Stream mining. Lab 2: Hoeffding trees and first drift detection

Maciej Grzenda, PhD, DSc
Maciej.Grzenda@pw.edu.pl

Faculty of Mathematics and Information Science
Warsaw University of Technology, Poland

Warszawa, 2025

Introduction

Lab objectives

The objective of this lab is to get hands on experience in the use of the first stream mining method, i.e. Hoeffding trees, and the use of drift detectors.

We will use both synthetic data streams and real data streams

We will look for changes in the prediction errors occurring over time

This will rely mostly on `river`, i.e. Python-based library

Note that to get some preliminary results or cross-check your findings, you can also use MOA. Will the results be identical under the same settings?

While we are going to use the `river` library, please note that further details on many stream mining algorithms implemented in the library can be found in [1].

Training and evaluating Hoeffding trees

Task 1: train-then-test for Hoeffding trees

Task objectives

What happens when we incrementally learn a tree model given instances representing stationary phenomenon, i.e. fixed $p(x)$ and $p(y|x)$? Will the accuracy increase? If so, will it increase monotonically?

To investigate such a setting:

1. Please implement a Python script using `river` library to generate 10,0000 instances from the `hyperplane` generator configured with stationary setting. You may assume 2D instances, i.e., $\mathbf{x}_i \in \mathbb{R}^2$
2. The script should use instances from the stream as input for training and testing Hoeffding trees in the immediate labelling setting, i.e. test-then-train evaluation
3. please investigate the accuracy of the model for the entire data stream and over a sliding window of 100 instances

First drift detection experiment

Task 2: drift detection

Task objectives

What happens when the distribution of our data changes (or does not)? Will the drift detector detect it? If so, will it immediately recognise that $p(x)$ has changed?"

To investigate such a setting:

1. Please implement a Python script using `river` library to use ADWIN detector `https://riverml.xyz/dev/api/drift/ADWIN/`) for detecting changes in real or synthetic data provided by you, starting from data in which $p(x)$ is constant
2. Next, please use one of the numeric features from the data set you like, ideally based on a time series. As an example, you may use the London travel mode choice data set.
3. Investigate different ADWIN `delta` and `grace_period` settings to observe their impact on drift detection

Drift detection in model errors

Task 3: detecting drift in model errors - stationary data

Task objectives

How can we detect significant changes in errors made by, e.g. a classifier? Will the drift detector detect it? If so, will it immediately recognise that $p(e)$ has changed, while e_i is the error made for i-th instance?"

To investigate such a setting:

1. Please implement a Python script using `river` library to use ADWIN detector this time for detecting changes in errors made by a model
2. To get the stream of errors:
   - use your script from Task 1 and apply the Hoeffding tree (HT) to stationary hyperplane data
   - use the stream of errors of the HT model as an input for the drift detector

Will it matter how we configure ADWIN `delta` and `grace_period` settings? What would the practical consequences be?

Task 4: detecting drift in model errors - non-stationary data

Task objectives

How do we detect significant changes in errors made by, e.g. a classifier? Will the drift detector detect it? If so, could we use drift detection to realise that drift has occurred?"

To investigate such a setting:

1. Please modify the script from the previous task to work with non-stationary hyperplane data

How could we use the fact that drift has been detected in the error stream? Should we create a new model after drift detection and use it instead of the original one?

Appendix I: installing river and CapyMOA

- To install `river` please follow `https://riverml.xyz/0.22.0/introduction/installation/`. Basically, it is enough to use `pip install river`
- To install CapyMOA:
  - please follow `https://capymoa.org/installation.html`.
  - Make sure, you also perform the steps described in **Install CapyMOA for Development**

References I

[1] Albert Bifet, Ricard Gavaldà, Geoff Holmes, and Bernhard Pfahringer. *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018. https://moa.cms.waikato.ac.nz/book/.