Stream mining. Machine learning methods for data streams. Part I: introduction

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

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

Warszawa, 2025

Introduction

Course objectives

In an increasing number of cases, data can **be constantly acquired and processed** as **data streams** rather than as (periodically) collected **static datasets** with a finite content. Examples include:

- data obtained from measurement devices in industrial environments
- vehicle location data streams
- Twitter/X news feed

The aim of the course is to:

- learn about machine learning methods dedicated to data streams
- focus on learning in non-stationary setting (concept drift)
- pay attention to real conditions such as e.g. limited availability of true labels and verification latency.

Bibliography - part I

Key references - machine learning:

1. Bifet A. et al, Machine Learning for Data Streams with Practical Examples in MOA, MIT Press, 2018
2. S. Putatunda. Practical Machine Learning for Streaming Data with Python: Design, Develop, and Validate Online Learning Models, Apress, 2021

Research papers on stream mining, such as:

1. Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. IEEE Computational Intelligence Magazine, 10(4):12-25, 2015.
2. H. M. Gomes, J. Read and A. Bifet, "Streaming Random Patches for Evolving Data Stream Classification," 2019 IEEE International Conference on Data Mining (ICDM), Beijing, China, 2019, pp. 240-249, doi: 10.1109/ICDM.2019.00034
3. Grzenda M., Gomes H. M., Bifet A.: Delayed labelling evaluation for data streams, Data Mining and Knowledge Discovery, 2020, vol. 34, pp.1237- 1266. DOI:10.1007/s10618-019-00654-y

Assessment rules

Assessment criteria

The assessment will be based on a analytical project and a final test, each marked on a scale of 0-50 points. The final grade depends on the total number of points obtained from both parts and is determined according to the following rules:

- 0-50 points - 2.0
- 51-60 points - 3.0
- 61-70 points - 3.5
- 71-80 points - 4.0
- 81-90 points - 4.5
- 91-100 points - 5.0

To pass this module you also have to:

- get at least 25 points from project
- and get at least 25 points from final test

The test will include open questions and will be arranged during lab no. 11 in this semester.

Project scope - informal introduction

Develop novel method(s) in stream mining and use partly new data streams.

This can be novel classification methods, evaluation methods, visualisation of evolving models, decision boundaries or changing feature importance.

Evaluate your methods as in the best papers in the field, i.e. using a selection of real and synthetic streams.

Develop a conference paper if your results go beyond what has been already done, if you like.

Contribute your code via pull requests to one of open source projects, if you like.

The goal is to develop a truly inspiring and interesting machine learning solution focusing on incremental learning and possibly changes in the data - such as changes in real decision boundaries!

Project scope - more formal summary

The analytical project includes:

- selecting real and synthetic data streams for your project
- performing drift detection for the streams
- developing novel methods in stream mining domain
- evaluating your methods and reference methods
- placing all data, code and results you have developed including persisted synthetic data streams in github

Report should document all that you have done, including drift detection results, novel method(s), the way they were evaluated and the outcomes of this evaluation.

Note that instead of report you can prepare a manuscript of a paper to be submitted for a conference from CORE ranking (or a journal).

The use of stream mining libraries and frameworks is encouraged. The same applies to contributing to open source projects.

Project milestones

Milestone	Description	Product	Deadline
MS1	Submission and presentation of the project idea	Project charter (slides)	Week 4
MS2	Submission of the initial version of the analytical project	Initial project (project outcomes will be also presented by the authors in weeks 10-11)	Week 10
MS3	Teacher's initial assessment of the analytical project	Scoring and comments	Week 12
MS4	Submission of the final result of the analytical project	Final project	Week 14

Project submission means submitting code, data, results and report and in the case of final project a list of changes made since the initial version of the project. You can get up to 10 points for project idea and up to 40 points for the rest of the project.

Project details

Project charter is developed in the form of slides and presented by the project team and should define team members, preliminary selection of streams, including novel streams, project idea and envisaged outcomes.

A project can be done by 1-3 students.

In the case of group projects, the scope of work to be done by each student has to be clearly documented.

Every team member has to:

- apply drift detectors for novel data streams and analyse whether the use of stream mining methods is justified
- contribute to selecting, applying and developing stream mining methods and analysing results obtained with these methods

Further comments

Successfully contributing to well-known stream mining projects such as MOA and river, i.e. having your pull requests accepted is strongly recommended. This will both build your experience and let you get a higher grade from the project.

Delay of at most one week in delivering milestone results:

- maximum number of points for this milestone is reduced by 20%
- Delay of more than one week in delivering milestone results: maximum number of points for this milestone is reduced by 40%.
- Note that the project has to be finished before the last teaching day of the semester to be graded.

Extra points are possible for getting outstanding results e.g. results, which could be published in conference proceedings and are documented in the manuscript. In such case, you can get up to 15 extra points, i.e. possibly up to 65 points together from your project.

Big Data processing and stream mining

The need for stream mining

More and more data are available as data streams.

This includes:

- sensor data including the data made available by smart objects comprising on IoT networks
- news feeds
- financial data

This raises a question of how to develop ML models using data streams. Similarly to general-purpose stream processing this can be done by:

1. splitting data streams into chunks of data and applying batch processing to these chunks
2. developing new algorithms and frameworks focused on continuous stream processing. In the case of ML methods, this means novel ML methods and frameworks dedicated for data streams i.e. *stream mining* methods

The difference between data sets and data streams

Table: Comparison of data set vs. data stream processing

	Bounded data	Unbounded data
Data available as	Data files a.k.a. *data-at-rest*	Data stream a.k.a. *data-in-motion*
Input for data processing	A data set $S = \{s_1, \ldots, s_N\}$ fully available before data processing starts	A sequence of constantly arriving tuples $\mathbf{s}_1, \mathbf{s}_2, \ldots$, typically describing some events

Data streams

Data streams unlike batches of data:

- Are of infinite size
- Can arrive at large speeds

Furthermore, data streams

- Can require data fusion i.e. joining different streams e.g. separately arriving streams of temperature and pressure, before ultimate processing can be done
- Can include *out-of-order data* i.e. some instances of a stream may arrive in a reversed order compared to the events they describe
- Can include instances arriving with non-negligible latencies
- Can be produced by IoT (Internet of Things) nodes and processed in distributed setting including IoT nodes, edge components and cloud
- Can be stored as data files on periodical basis and processed with batch processing, subject to processing latency constraints

Batch processing vs. stream processing

In the case of batch processing:

- The same records can be revisited e.g. by iterative optimisation algorithm
- When ML techniques are used, the features of the data set such as proportion of classes can be used to develop better ML models
- Usually, results of processing are expected within minutes or even days

In the case of stream processing:

- The input data has to be processed as it arrives
- There are little or no chances of revisiting the same data during the calculations
- The features of the data set such as proportion of classes can potentially evolve over time
- Usually, results of processing e.g. predictions are expected within a few seconds or less than a second

Big Data and stream mining

Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation [9]

Hence, ML methods for Big Data by definition should be *cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation*.

Stream processing benefits

In practice, benefits arising from stream processing also for not so large data assets (not Big Data scale) are also observed. These include continuous data processing rather than waiting for receiving data sets of acceptable size.

The (frequently unspoken) assumptions of batch learning

One of the most popular algorithms of constructing decision trees is classification and regression tree (CART) algorithm, proposed by Leo Breiman. The CART classifier is trained in batch mode:

1. First **(sufficiently large)** training data set has to be collected
2. Next, the classifier is created in **(possibly a time-consuming, and not time-constrained)** process, **(which may involve testing different parameter values)**
3. Only in the third stage, the classifier can be used i.e. can assign labels to new instances **(It is not clear whether and if so, when to update the classifier)**

This approach and the problems affecting it are common for batch learning techniques. By batch ML methods, we consider methods which use a *data set* rather than *data stream* as an input for the training.

Batch learning

Batch learning also known as *offline learning* [1] is the key group of ML algorithms. The input for batch learning algorithms is a single data set. Most frequently the set is assumed to:

- be of moderate size making it possible to:
  - allocate all the data fully in operating memory
  - iterate over the data set many times
- represent stationary phenomenon e.g. with static classification borders, not changing with time.

Typically, a model is developed in a time consuming process, performed off-line. No further learning is performed next.

Batch learning vs. Big Data

Batch learning is a natural approach when data is limited. Limited availability of the data was and still remains in many problems the key challenge.

Online learning

Online learning does **not** use an entire data set as an input.

Online learner has to match two conditions defined in [3] as follows:

1. has to make predictions about a sequence of instances, one after the other.
2. receives feedback after each prediction.

Other researchers refer to *online learning* in varied cases e.g. when [10]:

- the underlying process providing the data changes
- or volume of data is large
- or data flows continuously

Mini-batches and micro-batches

More recently:

- *Micro-batches* are used to adapt some of ML methods to streaming settings. In Apache Spark, data stream can be divided into chunks of e.g. 10 seconds data. Batch learning can be performed on these chunks. This can be difficult as such data chunks can be too large, too small, not representative.
- *Mini-batches* can be used when large data sets are processed with gradient descent methods. In such cases, small subsets of data can be used in every iteration to drive gradient-based methods.

Selecting subsets of the data

In both cases, a subset of data is used to update an ML model. In the case of a part of a data stream this can be difficult. The question arises whether a data set selected based on ingestion period provides a representative sample for ML purposes.

Stream mining

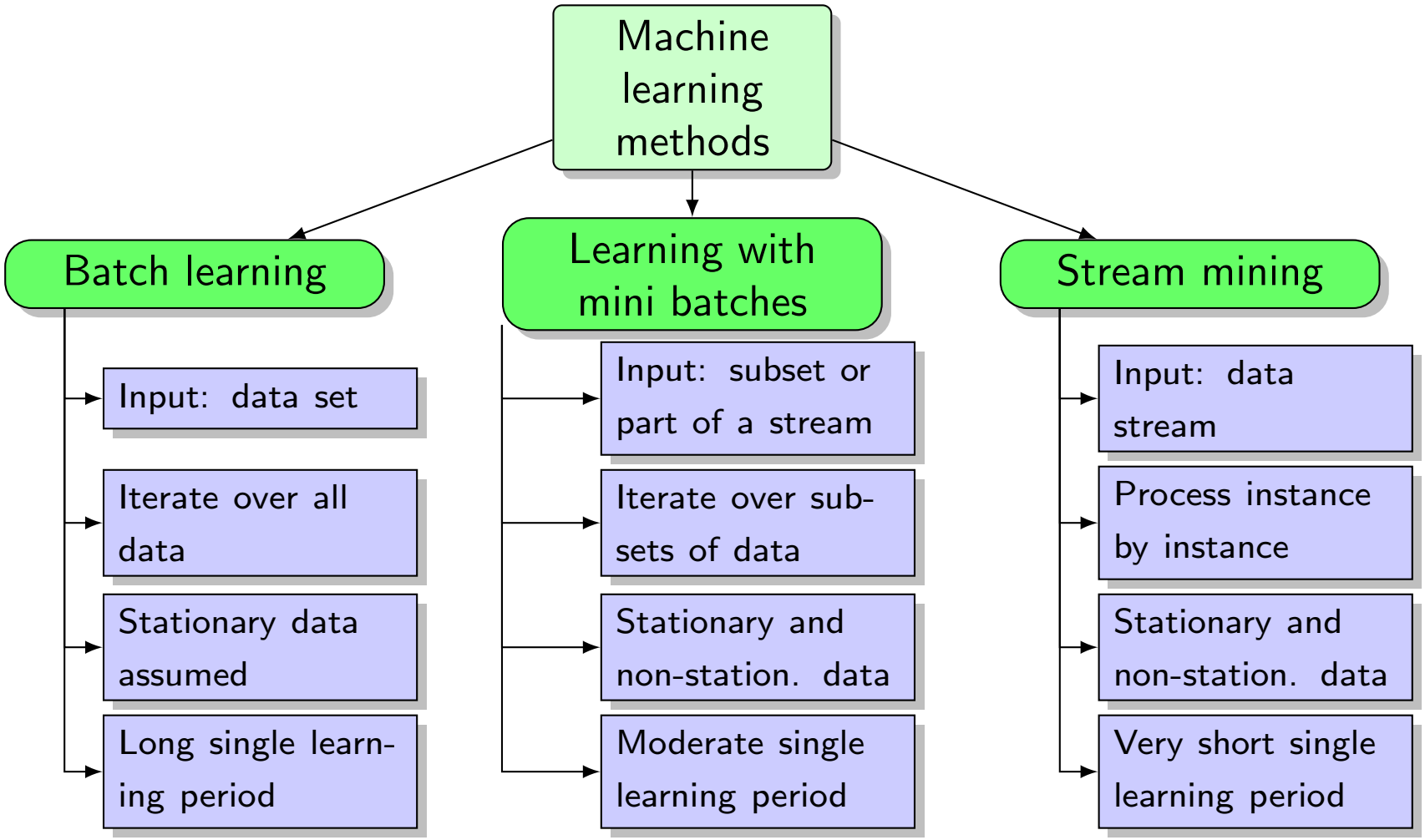
Another approach to model development with ML techniques is *stream mining* [5, 4]. Stream mining methods match the needs of ML performed with data streams, defined as follows [4]:

1. process an instance (at most) once i.e. avoid iterative processing of the same data
2. process every instance within limited amount of time
3. limit memory use, it should be limited irrespective of stream volume
4. enable response such as prediction, clustering or pattern at any time
5. adapt to temporal changes e.g. a change in real classification border

Stream mining vs. online learning

Stream mining is focused on high performance processing of large volume data streams without storing the data. Stream mining relies on underlying online learning methods [2]. Unlike online learning it pays more attention to limited processing overhead.

Batch learning vs. stream mining



Fundamentals of stream mining

Classification and regression

In the remainder of this lecture, we focus on classification and regression.

Under this assumption, a data stream can be defined as $\mathcal{S}_1, \mathcal{S}_2, ...,$
$\mathcal{S}_k = \{(\mathbf{x}_k, y_k)\}, \ldots,$
whereas $\mathbf{x}_k$ is a vector of input feature values, y_k is a true label.

In the simplest case, the objective is to perform for every instance the following steps:

1. predict a label $\tilde{y}_k = h_{k-1}(\mathbf{x}_k)$ with a model $h_{k-1}(\cdot)$ available prior to instance arrival
2. update performance indicators with the pair $(y_k, \tilde{y}_k)$
3. update a model $h_k(\cdot)$ with the new instance $\{(\mathbf{x}_k, y_k)\}$ i.e. $h_k = f(h_{k-1}, \{(\mathbf{x}_k, y_k)\})$

Test-then-train evaluation

Model evaluation performed above is named *test-then-train* evaluation and is most frequently applied, in spite of its obviously limiting assumptions.

Adapting batch learning to stream mining needs

To adapt an ML method for stream mining means to adapt at least the process of constructing a model (training method).

Batch model construction can be very time consuming (or even impossible) with large data sets. During this period, a model is not available for prediction, which is not acceptable from stream mining perspective.

However, prediction with batch-trained models can be frequently made while matching stream mining requirements. Once a model is ready, prediction can be:

- made with limited computational cost (e.g. CART models)
- or made by a parallel model implementation (e.g. random forest)

Traing a model efficiently is the key challenge under stream mining setting.

Adapting batch learning to stream mining needs

Some batch learning algorithms can be easily adapted to make them match stream mining requirements:

- instance-based classifiers e.g. kNN. Example: create an instance set out of $L \in \mathbb{N}, L \gg k$ most recent instances in a data stream
- Naive Bayes classifier: incrementally update posterior probabilities after processing every instance in a data stream
- baseline methods i.e. Majority Class (MC), i.e. predicting most frequently observed so far class label and NoChange - i.e. predicting last seen label i.e. $\tilde{y}_{k+1} = y_k$.

Difficulties in moving from batch to stream mining setting

Not every batch learning algorithm can be easily adapted. A counter example: CART trees. CART algorithm strongly relies on the availability of an entire data set for training.

Key aspects: incremental learning

A stream mining model has to process an instance at most once.

A model has to be constantly available.

Hence, stream mining models are built through incremental updates. Each update has to take a very limited computational effort.

Example: it is enough to drop one (most likely the oldest) instance from the exemplars used by kNN and add a new one (most likely the one that arrived most recently) to update a kNN classifier.

Online learning and incremental updates

Similarly, online models i.e. models developed with online learning are also known as incremental or sequential models [10].

Beyond IID data

As observed in [4], almost all developments in data mining, and particularly classification, assume that data are **independently and identically distributed (IID)**.

Thus, IID data are assumed to come from:

- stationary distribution i.e. distribution that does not change with time
- the distribution randomly producing data in no particular order

In a streaming setting, none of the two assumptions are typically met i.e.:

- we frequently observe labels to be correlated over some periods of time e.g. periods of intrusion attempts [4] or device failures
- and distribution changes over time i.e. virtual (change in $p(x)$) or real change in $p(y|x)$ drift occurs [7].

Most reference data streams well illustrate the challenges of stream mining.

Key aspects: concept drift

In line with [7], let P stand for a process, which provides a data stream $\{(\mathbf{x}_t, y_t)\}$ sampled from unknown distribution $p_t(\mathbf{x}, y)$.

For some processes, the probability $p_t(y|\mathbf{x})$ changes at some period(s). This means learning is performed in nonstationary environment and *concept drift* occurs.

Two cases can be distinguished [7]:

- *real drift* i.e. when $p_t(y|\mathbf{x})$ varies over time
- *virtual drift* i.e. when $p_t(\mathbf{x})$ varies over time, but without changing the probability of classes $p_t(y|\mathbf{x})$

Further categories

When abrupt concept drift occurs, it is also referred to as *concept change*. Furthermore, drifts can be permanent or transient.

Sample reference real data streams

Real data streams

Two categories of streams belong to this group:

- streams based on data sets in which no natural sequence of instances was present
- streams based on some time series. In these streams the sequence of instances reflects the sequence of some real events

Discussion

Can any data set be treated as a data stream? If so, can virtual drift occur in such a data stream? Can real drift happen?

Sample real data streams

Table: An overview of real data streams. #attrib - incl. label feature, (MOA) - `https://moa.cms.waikato.ac.nz/datasets/`, (UCI) - `https://archive.ics.uci.edu/`

Data	*#inst.*	*#attrib.*	#class	Task
Electricity	45,312	9	2	Based on data from Australian New South Wales Electricity Market predict price change (UP or DOWN) compared to a moving average of the last 24 hours. (MOA)
Airlines	539,383	8	2	Predict whether a flight will be delayed (1) or not (0) based on departure data. (MOA).
CovType	581,012	55	7	Also known as Forest Covertype. Based on data from US Forest Service Region 2 Resource Information System data such as slope and soil type. The task is to predict forest cover type from cartographic variables only. (UCI)

Sample real data stream - Electricity

The data set contains 45,312 instances. It has been described in detail in [8] and is frequently used in normalised form.

Each example refers to one 30-minute time slot of a day. The data covers the period from 7 May 1996 to 5 December 1998. The instances contain:

- `Date`: date between 7 May 1996 to 5 December 1998.
- `Day`: day of the week (1-7)
- `Period`: time slot index between 1 and 48
- `NSWprice`: New South Wales electricity price
- `NSWdemand`: New South Wales electricity demand
- `VICprice`: Victoria electricity price
- `VICdemand`: Victoria electricity demand
- `transfer`: scheduled electricity transfer between both states

The label (`class` feature) UP or DOWN indicates price change related to a moving average of the last 24 hours.

Larger real data streams

Table: An overview of real data streams. #attrib - incl. label feature, (UCI) - `https://archive.ics.uci.edu/`

Data	*#inst.*	*#attrib.*	#class	Task
Poker-hand	1,000,000	10		One instances is one example of a hand consisting of five playing cards drawn from a standard deck of 52 cards. The objective is to predict "Poker hand". (MOA)
AWS Prices	27,410,309	7	36,903	Amazon makes it possible to bid on spare server capacity. The task is to predict spot prices based e.g. on time and region. Avail. at `https://www.openml.org/d/41424`

Drift in real data streams

As pointed out in [6] in the context of inter alia CovType, airlines and electricity data, a common assumption for these data streams is that it is not possible to unequivocally state when drift occurs or if there is any drift.

Sample synthetic data stream generators

Introduction

Real data streams are available as data files.

This is not the case for synthetic ones.

Synthetic data streams are generated online by generators available in projects such as `MOA` and `river`.

This is because the generators can be parameterised to configure:

- the number of instances to be generated
- whether concept drift should occur and if so when, for how long and of what magnitude

Synthetic data stream vs. generator

In fact, one generator can generate a number of different streams following some pattern. Hence, mentioning just stream types is not enough to fully define the stream. **Generator is not a synonym for a data stream.**

Hyperplane data

	Description
Idea	Predict one of two classes separated by a hyperplane in d dimensional space. No real-life interpretation
Details	A hyperplane in d-dimensional space is the set of points matching condition $\sum_{i=1}^{d} w_i x_i = w_0$. Examples $\mathbf{x}$ above the hyperplane i.e. such that $\sum_{i=1}^{d} w_i x_i > w_0$ are labelled positive; remaining examples belong to negative class. By setting generator parameters we can change gradually the weights $(w_i := w_i \pm \delta)$ and rotate the plane
# features	d, which is the parameter set when initialising the generator
# classes	2
Key other parameters	• The number of attributes with drift • Magnitude of the change δ for every example • Percentage of noise (`noise_percentage`) to add to the data i.e. the percentage of instances for which class will be changed to the other one • Probability that the direction of change is reversed
Drift	Real drift possible. Depends on how the parameters are set.
Source and further details	[4], `https://riverml.xyz/0.11.0/api/synth/Hyperplane/`. Note that `river` implementation generates no noise for `noise_percentage`≤ 0.01

Generatic synthetic data - hyperplane generator

```
1
2 from river.datasets import synth
3
4 dataset = synth.Hyperplane(seed=42, n_features=2)
5
6 for x, y in dataset.take(10):
7     print( round(x[0],5), round(x[1],5),y)
```

We generate two-dimensional data (`n_features=2`) here and poll first 10 instances from the generator. What seed is used for?

```
1
2 0.73199 0.59866 1
3 0.86618 0.60112 1
4 0.83244 0.21234 0
5 0.52476 0.43195 0
6 0.29214 0.36636 0
7 0.51423 0.59241 1
8 0.06505 0.94889 1
9 0.09767 0.68423 1
10 0.03439 0.90932 1
11 0.52007 0.54671 1
```

Training classifiers with hyperplane data

```
1  from river import stream
2  from river import tree
3  from river.datasets import synth
4  from river import metrics
5
6  generator = synth.Hyperplane(seed=42, n_features=2,n_drift_features=1,mag_change
       =0.0,noise_percentage=0.05)
7  myStream = generator.take(10000)
8  metric = metrics.ClassificationReport()
9
10 # Hoeffding tree i.e. one of stream mining methods
11 model = tree.HoeffdingTreeClassifier()
12
13 # test-then-train evaluation
14 for x, y in myStream:
15     y_pred = model.predict_one(x)
16     model.learn_one(x, y)
17     if y_pred is not None:
18         metric.update(y, y_pred)
19
20 print(metric)
```

Metric displays values of accuracy and other performance measures. This code relies on the `river` library, which we will discuss during the labs.

Hyperplane data - no changes, different noise levels

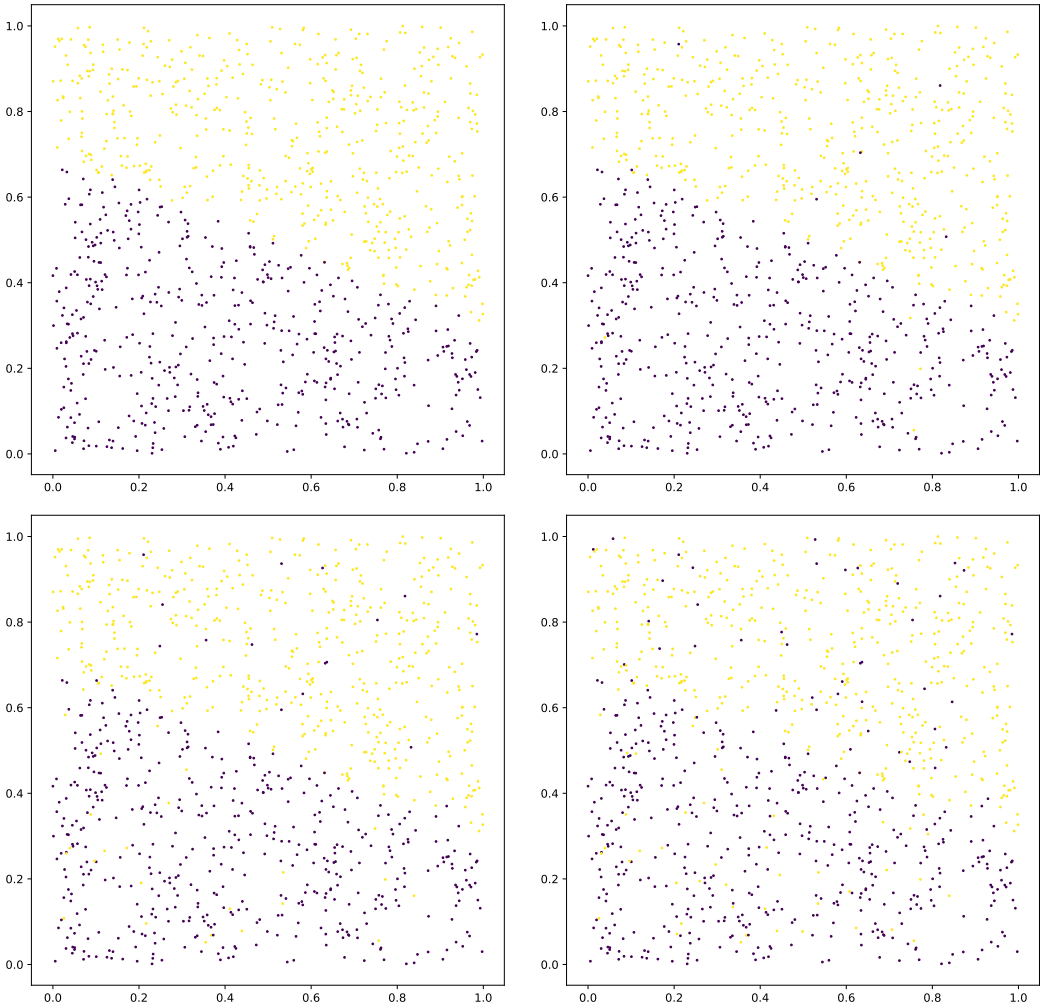


Figure: Sample hyperplane data. No changes in hyperplane parameters. First 1000 instances on a plane. Noise levels: 0.0 (top left), 0.02 (top right), 0.05 (bottom left), 0.1 (bottom right)

Results - no changes, different noise levels

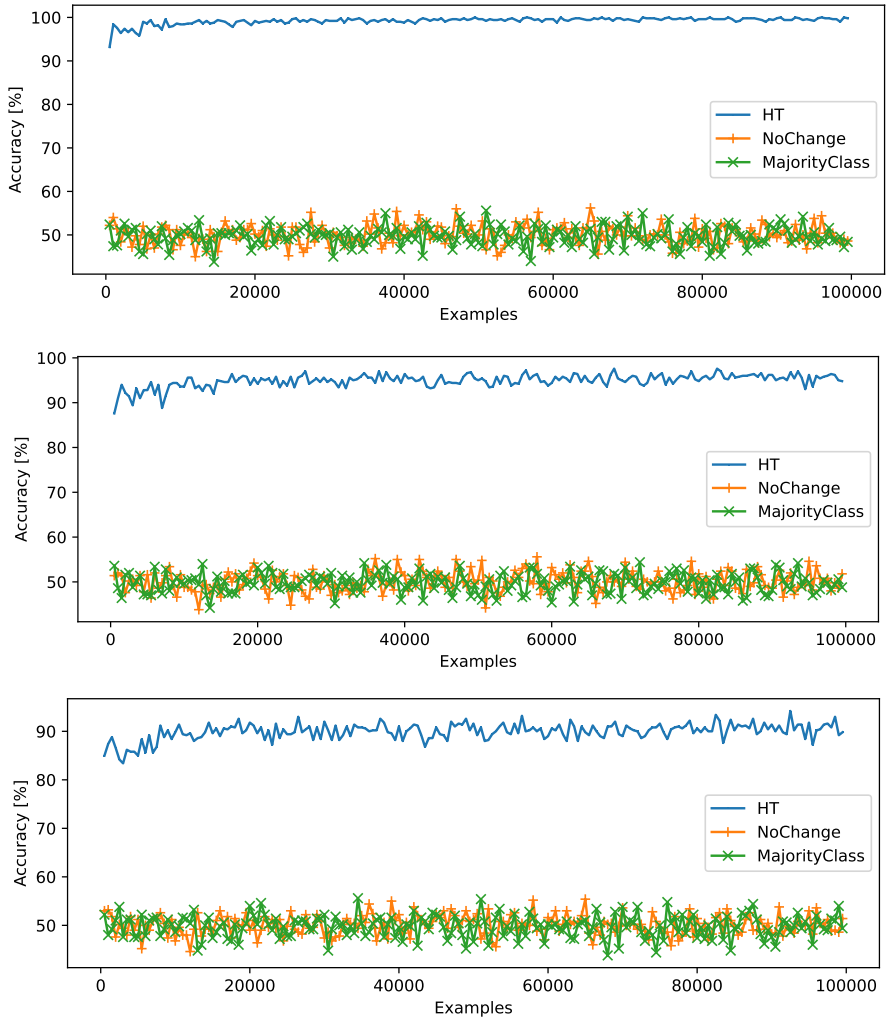


Figure: Sliding window accuracy for hyperplane data. Noise levels: 0.0 (top), 0.05 (middle), 0.1 (bottom)

Results - putting it together

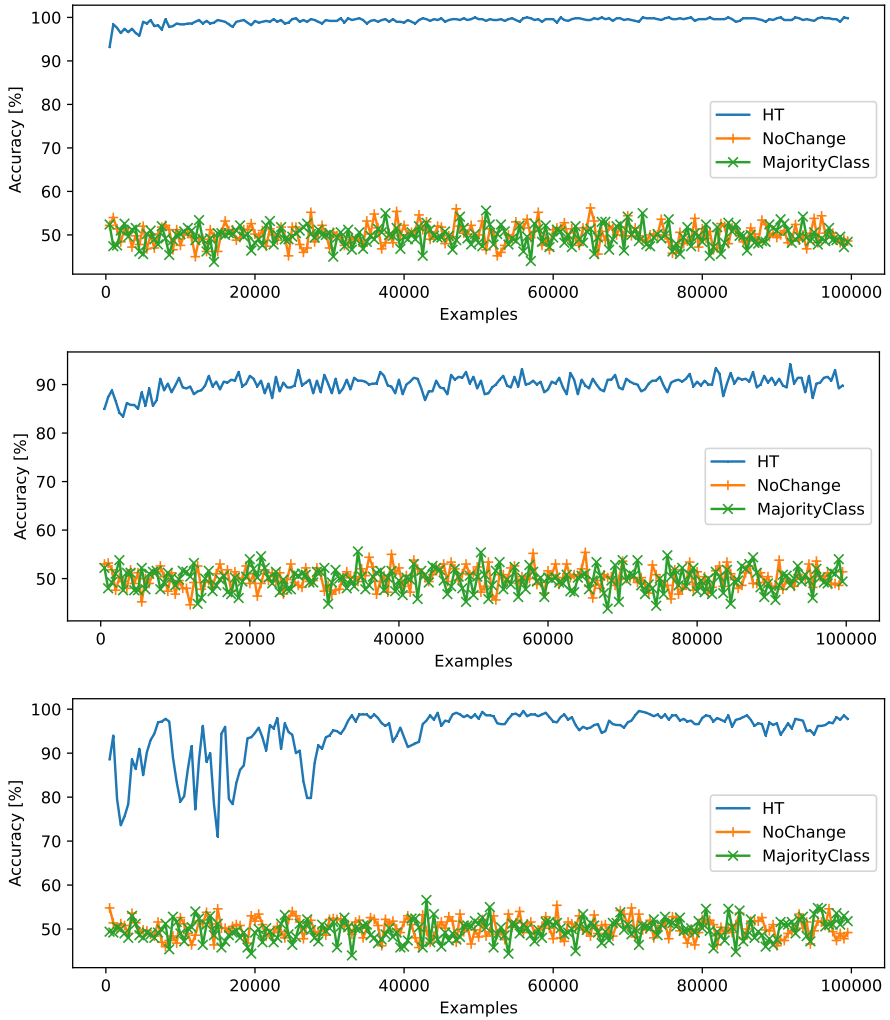


Figure: Sliding window accuracy for hyperplane data. No noise and change (top), no change and noise set to 0.1 (middle), change set to 0.1 and no noise (bottom)

Summary

Processing data streams rather than data sets requires new solutions and methods

This is true also for machine learning methods

Stream mining is the term used to describe machine learning methods for data streams

Stream mining models are evaluated with real and synthetic data which necessitate model updates

In the next step, we will focus on a sample machine learning method for online learning with data streams

References I

[1] Batch learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 98–99. Springer US, Boston, MA, 2017.

[2] Stream mining. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 1199–1200. Springer US, Boston, MA, 2017.

[3] Peter Auer. Online learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 929–937. Springer US, Boston, MA, 2017.

[4] 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/.

[5] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. *MOA Data Stream Mining - A Practical Approach*. Centre for Open Software Innovation COSI, 2011.

References II

[6] Dariusz Brzezinski and Jerzy Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, 2014.

[7] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[8] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence – SBIA 2004*, pages 286–295, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[9] Gartner. *Big Data definition*. Gartner, 2017.

[10] Beatriz Pérez-Sánchez, Oscar Fontenla-Romero, and Bertha Guijarro-Berdiñas. A review of adaptive online learning for artificial neural networks. *Artificial Intelligence Review*, 49(2):281–299, Feb 2018.