ECML - PKDD 2023

Politecnico di Torino & OGR, Torino, Italy

Opportunity and challenges in online deep learning

https://github.com/lucasczz/deep-river-demo-23

Cedric Kulbach
FZI, Karlsruhe, Germany

Lucas Cazzonelli
FZI, Karlsruhe, Germany

Hoang-Anh Ngo
AI Institute, University of Waikato

Minh-Huong Le-Nguyen
LCTI, Télécom Paris, IP Paris

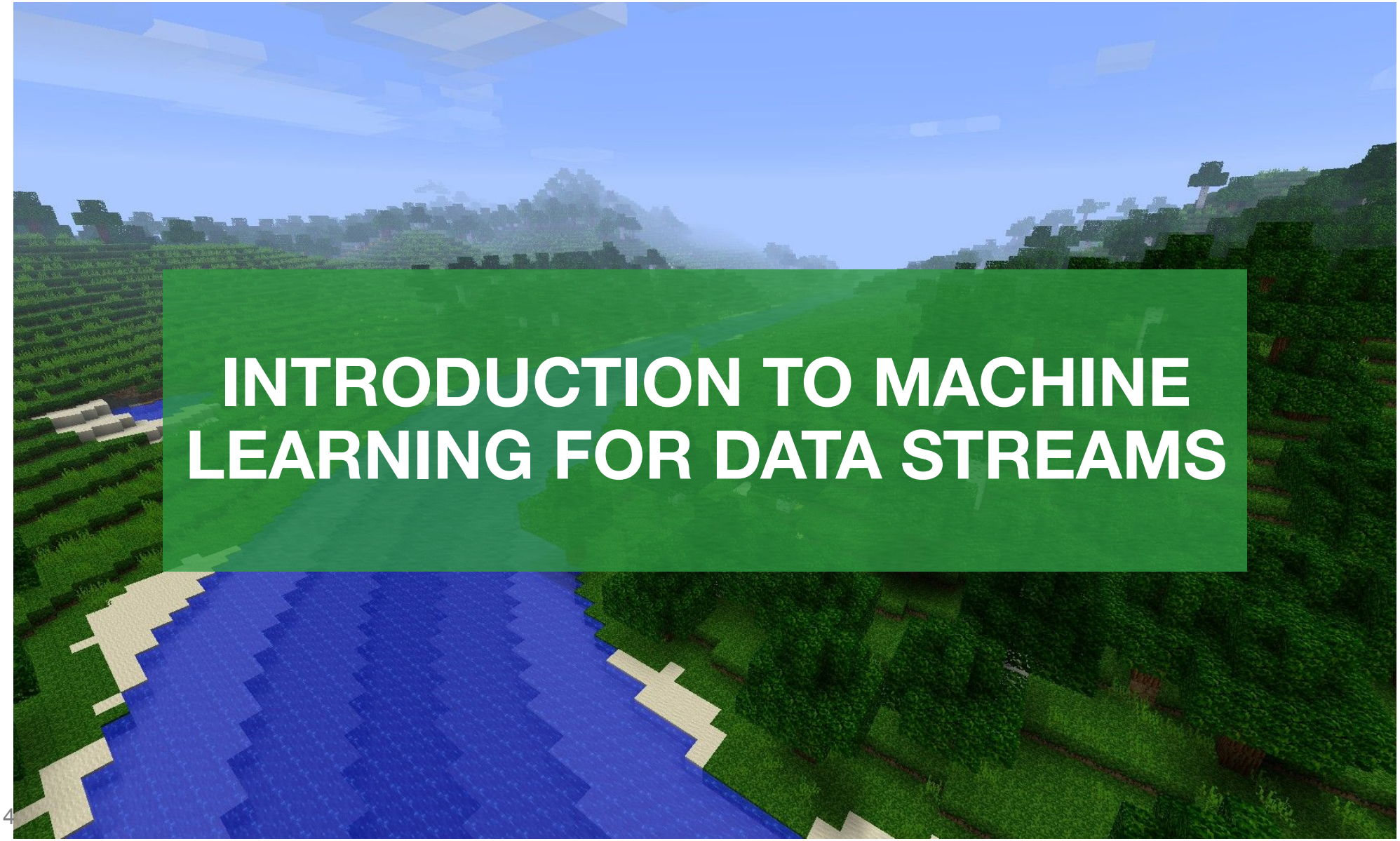
Albert Bifet
LCTI, Télécom Paris, IP Paris
AI Institute, University of Waikato

Outline (First Part)

Online Machine Learning in Practice

1. Introduction to ML for data streams
2. State-of-the-arts algorithms in online ML
3. How to induce fairness in online ML?
4. Introduction to **River**
5. Practical examples

Tutorial



INTRODUCTION TO MACHINE LEARNING FOR DATA STREAMS

GETTY

Artificial intelligence / Machine learning

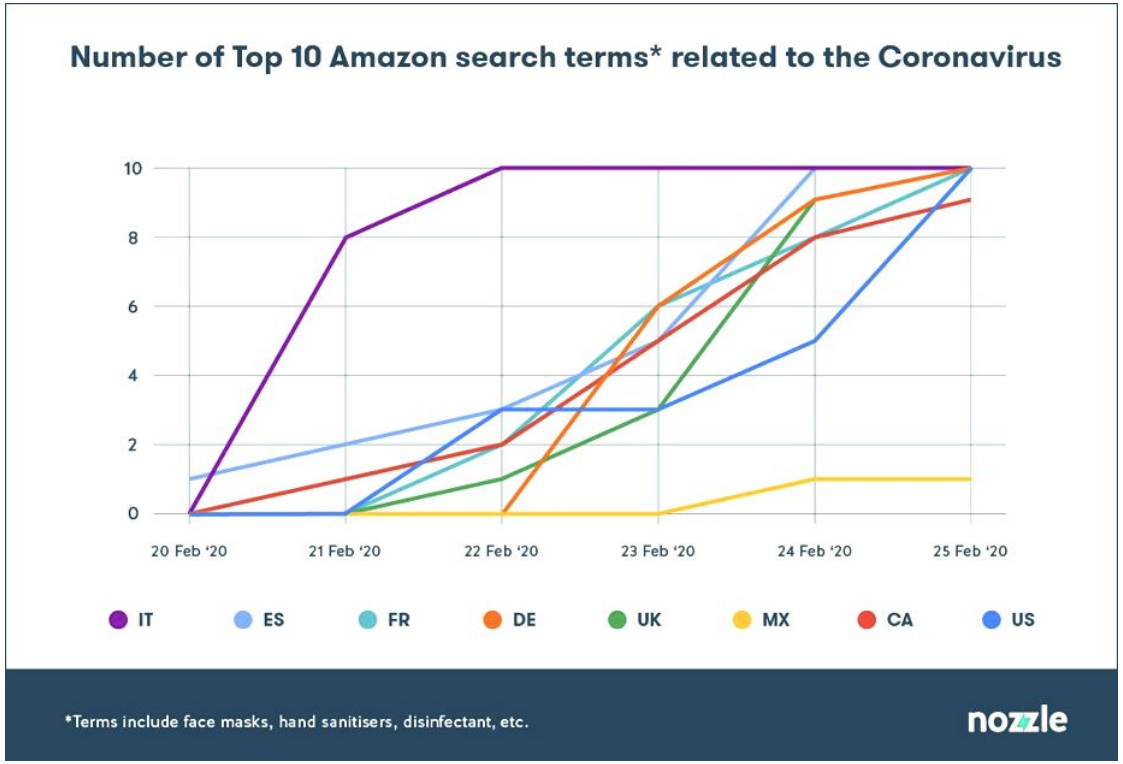
Our weird behavior during the pandemic is messing with AI models

Machine-learning models trained on normal behavior are showing cracks — forcing humans to step in to set them straight.

by **Will Douglas Heaven** May 11, 2020

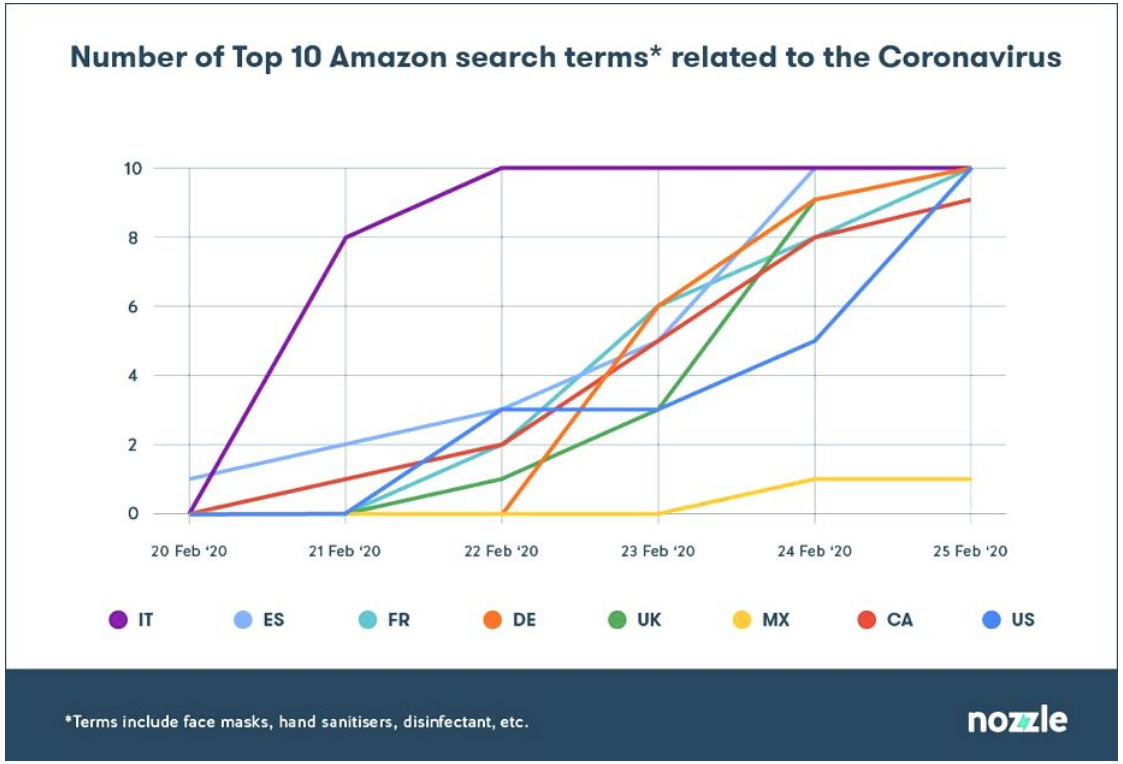
— EXPOSED WEAKNESSES OF ML MODELS DURING THE COVID-19 PANDEMIC

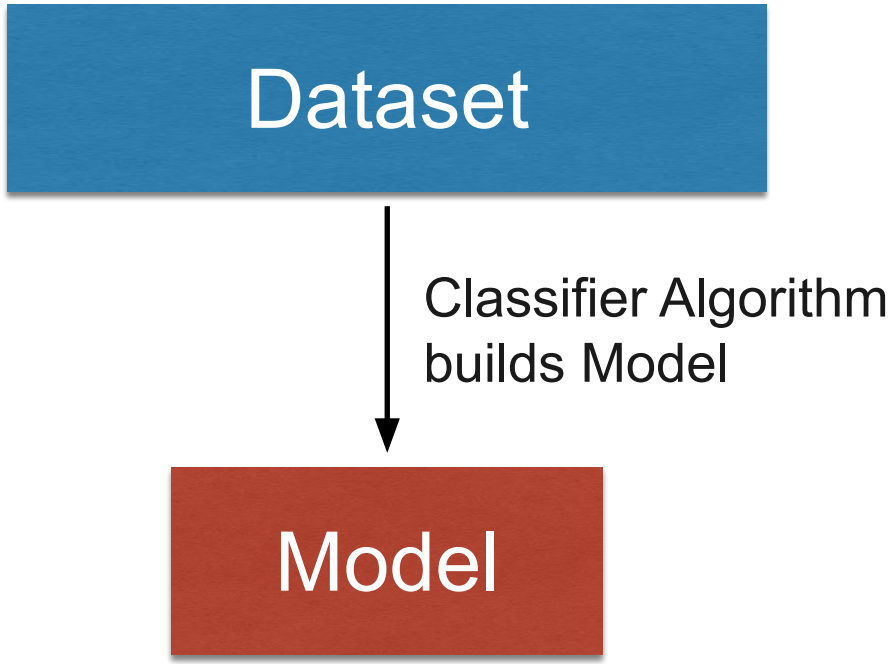
- "Normal has changed": less than a week is needed for top 10 search terms to be filled with COVID-19-related products, causing hiccups for models running behind-the-scene.
- Machine learning models perform badly when input data are too different from trained data.



— EXPOSED WEAKNESSES OF ML MODELS DURING THE COVID-19 PANDEMIC

- The pandemic situation is so “volatile”, yet if “A machine learning system doesn’t see what it’s expecting to see, then you will have problems.”
- Human input is extremely needed.





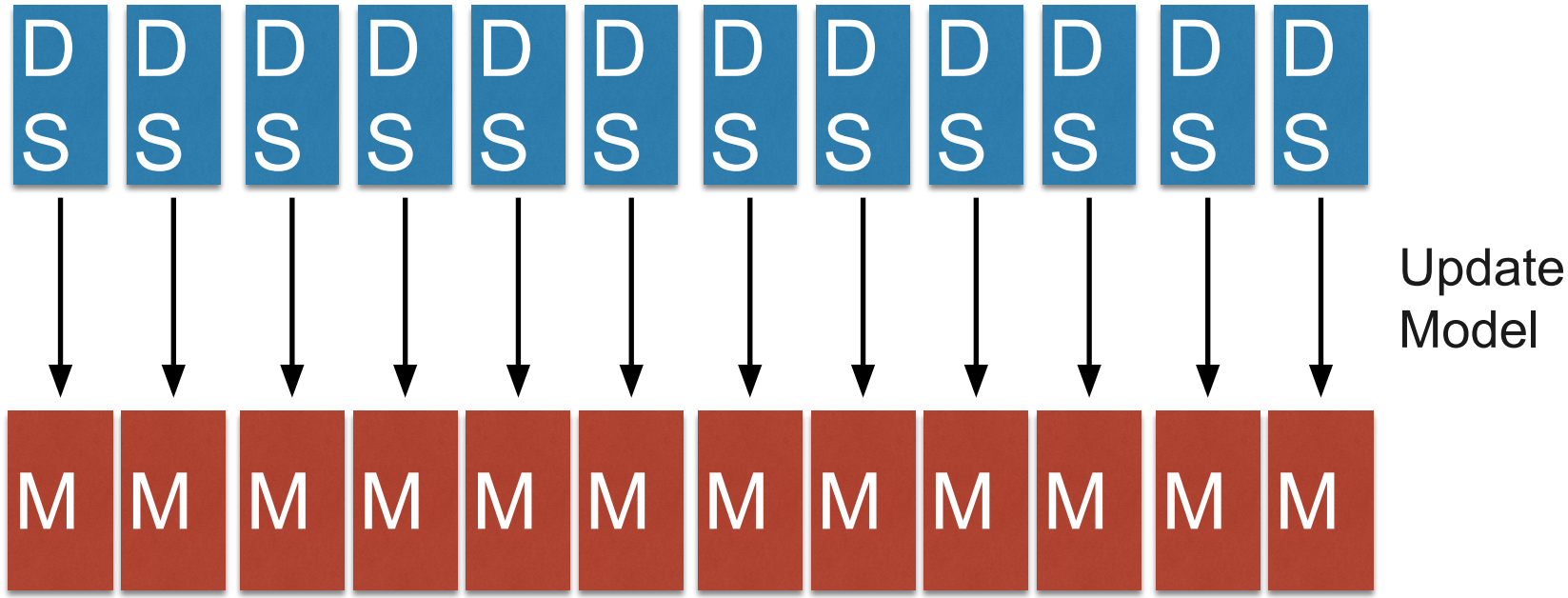
— ANALYTIC STANDARD APPROACH

- Finite training sets
- Static models

Problem: how often and how much data to use to build the model

— ML STANDARD APPROACH

- Finite training sets
  Static models



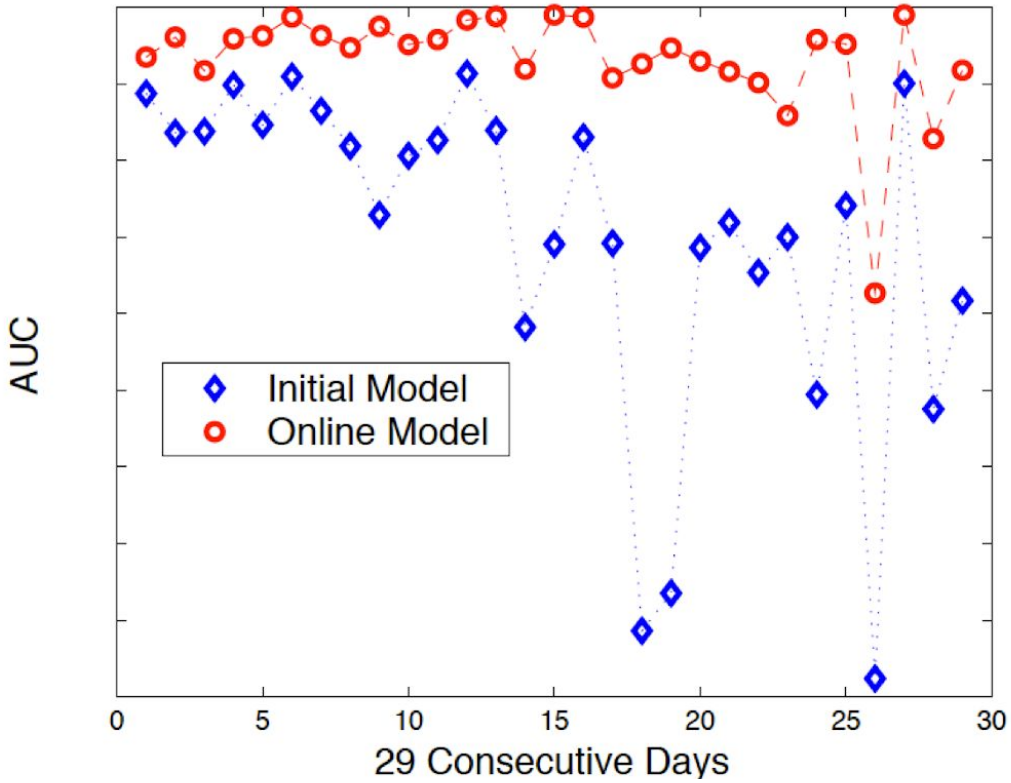
— DATA STREAM APPROACH

- Infinite training sets
- Dynamic models

*DS: Data set, M: Model

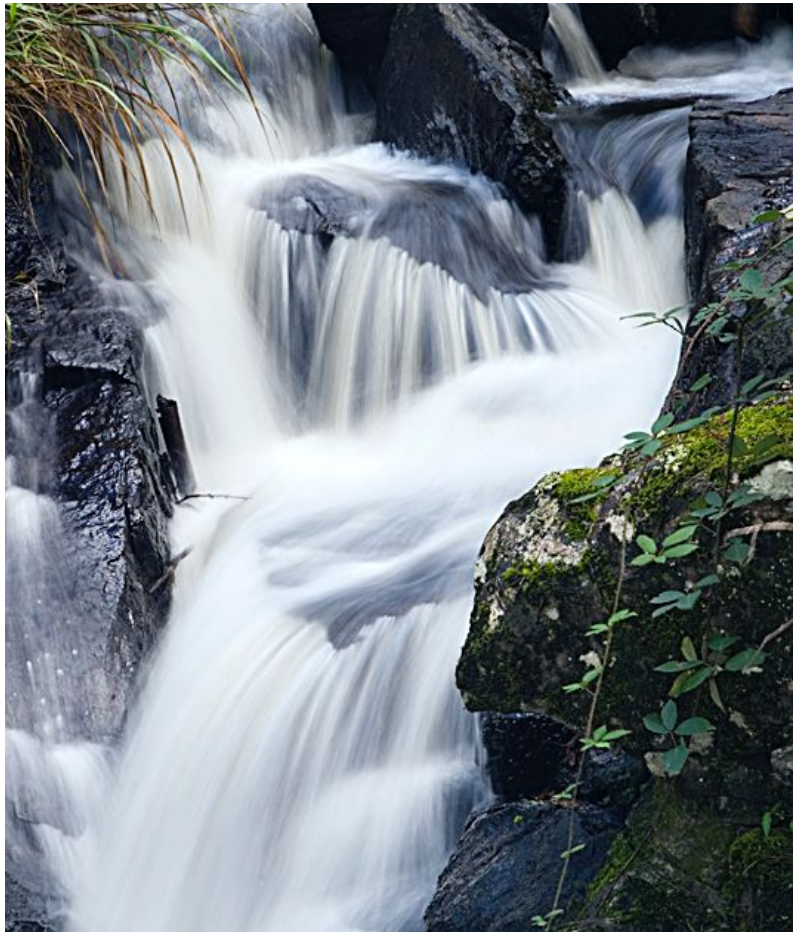
— PAIN POINTS

- Need to **retrain!**
  - Things change over time
  - How often?
- Data unused until next update!
  - Value of data wasted



STREAM MINING

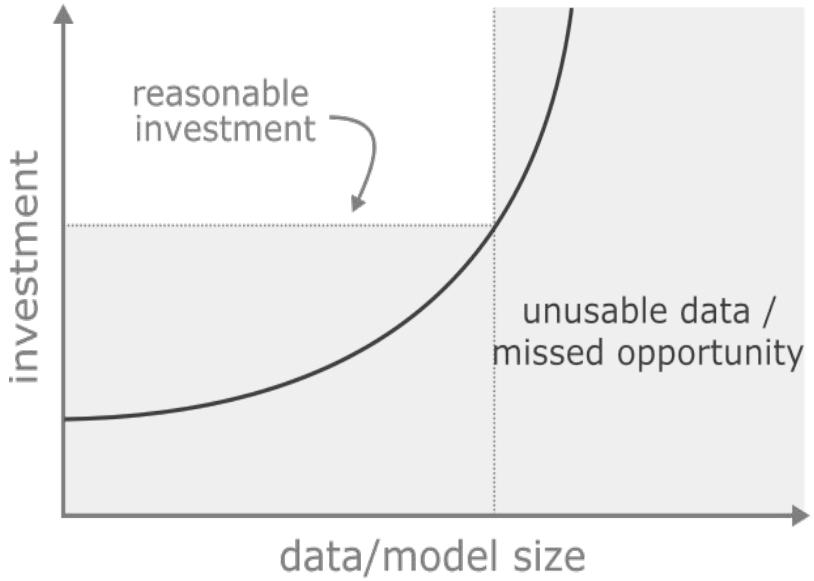
- Maintain models online
  - Incorporate data on the fly
  - Unbounded training sets
  - Resource efficient
  - Detect changes and adapts
  - Dynamic models



— INVESTMENT VS DATA SIZE

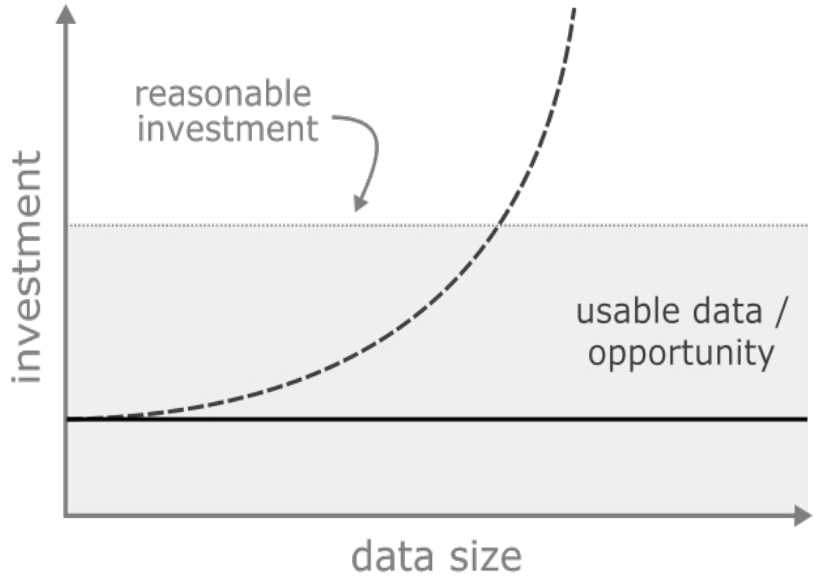
Batch learning

Often struggles to maintain investment (time, memory, cost) below reasonable level



Stream learning

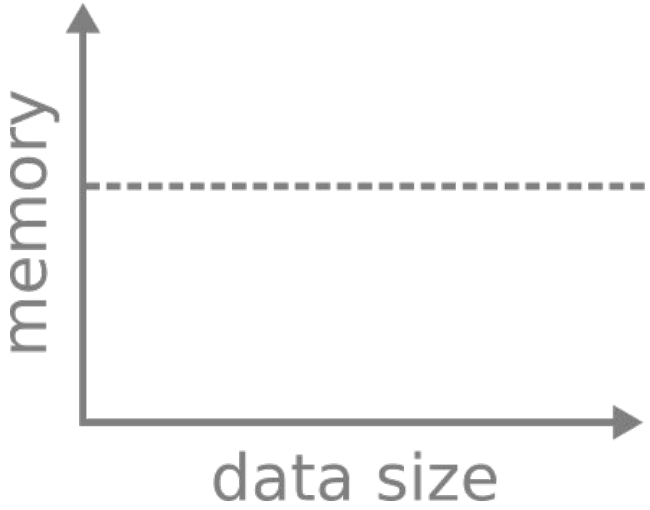
Efficiently generates incremental models from data streams



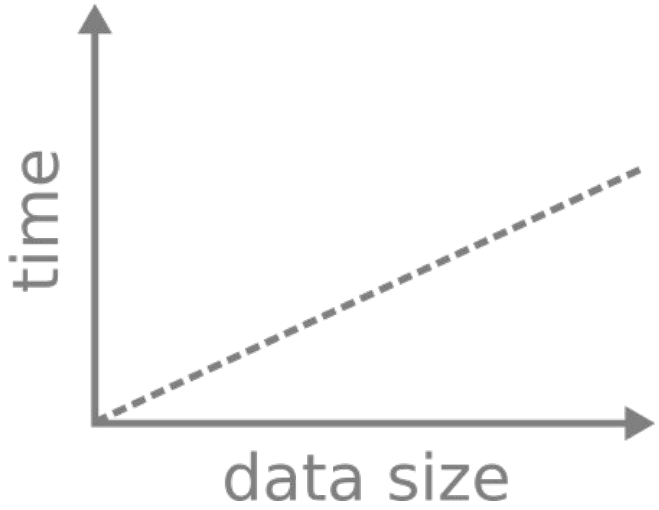
Source: AWS, 2018

— RESOURCES

Stream learning upper limits



constant memory



sub-linear processing time (training)

— REQUIREMENTS

Process one sample at a time, and inspect it only once

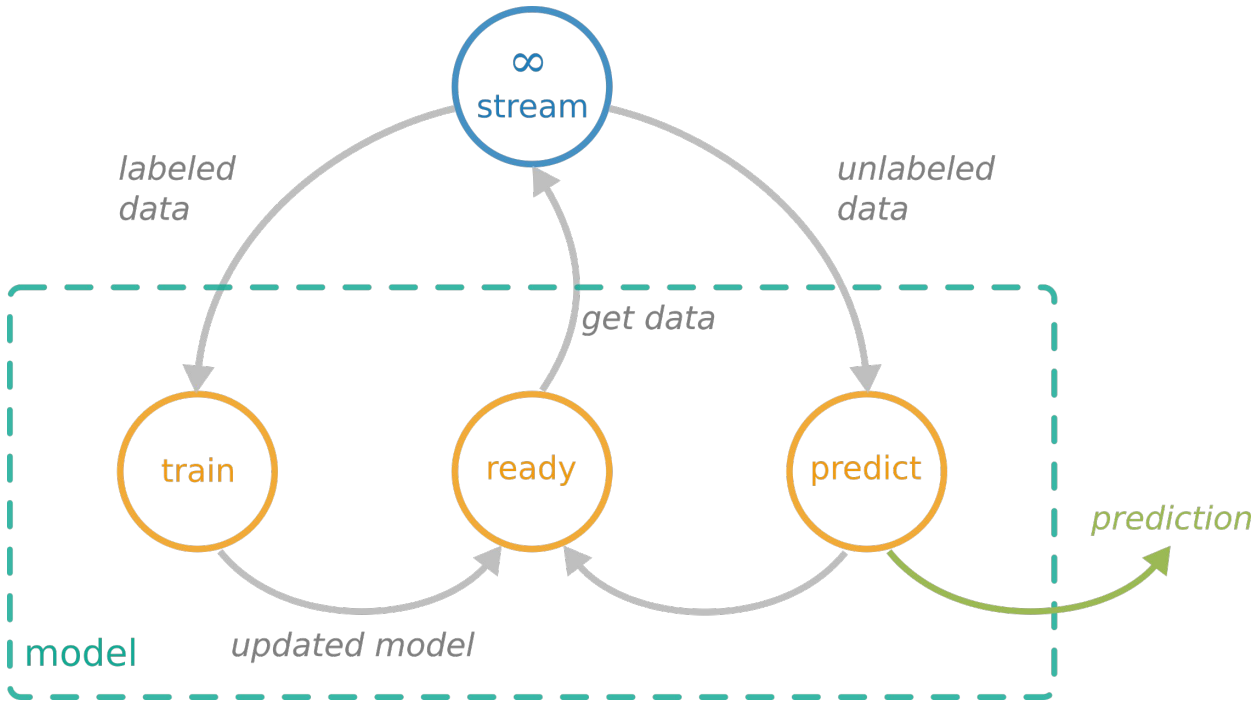
Use a limited amount of memory

Work in a limited amount of time

Always ready to predict

LEARNING FROM DATA STREAMS

Supervised learning



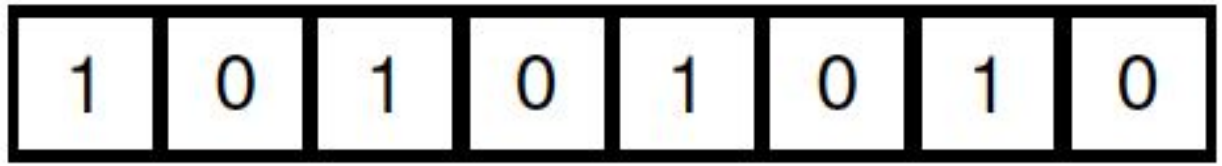
- One sample at a time
- Limited resources
- Predict at any time

— APPROXIMATION ALGORITHMS

- General idea, good for streaming algorithms
- Small error ϵ with high probability $1 - \delta$
  - True hypothesis H, and learned hypothesis $\widehat{\mathrm{H}}$
  - $\Pr\left[\left|\mathrm{H} - \widehat{\mathrm{H}}\right| < \epsilon|\mathrm{H}|\right] > 1 - \delta$

— APPROXIMATION ALGORITHMS

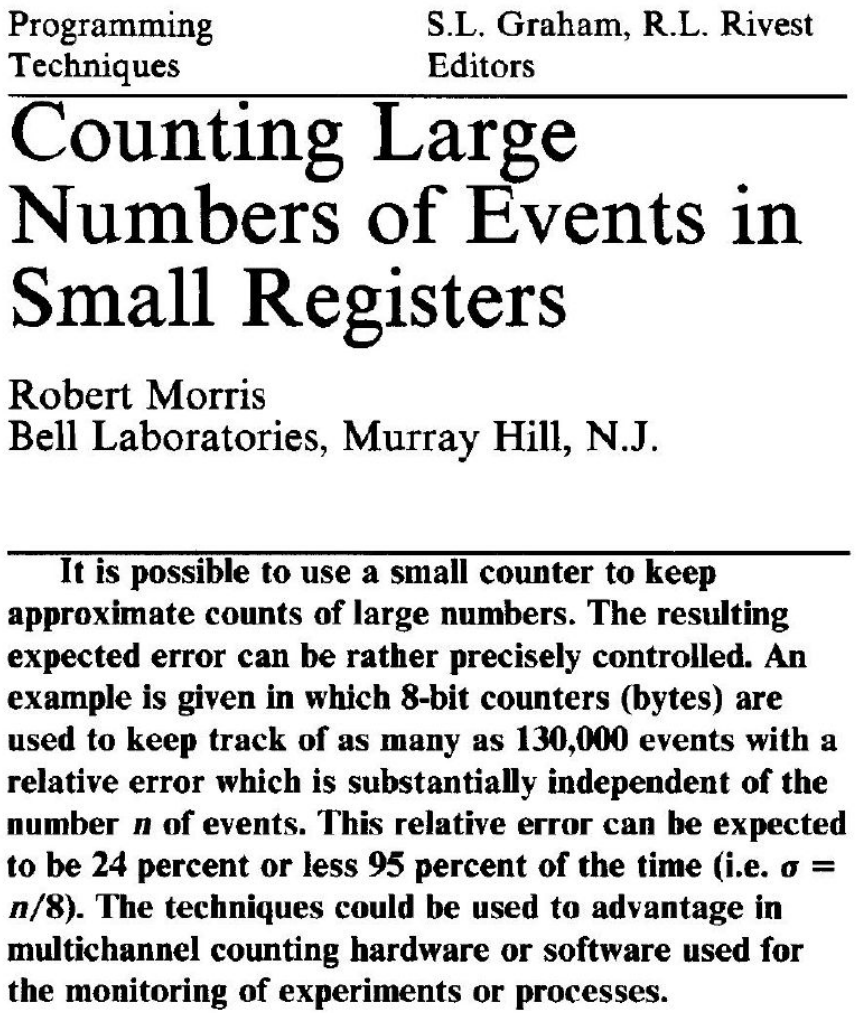
- What is the largest number that we can store in 8 bits?



— APPROXIMATION ALGORITHMS

- What is the largest number that we can store in 8 bits?

Robert Morris. 1978. *Counting large number of events in small registers*. **Communications of the ACM**, 21, 10 (1978), 840-842. https://doi.org/10.1145/359619.359627



Programming Techniques
S.L. Graham, R.L. Rivest Editors

Counting Large Numbers of Events in Small Registers

Robert Morris
Bell Laboratories, Murray Hill, N.J.

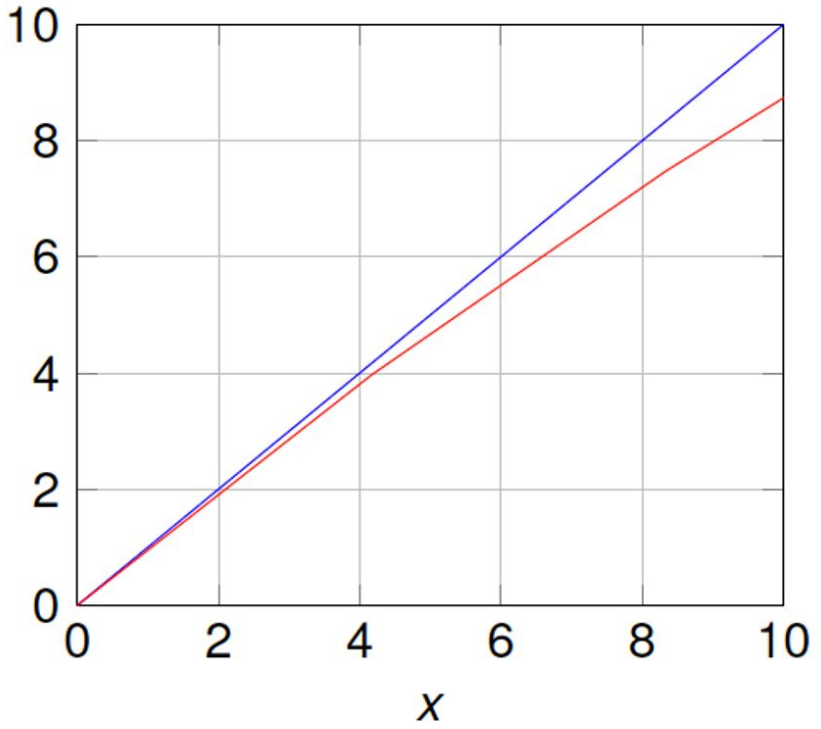
It is possible to use a small counter to keep approximate counts of large numbers. The resulting expected error can be rather precisely controlled. An example is given in which 8-bit counters (bytes) are used to keep track of as many as 130,000 events with a relative error which is substantially independent of the number n of events. This relative error can be expected to be 24 percent or less 95 percent of the time (i.e. $\sigma = n/8$). The techniques could be used to advantage in multichannel counting hardware or software used for the monitoring of experiments or processes.

largest

APPROXIMATION ALGORITHMS

- What is the largest number that we can store in 8 bits?

$$f(x) = \log(1 + x/30)/\log(1 + 1/30)$$

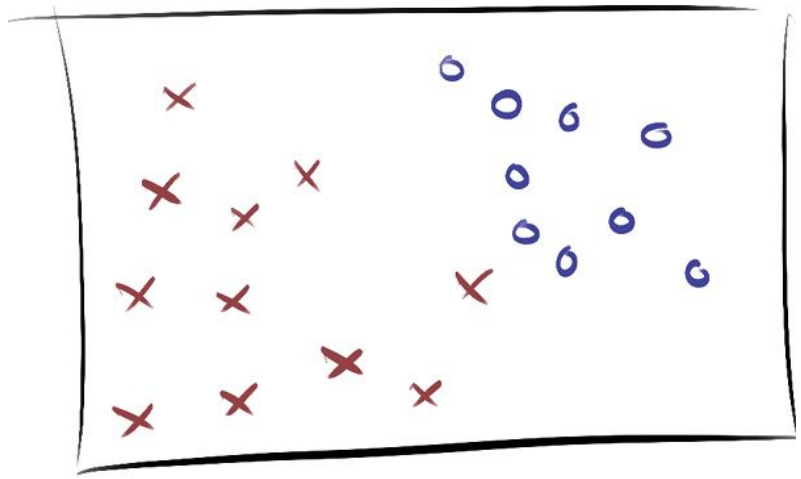


$$f(0) = 0, f(1) = 1$$

— CLASSIFICATION

— DEFINITION

Given a set of training examples belonging to $\mathbf{n_c}$ different classes, a classifier algorithm builds a model that predicts for every unlabeled instance x the class C to which it belongs



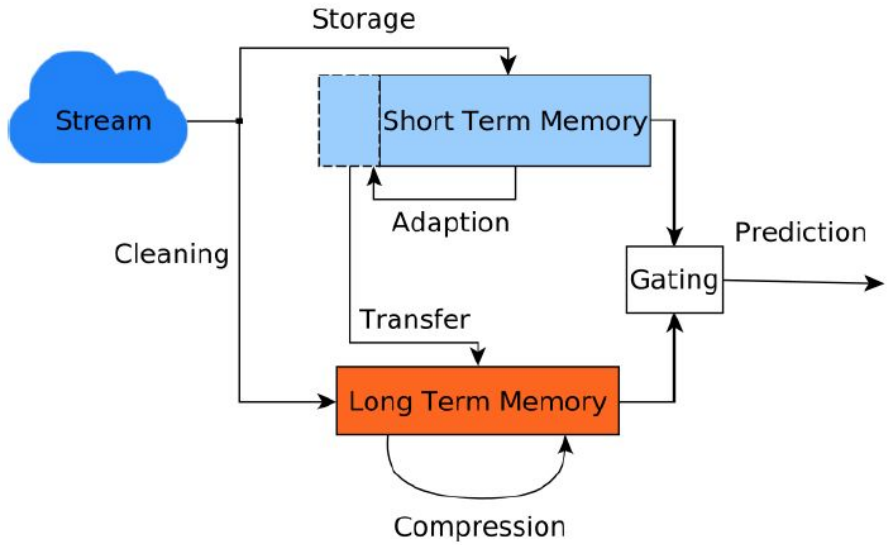
EXAMPLES

- Email spam filter
- Twitter sentiment analyzer

Photo: Stephen Merity http://smerity.com

SAM-kNN

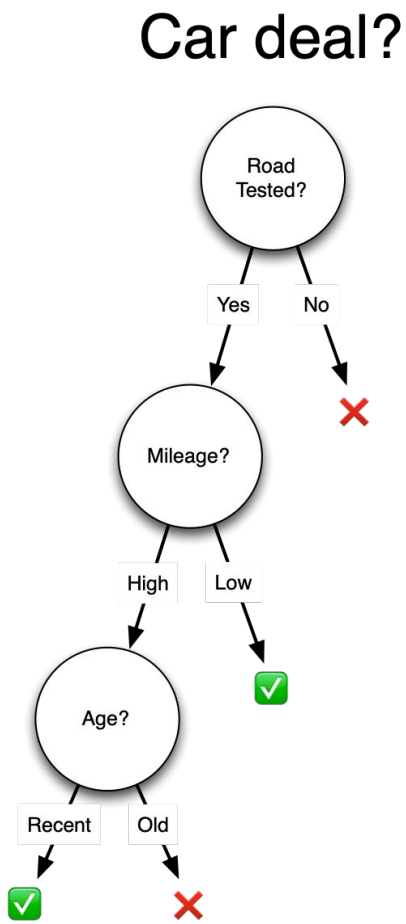
Viktor Losing, Barbara Hammer, and Heiko Wersing. 2016. **KNN Classifier with Self Adjusting Memory for Heteogenerous Concept Drift. In:** *Proceedings of the 16th International Conference on Data Mining (ICDM)*, (Barcelona, Spain), 291-300. https://doi.org/10.1109/ICDM.2016.0040



Received **Best Paper Awards**.

DECISION TREE

- Each node tests a features
- Each branch represents a value
- Each leaf assigns a class
- Greedy recursive induction
  - Sort all examples through tree
  - X_i = most discriminative attribute
  - New node for x_i, new branch for each value, leaf assigns majority class
  - Stop if no error | limit on number of instances



Hoeffding Tree

Pedro Domingos and Geoff Hulten. 2000. **Mining high-speed data stream.** In: *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Boston, Massachusetts, USA), Association for Computing Machinery, New York, NY, USA, 71-80. https://doi.org/10.1145/347090.347107

- Sample of stream enough for near optimal decision
- Estimate merit of alternatives from prefix of stream
- Choose sample size based on statistical principles
- When to expand a leaf?
  - Let x_1 be the most informative attribute, x_2 the second most informative one
  - **Hoeffding bound:** split if $G(x_1) - G(x_2) > \epsilon = 2 \times \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}}$ (*)

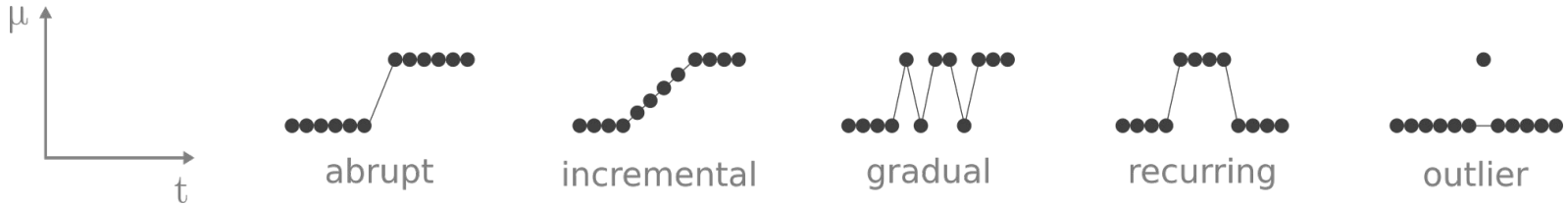
(*) This bound has been corrected in a recent paper: Pawel Matuszyk, Georg Krempl, and Myra Spiliopoulou. **Correcting the Usage of the Hoeffding Inequality in Stream Mining**. In: *IDA 2013: Advances in Intelligent Data Analysis*, 298-309. https://doi.org/10.1007/978-3-642-41398-8_26

CONCEPT DRIFT

Joao Gama and Indre Zliobaite, Albert Bifet, Mikola Pechenizkiy and Abdelhamid Bouchachia. **A survey on Concept Drift Detection**. *ACM Computing Surveys* 46, 4 (2014), 1-37. https://doi.org/10.1145/2523813

In dynamic and non-stationary environments, the data distribution can change over time.

- **Change detector:** Given an input sequence, raise an alarm signal at instant t if there is a change
- **Application:** Detect changes in model performance



ADAPTIVE RANDOM FOREST

- Why Random Forests?
  - Off-the-shelf learner
  - Good learning performance

Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes & Talel Abdessalem. 2017. **Adaptive random forests for evolving data streams classification.** *Machine Learning* 106 (2017), 1469-1495. https://doi.org/10.1007/s10994-017-5642-8

- Based on the original Random Forest by Leo Breiman

Leo Breiman. 2001. **Random Forests**. *Machine Learning* 45 (2001), 5-32. https://doi.org/10.1023/A:1010933404324

— ADAPTIVE RANDOM FOREST

1. Leveraging bagging

2. Random Hoeffding Trees
   - Random subsets of features for splits
   - Independent (can train in parallel)

3. Adaptive Learning
   - 1 drift and 1 warning detector per tree
   - Train trees in the background before adding them

— STREAMING RANDOM BATCHES

x_1	x_2	x_3	x_4	$x_{...}$	x_m
$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$v_{1,4}$	$v_{1,5}$	$v_{1,6}$
$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$v_{2,4}$	$v_{2,5}$	$v_{2,6}$
$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$v_{3,4}$	$v_{3,5}$	$v_{3,6}$
$v_{4,1}$	$v_{4,2}$	$v_{4,3}$	$v_{4,4}$	$v_{4,5}$	$v_{4,6}$
$v_{5,1}$	$v_{5,2}$	$v_{5,3}$	$v_{5,4}$	$v_{5,5}$	$v_{5,6}$
$v_{6,1}$	$v_{6,2}$	$v_{6,3}$	$v_{6,4}$	$v_{6,5}$	$v_{6,6}$
$v_{...,1}$	$v_{...,2}$	$v_{...,3}$	$v_{...,4}$	$v_{...,5}$	$v_{...,6}$

Heitor Murilo Gomes, Albert Bifet, and Jesse Read. **Streaming Random Patches for Evolving Data Stream Classification**. In: *Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM)*, 240-249. https://doi.org/10.1109/ICDM.2019.00034

— STREAMING RANDOM BATCHES

TABLE II: Test-then-train accuracy (%) using $n = 100$ base models. Underlined results means the performance increased in comparison to $n = 10$ version. BAG and LB did not finish execution for SPAM dataset.

Data set	*LB*	*OAUE*	*DWM*	*OB*	*ARF*	**SRP**	*SRS*	*BAG*
LED(A)	73.953	73.393	73.958	72.475	73.96	74.027	**74.04**	73.975
LED(G)	73.225	72.582	73.031	72.117	73.094	**73.233**	73.179	73.215
AGR(A)	88.717	90.164	88.299	90.374	87.929	**92.869**	92.807	86.663
AGR(G)	83.713	85.244	79.437	87.834	82.288	89.651	**90.259**	82.52
RBF(M)	84.338	84.262	60.977	74.514	**86.958**	86.039	84.821	86.671
RBF(F)	76.771	57.147	54.531	48.698	76.291	76.375	61.622	**77.686**
AIRLINES	62.82	65.229	64.025	64.556	66.417	**68.564**	68.303	62.093
ELEC	89.508	87.407	87.754	89.515	89.672	89.859	**90.267**	89.822
COVTYPE	95.104	92.857	88.519	92.695	94.967	**95.348**	93.461	95.288
KDD99	99.965	2.445	99.951	99.936	99.972	**99.981**	99.973	99.974
ADS	99.634	15.401	97.499	90.393	99.695	**99.726**	98.353	99.634
NOMAO	97.072	58.233	95.462	96.393	97.197	**97.383**	96.57	97.226
SPAM	NA	80.781	89.361	86.519	97.319	**97.437**	95.924	NA
Avg Rank	4.46	6.33	6.67	6.17	4	**1.58**	3.17	3.63
Avg Rank Synt.	4.17	5.67	6.67	6.17	4.67	**2**	2.83	3.83
Avg Rank Real	4.75	7	6.67	6.17	3.33	**1.17**	3.5	3.42

Heitor Murilo Gomes, Albert Bifet, and Jesse Read. **Streaming Random Patches for Evoving Data Stream Classification**. In: *Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM)*, 240-249. https://doi.org/10.1109/ICDM.2019.00034

FAIRNESS?

- Despite several notions of fairness have been proposed recently, there is no agreement on what to be applied in each situation.
- Most common notion: **statistical parity**, which checks whether the common and deprived communities are assigned to the positive class with equal probability.

APPLIED FAIRNESS METHODS FOR STATIC MACHINE LEARNING

- Static fairness-aware approaches can basically categorized into three categories:
  - Pre-processing methods;
  - In-processing methods; and
  - Post-processing methods.

APPLIED FAIRNESS METHODS FOR STATIC MACHINE LEARNING

- **Pre-processing methods** work under the assumption that training data should be discrimination-free. These methods are *model-agnostic* (i.e after process, any model would be applicable). To obtain such state, the following approaches are adapted:
  - ***Massaging:*** Re-label certain instances residing close to the decision boundary to neutralize discriminatory effects;
  - ***Re-weighting:*** Assign different weights to different groups.

APPLIED FAIRNESS METHODS FOR STATIC MACHINE LEARNING

- **In-processing methods:** directly modify the learning algorithm to ensure that they will produce fair results. Contrary to pre-processing methods, these methods are algorithm-specific. Examples include
  - Decision tree with fairness encoded by applying a modified entropy-based attribute splitting criterion (Kamiran et al., 2010)
  - Sensitive attributes are included within the learning process by using a joint loss function to represent a trade-off between fairness and accuracy (Dwork et al. 2018)

APPLIED FAIRNESS METHODS FOR STATIC MACHINE LEARNING

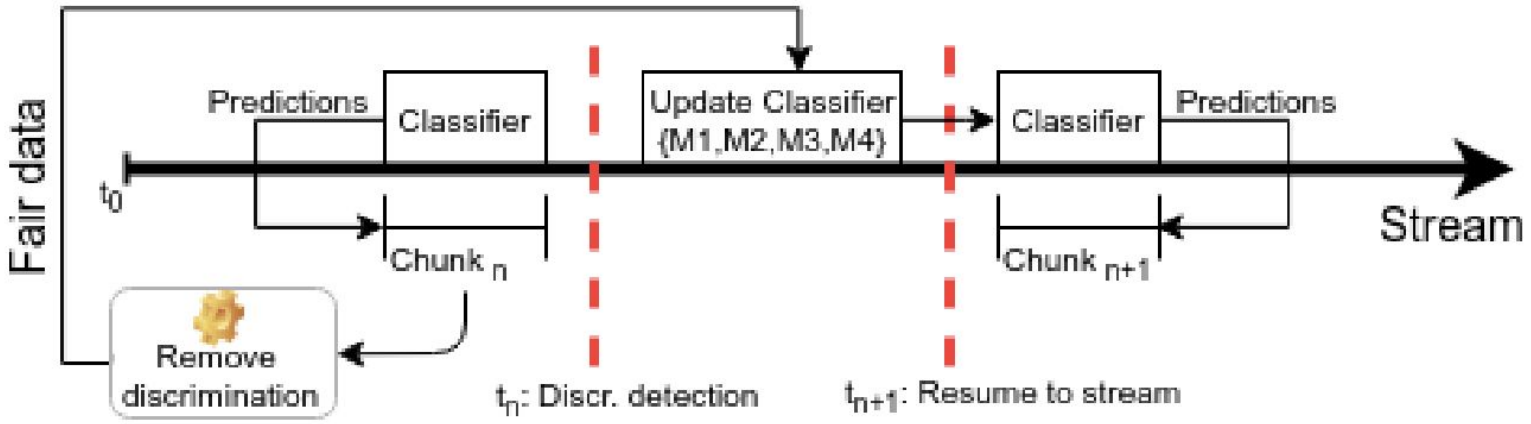
- **Post-processing methods:** modify the result of a trained model to ensure that the fairness criterion is met. Examples include
  - (Kamiran et al., 2010) ***Modifying*** the leaf labels of a decision tree;
  - (Pedreschi et al, 2009) ***Changing*** the confidence values of classification rules; and
  - (Fish et al, 2016) ***Shifting*** the decision boundary of a learner until the criterion is fulfilled.

CONCEPT OF SEQUENTIAL FAIRNESS

- When a series of predictions have to be made, a sequence of decisions has to be taken, the notion of sequential fairness is relevant and of importance now. There have been several work related to the concept, including:
  - Fair online item ranking for groups (Stoyanovich et al., 2018)
  - How fairness criteria interact with temporal indicators of well-being and affect discriminated populations on the long term (Luy et al, 2018).

HOW IS SEQUENTIAL FAIRNESS MAINTAINED

- There are several approaches to maintain fairness in data streams, including:
  - Combining pre-processing methods for fairness and stream classification methods (Iosifidis et al, 2019).



HOW IS SEQUENTIAL FAIRNESS MAINTAINED

- There are several approaches to maintain fairness in data streams, including:
  - **FABBOO (Online fairness and class imbalance-aware boosting)**, consisting of a *class-imbalance monitoring component* while adapting to concept drifts via *blind model adaptation* (Iosifidis et al, 2020)..

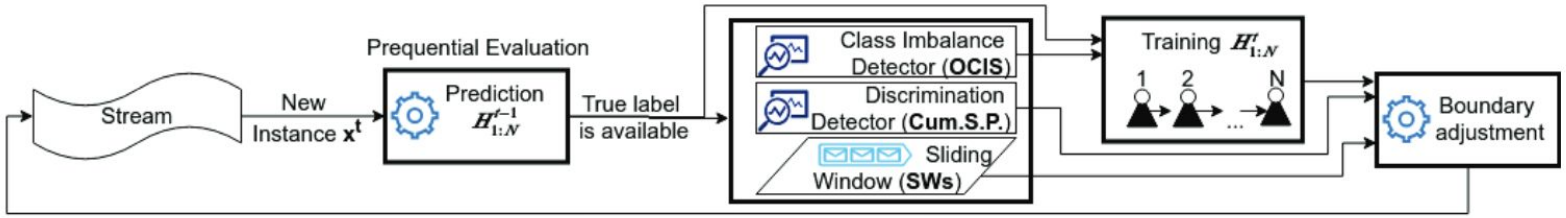


Fig. 1. An overview of FABBOO

HOW IS SEQUENTIAL FAIRNESS MAINTAINED

- There are several approaches to maintain fairness in data streams, including:
  - **FAHT (Fairness Aware Hoeffding Tree)**: resolved fairness issues by considering fairness gain along with the information in the splitting criterion of the decision tree (Zhang et al. 2019).

    The idea of Fairness Gain (FG) aligns with the idea of information gain (IG); for both, it holds that the higher the reduction, the merrier.

$$FG(D, A) = |Disc(D)| - \sum_{v \in dom(A)} \frac{|D_v|}{|D|} |Disc(D_v)|$$

River

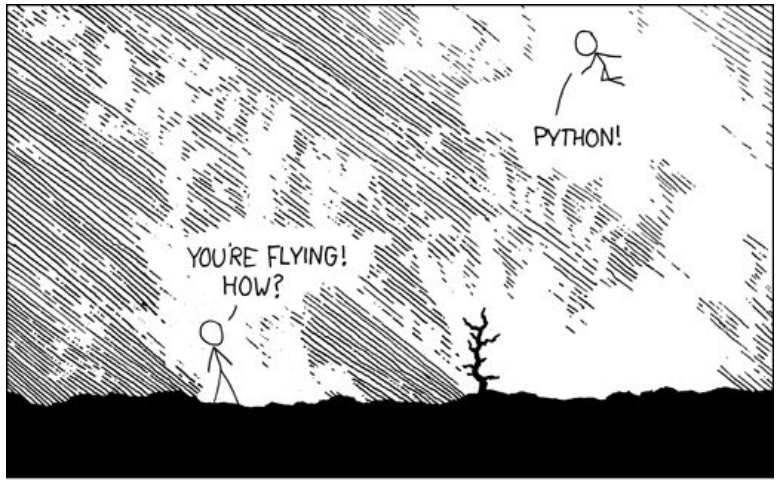
Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. 2021. **River: machine learning for streaming data in Python**. *Journal of Machine Learning Research* 22 (April 2021), 1–8. http://jmlr.org/papers/v22/20-1380.html

A Python library for **stream learning**

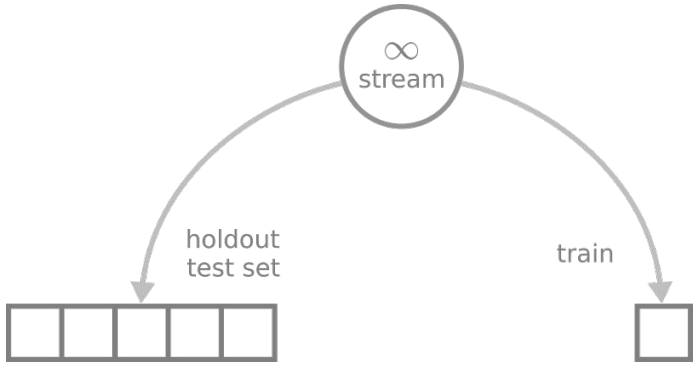
- Incremental + adaptive methods
  - *Supervised learning*
    Classification, Regression
  - *Unsupervised learning*
    Anomaly detection*, Clustering
- Drift detection
- Pipelines / Data transformers
- Evaluation / Metrics
- etc.

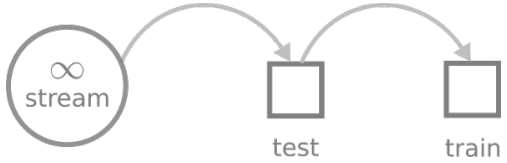
DESIGN PRINCIPLES

- Pythonic
- Easy to use (any expertise level)
- Easy to extend
- Intended to work with other tools in the Python ecosystem
- Users: researchers *and* practitioners



EVALUATION





Holdout an independent test set

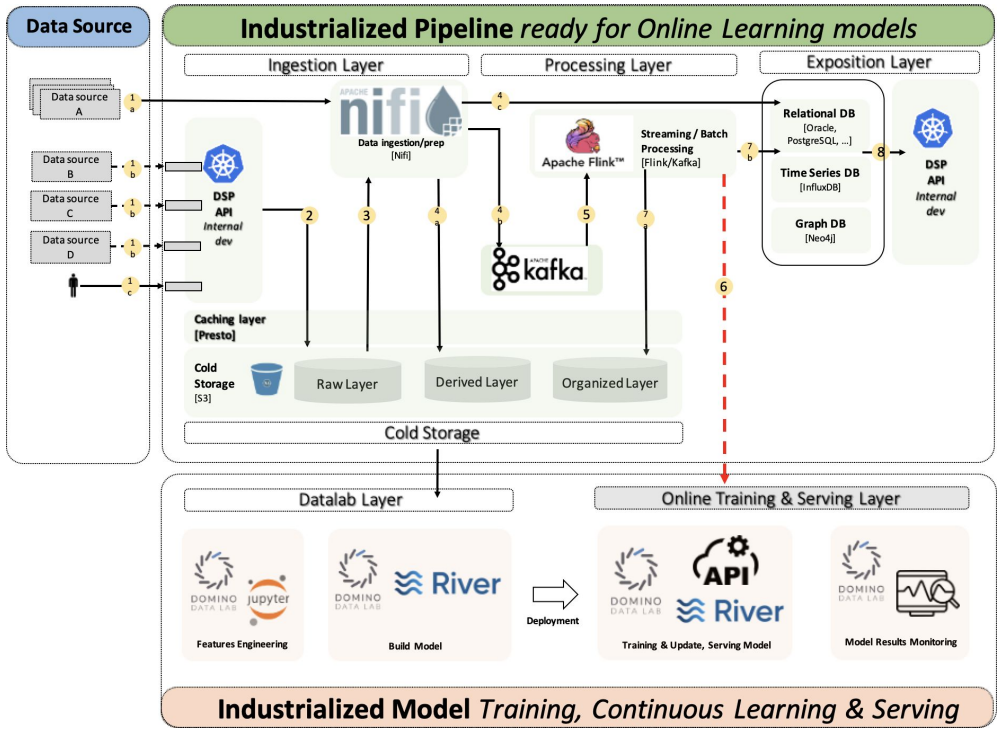
- Apply the current model to the test set, at regular time intervals
- *Unbiased* performance estimation
- Popular in *batch* and *stream* learning

Prequential

- Test *then* train each new instance
  - Order matters!
  - All data is used for training
- Performance is estimated on the sequence
- Popular in the *stream* setting

USE CASE: BNP PARIBAS

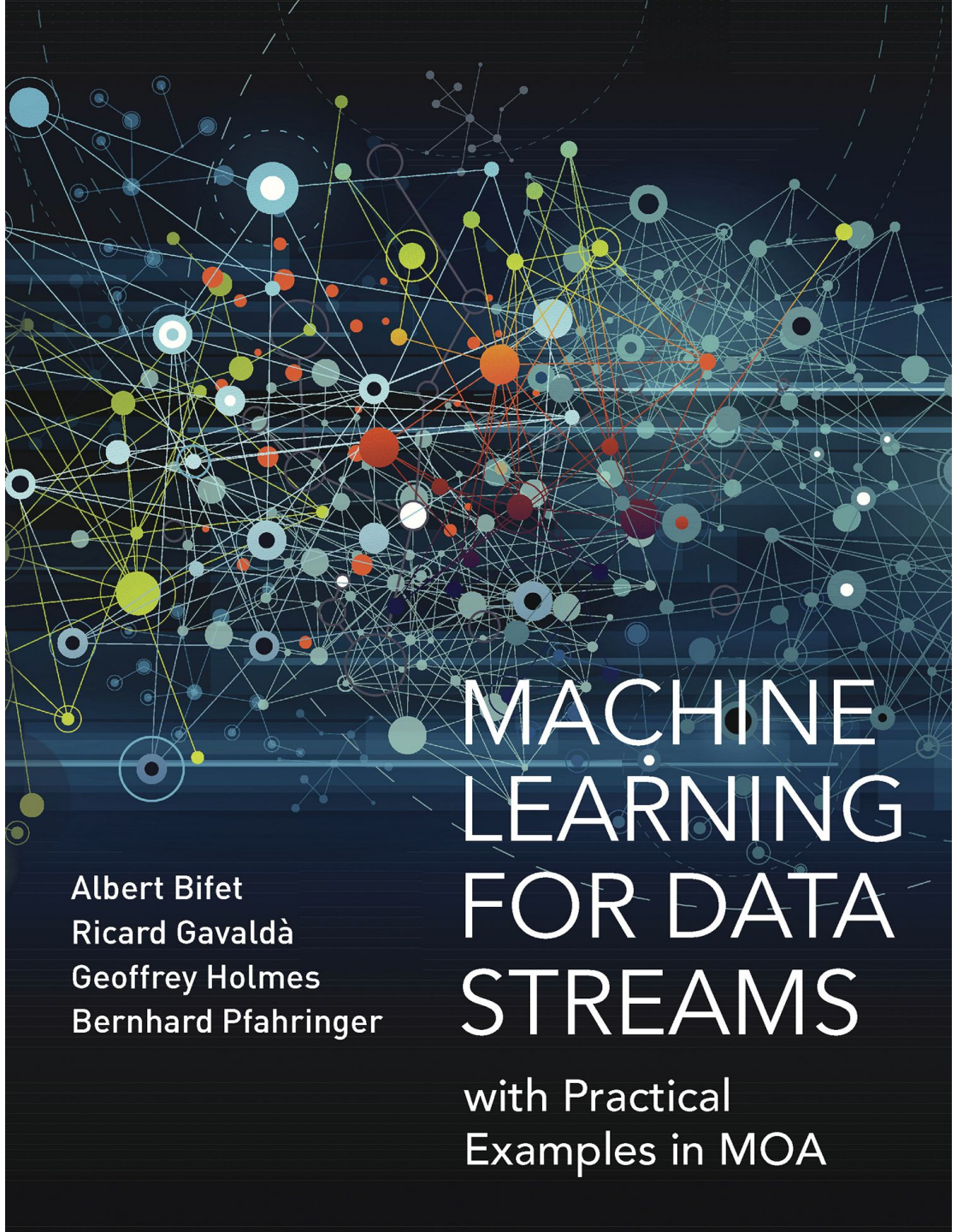
Marriam Barry and Albert Bifet and Raja Chiky and Jacob Montiel and Vinh-Thuy Tran. **Challenges of Machine Learning for Data Streams in Banking Industry**. In: *BDA 2021: Big Data Analytics*. Springer, 106-118. https://doi.org/10.1007/978-3-030-93620-4_9



Proof of concept.

"*...a platform architecture used to deploy online learning models in a production environment and at a large scale.*"

— CLUSTERING



The MIT Press

FZI

TELECOM Paris

IP PARIS

Te Ipu o te Mahara Artificial Intelligence Institute
THE UNIVERSITY OF WAIKATO

DEFINITION

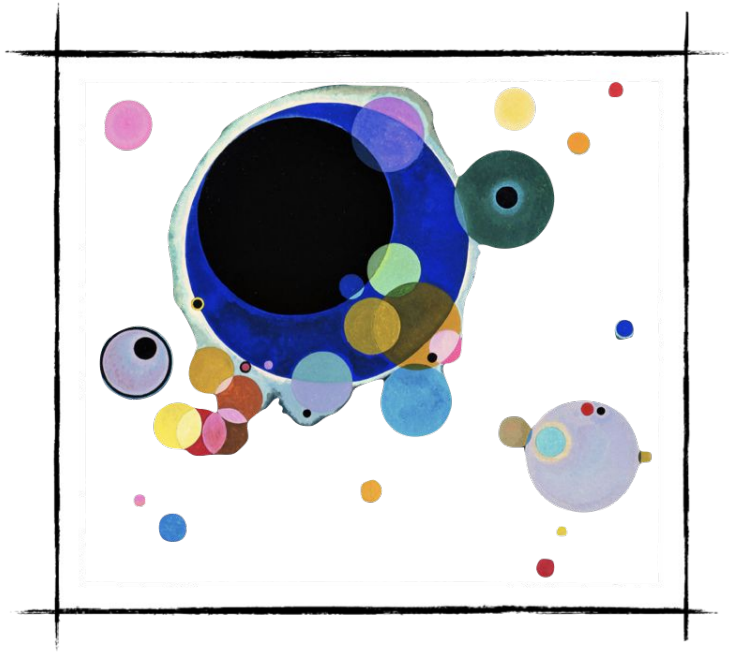


Photo source: W. Kandinsky - Several Circles (edited)

- Given a set of unlabeled instances, distribute them into homogeneous groups according to some common relations or affinities

EXAMPLES
- Market segmentation
- Social network communities

— CLUSTERING

Given

- a set of instances I
- a number of clusters K
- an objective function $cost(I)$

a clustering algorithm computes an assignment of a cluster for each instance

$$f : I \rightarrow \{1, \ldots, K\}$$

that minimizes the objective function $cost(I)$

— CLUSTERING

Given

- a set of instances I
- a number of clusters K
- an objective function $cost(C, I)$

a clustering algorithm computes a set C of instances with $|C| = K$ that minimizes the objective function

$$cost(C, I) = \sum_{x \in I} d^2(x, C)$$

where

- $d(x, c)$: distance function between x and c
- $d^2(x, C) = min_{c \in C} d^2(x, c)$: distance from x to the nearest point in C

— AVAILABLE SOFTWARES

Very few implementations (only most prominent ones) and unified frameworks with multiple algorithms co-existing:

- **Massive Online Analysis (MOA):** Most popular framework, written by Bifet et al. (2010) in Java, including the most number (7) of clustering algorithms. However, one major **disadvantage**: only works well when information of data streams are previously known.
- **stream package**: Written in R by Hahsler et al. (2018), with newer algorithms including D-Stream, DBSTREAM and evoStream.

— AVAILABLE SOFTWARES

Very few implementations (only most prominent ones) and unified frameworks with multiple algorithms co-existing:

- **Subspace MOA framework:** An extension of MOA from Java into R, written by Hassani et al. (2016), with extra algorithms including HDDStream and PreDeConStream.
- **streamDM:** Written by Huawei Noah's Ark Lab (2015) with Spark Streaming, an extension of Spark engine. Including CluStream and StreamKM++, but no plans for any further implementation

— SOLUTION – RIVER

→ **River** comes into play, with a neat implementation that allows:

- Works with any arbitrary numerical data stream;
- Well-maintained, documented and includes various algorithms of different types.

Currently, River offers **7** clustering algorithms, including incremental K-Means, CluStream, DenStream, DBSTREAM, StreamKMeans (O'Callaghan et al., 2002), evoStream and Hierarchical Clustering using OTD (Rajagopalan et al. 2019) with a clear plan of further implementations.

Includes the **most** number of clustering algorithms apart from MOA.

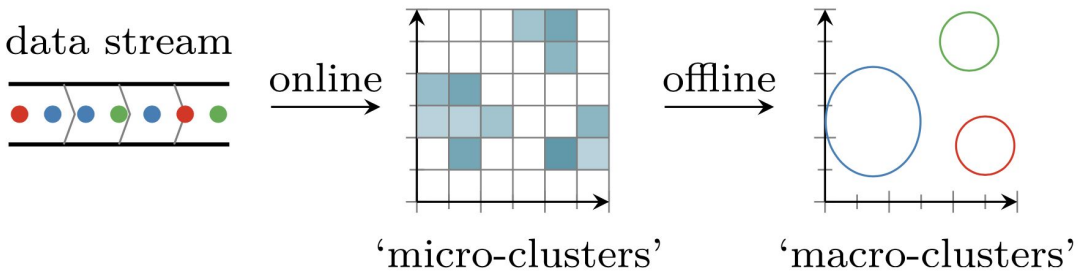
— STRATEGY

Basically, finding clustering solutions is an optimization task, with the following principle strategies:

- Minimizing intra-cluster distances or radii of clusters (ensuring that objects within the same cluster are similar);
- Maximizing inter-cluster distances or heterogeneity (ensuring that objects within different clusters are well-separated);
- Maximizing likelihood estimates

— ARISING PROBLEMS

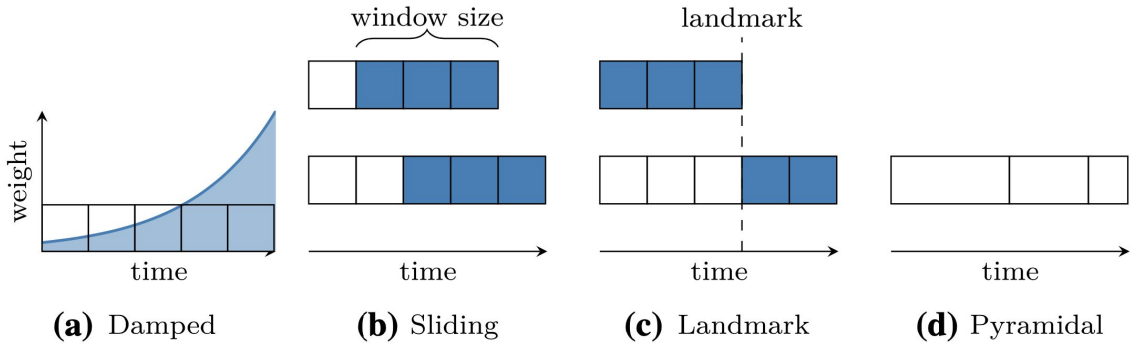
- In online clustering, historical data will be discarded and only information of formed clusters (cluster centers, number of points, linear sum, sum of squares, etc.) will be saved → Clustering algorithms are divided into **two** phases: ONLINE phase and OFFLINE phase.



Two-phase stream clustering with grid-based approach
(Source: Matthias Carnein et al. 2017. An empirical comparison of stream clustering algorithms.)

— ARISING PROBLEMS

– Through time, the distribution of the stream will change. (also known as drift or concept drift) → Models can employ time-window models, which only keeps the most few recent data points to avoid bias. This approach can include damped, sliding, landmark or pyramidal models.



Two-phase stream clustering with grid-based approach
(Source: Zhu Y. and Shasha D. 2002. Statstream: statistical monitoring of thousands of data streams in real life and Silva J. A. et al. 2013. Data stream clustering: a survey.)

— APPROACHES

Matthias Carnein and Heike Trautmann. 2019. **Optimizing Data Stream Representation: An Extensive Survey on Stream Clustering Algorithms**. *Business and Information Systems Engineering* 61, 3 (2019), 277-297. https://doi.org/10.1007/s12599-019-00576-5

- **Distance-based approach:** threshold the distance of the new observation to existing clusters, either to insert or initialize new clusters, including:
  - ***Clustering Features (CFs), Extended CFs, Time-Fading CFs:*** BIRCH, CluStream, SDStream, ClusTree;
  - ***Centroids, Medoids:*** StreamKM++, STREAM;
  - ***Competitive Learning:*** DBSTREAM.
- **Density-based (Grid-based) approach:** capture the density of observation in a grid, by separating the data space among all dimension, including:
  - ***One-time or recursive partitioning:*** DUCStream, D-Stream, Stats-Grid;
  - ***Hybrid Grid-Approach:*** HDCStream, Mudi-Stream;

— APPROACHES

Matthias Carnein and Heike Trautmann. 2019. **Optimizing Data Stream Representation: An Extensive Survey on Stream Clustering Algorithms**. *Business and Information Systems Engineering* 61, 3 (2019), 277-297. https://doi.org/10.1007/s12599-019-00576-5

- **Model-based approach:** Summarize the data stream as a ***statistical model***, with a common area of research based on the Expectation Maximization (EM) algorithm. Including CluDisStream, SWEM, COBWEB, Wstream, etc.

- **Projected approach:** This special approach deals with ***high dimensional data stream***, addressing the curse of dimensionality. Including HPStream, HDDStream, and PreDeConStream along with their extensions.

K-MEANS

K-MEANS(P, k)

Input: a dataset of points $P = \{p_1, \ldots, p_n\}$, a number of clusters k
Output: centers $\{c_1, \ldots, c_k\}$ implicitly dividing P into k clusters

```
1  choose k initial centers C = {c_1, ..., c_k}
2  while stopping criterion has not been met
3      do ▷ assignment step:
4          for i = 1, ..., N
5              do find closest center c_k ∈ C to instance p_i
6                  assign instance p_i to set C_k
7          ▷ update step:
8          for i = 1, ..., k
9              do set c_i to be the center of mass of all points in C_i
```

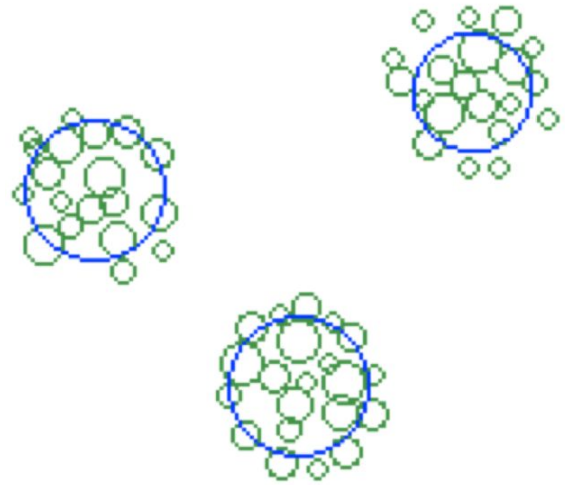
— SIMPLEST IMPLEMENTED ALGORITHM: INCREMENTAL K-MEANS

- Basically the simplest, naive approach that we can think of.
- Most noticeable parameter: halflife, which decides how much to move the cluster to the new point.
- Obtains comparable performance compared to more complicated algorithms, while maintaining a constantly low memory usage.

— MICRO-CLUSTERS

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. **BIRCH: An Efficient Data Clustering Method for Very Large Databases**. In *SIGMOD'96: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/233269.233324

- Cluster Features $\overrightarrow{CF}$ (statistical summary structure)
- Maintained in online phase, input for offline phase
- Data stream $\langle \overrightarrow{x_i} \rangle$, d dimensions
- Cluster Features vector includes
  - N: number of points
  - LS_i: sum of values (for dimension j)
  - SS_i: sum of squared values (for dimension j)
- Easy to update, easy to merge
- **Constant space irrespective to the number of examples!**



Properties:

- Centroid $= LS/N$
- Radius $= \sqrt{SS/N - (LS/N)^2}$
- Diameter $= \sqrt{\frac{2 \times N * SS - 2 \times LS^2}{N \times (N-1)}}$

HANDLING ADDITION AND SUBTRACTION

- **Addition:**
  - $n_{AB} = n_A + n_B$
  - $\overline{x_{AB}} = \frac{n_A\overline{x_A} + n_B\overline{x_B}}{n_{AB}}$
  - $M_{2,AB} = M_{2,A} + M_{2,B} + \frac{\delta^2 n_A n_B}{n_{AB}}$
    $(\delta = \overline{x_B} - \overline{x_A})$

- **Subtraction:**
  - $n_A = n_{AB} - n_B$
  - $\overline{x_{AB}} = \frac{n_{AB}\overline{x_{AB}} - n_B\overline{x_B}}{n_A}$
  - $M_{2,A} = M_{2,AB} - M_{2,B} - \frac{\delta^2 n_A n_B}{n_{AB}}$

— CLUSTREAM

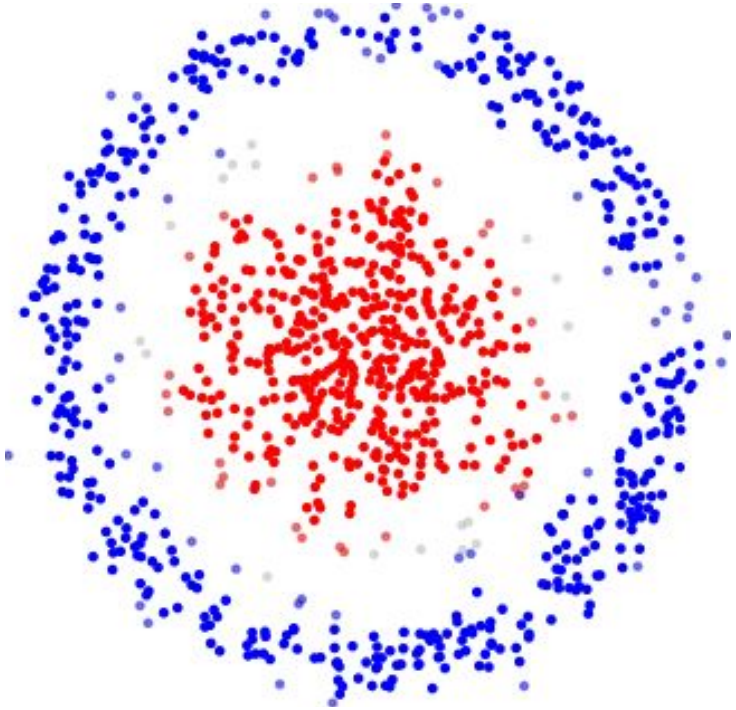
Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Phillip S. Yu. 2003. **A Framework for Clustering Evolving Data Streams**. In: *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29* (Berlin, Germany) (VLDB '03). VLDB Endowment, Berlin, Germany, 81–92.

- Time - stamped data stream $\langle t_i, \overrightarrow{x_i} \rangle$, represented in $d + 1$ dimensions
- Seed algorithm with q micro-clusters (K-Means on initial data)
- **ONLINE phase.** For each new point, either:
  - Update one micro-cluster (point within maximum boundary)
  - Create a new micro-cluster (delete/merge other micro-clusters)
- **OFFLINE phase**. Determine k macro-clusters on demand:
  - K-Means on micro-clusters (weighted pseudo-points)
  - Time-horizon queries via pyramidal snapshot mechanism

— DBSCAN

Martin Ester and Hans-Peter Kriegel and Jörg Sander and Xiaowei Xu. 1996. **A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise**. In *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 226-231. https://doi.org/10.5555/3001460.3001507

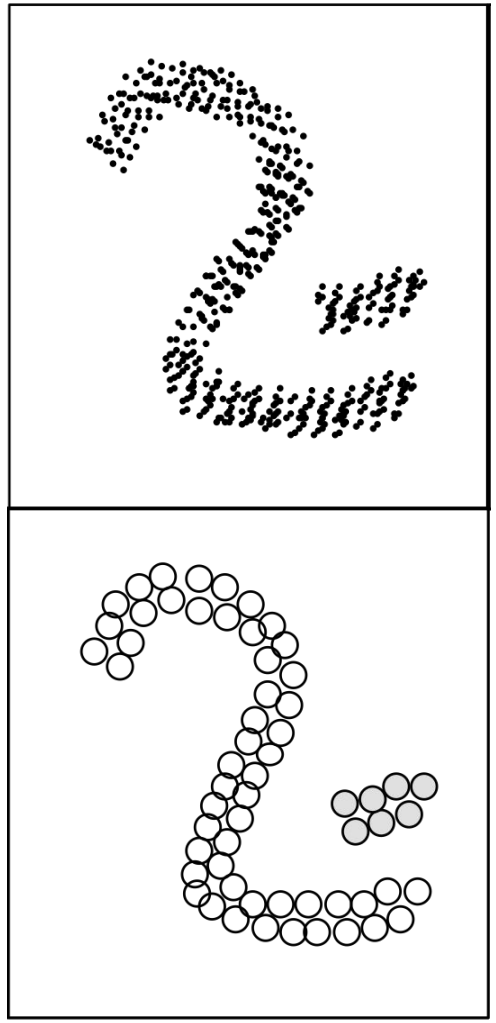
- $N_\epsilon(p)$: set of points at distance $\leq \epsilon$ from p
- Core object p **iff** $N_\epsilon(p)$ has weight $\geq \mu$
- p is directly density-reachable from q **iff** p $\in N_\epsilon(q)$ **and** q is a core object
- p_n is density-reachable from p_1 **iff** there exists chain of points $p_1, \ldots, p_n$ such that p_{i+1} is directly $d - r$ from p_i
- **Cluster**: set of points that are mutually density-connected



— DENSTREAM

Feng Cao and Martin Ester and Weining Qian and Aoying Zhou. **Density-Based Clustering over an Evolving Data Stream with Noise**. In: *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*. Society for Industrial and Applied Mathematics (SIAM), 328-339. https://doi.org/10.1137/1.9781611972764.29

- Based on DBSCAN
- Core-micro-cluster: CMC(ω, c, r) with weight $\omega > \mu$, center c, radius r $< \epsilon$
- Potential/Outlier micro-clusters
- **Online phase:** merge point into p (or o) micro-cluster if new radius r$' < \epsilon$
  - Promote outlier to potential if $\omega > \beta\mu$; else, create a new o-micro-cluster
- **Offline phase:** DBSCAN



— DENSTREAM

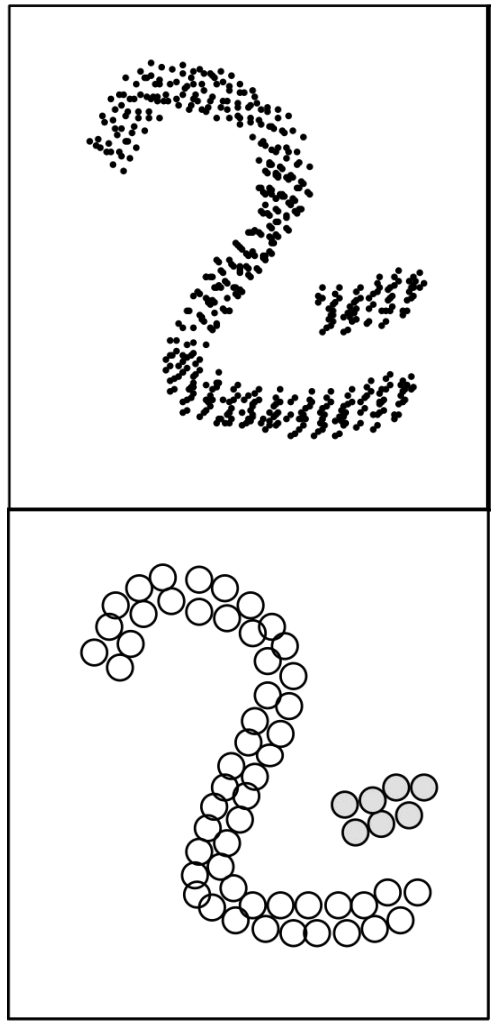
Feng Cao and Martin Ester and Weining Qian and Aoying Zhou. **Density-Based Clustering over an Evolving Data Stream with Noise**. In: *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*. Society for Industrial and Applied Mathematics (SIAM), 328-339. https://doi.org/10.1137/1.9781611972764.29

DEN-STREAM(*Stream*, λ, μ, β)

Input: a stream of points, decaying factor λ,
core weight threshold μ, tolerance factor β

▷ Online phase
$T_p \leftarrow \lceil \frac{1}{\lambda} \log(\frac{\beta\mu}{\beta\mu - 1}) \rceil$
for each new point that arrives
  do try to merge to a p-microcluster; if not possible,
    merge to nearest o-microcluster
      if o-microcluster weight $> \beta\mu$
        then convert the o-microcluster to p-microcluster
        else create a new o-microcluster

▷ Offline phase
if ($t \bmod T_p = 0$)
  then for each p-microcluster c_p
    do if $w_p < \beta\mu$
      then remove c_p
    for each o-microcluster c_o
      do if $w_o < (2^{-\lambda(t - t_o + T_p)} - 1)/(2^{-\lambda T_p} - 1)$
        then remove c_o
    apply DBSCAN using microclusters as points



DBSTREAM

Michael Hashler and Matthew Bolanos. 2016. **Clustering Data Streams Based on Shared Density between Micro-Clusters**. *IEEE Transactions on Knowledge and Data Engineering* 28, 6 (2016), 1449-1461. https://doi.org/10.1109/TKDE.2016.2522412

- DBSTREAM is the first micro-cluster-based algorithm that explicitly captures the density between micro clusters via a shared density graph.
- **Online phase:**
  - A new micro cluster is generated from the newly coming point. If one or more micro clusters are found within a fixed radius from it, these micro clusters will be updated. Else, the newly generated one will be added into the list of existing micro clusters.
  - The density graph is then updated. Also, to prevent micro-clusters from collapsing, movement will be restricted.
  - Cleanup process is then initiated after each t_{gap} interval.

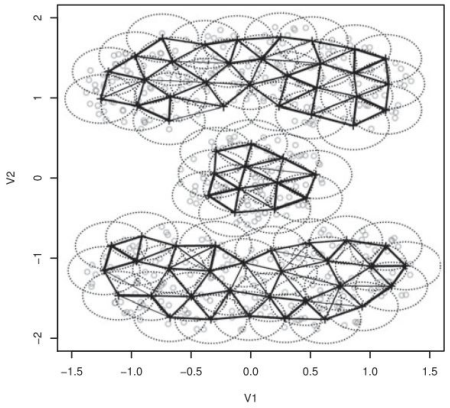
DBSTREAM

Michael Hashler and Matthew Bolanos. 2016. **Clustering Data Streams Based on Shared Density between Micro-Clusters**. *IEEE Transactions on Knowledge and Data Engineering* 28, 6 (2016), 1449-1461. https://doi.org/10.1109/TKDE.2016.2522412

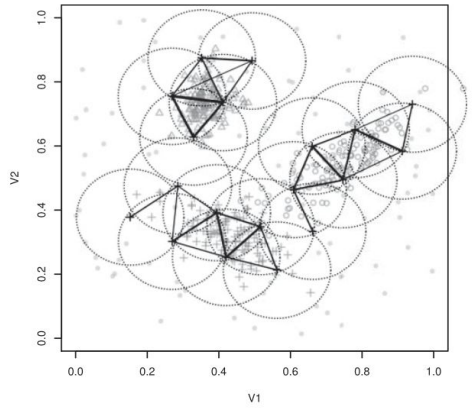
- DBSTREAM is the first micro-cluster-based algorithm that explicitly captures the density between micro clusters via a shared density graph.
- **Offline phase**:
  - The connectivity graph is constructed using shared density entries between strong micro clusters. The edges in this connectivity graph with a connectivity value greater than the intersection threshold (α) are used to find connected components representing the final cluster.
  - After the connectivity graph is generated, a version of DBSCAN by Ester et al. is applied to form all macro-clusters from α-connected micro clusters.

DBSTREAM

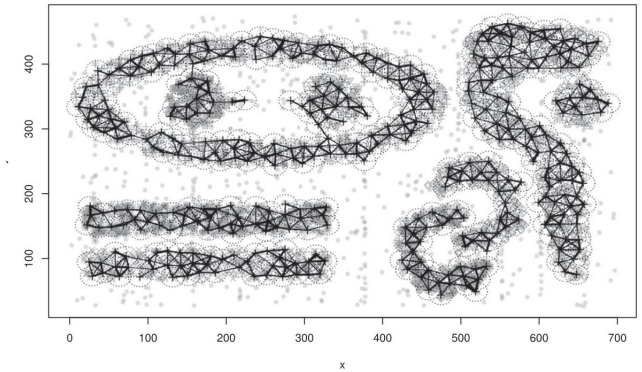
Michael Hashler and Matthew Bolanos. 2016. **Clustering Data Streams Based on Shared Density between Micro-Clusters**. *IEEE Transactions on Knowledge and Data Engineering* 28, 6 (2016), 1449-1461. https://doi.org/10.1109/TKDE.2016.2522412



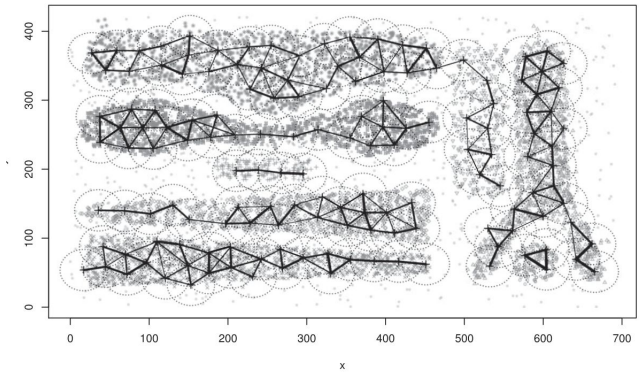
(a) Cassini



(b) Noisy mixture of Gaussians



(c) Chameleon dataset DS3



(d) Chameleon dataset DS4

EVOSTREAM

Matthias Carnein and Heike Trautmann. 2018. **evoStream – Evolutionary Stream Clustering Utilizing Idle Time**. *Big Data Research* 14 (2018), 101-111. https://doi.org/10.1016/j.bdr.2018.05.005

- A fairly new online clustering algorithm.
- evoStream employs an evolutionary algorithm, first introduced by Maulik U. and Bandyopadhyay S. (2000), to utilize “idle” time efficiently to find better macro-cluster solutions.
- In the evolution algorithm, promising solutions are combined to create offsprings which can combine the best attributes of both parents.
- Include two phases: **Micro-cluster maintenance** (online learning phase) and **evolutionary step of micro-cluster generation** (offline phase)

TEXTCLUST

Assenmacher, D., Trautmann, H. (2022). Textual One-Pass Stream Clustering with Automated Distance Threshold Adaption. In: Nguyen, N.T., Tran, T.K., Tukayev, U., Hong, TP., Trawiński, B., Szczerbicki, E. (eds) Intelligent Information and Database Systems. ACIIDS 2022. Lecture Notes in Computer Science(), vol 13757. Springer, Cham. https://doi.org/10.1007/978-3-031-21743-2_1

- TF-IDF (Term frequency – inverse document frequency) based text stream clustering algorithm, using an automated distance threshold adaptation technique for document insertion and cluster merging.
- textClust uses the popular two-phase approach: The online phase will be used to summarize data coming in a real time manner into micro-clusters, while the offline phase will be used to create macro-clusters on demand.
- To keep micro-clusters up-to-date, a fading strategy is applied where micro-clusters that are not updated regularly gradually loses relevance.

TEXTCLUST

Assenmacher, D., Trautmann, H. (2022). Textual One-Pass Stream Clustering with Automated Distance Threshold Adaption. In: Nguyen, N.T., Tran, T.K., Tukayev, U., Hong, TP., Trawiński, B., Szczerbicki, E. (eds) Intelligent Information and Database Systems. ACIIDS 2022. Lecture Notes in Computer Science(), vol 13757. Springer, Cham. https://doi.org/10.1007/978-3-031-21743-2_1

- The algorithm can be described as follows:
  - First, the algorithm reads a new text x, which will go through the preprocessing step and be split into individual tokens.
  - The remain list of tokens will be used to construct n-grams, and a new virtual MC is constructed. Then, IDF vector across all documents are computed, which will then be used to calculate the TF-IDF representations of each existing and the new virtual MC. The closest MC becomes the candidate for merging

TEXTCLUST

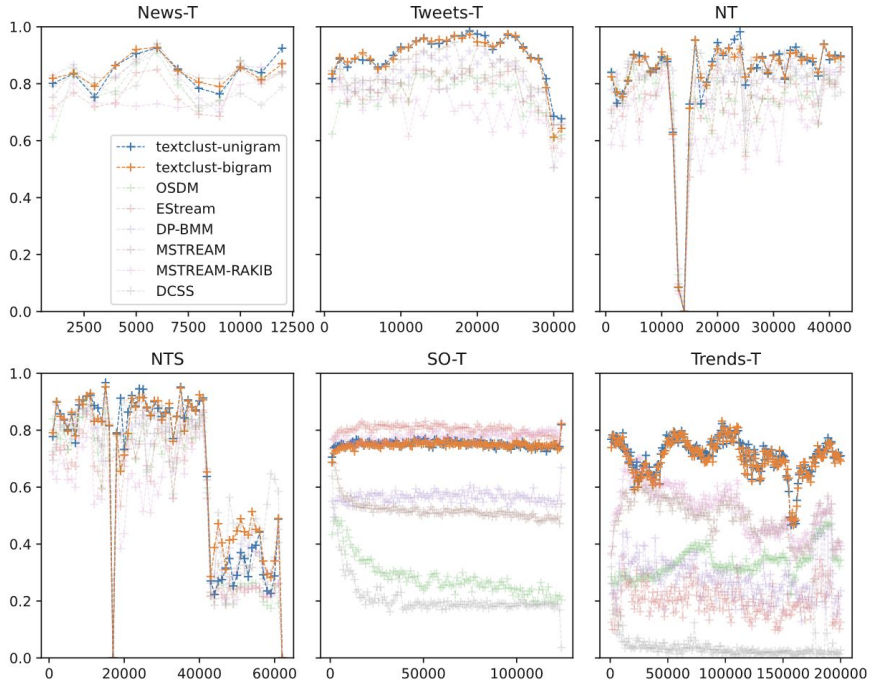
Assenmacher, D., Trautmann, H. (2022). Textual One-Pass Stream Clustering with Automated Distance Threshold Adaption. In: Nguyen, N.T., Tran, T.K., Tukayev, U., Hong, TP., Trawiński, B., Szczerbicki, E. (eds) Intelligent Information and Database Systems. ACIIDS 2022. Lecture Notes in Computer Science(), vol 13757. Springer, Cham. https://doi.org/10.1007/978-3-031-21743-2_1

- The algorithm can be described as follows:
  - Next, check whether the candidate is close enough to the new MC for both to be merged.
    - If yes, the new virtual MC is merged into the candiate cluster, and the cluster's weight will be decayed to the current time.
    - If no, the temporary MC will be added to all the clusters.
  - Finally, clean-up after each time gap.

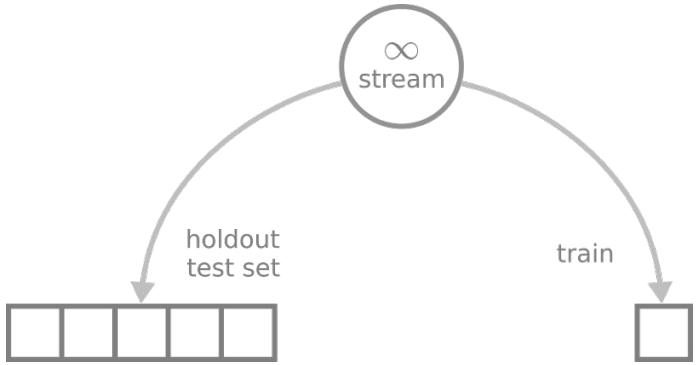
TEXTCLUST

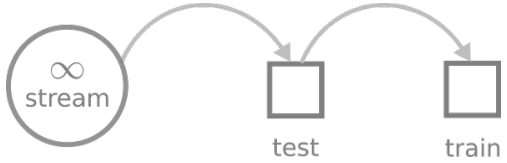
Assenmacher, D., Trautmann, H. (2022). Textual One-Pass Stream Clustering with Automated Distance Threshold Adaption. In: Nguyen, N.T., Tran, T.K., Tukayev, U., Hong, TP., Trawiński, B., Szczerbicki, E. (eds) Intelligent Information and Database Systems. ACIIDS 2022. Lecture Notes in Computer Science(), vol 13757. Springer, Cham. https://doi.org/10.1007/978-3-031-21743-2_1

- textClust outperforms state-of-the-art one-pass and batch-based stream clustering algorithms (Estream, MSTREAM, OSDM, etc.)



EVALUATION





Holdout an independent test set

- Apply the current model to the test set, at regular time intervals
- *Unbiased* performance estimation
- Popular in *batch* and *stream* learning

Prequential

- Test *then* train each new instance
  - Order matters!
  - All data is used for training
- Performance is estimated on the sequence
- Popular in the *stream* setting

— CLASSICAL STATIC EVALUATION

4 basic internal (validation) metrics include

- **Cohesion**: The average distance from a point in the dataset to its assigned cluster centroid. The smaller the better.
- **SSQ**: The sum of squared distances from data points to their assigned centroids. Closely related to cohesion. The smaller the better.
- **Separation**: Average distance from a point to the points assigned to other clusters. The larger the better.
- **Silhouette coefficient:** the ratio between cohesion and the average distances from the points to their second-closest centroid.

CLASSICAL STATIC EVALUATION

External validation metrics, requiring ground truth values, is mostly based on the following concepts

- **Accuracy:** Fraction of the points assigned to their “correct” cluster.
- **Recall:** Fraction of the points of a cluster that are in fact assigned to it.
- **Precision:** Fraction of the points assigned to a cluster that truly belong to it.
- **Purity:** In a maximally pure clustering, all points in the cluster belong to the same ground-truth class or cluster. Formally, purity is

$$\frac{1}{N}\sum_{c=1}^{k}(\text{number of points in cluster } c \text{ in the majority class for } c).$$

INTERNAL METRICS: ARE THEY REALLY ONLINE?

Leonardo Enzo Brito Da Silva, Niklas Max Melton and Donald C. Wunsch. 2022. **Incremental Cluster Validity Indices for Online Learning of Hard Partitions: Extensions and Comparative Study**. *IEEE Access*, 8 (2020), 22025-22047. https://doi.org/10.1109/ACCESS.2020.2969849.

- In 2020, Leonardo Enzo Brito Da Silva et al. introduced a new approach for incremental validity indices. This allows an update to a new value from a previous old value.
- However, this approach still has one huge limitation: ALL information of each previous data points still have to be available.
- As such, there is a requirement to come up with metrics that are truly incremental (facilitating the fashion of learning one sample at a time).

HOW TO DESIGN AN INCREMENTAL INTERNAL METRIC?

INCREMENTAL EVALUATION

In **River**, static evaluation metrics can be modified to continuously evaluate data streams as follows:

- **External metrics:** All external metrics implemented in River share the same **confusion matrix**, thus reducing the amount of occupied storage and computational time. This confusion matrix can easily be updated once a new predicted label and its ground truth arrive.

- **Internal metrics:** Since information on previous data points are **totally discarded** once a new data point arrives, traditional internal metrics **cannot be used incrementally**. Instead, they are slightly modified by saving the most essential information based on requirement, for example linear sum and sum of squares of data points in the same cluster, distance from data point to assigned and/or second closest cluster centre at the time of update, etc.

INCREMENTAL EVALUATION

With **20 internal metrics** and **18 external metrics**, River is currently the package with the highest number of metrics offered for data stream continuous or incremental validation.

- Internal metrics: Cohesion, SSB, SSW, Separation, Silhouette, Ball-Hall, CH, Hartigan, WB, Xie-Beni, Xu, (Root) Mean Squared Standard Deviation, R-Squared, I Index, Davies-Bouldin, Partition Separation, Dunn's indices 43 and 53, SD Validation Index, and Bayesian Information Criterion.

- External metrics: Completeness, Homogeneity, VBeta, (Adjusted, Expected, Normalized) Mutual Information, Q0 and Q2, Fowlkes-Mallows, Markedness, Informedness, Matthews Correlation Coefficient, (Adjusted) Rand Index, Purity, Prevalence Threshold, and Sorensen-Dice index.

INCREMENTAL EVALUATION

Every metric (both internal and external) in River contains the following attributes:

- `cm`: Confusion matrix;
- `update` and `revert`: Allow the metric to be updated with a new observation, or reverted to the previous state;
- `get`: Obtain the exact value of the metric;
- `bigger-is-better`: Indicate whether the metric has the property of the bigger, the better the clustering solution is;
- `work_with`: Indicate whether the metrics work with algorithms of which type (clustering, classification, regression, etc.);

STREAMING EVALUATION

Hardy Kremer, Philipp Kranen, Timm Jansen, Thomas Seidl, Albert Bifet, Geoff Holmes, Bernhard Pfahringer. 2011. **An effective evaluation measure for clustering on evolving data streams**. In: *KDD '11 Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, New York, NY, USA, 868-876. https://doi.org/10.1145/2020408.2020555

- Clusters may: appear, fade, move, merge
  - Missed points (unassigned)
  - Misplaced points (assigned to different cluster)
  - Noise
- Cluster Mapping Measure CMM
  - External (ground truth)

FURTHER STEPS

- Benchmarking
- Implementation of more clustering algorithms and/or improvement in performance of current ones
  - Rust (light-river) (work in progress)
  - Mojo (which claims to combine the usability of Python with performance of C)
- Fairness? Interpretability?

Pause,
See you in 30 min!

Outline (Second Part)

Online Deep Learning

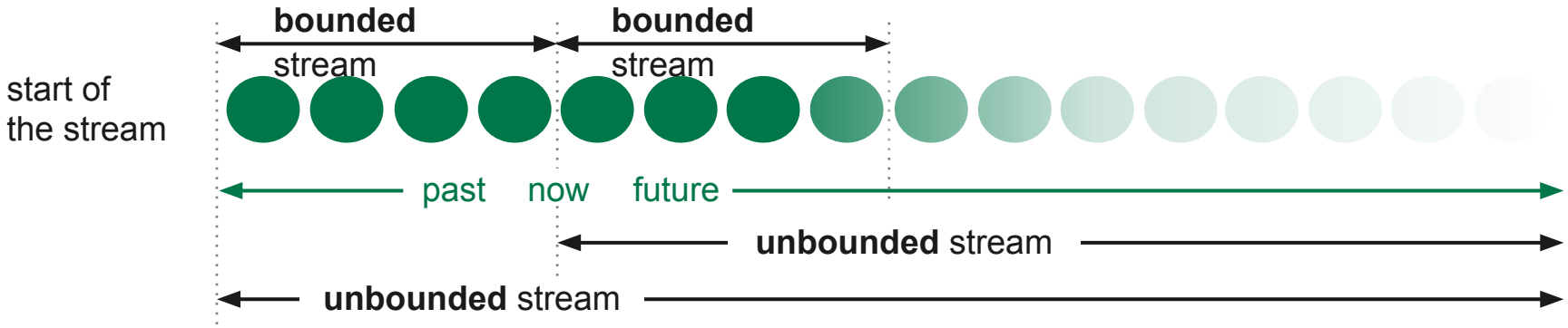
1. Introduction to Deep ML for data streams
2. A brief Introduction to deep-river
3. Solving ML Tasks on Data Streams with Deep Learning
4. Chances and Pitfalls of online deep learning
5. Conclusion

Tutorial

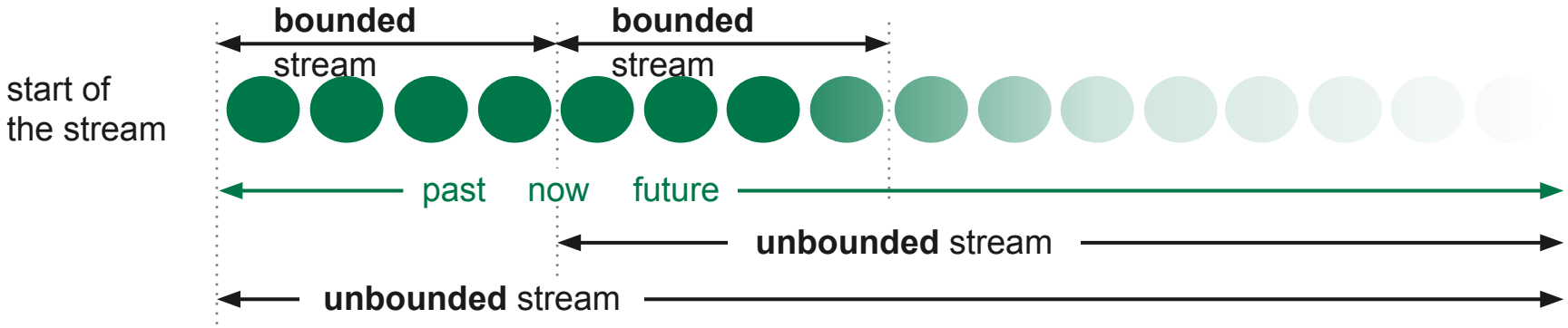
01

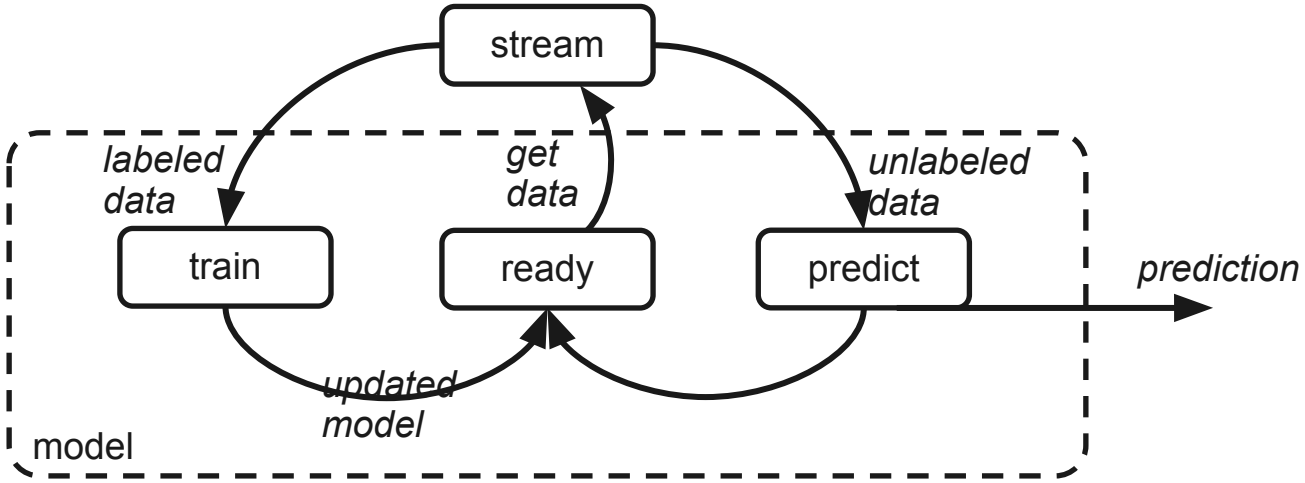
Introduction to Deep ML on Data Streams

Characteristics of Data Streams



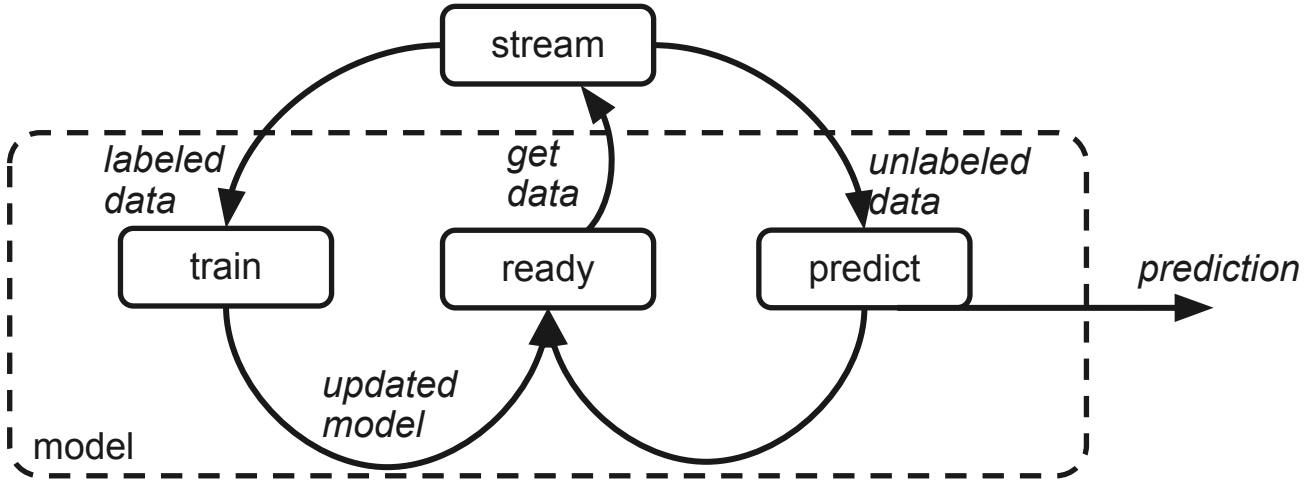
Characteristics of Data Streams





Characteristics of Data Streams

How to monitor the performance



- **Error Estimation**
  - **Holdout:** Create holdout sets to test an train.
  - **Interleaved test-then-train:** Each sample is used to test and then train**.**
  - **Prequential:** test-then-train with sliding window so that recent examples are more important.
  - **Interleaved chunks:** test-then-train with chunks in sequence.

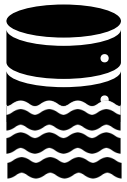
Characteristics of Data Streams

Requirements

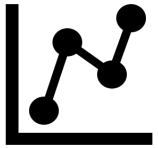
- Process one sample at a time, and inspect it only once
- Use a limited amount of memory
- Work in a limited amount of time
- Always ready to predict

Characteristics of Data Streams

Data Stream

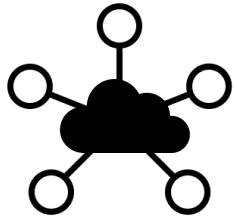


Frequent



Incremental

Changing



Big

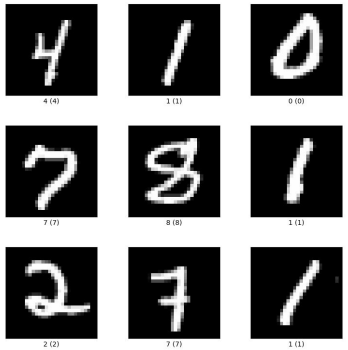
[1]

[2]

[2]

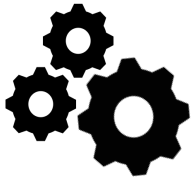
Characteristics of neural networks

Incremental Torch

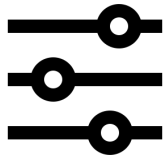


Neural Networks

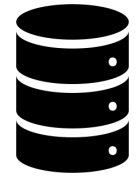
Strong performance



Complex problems

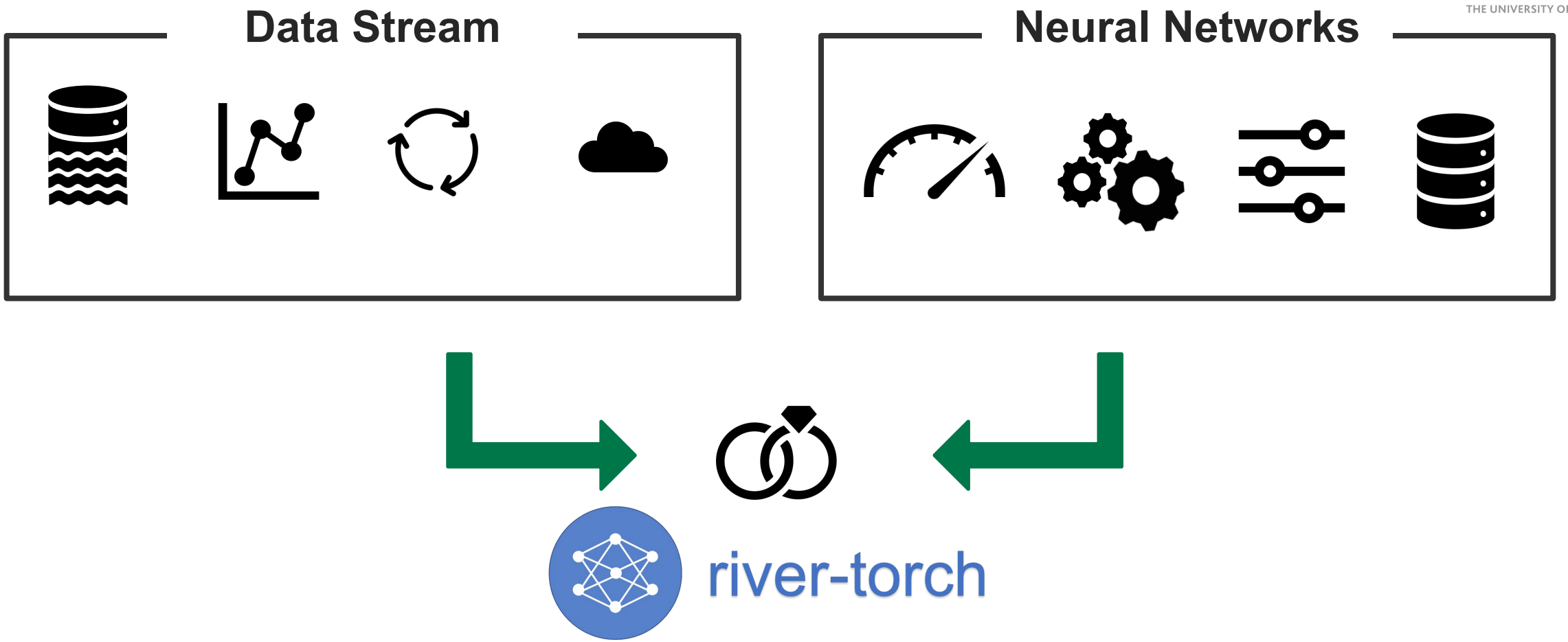


Parameter-Tuning



Large data sets

Goin‘ Deeper: river-torch



— A brief Introduction to deep-river

02

- What is deep-river?
- Design Principles
- Demo: Solving ML Tasks with deep-river
- How efficient is online learning with PyTorch?

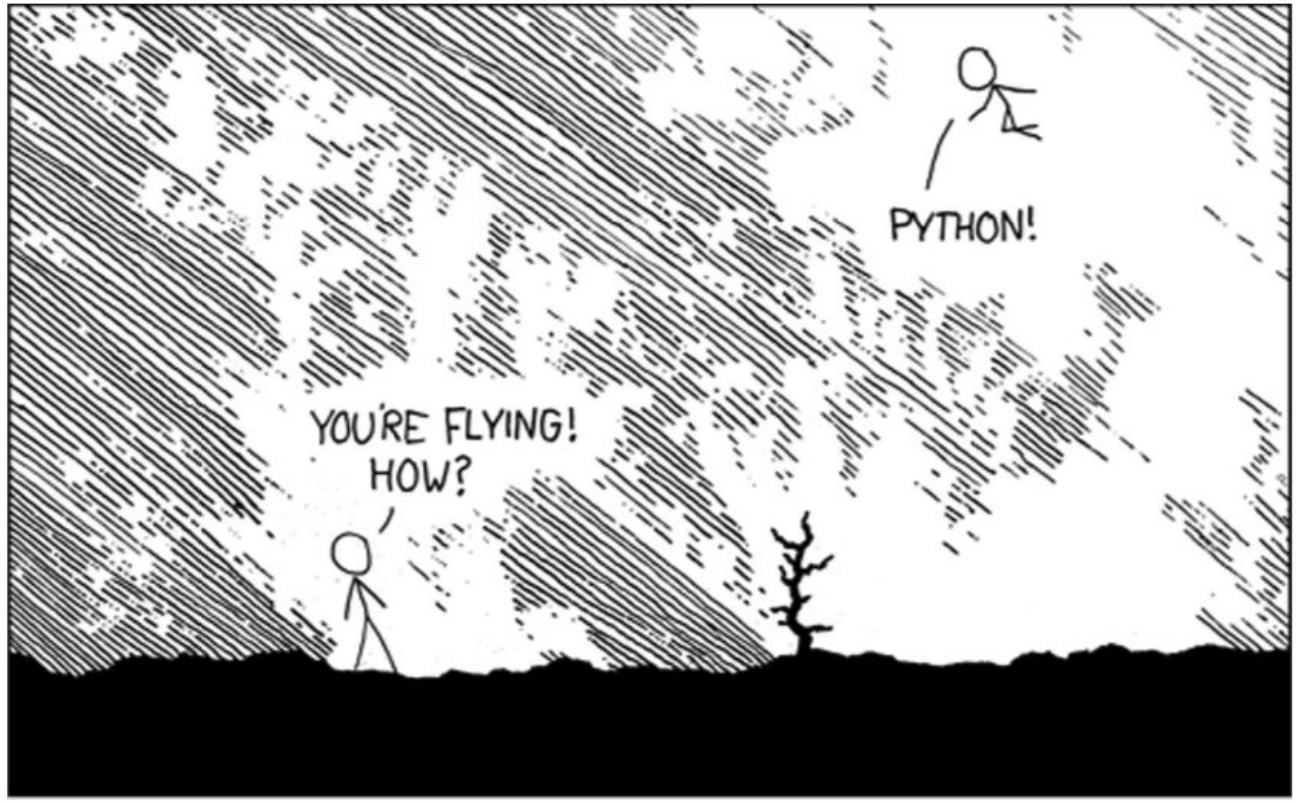
Design Principles

River

- Pythonic
- Easy to use (any expertise level)
- Easy to extend
- Intended to work with other tools in the Python ecosystem

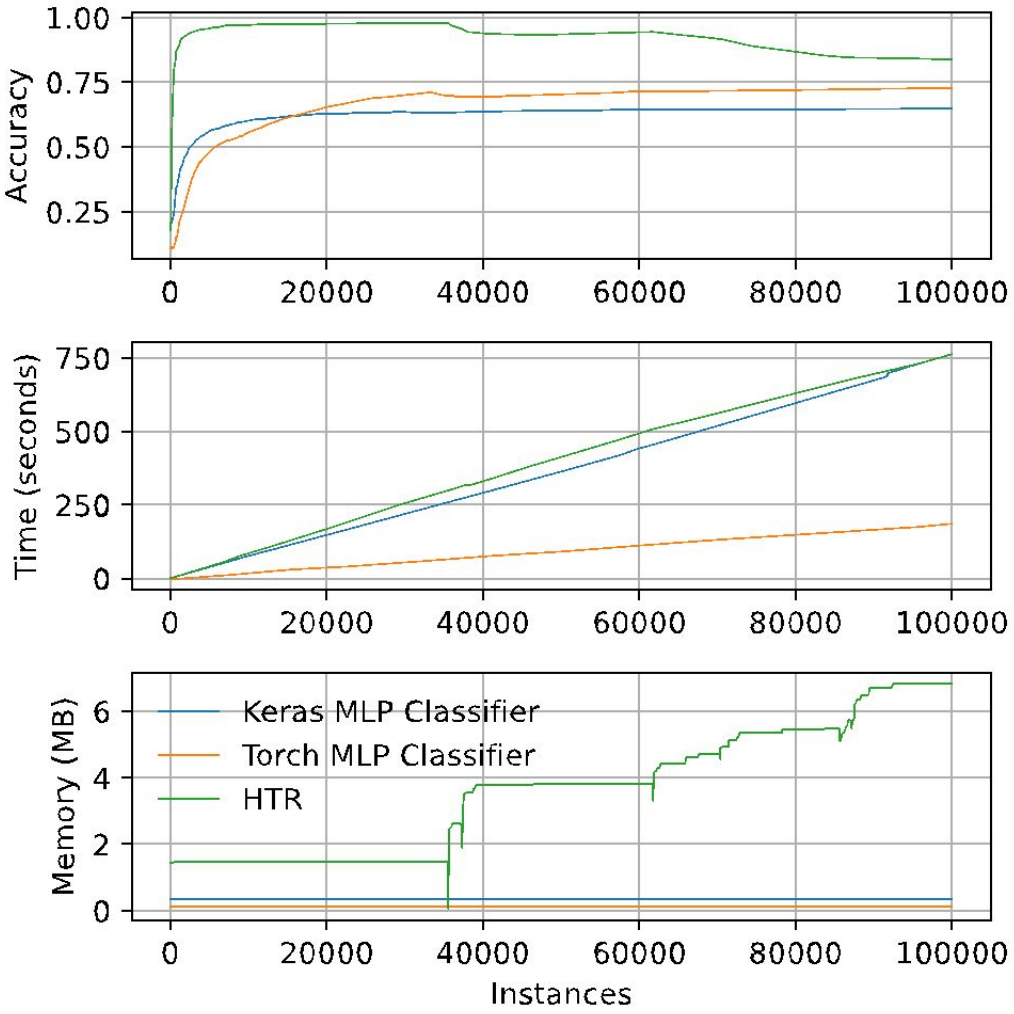
Sklearn

- Consistency (Interface)
- Inspection (Hyperparameters)
- Nonproliferation of classes (Numpy)
- Composition
- Sensible Defaults



Solving ML Tasks with deep-river

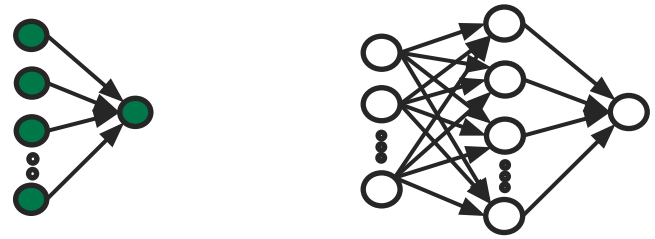
Keras vs. Torch



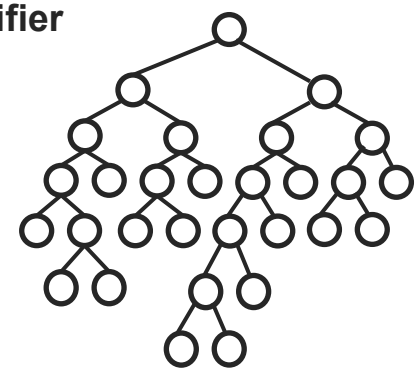
Random RBF Generator (#Features: 20, #Classes: 10)

- *Keras MLP*
- *Torch MLP*
- *Hoeffding Tree*

Neural Networks



Hoeffding Tree Classifier



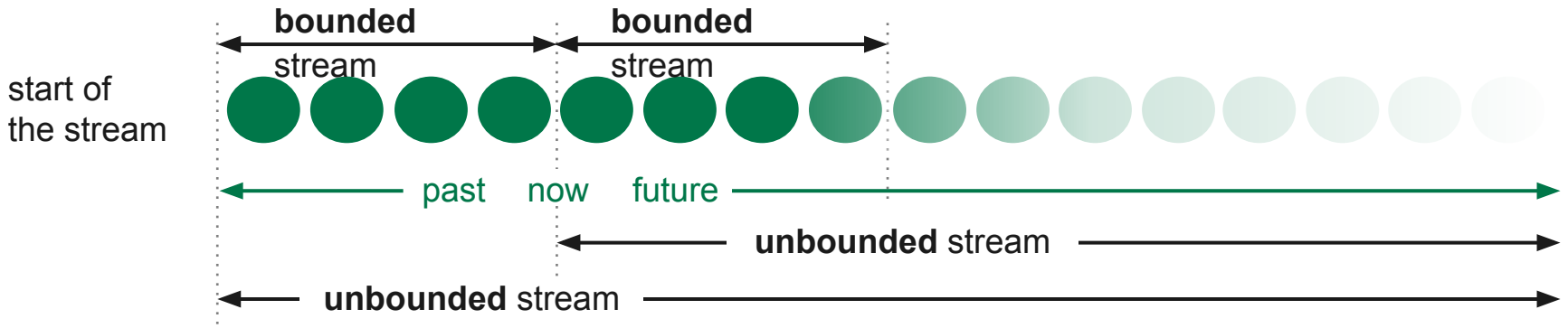
How efficient is online learning with PyTorch?

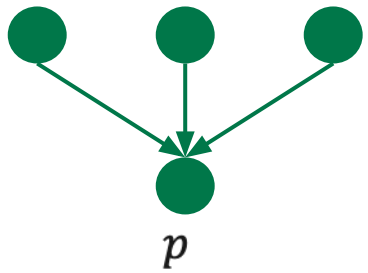
— Chances and Pitfalls of online deep learning

04

- Classification Module with new classes in Stream
- Demo: Does the usage of GPUs influence the throughput?
- Demo: Goin’ deeper

How to adapt to classes?



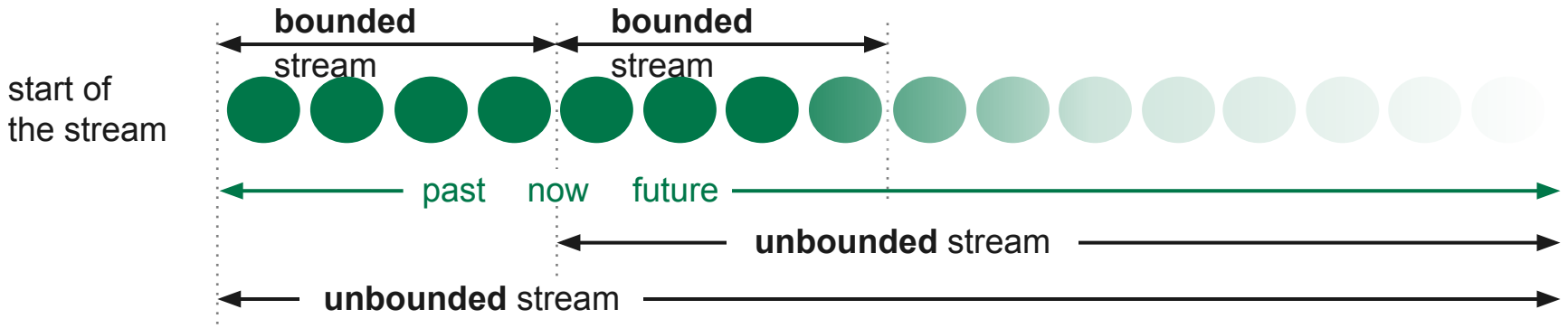


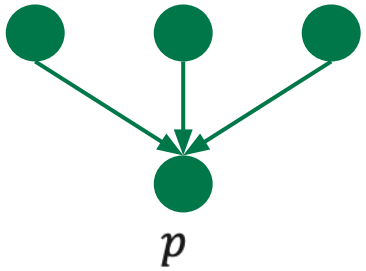
Binary classification	

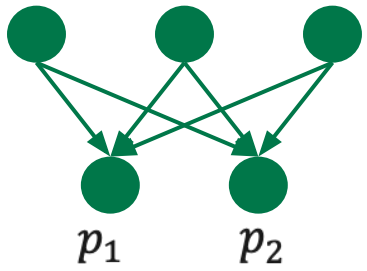
Old weights

Avg. weights from other classes

How to adapt to classes?







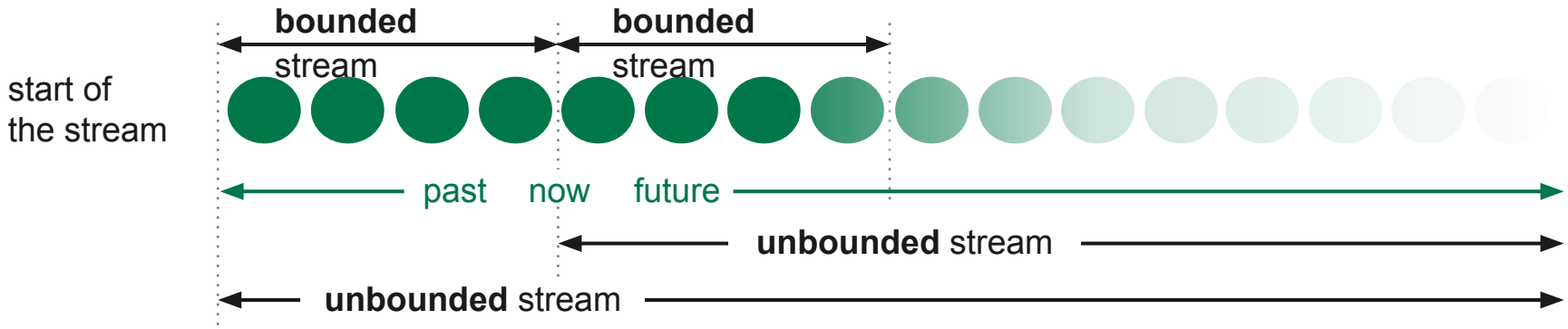
Binary classification	

Binary classification	

Old weights

Avg. weights from other classes

How to adapt to classes?

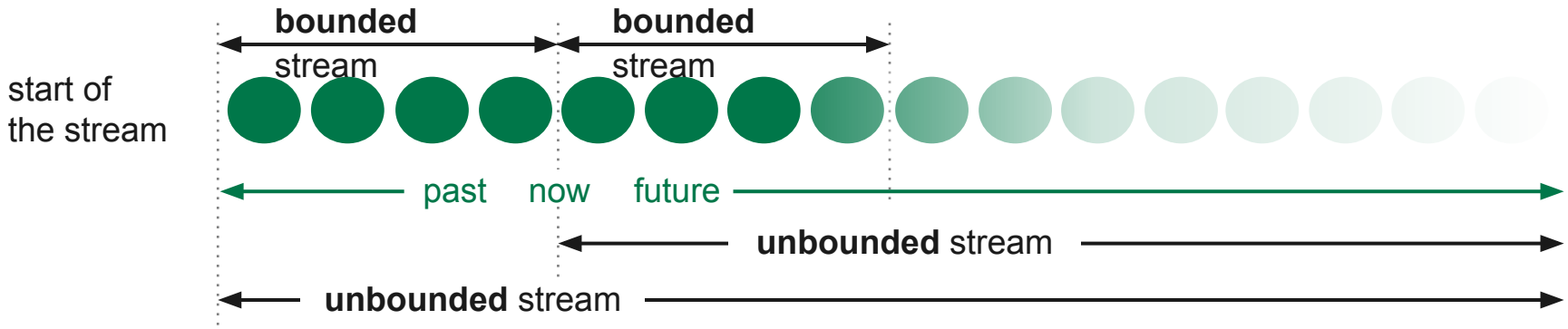


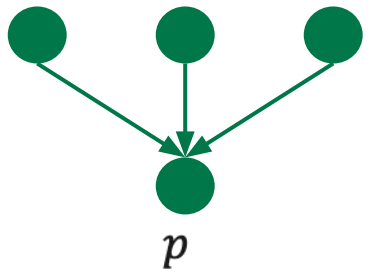
Binary classification		Binary classification		Multiclass classification	

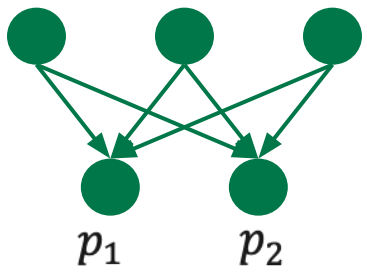
Old weights

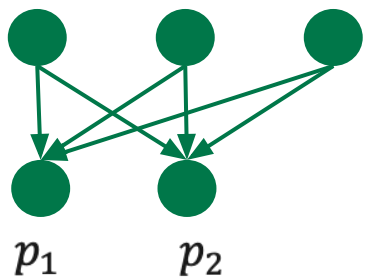
Avg. weights from other classes

How to adapt to classes?









Binary classification	

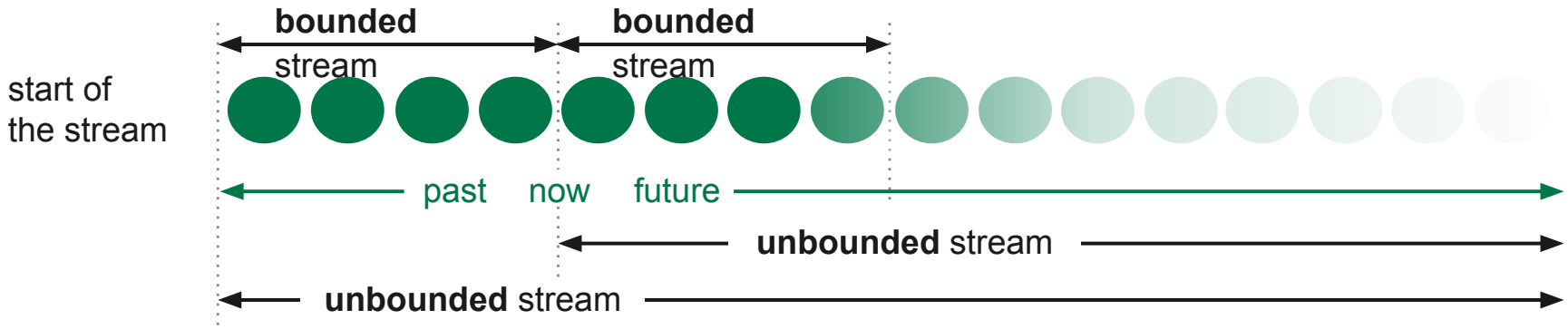
Binary classification	

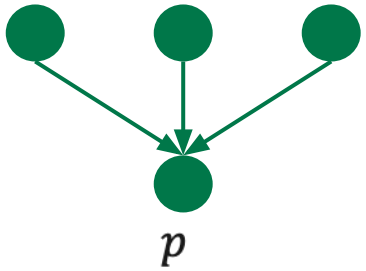
Multiclass classification	

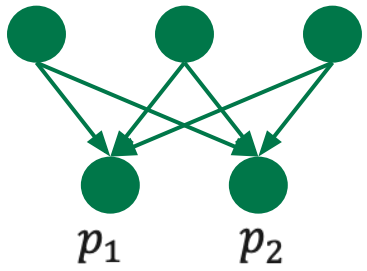
Old weights

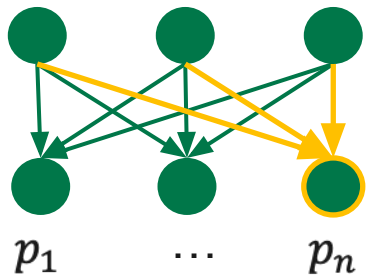
Avg. weights from other classes

How to adapt to classes?









Binary classification		Binary classification		Multiclass classification	

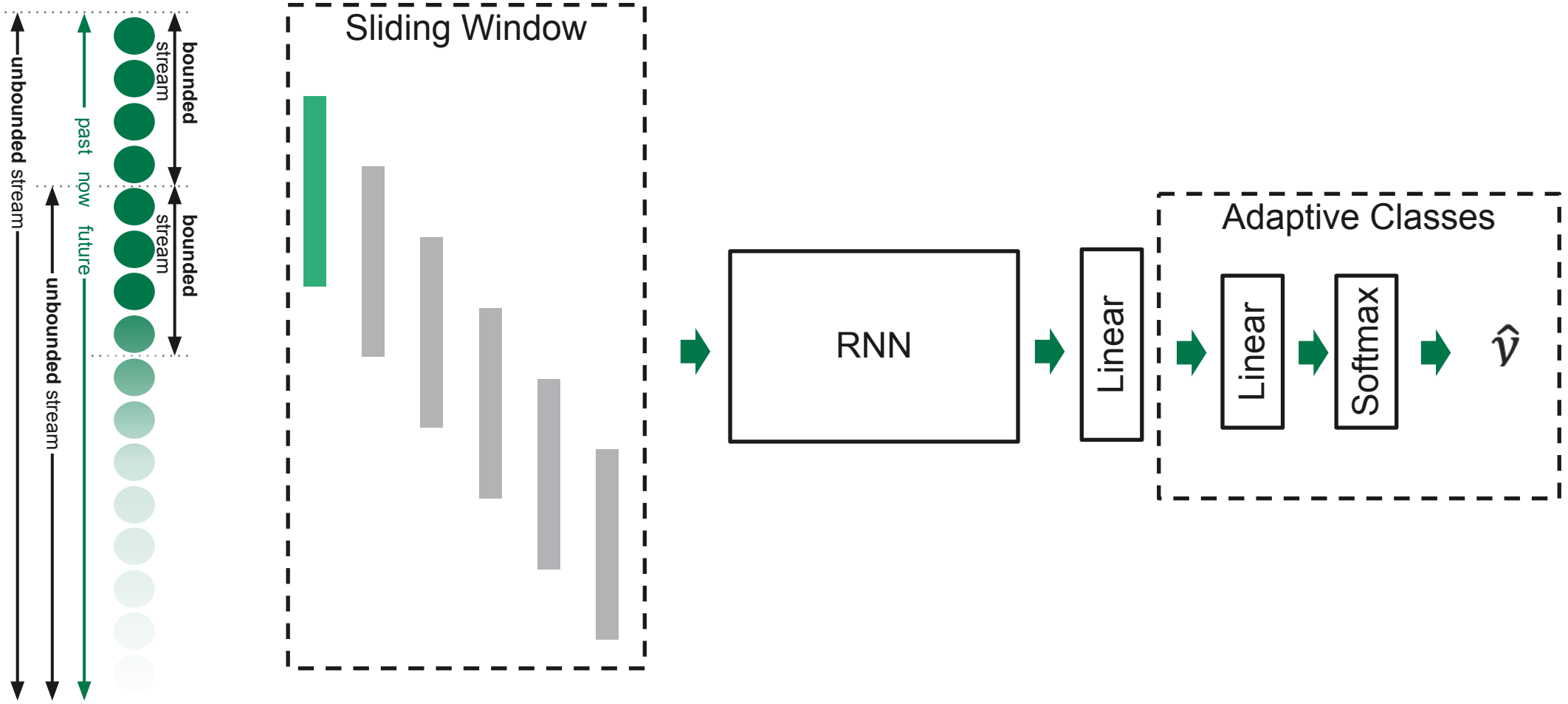
Old weights

Avg. weights from other classes

Does the usage of GPUs increase the throughput of the deep learning model?

Goin‘ Deeper

Building LSTM Models for Streaming Data



How does the integration of `PyTorch` influence the models throughput?

- From Nowcasting to Forecasting

— Conclusion

05

Chances

- Variability in Architecture leads to better predictive performance
- Adaptation Strategies are scalable
- Composition of Modules and compatibility to other Frameworks
- Optimization techniques

Pitfalls

- Data conversion overhead
- No advantage in using GPU’s (for now)

Chances

- Variability in Architecture leads to better predictive performance
- Adaptation Strategies are scalable
- Composition of Modules and compatibility to other Frameworks
- Optimization techniques

HUGE Research potential!!

Pitfalls

- Data conversion overhead
- No advantage in using GPU's (for now)

What's next

River

- Looking for Funding
- Mini-River in Rust

river-torch

- Adding model zoo of suitable NN modules
- Boost throughput of NNs
- Add visibility

BACKUP