3. Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI (Dec 2019)
4. Baena-Garcıa, M., Gavalda, R., Morales-Bueno, R.: Early Drift Detection Method. Fourth international workshop on knowledge discovery from data streams **6**, 77–86 (Sep 2006)
5. Bayram, F., Ahmed, B.S., Kassler, A.: From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors (Mar 2022). https://doi.org/10.48550/arXiv.2203.11070
6. Biesialska, M., Biesialska, K., Costa-jussà, M.R.: Continual Lifelong Learning in Natural Language Processing: A Survey. Proceedings of the 28th International Conference on Computational Linguistics pp. 6523–6541 (2020). https://doi.org/10.18653/v1/2020.coling-main.574
7. Bifet, A., Gavaldà, R.: Learning from Time-Changing Data with Adaptive Windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM), pp. 443–448. Proceedings, Society for Industrial and Applied Mathematics (Apr 2007). https://doi.org/10.1137/1.9781611972771.42
8. Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., Seidl, T.: MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. Proceedings of the first workshop on applications of pattern analysis pp. 44–50 (Sep 2010)
9. Bracewell, R.: Pentagram notation for cross correlation. the fourier transform and its applications. New York: McGraw-Hill **46**, 243 (1965)
10. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with Drift Detection. In: Bazzan, A.L.C., Labidi, S. (eds.) Advances in Artificial Intelligence – SBIA 2004. pp. 286–295. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28645-5_29
11. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys **46**(4), 44:1–44:37 (Mar 2014). https://doi.org/10.1145/2523813
12. Gupta, U., Babu, M., Ayoub, R., Kishinevsky, M., Paterna, F., Ogras, U.Y.: STAFF: Online learning with stabilized adaptive forgetting factor and feature selection algorithm. In: Proceedings of the 55th Annual Design Automation Conference. pp. 1–6. ACM, San Francisco California (Jun 2018). https://doi.org/10.1145/3195970.3196122
13. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge

Discovery and Data Mining. pp. 97–106. ACM, San Francisco California (Aug 2001). https://doi.org/10.1145/502512.502529
14. Kemker, R., Kanan, C.: FearNet: Brain-Inspired Model for Incremental Learning. In: International Conference on Learning Representations (Feb 2022)
15. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (Mar 2017). https://doi.org/10.1073/pnas.1611835114
16. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. International Conference in Machine Learning p. 10 (2019)
17. Lin, J.: The lambda and the kappa. IEEE Internet Computing **21**(5), 60–66 (2017)
18. Lopes, R.H.C.: Kolmogorov-Smirnov Test. In: Lovric, M. (ed.) International Encyclopedia of Statistical Science, pp. 718–720. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-04898-2_326
19. Masana, M., Tuytelaars, T., van de Weijer, J.: Ternary Feature Masks: Zero-forgetting for task-incremental learning. arXiv:2001.08714 [cs] (Apr 2021)
20. McCloskey, M., Cohen, N.J.: Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In: Psychology of Learning and Motivation, vol. 24, pp. 109–165. Elsevier (1989). https://doi.org/10.1016/S0079-7421(08)60536-8
21. Nguyen, C.V., Achille, A., Lam, M., Hassner, T., Mahadevan, V., Soatto, S.: Toward Understanding Catastrophic Forgetting in Continual Learning. arXiv:1908.01091 [cs, stat] (Aug 2019)
22. Papoulis, A.: The fourier integral and its applications. Polytechnic Institute of Brooklyn, McCraw-Hill Book Company Inc., USA, ISBN: 67-048447-3 (1962)
23. Raab, C., Heusinger, M., Schleif, F.M.: Reactive Soft Prototype Computing for Concept Drift Streams. Neurocomputing **416**, 340–351 (Nov 2020). https://doi.org/10.1016/j.neucom.2019.11.111
24. Ramasesh, V.V., Dyer, E., Raghu, M.: Anatomy of Catastrophic Forgetting: Hidden Representations and Task Semantics. arXiv:2007.07400 [cs, stat] (Jul 2020)
25. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: Incremental Classifier and Representation Learning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5533–5542. IEEE, Honolulu, HI (Jul 2017). https://doi.org/10.1109/CVPR.2017.587
26. Ring, M.B.: Continual Learning in Reinforcement Environments. GMD-Bericht (1994)
27. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual Learning with Deep Generative Replay. arXiv:1705.08690 [cs] (Dec 2017)
28. Tjoa, E., Guan, C.: A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. IEEE Transactions on Neural Networks and Learning Systems **32**(11), 4793–4813 (Nov 2021). https://doi.org/10.1109/TNNLS.2020.3027314
29. Wang, S., Choi, Y., Chen, J., El-Khamy, M., Henao, R.: Toward Sustainable Continual Learning: Detection and Knowledge Repurposing of Similar Tasks (Oct 2022)
30. Yu, H., Webb, G.I.: Adaptive online extreme learning machine by regulating forgetting factor by concept drift map. Neurocomputing **343**, 141–153 (May 2019). https://doi.org/10.1016/j.neucom.2018.11.098