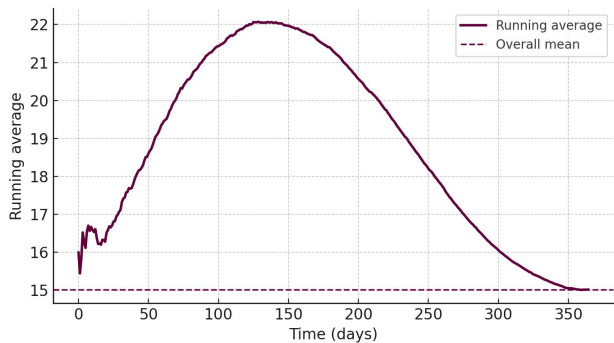


Online Shuffling for Online Machine Learning and Progressive Visualization

Pavlo Poliuha, Francesca Bugiotti, Benoit Groz, Jean-Daniel Fekete
Université Paris-Saclay, CNRS, Inria, LISN

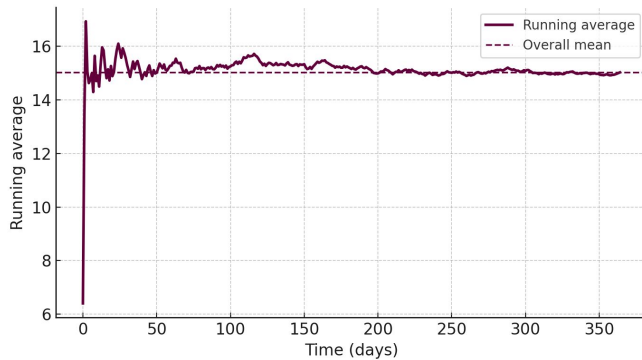
Problem



Original (sorted) order → bias

- Ordered online data:
 - biased early results
 - slow convergence.

Solution



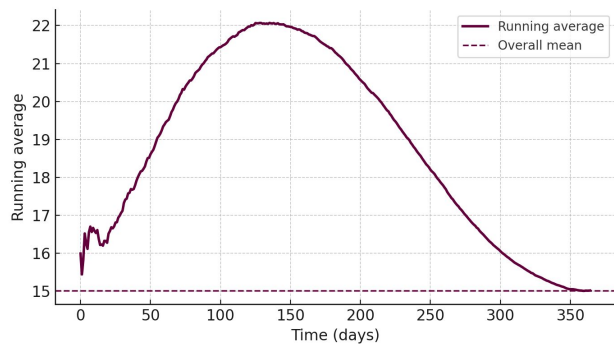
Shuffled order → fast convergence

- Middleware that shuffles remote data 'on the fly'
- Streams progressively unbiased samples to the downstream system.

Applications & Impact

- **Online ML:** faster, less biased updates.
- **Visualization:** early previews, quicker insights.
- **Efficiency:** fewer transferred bytes, faster computations → reduce energy and CO2.

Problem



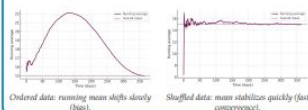
Original (sorted) order → bias

- Ordered online data:
 - biased early results
 - slow convergence.

MOTIVATION

- We want to **analyze and visualize large datasets during download** to obtain useful results early with low resources.
- When data is **progressively fetched** from remote repositories, it often arrives in **chronological order**, which biases early statistics, learning, and slows convergence.

Goal: provide a Middleware service that shuffles incoming rows "on the fly", removing bias and accelerating convergence.



IMPLEMENTATION

Middleware Model

- Input: url, time to first chunk t_0 , update period Δt
- Measures: output chunk size, throughput.



Middleware Pipeline

Timed window (TW)

- Fill reservoir until t_0 , shuffle, emit first block.
- During Δt , read new rows, shuffle that window, emit.
- Quality is limited by the window size (block boundaries).

Sliding reservoir (SR)

- Fill reservoir until t_0 , of size B .
- During Δt , pop random rows from the reservoir into a chunk and send it, and add new rows.
- The larger B (thus t_0) → the higher Q .

RESULTS

- Both strategies **reduce early bias** against the original order.
- Shuffle quality Q **grows** with the size of the window (TW) or reservoir (SR).
- The service provides a tunable balance between **responsiveness** and **shuffle quality**.

REFERENCES

- [1] Steven, C.H. Hot et al., Online learning: A comprehensive survey, 2021
- [2] Fekete, J.-D. et al., Progressive Data Analysis: Roadmap and Research Agenda, 2024
- [3] Fisher, R.A. and Yates, F., Statistical Tables for Biological, Agricultural and Medical Research, 1948
- [4] Kolmogorov, A., Sulla determinazione empirica di una legge di distribuzione, 1903
- [5] Csiszar, I., f-divergence geometry of probability distributions and minimization problems, 1975
- [6] Bhattacharyya, A., On Approximating Total Variation Distance, 2022

APPLICATIONS & IMPACT

- Online ML: faster, less biased early updates.
- Progressive visualization: early previews; stop sooner.
- Efficiency: fewer transferred bytes and faster computations → lower energy/CO₂.

SHUFFLE MODEL

- We use the Empirical shuffle matrix from an algorithm:
 $M \in \mathbb{R}^{N \times N}$ tracks, for each input index i (out of N elements), the output position j , after T independent runs:

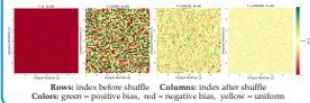
$$M_{i,j} = \frac{1}{T} \sum_{k=1}^T \mathbb{1}_{i \mapsto j}.$$

- Compare it to the Uniform baseline:
 $U \in \mathbb{R}^{N \times N}$ for an ideal unbiased shuffle, where every element is equally likely:

$$U_{i,j} = \frac{1}{N} \quad \forall i, j.$$

Fair Shuffle Simulation (Fisher-Yates)

As the number of runs T increases, M should converge toward U , i.e., the empirical distributions approach uniformity.



SHUFFLE QUALITY

- Existing metrics: Kolmogorov-Smirnov Test, Kullback-Leibler Divergence, Total Variation Distance (TVD)
- TVD (difference between M and U) is selected for its simplicity and effectiveness:

$$D_{TVD}(M, U) = \frac{1}{2N} \sum_{i=1}^N \sum_{j=1}^N |M_{i,j} - U_{i,j}|$$

- Shuffle quality (higher is better):

$$Q = 1 - D_{TVD}(M, U) \in [0, 1].$$

FUTURE WORK

- Systematic evaluation of the middleware on public datasets.
- Study dynamic reservoirs, hybrid scheduling.
- Benchmark $(t_0, \Delta t, B, c, P)$ and report Q with downstream convergence speed.

Applications & Impact

- Online ML: faster, less biased updates.
- Visualization: early previews, quicker insights.
- Efficiency: fewer transferred bytes, faster computations → **reduce energy and CO₂**.