

INTRODUCTION

- In a production environment, machine learning models are trained over historical data and then deployed. Over time, the accuracy of such a model decreases since the distribution of real-world data always changes. When this happens, we say that concept drift (or data shift) occurred.
- Concept drift can be abrupt (sudden, instantaneous), or gradual (it takes a certain number of samples until the distribution changes and becomes stationary again). The two types are shown in Fig. 1.
- In this study, we focus on label-independent concept drift detectors to spot the moment when this change in the distribution occurs.
- A framework for detecting concept drift, separated into four stages, is summarized in Fig. 2.
- Concept drift detectors are of different types, based on *data distribution*, *error rate*, *multiple hypothesis*, or *mixed*.

Fig 1. Types of drift [1]

Fig 2. Framework for drift detection [2]

RESEARCH QUESTIONS

- RQ1: How well do mixed concept drift detectors identify drift in the case of synthetic/ real-world data?
- RQ2: How do mixed concept drift detectors perform compared to other label-dependent detectors?

REFERENCES

[1] Lorena Poenaru-Olaru, Luis Cruz, Arie van Deursen, and Jan S. Rellermeyer. Are concept drift detectors reliable alarming systems? – a comparative study, 2022.

[2] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, andGuangquan Zhang. Learning under concept drift: A review. IEEE Transactions on KnowMedge and Data Engineering, 31(12):2346– 2363, 2019.

[3] Sulaimon Adebayo Bashir, Andrei Petrovski, and Daniel Doolan. A framework for unsupervised change detection in activity recognition. International Journal of Pervasive Computing and Communications, 13(2):157– 175, 2017.

[4] Andre G. Maletzke, Denis M. dos Reis, and Gustavo E.A.P.A. Batista. Quantification in data streams:Initial results. 2017 Brazilian Conference on Intelligent Systems (BRACIS), 2017.

Analysis of Mixed Concept Drift Detectors in Deployed Machine Learning Models

METHODOLOGY

- We define a specific data setup, where we separate the data into training and testing. The testing data is then divided into equal-size batches, similarly to Stage 1 of Fig. 1.
- Synthetic data sets: *SEA*, *AGRAW1*, *AGRAW2*. For each type, we have a data set with abrupt drift and five with gradual drift, with widths of 500, 1000, 5000, 10000, 20000. The size of the data sets is 100k samples, 30k for training and 7 batches x 10k for testing. We know that drift starts at sample 55000, meaning in batch 3, as shown in Fig 4. For evaluation, we define false positive rate (FPR\_S) and latency (L).
- Real-world data sets: *Airlines*, *spam*, *ELECT2*, and *Weather*. For evaluation, we define different metrics than for the synthetic data sets, namely *false positive rate* (FPR\_R) and *drift detection rate* (DDR). In order to perform the experiments, we also need some batches that we can consider as *actual drift*. To do this, we defined a method for finding drift based on the error-rate increase using cross-validation on the training data. We consider two types of splits for this method: *time-based splitting* and *sequential splitting*. As shown in Fig. 5, the two methods signal drift at different batches, so we considered both during the experiments.
- Two label-independent drift detectors, SQSI [3], and UDetect [4], are implemented from scratch and evaluated on both synthetic and real-world data. It is important to mention that the detectors are not updated after drift is signaled. The implementation is available on GitHub (<https://github.com/tzamfrescu/concept-drift-detection>).
- Five label-dependent drift detectors are imported from the *river* Python library and evaluated only on real-world data.

Table 1: Results of SQSI and UDetect on synthetic data sets with abrupt drift

Table 2: Results of different label-independent concept drift detectors, applied on real-world data sets

CONCLUSIONS

- The two mixed label-independent concept drift detectors, SQSI and UDetect, were difficult to implement and adapt to our data setup, mainly because implementation details were missing.
- These detectors performed well on synthetic data sets, for both abrupt and gradual drift.
- The evaluation of detectors on real-world data is a challenge - it is difficult to reliably calculate the *reference drifted batches* for evaluation. Also, the fact that we do not update the detectors as soon as drift occurs influences the results in a negative way. Therefore, in our setup, the detectors are not reliable on real-world data sets.
- The evaluation of label-dependent concept drift detectors requires a different data setup.
- More computational power would improve the results of this study.

This research paper was part of the 2022-2023 Q2 edition of the *CSE3000 Research Project* course in TU Delft's BSc Computer Science and Engineering. The project proposes a comparison study that analyzes the performances of different unsupervised concept drift detectors in a setup suited for deployed machine learning models.

Fig. 4 Correct drift detection for the synthetic data sets

Fig. 5 Drifted batches in the spam data set, using the two split methods

RESULTS

- The label-independent detectors SQSI and UDetect performed very well on synthetic data, with average FPR\_S = 0.1 and L = 0.1. Table 1 shows the results for detectors on the synthetic data sets with abrupt drift. These two detectors only work on numerical features, so they rely on scalars and encoders for the features. Also, they are sensitive to gradual drift - the latency increases when the width of the drift increases.
- The same two detectors performed poorly on real-world data, detecting drift in most batches (FPR\_R and DDR were both close to 1), which is expected since the detectors are not updated once drift occurs. Furthermore, it is difficult to reliably compute the *actual drifts* for the real-world data; even the type of splitting in the method produces different results. The results of the experiments on real-world data are shown in Table 2.
- The label-dependent concept drift detectors from the *river* Python library produced inconsistent results on this setup. This is also expected since those detectors are updated as soon as drift occurs, therefore relying on testing data. Our detectors and setup rely only on the training data for detection.