ONLINE CLUSTERING: ALGORITHMS, EVALUATION, METRICS, CHALLENGES, APPLICATIONS AND BENCHMARKING WITH RIVER

Hoang-Anh Ngo

@Télécom Paris, IP Paris, September 2022

OUTLINE

- A (very brief) introduction to River
- Online clustering algorithms:
  - Challenges and Solutions
  - Approaches
  - State-of-the-art algorithms
  - Further steps and personal insights

River

https://hoanganhngo610.github.io/river-clustering.kdd.2022/

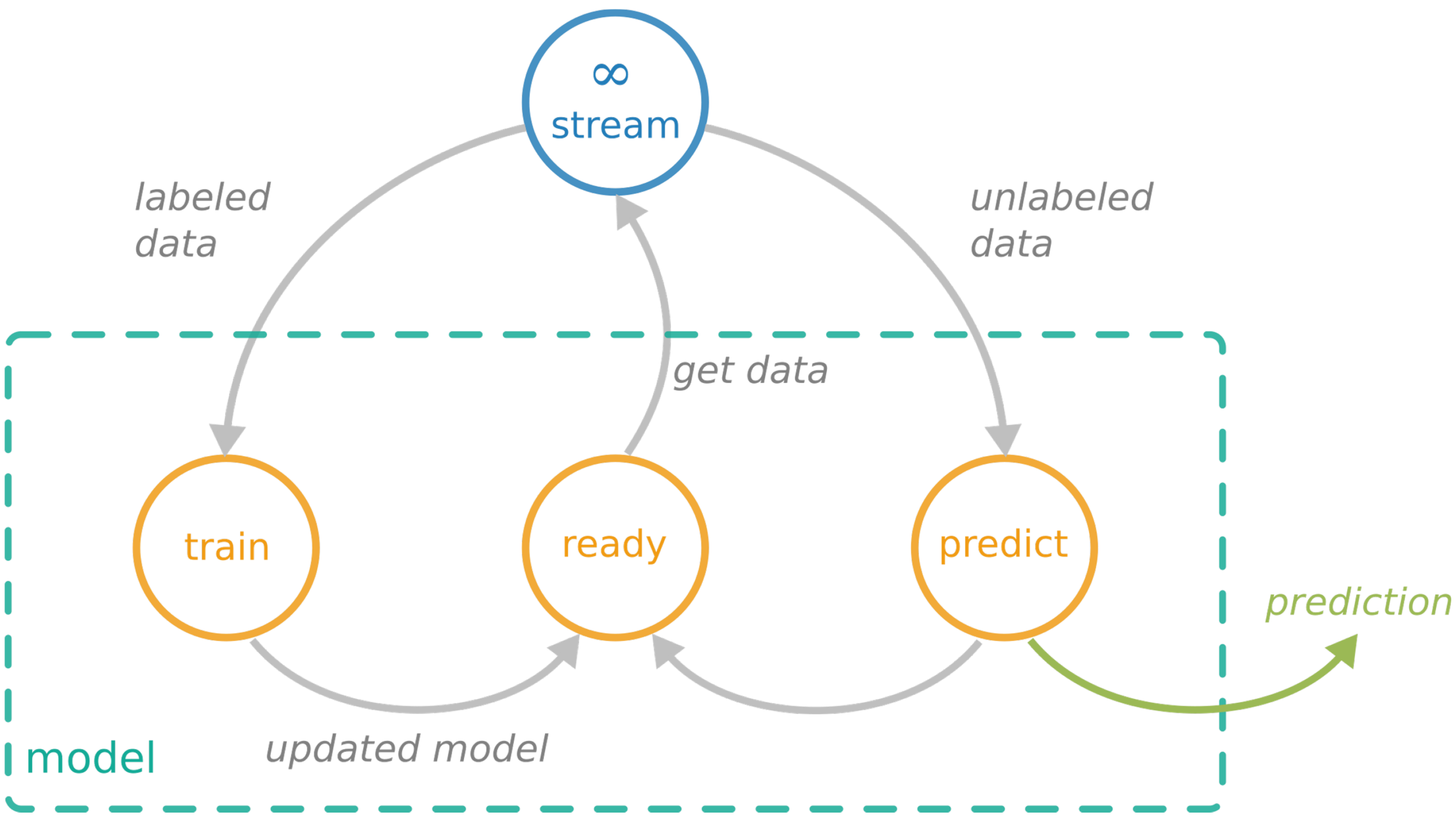
https://doi.org/10.1145/3534678.3542600

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

River

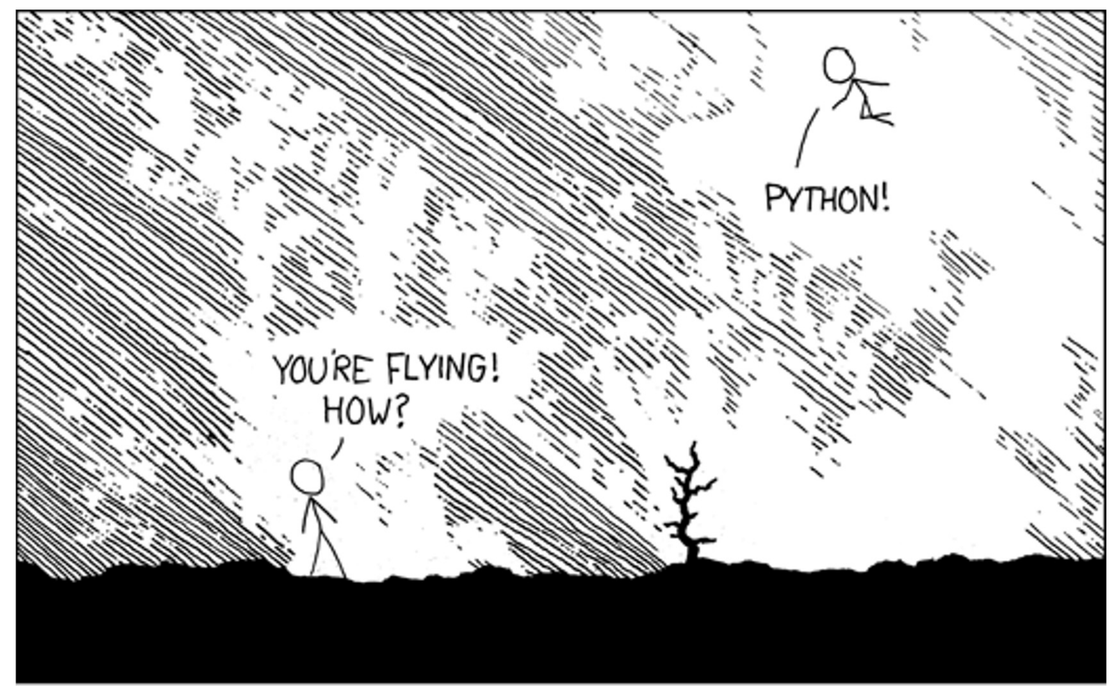
Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. 2021. **River: machine learning for streaming data in Python**. *Journal of Machine Learning Research* 22 (April 2021), 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



AVAILABLE SOFTWARES FOR ONLINE CLUSTERING

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 FOR ONLINE CLUSTERING

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 6 clustering algorithms, including incremental K-Means, CluStream, DenStream, DBSTREAM, StreamKMeans (O'Callaghan et al., 2002) and evoStream (under a fully functional Pull Request) with a clear plan of further implementations.

Includes the most number of clustering algorithms apart from MOA.

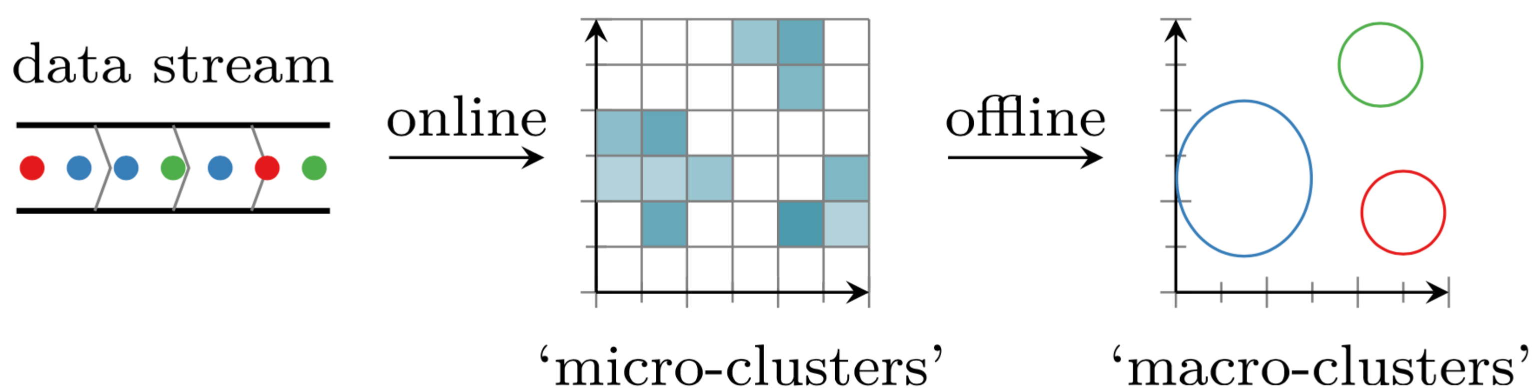
STRATEGY

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

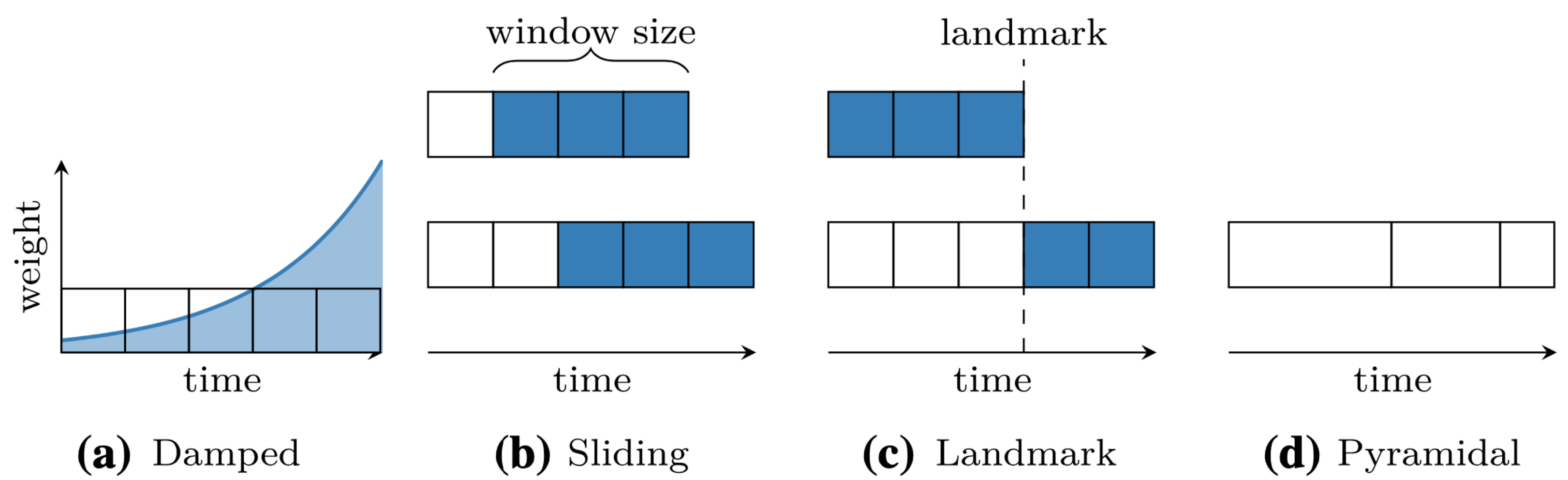
- In online clustering, historical data will be discarded and only information of formed clusters (cluster centres, number of points, linear sum, sum of squares, etc.) will be saved → Clustering algorithms are divided into **two** phases: ONLINE phase and OFFLINE phase.



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++ (coreset), STREAM, RepStream (graph of nearest neighbors);
  - *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

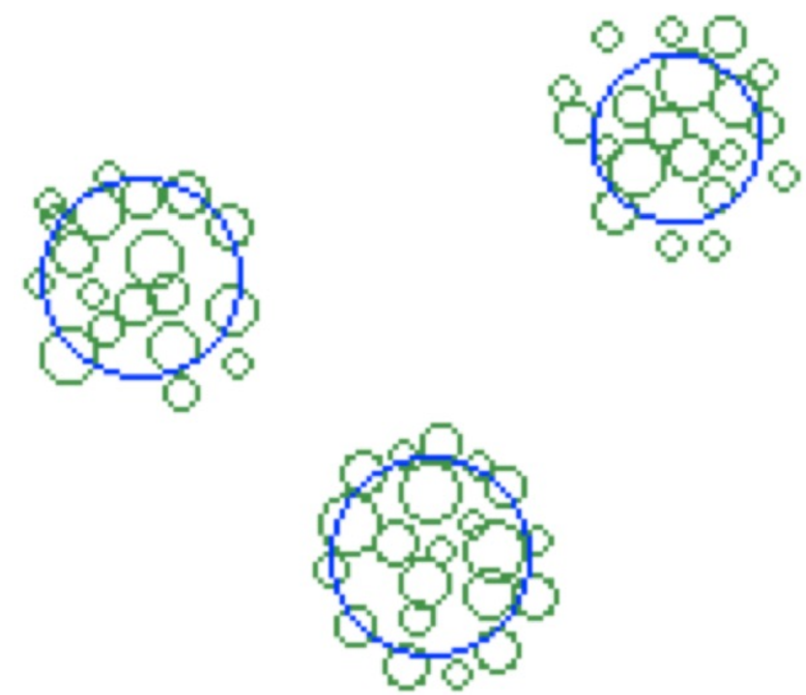
- **Model-based approach:** Summarize the data stream as a ***statistical model***, with a common area of research based on the Expectation Maximization (EM) algorithm. Others include the use of an incrementally-built classification tree or concepts from linear regression. Including CluDisStream, SWEM, 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.

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 $\overrightarrow{\mathrm{CF}}$ (statistical summary structure)
- Maintained in online phase, input for offline phase
- Data stream $\langle \overrightarrow{\mathrm{x}_\mathrm{i}} \rangle$, d dimensions
- Cluster Features vector includes
  - N: number of points
  - $\mathrm{LS_i}$: sum of values (for dimension j)
  - $\mathrm{SS_i}$: 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 $= \sqrt{\frac{2 \times N * SS - 2 \times LS^2}{N \times (N-1)}}$

WELFORD'S ALGORITHM

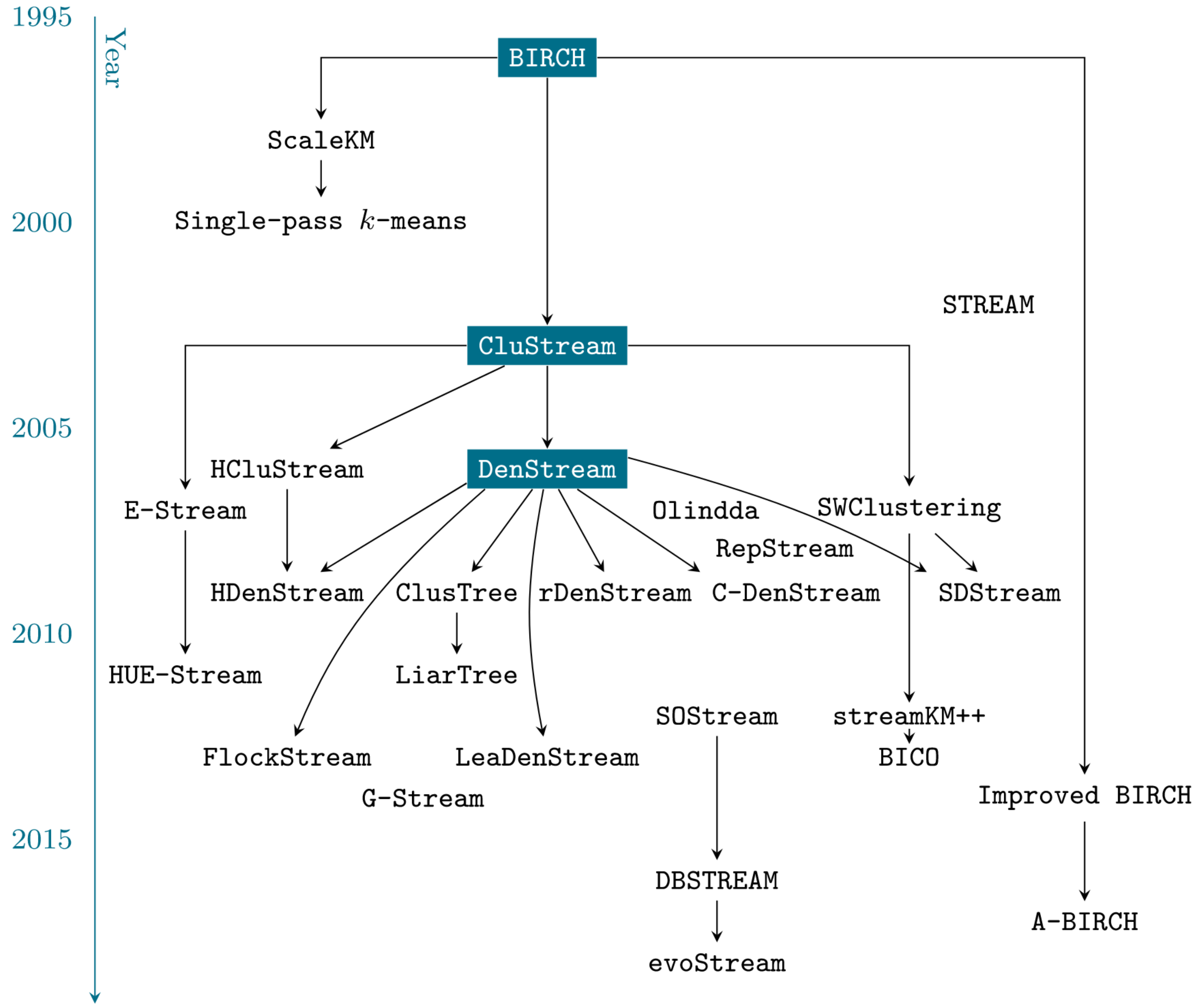
- Used as an alternative to cluster feature vectors, by calculating the variance incrementally
- Less prune to errors with large values and/or large number of observations
- Algorithm:
  - Intitialize: $\overline{x_1} = 0, \mathrm{M}_{2,1} = 0$. For any $\mathrm{n} > 1$:
    - $\overline{\mathrm{x_n}} = \overline{\mathrm{x_{n-1}}} + \frac{\overline{\mathrm{x_n}} - \overline{\mathrm{x_{n-1}}}}{\mathrm{n}}$
    - $\mathrm{M}_{2,\mathrm{n}} = \mathrm{M}_{2,\mathrm{n}-1} + (\mathrm{x_n} - \overline{\mathrm{x_{n-1}}})(\mathrm{x_n} - \overline{\mathrm{x_n}})$
    - Variance: $\frac{\mathrm{M}_{2,\mathrm{n}}}{\mathrm{n}-1}$

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

EVOLUTION OF ONLINE CLUSTERING ALGORITHS

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



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 off-springs 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)

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

Require: radius r, decay rate λ, cleanup interval t_{gap}, initialization threshold γ, Population size P, number of clusters k

Initialize: $t = 0$, $MC = \emptyset$, $\mathbf{C} = \emptyset$

while stream is active **do**
    read $\boldsymbol{x}$ from stream
    $t \leftarrow t + 1$
    $new \leftarrow (\boldsymbol{x}, t, 1)$ ▷ Temporary micro-cluster
    for $mc \in MC$ **do** ▷ $mc := (\mathbf{c}, t, \omega)$
        if $\text{DIST}(mc, new) < r$ **then** ▷ Absorb observation
            $mc[\mathbf{c}] \leftarrow mc[\mathbf{c}] + h(new[\mathbf{c}], mc[\mathbf{c}]) \cdot (new[\mathbf{c}] - mc[\mathbf{c}])$ ▷ h as in Equation (1)
            $mc[t] \leftarrow t$
            $mc[\omega] \leftarrow mc[\omega] \cdot 2^{-\lambda(t - mc[t])} + 1$
    if new has not been absorbed by any $mc \in MC$ **then** ▷ Initialize new micro-cluster
        $MC \leftarrow MC \cup new$
    if $t \mod t_{gap} = 0$ **then** ▷ Periodic adjustments
        CLEANUP$(\cdot)$ ▷ see Algorithm 2
    if $|MC| = \gamma$ **and not** initialized **then** ▷ Initialize macro-clusters
        for $i \leftarrow 1, \ldots, P$ **do**
            $C_i \leftarrow k$ randomly chosen micro-cluster
    while idle **do** ▷ Until new example available
        EVOLUTION$(\cdot)$ ▷ Repeat evolutionary step, see Algorithm 3

evoStream algorithm (with both online and offline phase)

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

function CLEANUP($\cdot$)
  for each $mc \in MC$ **do**
    $mc[\omega] \leftarrow mc[\omega] \cdot 2^{-\lambda(t - mc[t])}$ ▷ Update weight
    if $mc[\omega] \leq 2^{-\lambda t_{gap}}$ **then**
      Remove mc from MC ▷ Remove outdated
  Merge all mc_i, mc_j where DIST(mc_i, mc_j) $\leq r$ ▷ Merge colliding clusters

Cleanup phase (after each t_{gap} time interval)

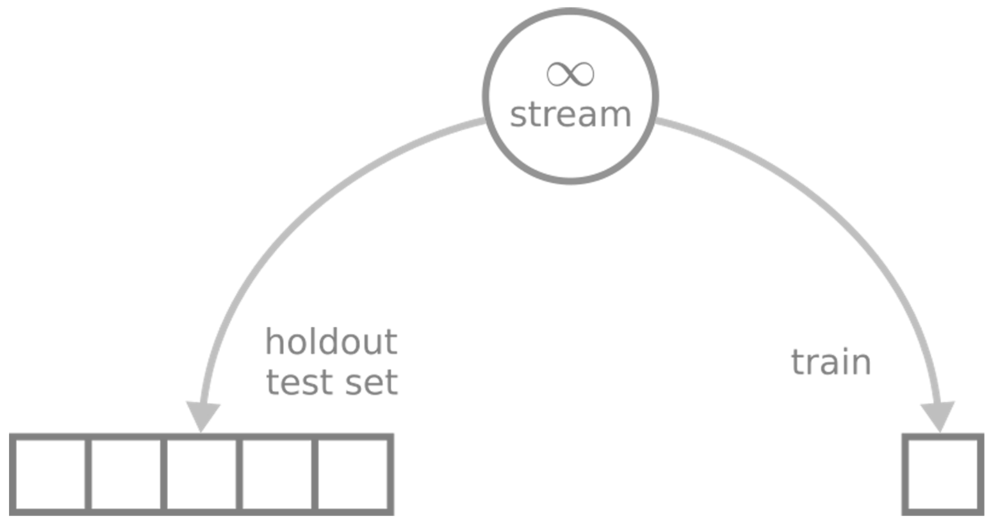
EVOSTREAM

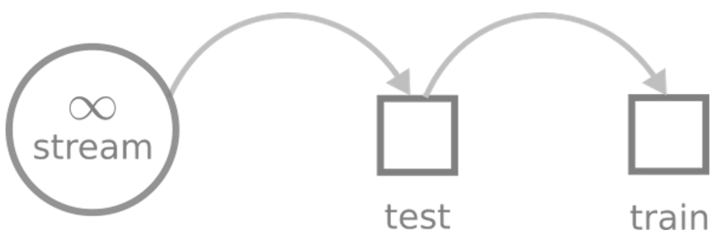
Matthias Carnein and Heike Trautmann. 2018. evoStream – Evolutionary Stream Clustering Utilizing Idle Time. *Big Data Research* 14 (2018), 101-111. https://doi.org/10.1016/j.bdr.2018.05.005

```
function EVOLUTION(·)
    p_1, p_2 ← Select two solutions proportionally to their fitness from C
    o_1, o_2 ← Create offsprings of p_1, p_2 using binary crossover
    for each g_i in o_1, o_2 do                                ▷ For each child-gene
        if RANDOM(0,1) < P_m then                              ▷ Mutate with probability P_m
            if g_i = 0 then
                g_i ← 2δ
            else
                g_i ← 2δ · g_i
    Add o_1, o_2 to C and discard the two least fittest solutions
```

Evolution algorithm (during idle time)

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).

**Work to be submitted to PAKDD 2023*

HOW TO DESIGN AN INCREMENTAL INTERNAL METRIC?

- Save the information that are needed, finite and require low computational time/resources:
  - Linear sum and/or sum of squares of distances of point x^{t} at time t to the nearest cluster center at the same time, i.e $\sum_{\mathrm{t}} \left\| \mathrm{v}_{\mathrm{c_t}}^{\mathrm{t}} - \mathrm{x}^{\mathrm{t}} \right\|_{1/2}$
  - Number of points passed (in total and/or within each cluster)
  - Centers and centroids of clusters (using incremental means)
  - Centers of the whole dataset
  - 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.

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.);

FURTHER STEPS

- Benchmarking
- Text-specific clustering algorithms (although this can currently be done using TFIDF + Any clustering model pipeline in River)

PERSONAL THOUGHTS

- The world of online clustering is still “chaotic”, with a lot of papers having no official implementation or implementations scattered in different frameworks/languages → Hard to evaluate.
- Are we too much dependent on the concept of online and offline phase while doing online clustering?
- Online deep clustering (ODC) with the assistance of `river-torch`? (https://arxiv.org/abs/2006.10645)

Thank you for your attention!