Employing dropout regularization to classify recurring drifted data streams

Filip Guzy
Department of Systems and Computer Networks
Wrocław University of Science and Technology
Wrocław, Poland
0000-0002-8619-7080

Michał Woźniak
Department of Systems and Computer Networks
Wrocław University of Science and Technology
Wrocław, Poland
0000-0003-0146-4205

***Abstract*—Streaming data analysis is currently a rapidly growing research direction. One of the serious problems hindering the data stream classification is the fact that during the exploitation of the model, its probabilistic characteristics may change. This phenomenon is called concept drift. Until today, multiple methods have been proposed to overcome their negative influence on model performance during learning in dynamic environments. This work introduces a new streaming data classifier based on a dropout technique that can significantly reduce model restoration time and performance loss and can improve its overall score in the presence of recurring concept drifts. The usefulness of the proposed algorithm is evaluated based on extensive experimental study and backed-up with thorough statistical analysis.**

***Index Terms*—Dropout, recurring concept drift, data streams, restoration time, performance loss, multilayer perceptron**

I. Introduction

Machine learning models that work in dynamic environments, like weather predictors, stock analyzers, or email filters, have to handle multiple changes in upcoming data distribution without losing their predictive performance, i.e., each model should adapt to the changes as quick as possible without significant deterioration of its quality [1].

The mentioned phenomenon of data distribution change is unpredictable and is called *concept drift* [2]. The development of methods that can deal with the analysis of non-stationary data streams is currently the subject of intensive research. The main challenge is to propose mechanisms for adapting the model to changing data distributions. The most commonly used approaches are based on *concept drift* detection and subsequent model rebuilding when it occurs, or the built-in adaptability of classification algorithms that are based on forgetting mechanisms using sliding windows or instance weighting. A group of algorithms often used is classifier ensembles [3], which can adapt to concept drift due to a change in the composition of the ensemble, or a change in the so-called *combination rule*.

The paper will propose a method of non-stationary data stream classification using the dropout [4] regularization technique. Models that use this technique drop some of their neurons during the training process for each sample, which helps them to generalize data accurately. We will use this approach to construct a repository of models to classify a data stream with the so-called *reoccurring concept drift*. Harries et al. [5] observed that context recurrence frequently appears in financial prediction, dynamic control, or cyclic phenomena caused, e.g., by season changes. We propose to generate a new model for each new concept using the same neural network architecture but dropping out a few neurons for each training. We observed that when using this method, models can significantly reduce the crucial metrics (as the restoration time and the performance loss) during the classification of the data stream affected by *recurring concept drift* and also keep its accuracy on a satisfactory level.

In a nutshell, the main contributions of this work are as follows:

- Explanation of the dropout usefulness in the *recurring concept drift* environment.
- Proposition of a neural network model with *dropout* regularization for the data stream learning.
- Evaluation of the proposed method based on the diverse synthetic data streams.

II. Related works

Some of the articles on data stream classification concentrate on incremental learning [6]. In this task, the classifier is either updated or needs to retrain [7]. We consider the data stream in the form of data chunks (batch learning). We will skip the problem of how to properly adjust the size of a chunk, but obviously, some advanced algorithms change it dynamically [2], or they can process data using several windows [8].

Let's define a data stream as an ordered sequence of values $S = \{S_1, S_2, ..., S_n\}$, which are usually processed in a form of chunks. Each chunk $S_i = \{(x_i^{(1)}, y_i^{(1)}), (x_i^{(2)}, y_i^{(2)}), ..., (x_i^{(n)}, y_i^{(n)})\}$ consists of sequence of n observations, i.e., $x_i^{(k)}$ stands for the values of attributes describing the kth instance in the S_i chunk, while $y_i^{(k)}$ denotes its label.

The probabilistic characteristics of data distribution can change over time, and as we mentioned above, this phenomenon is known as *concept drift*. Proper classification of a data stream is dependent on an occurring drift type. Gama et al. [9] discussed the different types of concept drift, and they distinguished two main *concept drift* types: virtual and real. The first one happens when data distribution change does not

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

affect model performance; i.e., it does not affect the shapes of the decision boundaries. The second one forces our model to re-adapt to the incoming data. Another taxonomy distinguishes concept drift types because of its suddenness and enumerates sudden and incremental drifts. The *sudden concept drift* results in a significant model performance deterioration in a short time because data distribution changes very fast. The variation of the sudden drift is the *gradual concept drift*, i.e., it happens if we may observe that the instances from two distributions appear at the same before concept switches to new distribution. The *incremental concept drifts* are smoother and can be handled with adaptive learning methods. The *recurring concept drifts* are usually highly predictable - we can prepare proper defensive mechanisms before they come. They may occur both suddenly or incrementally. Fig. 1 illustrates different types of the *concept drift* for a simple one-dimensional data.

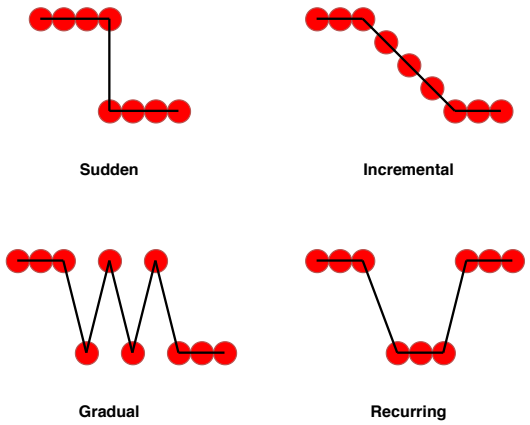

Fig. 1. Illustration of the different types of concept drift.

One of the methods of dealing with the occurrence of changes in data distribution is the use of so-called drift detectors [10] or employ the adaptive skills of the classification models. Bifet and Gavald introduced the drift detection method based on adaptive windowing [11]. The main goal of this approach was to keep statistics from a window of variable size while detecting *concept drift*. Gama et al. [10] proposed DDM (*Drift Detection Method*), which is the most well-known representative of the detectors based on statistical method control. It estimates a predictive performance of a classifier (and its standard deviation), which (assuming the convergence of the classifier training method) has to decrease as more training examples are received. Modification of this approach, called EDDM (*Early Drift Detection Method*), was proposed by Garcia et al. [12]. It keeps a good performance against *sudden concept drift* and improves the detection rate of *gradual concept drift* as well. The work by Gama et al. [9] presents an interesting overview of the most popular adaptive learning methods that help handle different types of *concept drift* and among others, it also describes several windowing algorithms that use the forgetting mechanism.

Another approach to dealing with *concept drift* is to use the classifiers' ability to adapt to changes in the data distribution. The most commonly used models are the classifier ensembles. Kuncheva [3] reviewed different ensemble methods for the *concept drift* detection. Krawczyk et al. also presented a survey about classifier ensembles for data stream classification and regression [13]. One of the earliest well-known classifier ensembles dedicated to data stream classification are AWE (*Accuracy Weighted Ensemble*) developed by Wang et al. [14] and SEA (*Streaming Ensemble Algorithm*) proposed by Street and Kim [15]. It was also worth mentioning Learn++ dedicated to stationary stream classification [16]. Learn++ constructs new neural network models on each incoming data chunk and then combines their outputs using majority voting. Its modification Learn++.NSE [17] was also proposed for the classification of the non-stationary data streams. Katakis et al. [18] proposed another example of an ensemble algorithm for data streams containing recurring concept drifts. It assumed building a conceptual feature space for batches of examples and using the clustering algorithm for the identification of different concepts. Brzeziński and Stefanowski [19] presented an accuracy updated ensemble algorithm adjusted for different types of concept drift. In another work, they also proposed an ensemble of combined block-based and online methods for concept-drifting data streams [20].

Most of the classification models, even if they form a complex model, experience a phenomenon of catastrophic forgetting [21]. It is highly undesirable in the case of *recurring concept drifts* occurrence because the model has to learn the same task several times. There were multiple approaches to reduce the impact of *catastrophic forgetting*. Kirkpatrick et al. proposed the Elastic Weight Consolidation algorithm that helps to keep already gathered knowledge in neural network [22]. Ksieniewicz et al. proposed to benefit from the forgetting in overcoming *concept drift* related problems in an online active learning approach [23]. The dropout-based technique introduced in this work should not be affected much by *catastrophic forgetting* because it is using submodels generated for each task.

III. Methods

Dropout, due to its similarity to ensemble methods, can be beneficial in the learning from the data stream if we have some prior knowledge about the data. Assuming that the streaming samples are drawn from the fixed number of distributions, we can expect the recurring drift appearance in the learning process. To detect them we use DDM, mentioned above drift detector introduced by Gama et al. [10], with an increased sensitivity basing on assumption that the learner's error rate will decrease if the number of analyzed samples increase, as long as the data distribution is stationary, and that there is no disruptive noise in generated data streams. We assume that the drift is detected if:

$$p_i + s_i \geq p_{min} + 1.5 \cdot s_{min} \tag{1}$$

where p_i is the error rate of the learning algorithm for i sample, and s_i is the standard deviation. To properly handle concept

drifts, the model can generate submodels by dropping out the specified percent of its neurons on each detected distribution change during the training process. Generated submodels configurations can be stored in the models' memory and used to handle further concept drifts. Submodels are picked in a way of internal accuracy evaluation using the lastly achieved data chunk. Stream was processed using *test-then-train* method [24], which was presented in Fig. 2.

We propose to drop the chosen neurons only in the training phase. During the testing phase, all neurons are active, but the weights of previously dropped neurons are multiplied by the dropout retain probability. With this approach, we have to make sure that our model has enough capacity to generate a desired number of submodels. In the best-case scenario with all data changes appropriately detected, the resulting number of submodels should be equal to the number of distributions available in the stream. The pseudocode of the proposed method is presented in Alg. 1.

Algorithm 1 Dropout algorithm for the data stream
Input: *m* - model
s - data stream
ddm - drift detector (DDM)
n - number of chunks
test() - procedure that tests model with a chunk and returns a score
train() - procedure that trains model with a chunk
change_detected() - procedure that informs about drift occurrence using drift detector and last score
set_submodel() - procedure that evaluates all internal submodels with the last chunk and returns the best one

```
for i = 0 to n do
    chunk ← s(i)
    score ← test(m, chunk)
    if change_detected(ddm, score) then
        set_submodel(m, chunk)
    end if
    train(m, chunk)
end for
```

There are two different ways to apply dropout on each distribution change. The first way assumes making no additional modifications to weights during model switching. When the new submodel is trained, its weights evolve separately from the previous one. Further changes in other submodel weights do not influence it. The second approach assumes that some weights are being updated. It means that after each switch, a newly picked submodel, just after dropout application, updates its inactive weights that were active in the previous submodel. We have to remember that all neurons are used in the testing phase, so this method allows sharing some knowledge among different submodels. These two approaches are presented in Fig. 3.

Shaker and Hüllermeier [25] proposed the performance metrics: restoration time and maximum performance loss, which are very useful when we analyze the classifier reaction to *concept drift*. The restoration time is defined as:

$$T_R = \frac{t_2 - t_1}{T} \in [0, 1] \tag{2}$$

where in our case, t_1 is a chunk number for which model learning curve drops below 95% of the achieved accuracy, t_2 is a chunk number for which model learning curve restores to 95% of achieved accuracy, and T is the total number of chunks in a single stream. For each drift appearance, the maximum performance loss is defined as:

$$L = \frac{Acc(t) - Acc_{12}(t)}{Acc(t)} \tag{3}$$

where $Acc(t) = min\{Acc_1(t), Acc_2(t)\}$ is the lower accuracy value of two learning curves surrounding the drift area, and Acc_{12} is the lowest accuracy achieved in a drift area. Apart from the above metrics, we also calculate a mean accuracy for all models. Gathered values are compared using Wilcoxon sign-ranked test [26].

IV. Experiment

A. Research hypothesis

We expect that models using dropout should behave much better in a recurring drift environment than the unmodified perceptron. Because they are using submodels for different data distributions, they should achieve lower restoration times and consequently higher mean accuracy rates. In the case of the normal dropout approach and its weights sharing modification, we expect to notice some differences in achieved accuracy and performance loss, but not in restoration times. We assume that statistically significant differences will be visible for scores obtained by the different classifiers.

B. Experimental setup

Three multilayer perceptrons were chosen to be compared among themselves: with no dropout, 25% dropout, and 25% dropout with weights sharing. Neural network layers contained: 2, 200, and 1 neuron respectively. They were trained using *test-then-train approach* [24] on 150 replicable data streams generated with *stream-learn library* [27]. Each data stream contained 1000 chunks and was built of two recurring data distributions with six sudden concept drifts. Single data chunk contained 100 balanced data samples belonging to 2 different classes. Variability of data streams was guaranteed by using unique random seeds in generator. Example of *concept drift* generated in datasets is presented in Fig. 4. Learning from a small number of generated data streams, because of their complexity, resulted in model underfitting and drift detection problems. These misleading results produced no added value and were not contained in the final comparison. Achieved scores: performance loss, restoration time and accuracy were compared using Wilcoxon sign-ranked statistical test with a 99% confidence level.

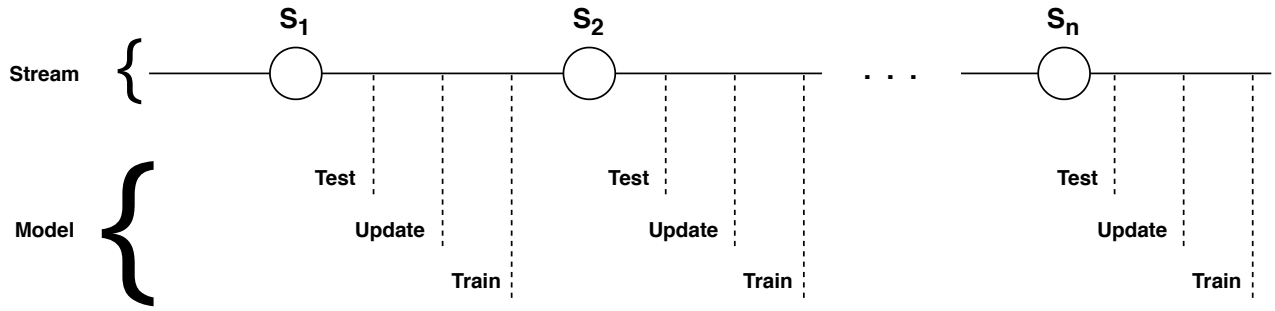

Fig. 2. Illustration of the *Test-then-train* evaluation scheme.

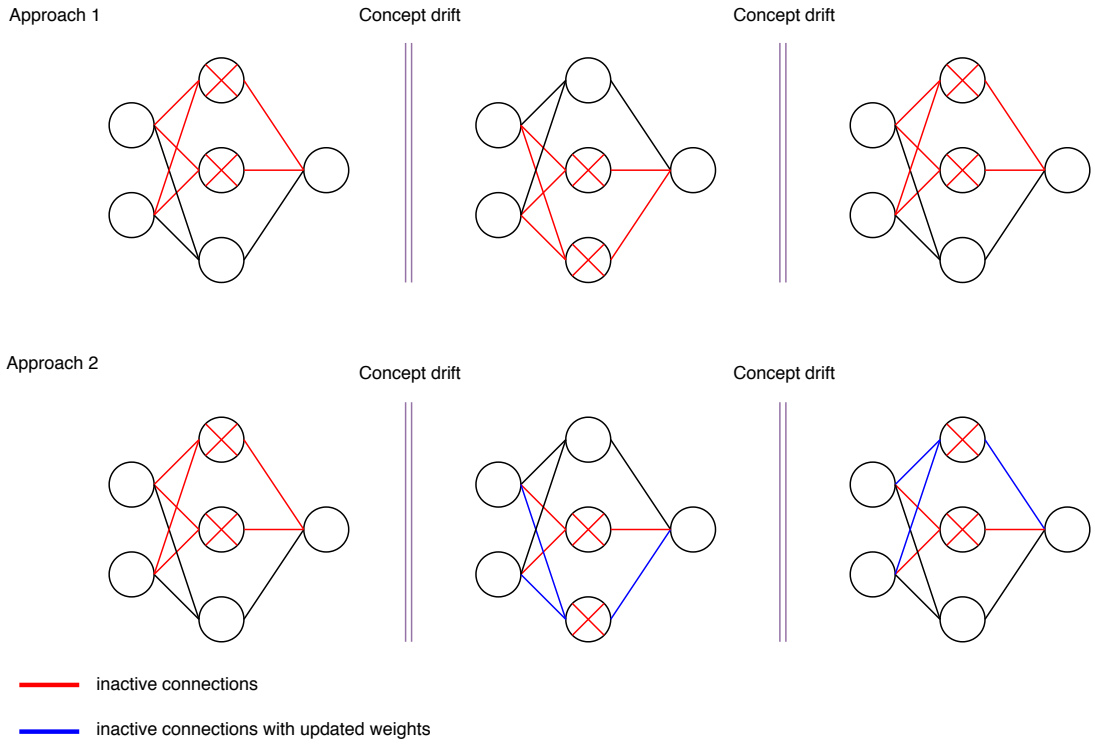

Fig. 3. Two ways of dropout application. In Approach 1 we only switch submodels, in Approach 2 we additionally perform weights update.

C. Results

We mark models with shortcuts:

- MLP - multilayer perceptron with no dropout
- MLP D25 - multilayer perceptron with 25% dropout
- MLP DW25 - multilayer perceptron with 25% dropout and weights sharing

Tab. I shows the mean performance loss, restoration time, and accuracy values for all streams. Fig. 5 shows generated learning curves for selected streams. Tab. II presents the model comparison according to results achieved with the Wilcoxon test. Plus sign in the cell means that the model in the column is statistically significantly better than the model in the row. All results can be found in the related public code repository [28]. We can see that MLP D25 performs better than MLP and MLP DW25. MLP DW25 achieves better scores than MLP, but worse than MLP D25.

D. Results discussion

Achieved scores prove the hypothesis that the dropout-based models behave better than the normal ones. Submodels separation makes it possible to keep their weights unchanged when they are not in use, which reduces the chance that they will have to learn data again from scratch. Using the proposed approach, we should keep in mind to construct models that will have enough capacity to generate the proper number of

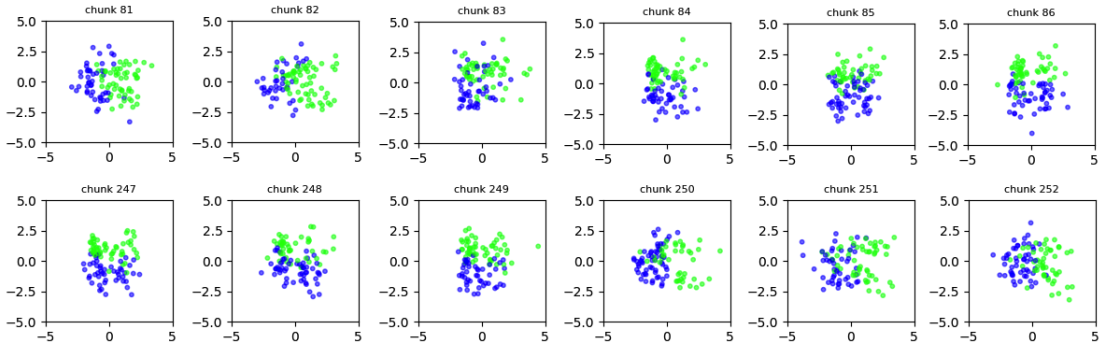

Fig. 4. Example of sudden recurring drifts in the generated data stream.

TABLE I
PERFORMANCE METRICS FOR THE THREE TESTED MODELS.

Model type	**Mean scores**		
	Performance loss	***Restoration time***	***Accuracy***
MLP	0.5000	0.0323	0.8224
MLP D25	0.4705	0.0068	0.8687
MLP DW25	0.4618	0.0068	0.8648

TABLE II
RESULT OF WILCOXON'S SIGNED-RANK TEST. + MEANS THAT THE MODEL IN THE COLUMN IS STATISTICALLY SIGNIFICANTLY BETTER THAN THE MODEL IN THE ROW.

Compared to	**Models**		
	MLP	MLP D25	MLP DW25
MLP	-	+	+
MLP D25	-	-	-
MLP DW25	-	+	-

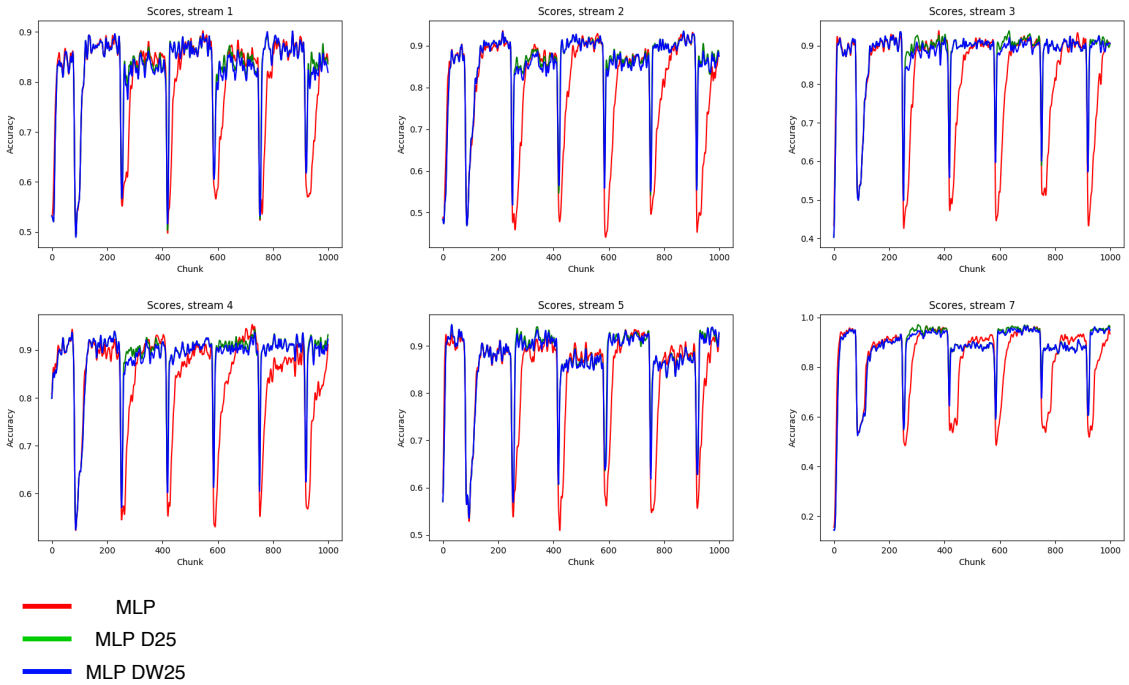

Fig. 5. Learning curves for the selected data streams.

submodels. If concept drifts are detected early enough, it also helps to avoid the *catastrophic forgetting* phenomenon. On the other hand, the gathered results show that the dropout approach with weights sharing is not so good as we thought it would be. The scores are even worse than in a classic variant. The performance loss is lower for the weights sharing model, but it is directly related to the achieved accuracy value. It looks like weights update introduces an undesired noise to the learning process.

V. Conclusions

The main aim of this research was to propose a novel, effective dropout-based algorithm for data stream classification affected by the recurring concept drift. The proposed method, which builds a pool of neural networks for each appearing concept, could react very fast for a concept drift appearance. The computer experiments confirmed the usefulness of the proposed approach and based on a thorough statistical analysis. We may assert that this method allows us to shorten the restoration time as well as maximum performance loss in comparison to the canonical model of neural networks. Additionally, we also observed that the prediction performance is significantly better than the baseline model.

In summary, lessons learned are as follows:

- Using a dropout-based model significantly improves the overall performance for recurring drifted data streams.
- Models using dropout has to provide enough capacity to handle multiple submodels.
- Frequently occurring drifts may lead to the generation of multiple poorly performing models.
- In the best-case scenario, the number of generated submodels should be less or equal to the number of distributions in upcoming data.
- Weight sharing introduces a noise to the learning process, and it is not an effective approach.

This work is a step forward towards the use of the neural networks for non-stationary data streams. Results obtained in this study encourage us to continue works on alternative approaches with the special focus on the following issues:

- Evaluate how the dropout-based method is robust to data noise.
- Evaluate how it reacts to other types of concept drift.
- Develop the methods of classifier ensemble forming based on the proposed dropout-based approach.
- Propose novel metrics of restoration time as well as maximum performance, which can take into consideration a higher number of drifts, as well as their frequency.
- Extend the experimental study on real data streams.

References

[1] S. Ramirez-Gallego, B. Krawczyk, S. Garcia, M. Wozniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39 – 57, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217302631

[2] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[3] L. I. Kuncheva, "Classifier ensembles for detecting concept change in streaming data: Overview and perspectives," in *Proc. 2nd Workshop SUEMA 2008 (ECAI 2008)*, 2008, pp. 5–10.

[4] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: http://www.cs.toronto.edu/ rsalakhu/papers/srivastava14a.pdf

[5] M. B. Harries, C. Sammut, and K. Horn, "Extracting hidden context," *Mach. Learn.*, vol. 32, no. 2, p. 101–126, Aug. 1998. [Online]. Available: https://doi.org/10.1023/A:1007420529897

[6] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn^{++}.nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 152–168, 2009.

[7] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 97–106.

[8] M. M. Lazarescu, S. Venkatesh, and H. H. Bui, "Using multiple windows to track concept drift," *Intell. Data Anal.*, vol. 8, no. 1, pp. 29–59, Jan. 2004.

[9] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014. [Online]. Available: http://doi.acm.org/10.1145/2523813

[10] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proceedings of the Seventh Brazilian Symposium on Artificial Intelligence (SBIA'04) - Lecture Notes in Computer Science*, vol. 3171. São Luiz do Maranhão, Brazil: Springer, 2004, pp. 286–295.

[11] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. 7th SIAM Int. Conf. Data Min.*, 2007.

[12] M. Baena-García, J. Campo-Ávila, R. Fidalgo-Merino, A. Bifet, R. Gavald, and R. Morales-Bueno, "Early drift detection method," 01 2006.

[13] B. Krawczyk, L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, p. 132–156, 09 2017.

[14] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 226–235. [Online]. Available: https://doi.org/10.1145/956750.956778

[15] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 377–382. [Online]. Available: https://doi.org/10.1145/502512.502568

[16] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *Trans. Sys. Man Cyber Part C*, vol. 31, no. 4, p. 497–508, Nov. 2001. [Online]. Available: https://doi.org/10.1109/5326.983933

[17] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, Oct 2011.

[18] I. Katakis, G. Tsumakas, and I. Vlahavas, "An ensemble of classifier for coping with recurring contexts in data streams," in *Frontiers in Artificial Intelligence and Applications (ECAI 2008)*, 2008, pp. 763–764.

[19] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. on Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 81–94, 2014.

[20] ——, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Information Sciences*, vol. 265, pp. 50–67, 2014.

[21] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4652–4662. [Online]. Available: http://papers.nips.cc/paper/7051-overcoming-catastrophic-forgetting-by-incremental-moment-matching.pdf

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, 12 2016.

[23] P. Ksieniewicz, M. Wozniak, B. Cyganek, A. Kasprzak, and K. Walkowiak, "Data stream classification using active learned neural networks," *Neurocomputing*, 03 2019.

[24] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, p. 1601–1604, Aug. 2010.

[25] A. Shaker and E. Hüllermeier, "Recovery analysis for adaptive learning from non-stationary data streams: Experimental design and case study," *Neurocomputing*, vol. 150, pp. 250–264, 2015.

[26] F. Wilcoxon, "Individual comparisons by ranking methods," in *Biometrics Bulletin*, vol. 1. International Biometric Society, 1945, pp. 80–83.

[27] P. Ksieniewicz and P. Zyblewski, "stream-learn - open-source python library for difficult data stream batch analysis," *ArXiv*, vol. abs/2001.11077, 2020.

[28] "ds-dropout-submodels," 2020. [Online]. Available: https://gitlab.com/filipmg/ds-dropout-submodels