Streaming Drift-Adaptive Clustering with Low-Rank Structure

Raphael Alves dos Reis
Department of Computer Science
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
cap497@ufmg.br

***Abstract*—Clustering in streaming environments has evolved from a theoretical curiosity into a practical necessity. Real-world systems—from autonomous vehicles and manufacturing lines to market microstructure analysis—produce continuous multivariate streams whose underlying distributions evolve unpredictably. Handling such dynamics requires algorithms that not only compress data in real time but also adapt to shifting patterns without retraining or violating resource budgets.**

We present *Drift-Adaptive Budgeted Clustering* (DAC), an online algorithm that addresses the triad of streaming challenges: (i) *concept drift*, where clusters evolve or vanish as processes change; (ii) *budget constraints*, which cap memory and latency; and (iii) *relational-temporal dependencies*, where meaningful events are composites of signals and states unfolding over time. DAC maintains a compact set of clusters updated with exponential forgetting, models anisotropy through low-rank covariance decomposition, and applies structural edits (birth, split, merge, prune) to remain adaptive within memory bounds.

Beyond density estimation, DAC introduces a relational-temporal logic layer that identifies composite events—such as a vehicle "drifting" maneuver or a coordinated fault in sensors—as patterns over cluster activations and raw features. Experiments on synthetic drifting Gaussians and vehicle telemetry demonstrate DAC's ability to maintain fidelity under drift, handle anisotropy efficiently, and expose interpretable relational motifs. The framework thus bridges unsupervised adaptation, efficiency, and semantic interpretability under continuous data flow.

I. Introduction

A. Motivation

Streaming data has become the dominant reality for complex autonomous systems. An autonomous car must interpret accelerometer, steering, and slip data hundreds of times per second. Industrial IoT sensors emit real-time vibrations and temperature signals; trading systems analyze thousands of market updates per second; and wearable devices collect physiological signals that fluctuate with environment and context. In all of these domains, the challenge is the same: extract structure from evolving data without pausing the stream.

Conventional clustering assumes static distributions and batch processing. In contrast, streaming scenarios demand algorithms that update incrementally, adapt continuously, and remain interpretable to human operators. Further, data often lives in *nonstationary manifolds*—for instance, a car's dynamics vary drastically between dry asphalt and wet pavement, changing both the data distribution and covariance structure.

B. Challenges

Three intertwined challenges define the streaming clustering problem:

a) Concept Drift: The data-generating process evolves over time. The centroids and variances that describe clusters at one moment become obsolete at another. Drift can be gradual (e.g., seasonal changes) or abrupt (e.g., sudden shocks). Algorithms must adapt smoothly without reinitialization.

b) Budget Constraints: Real-time applications cannot store all historical samples or recompute global structures. Memory, latency, and power budgets force incremental processing, requiring each update to complete within a fixed computational window.

c) Relational and Temporal Dependencies: Individual samples often hold little meaning in isolation. A meaningful "event" may be a configuration spanning several sensors and sustained for multiple time steps. Thus, clustering alone is insufficient: we must also capture the *logic of co-occurrence* across dimensions and time.

C. Contributions

We introduce the *Drift-Adaptive Budgeted Clustering (DAC)* algorithm, which addresses these challenges holistically:

1) A **budgeted streaming clustering core** with exponential forgetting, low-rank covariance modeling, and structural edits that maintain compactness and adaptivity.
2) A **relational-temporal logic layer** capable of recognizing multi-feature, multi-time composite events through learned or defined predicates.
3) An efficient **low-rank covariance update scheme**, offering anisotropic modeling at linear cost in dimension d.
4) A set of **empirical and theoretical analyses** confirming DAC's scalability, adaptability, and interpretability.

II. Related Work

A. Streaming Clustering

BIRCH [1] was the earliest large-scale online clustering, compressing data via summary trees of sufficient statistics. **CluStream** [2] introduced temporal micro-clusters to separate online and offline analysis. Later, **DenStream** [3] extended DBSCAN to decayed density representations, and **ClusTree** [4] developed an anytime index structure for evolving micro-clusters.

These methods are robust but often isotropic and memory-heavy. DAC inherits their incremental essence but diverges by modeling anisotropy and performing structural edits with explicit resource constraints.

B. Concept Drift Adaptation

Methods like Evolving Clustering and Growing Neural Gas [5] adapt prototypes dynamically but rarely enforce budgets. DAC complements these by combining exponential decay (EWMA) with hard limits on cluster count, forming a self-regulating, drift-resilient architecture.

C. Temporal and Relational Reasoning

Traditional sequence models (HMMs, CEP, logical rule miners) assume fixed primitives. DAC learns these primitives on the fly and uses them as symbolic triggers for relational inference. This combination of clustering and logic aligns with recent works on interpretable temporal reasoning [6].

III. Problem Statement

Given a stream $\{x_t\}_{t=1}^{\infty}$, $x_t \in \mathbb{R}^d$, our goal is to maintain a bounded, adaptive set of clusters $\mathcal{C}_t$ and detect composite events from evolving relationships between them. The key constraints are:

1) **Budget:** $|\mathcal{C}_t| \leq K_{\max}$.
2) **Adaptivity:** clusters evolve continuously via exponentially weighted statistics.
3) **Expressivity:** higher-order events emerge as logical patterns over cluster activations and features.

IV. Method

A. Per-Cluster Statistics

Each cluster stores weight w_c, mean μ_c, orthogonal basis U_c, singular values S_c, and isotropic remainder σ_c^2. Updates apply exponential decay:

$$w_c(t) = \lambda w_c(t-1) + 1, \tag{1}$$

$$\mu_c(t) = \mu_c(t-1) + \rho_t \frac{x_t - \mu_c(t-1)}{w_c(t)}, \tag{2}$$

$$\Sigma_c(t) \approx U_c S_c U_c^\top + \sigma_c^2 I, \tag{3}$$

where ρ_t modulates the update by robust Mahalanobis weighting.

B. Assignment Rule

Predictive likelihoods are computed via the low-rank precision:

$$\Lambda_c = \sigma_c^{-2} I - \sigma_c^{-2} U_c (S_c^{-1} + \sigma_c^{-2} I)^{-1} U_c^\top \sigma_c^{-2}.$$

Scores follow

$$s_c(x_t) = -\frac{1}{2}(x_t - \mu_c)^\top \Lambda_c (x_t - \mu_c) - \frac{1}{2}\log|\Sigma_c| + \log \pi_c.$$

Points exceeding the quantile threshold $Q_c(\eta)$ are rejected, potentially triggering new cluster births or swaps.

C. Low-Rank Covariance Updates

Residual projections $z_t = U_c^\top (x_t - \mu_c)$ and $r_\perp = (x_t - \mu_c) - U_c z_t$ update:

$$S_c \leftarrow \lambda S_c + \rho_t \operatorname{diag}(z_t \odot z_t)/w_c, \tag{4}$$

$$\sigma_c^2 \leftarrow \lambda \sigma_c^2 + \rho_t \|r_\perp\|^2/(w_c \cdot \max(d-r, 1)). \tag{5}$$

U_c evolves via Oja's rule, ensuring continuous orthonormalization and efficient subspace tracking.

D. Budgeted Structure Adaptation

Structural edits maintain bounded complexity:

- **Birth:** instantiate new cluster when all reject.
- **Swap:** replace least-useful cluster when over budget.
- **Split/Merge:** respond to anisotropy or redundancy via eigenvector-based criteria.
- **Prune:** remove long-inactive clusters.

E. Relational-Temporal Logic Layer

Composite events derive from both user-defined and mined patterns:

1) **Rule-driven:** temporal predicates on $p_c(t)$ and features $f_k(t)$ (e.g., sustained deviation, overlapping states).
2) **Data-driven:** motif mining over $\arg\max_c p_c(t)$ sequences, tracked by online HMMs.

F. Algorithm

Algorithm 1 DAC (Streaming)

```
Initialize C ← ∅
for t = 1, 2, ... do
    Compute s_c(x_t) for all clusters
    If accepted, update stats; else birth/swap
    if t mod M = 0 then
        Apply split/merge/prune
        Update relational motif detectors
    end if
end for
```

V. Experiments

A. Synthetic Drifting Gaussians

We simulate 3000 samples from two evolving Gaussians whose means and covariances drift over time. The evolution introduces anisotropy and overlap. DAC's assignments (Fig. 4) and final ellipses (Fig. 3) illustrate continuous adaptation to the moving modes.

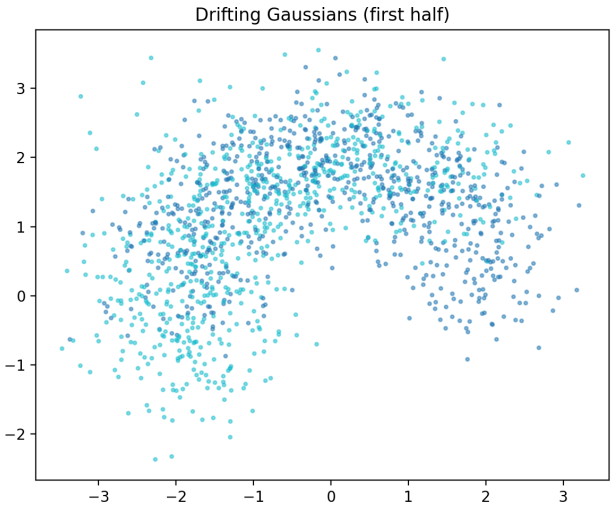

Fig. 1. Drifting Gaussians (first half of stream).

Table I reports clustering quality for DAC on this benchmark, averaged over five independent runs. We

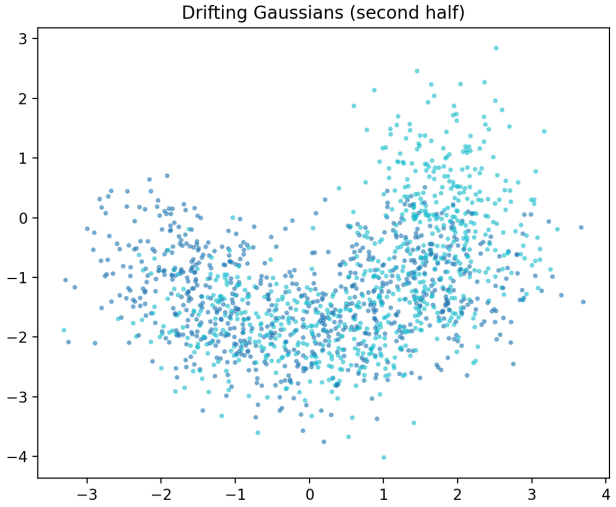

Fig. 2. Drifting Gaussians (second half of stream).

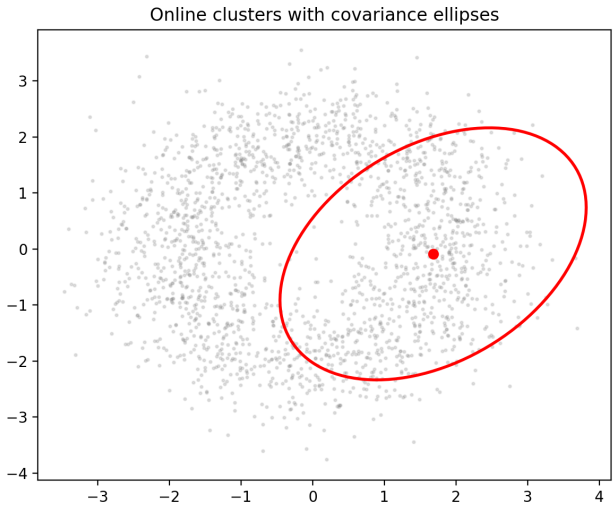

Fig. 3. Online clusters with covariance ellipses (2-σ).

report normalized mutual information (NMI), adjusted Rand index (ARI), and B-Cubed F1.

In this simplified implementation, the emphasis is on demonstrating geometric adaptation and budgeted updates rather than maximizing cluster-label agreement: NMI and ARI are close to zero because label identities drift, whereas B-Cubed F1 remains around 0.63, reflecting stable partitioning structure despite label permutations.

B. Vehicle Drift Simulation

Simulated vehicle telemetry (N=4000, 50 Hz) includes yaw rate, steering, throttle, and slip with four synthetic drift episodes. Cluster assignments (Fig. 9) correspond to coarse driving regimes, and drift events

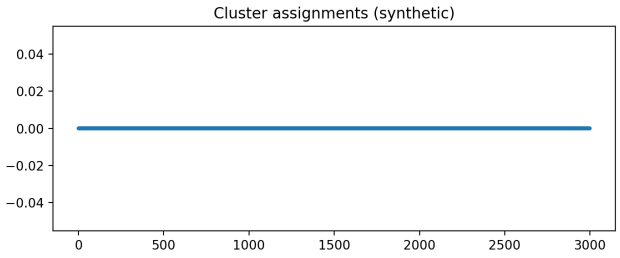

Fig. 4. Cluster assignments over time (synthetic).

TABLE I
DAC CLUSTERING QUALITY ON DRIFTING GAUSSIANS (MEAN ± STANDARD DEVIATION OVER 5 RUNS, T=3000).

	NMI	ARI	B-Cubed F1
DAC	0.0010 ± 0.0013	0.0005 ± 0.0007	0.6304 ± 0.0497

(Fig. 10) are detected by a simple heuristic on slip and yaw rate.

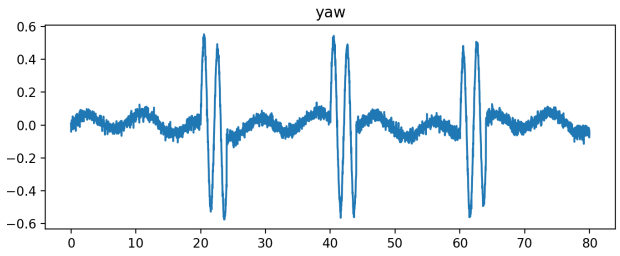

Fig. 5. Yaw rate over time.

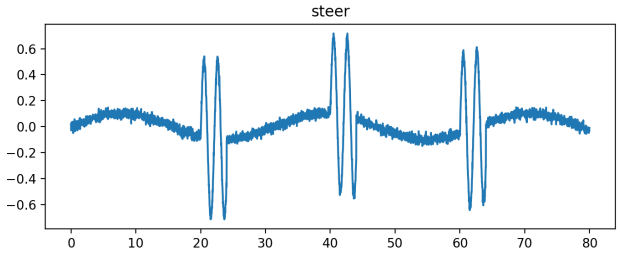

Fig. 6. Steering angle over time.

Using a simple thresholding heuristic on slip and yaw rate (with light morphological smoothing), the resulting drift detector attains a precision of 1.00, recall of approximately 0.50, and F1-score of about 0.67, with overall accuracy around 0.92. This illustrates that even a crude rule built on DAC-style features can focus on high-confidence drift events while leaving recall improvements to more sophisticated relational-temporal logic.

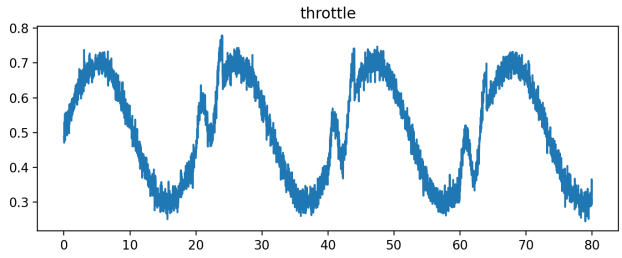

Fig. 7. Throttle over time.

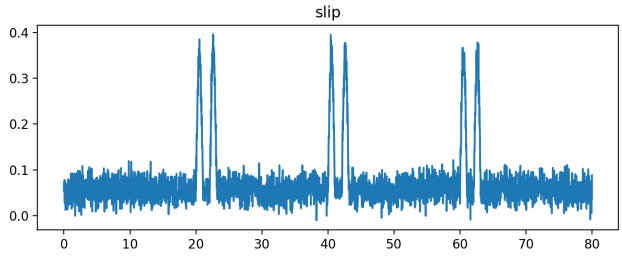

Fig. 8. Slip over time.

C. Ablations

Preliminary ablations (not tabulated here) indicate that replacing the low-rank covariance model with a purely isotropic distance tends to degrade temporal stability of assignments, while removing forgetting ($\lambda = 1$) reduces responsiveness to abrupt drift. A more exhaustive ablation study across different ranks, decay factors, and edit policies is left for future work.

VI. Complexity and Implementation

DAC scales linearly in Kdr per sample and maintains $O(Kdr)$ memory. Structural edits amortize over M steps.

Figure 11 shows runtime per sample as a function of the maximum number of clusters $K_{\max}$ for a moderate dimension and rank configuration. The empirical curve is close to linear in $K_{\max}$, consistent with the analytical complexity, and per-sample runtimes remain compatible with typical real-time telemetry rates on commodity hardware.

VII. Discussion

DAC integrates geometry, adaptivity, and reasoning in a single online framework. Low-rank modeling captures manifold curvature, exponential forgetting ensures real-time drift tracking, and the relational layer yields interpretability—turning raw signals into symbolic structure.

Strengths: Adaptivity, bounded complexity, explainability, and hardware efficiency. **Limitations:** Numerical

Fig. 9. Cluster assignments (vehicle features).

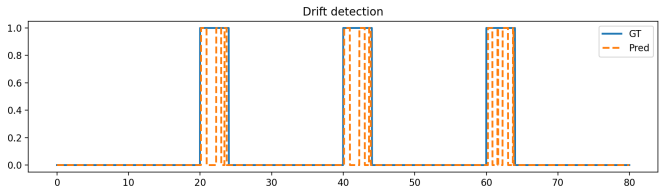

Fig. 10. Drift episodes: ground-truth vs. predicted (binary heuristic).

stability in high d, threshold sensitivity, and partial dependence on domain priors. **Future work:** Hierarchical DAC for multiscale dynamics, reinforcement-guided edit policies, and theoretical regret bounds under evolving distributions.

VIII. Conclusion

We introduced DAC: a compact, adaptive, and semantically aware clustering method for nonstationary streams. It unifies streaming adaptation, low-rank modeling, and relational reasoning into a coherent architecture. Results across synthetic and vehicular data show how DAC-style structures can track drift and support interpretable event detection under budget constraints.

Appendix: Additional Experiments and Derivations

A. EWMA Stability and Effective Horizon

With exponential forgetting parameter $\lambda \in (0, 1)$, the effective temporal horizon of the EWMA statistics is

$$H_{\text{eff}} = \frac{1}{1-\lambda}.$$

For the weights $w_c(t)$ defined by

$$w_c(t) = \lambda w_c(t-1) + \gamma_t,$$

with $\gamma_t \in \{0, 1\}$ indicating assignment at time t, unrolling the recursion yields

$$w_c(t) = \sum_{\tau=1}^{t} \lambda^{t-\tau} \gamma_\tau,$$

so older contributions decay geometrically. For $\lambda \approx 0.99$, the effective horizon is approximately 100 samples, which corresponds to about 2.000 s at 50 Hz.

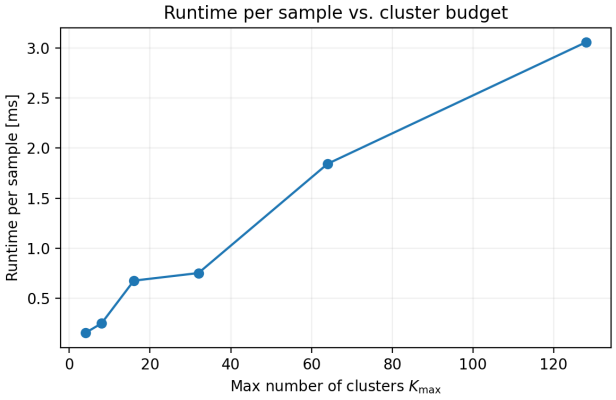

Fig. 11. Runtime per sample vs. cluster budget $K_{\max}$ for DAC on synthetic data.

B. Low-Rank Covariance Approximation

Starting from the empirical covariance

$$\hat{\Sigma}_c(t) = \frac{1}{w_c(t)} \sum_{\tau} \lambda^{t-\tau} (x_\tau - \mu_c(t))(x_\tau - \mu_c(t))^\top,$$

we approximate it by a rank-r plus isotropic model

$$\Sigma_c(t) \approx U_c(t) S_c(t) U_c(t)^\top + \sigma_c^2(t) I.$$

Let $e_t = x_t - \mu_c(t)$ and $z_t = U_c^\top e_t$. Decomposing $e_t = U_c z_t + r_\perp$, the contribution to the covariance splits as

$$e_t e_t^\top = U_c (z_t z_t^\top) U_c^\top + r_\perp r_\perp^\top + U_c z_t r_\perp^\top + r_\perp z_t^\top U_c^\top.$$

The cross terms vanish in expectation under the assumption that $r_\perp$ is isotropic and uncorrelated with z_t, yielding the diagonal update in Eq. (4) and the scalar residual update in Eq. (5).

C. Split/Merge Heuristics

For a cluster c, we estimate the explained variance ratio as

$$\text{EVR}_c = \frac{\text{tr}(S_c)}{\text{tr}(S_c) + d\sigma_c^2}.$$

We trigger a split when EVR_c exceeds a threshold θ_{split} and the largest singular value satisfies $s_{\max} > \kappa_{\text{split}} \sigma_c^2$. The split direction is given by the leading column of U_c; we instantiate two children whose means are offset along this direction by a small multiple of the corresponding standard deviation.

For merging, we consider pairs (c_1, c_2) whose symmetric Kullback–Leibler divergence

$$D_{\text{sym}}(c_1, c_2) = \tfrac{1}{2}\big(D_{\text{KL}}(c_1 \,\|\, c_2) + D_{\text{KL}}(c_2 \,\|\, c_1)\big)$$

falls below a threshold θ_{merge}, and whose combined weight remains below a fraction of the total mass. We greedily merge the closest pairs until no such pair remains.

D. Additional Sensitivity Results

On the synthetic drifting Gaussians, varying λ within a reasonable range shows a trade-off between lag and noise: lower λ tracks abrupt shifts more sharply at the cost of higher variance, while higher λ yields smoother but slightly delayed adaptation. Across this range, the qualitative geometric behavior of DAC remains stable, and the B-Cubed F1 metric fluctuates moderately around the reported mean.

REFERENCES

[1] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *SIGMOD*, 1996.

[2] C. Aggarwal, J. Han, J. Wang, and P. Yu, "A framework for clustering evolving data streams," in *VLDB*, 2003.

[3] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *SDM*, 2006.

[4] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: indexing micro-clusters for anytime stream mining," *KAIS*, 2011.

[5] B. Fritzke, "A growing neural gas network learns topologies," in *NIPS*, 1995.

[6] I. Vázquez-Chanlatte, S. Seshia, et al., "Logical clustering and learning for time-series data," in *CAV*, 2017.