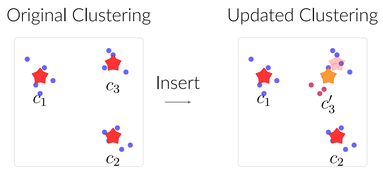
Online Clustering with Nearly Optimal Consistency

T-H. Hubert Chan[1] Shaofeng H.-C. Jiang[2] Tianyi Wu[2] Mengshi Zhao[1]

[1]The University of Hong Kong [2]Peking University

Consistent Clustering

- k-Means:
  - **Input:** a data set $P \subseteq \mathbb{R}^d$ with an aspect ratio Δ.
  - **Goal:** find a set of k centers $C \subset \mathbb{R}^d$ to minimize the cost function
$$\mathrm{cost}(P, C) := \sum_{x \in P} (\mathrm{dist}(x, C))^2.$$
- Online k-Means:
  - Data points arrive in an arbitrary order (assume points are in $[\Delta]^d$). The algorithm must decide whether and where to define a new center when a new point arrives.
  - **Competitive ratio:** the ratio between the algorithm's k-Means cost and the optimal k-Means cost (with full information).
  - **Lower bound**: No bounded competitive ratio [LSS16].
- Consistent online k-Means: Recourse of decisions is allowed [LV17].
  - **Consistency:** $\sum_i |C_i \setminus C_{i-1}|$ where C_i is the center set after the algorithm processes the i-th input point.
  - **Goal:** Minimizing the consistency while maintaining a bounded competitive ratio.
  - **Lowerbound:** any $O(1)$-competitive algorithms for k-Means must be $\Omega(k \log(n))$-consistent [LV17].



Related Works and Our Contribution

Algorithm	Competitiveness	Consistency
[LV17](k-Median, k-Means)	$O(1)$	$\tilde{O}(k^2)$
[FLNS21](k-Median)	$O(1)$	$\tilde{O}(k)$
Our work (k-Median, k-Means)	$1+\epsilon$	$\tilde{O}_\epsilon(k)$

Our work closes the following gap:

- k-Means also admits $O(1)$-competitive ratio with $O(k\,\mathrm{polylog}(n))$-consistency.
- Moreover, it can achieve $(1+\epsilon)$-competitive ratio and $O(k\,\mathrm{polylog}(n))$-consistency (might require an exponential running time).
- There is a framework that turns any offline clustering algorithm into an online one with near-optimal consistency.

Main Theorem

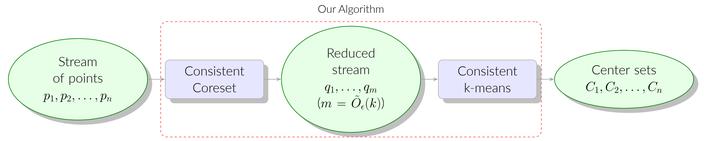
Given an offline α-approximate algorithm for k-Means that runs in $T(n)$ time for n points, there exists an $\tilde{O}_\epsilon(k)^a$-consistent $(1+\epsilon)\alpha^2$-competitive algorithm for online k-Means, and the running time is $\tilde{O}_\epsilon(nk + k^3 \cdot T(\tilde{O}_\epsilon(k)))$.

- This result more generally works for (k, z)-Clustering (which particularly contains k-Median), in addition to k-Means.
- Plugging in the brute-force exact offline algorithm leads to a $(1+\epsilon)$-competitive $O(k\,\mathrm{polylog}\,n)$-consistent algorithm for k-Means.

[a] $\tilde{O}_\epsilon(k)$ hides $\mathrm{poly}(\epsilon^{-1} d \log(n))$ factor.

Our Algorithm

- Consistent Coreset Construction: reduce the number of points to be considered: from n points to $\tilde{O}_\epsilon(k)$ weighted points.
- Algorithms for Bounded Input: achieving $\tilde{O}_\epsilon(1)$-amortized consistency.
- $\tilde{O}_\epsilon(1)$-amortized consistency + $\tilde{O}_\epsilon(k)$ points $\to$ $\tilde{O}_\epsilon(k)$-consistent.



Step1: Consistent Coreset

- ϵ-coreset: a weighted set $S \subseteq \mathbb{R}^d$ such that $\forall C \subseteq \mathbb{R}^d, |C| = k$, $\mathrm{cost}(S, C) \in (1+\epsilon)\,\mathrm{cost}(P, C)$.

Theorem for Consistent Coreset

There is an algorithm that given $\epsilon \in (0,1)$ and an online sequence $(p_1, \ldots, p_n)$ in $[\Delta]^d$, outputs a weighted sequence $\vec{D}_1, \ldots, \vec{D}_t$ where, w.p. $1 - \mathrm{poly}(\frac{1}{n})$, $\vec{D}_i$ is an ϵ-coreset for $P_i = \{p_1, \ldots, p_i\}$, and the sequence is $\tilde{O}_\epsilon(k)$-consistent. It runs in $\tilde{O}_\epsilon(nk)$ time.

- It is always a coreset for any prefix of the stream;
- Points sampled into the data structure are never deleted from the incremental coreset.

Step 2: Consistent Clustering on the Coreset

Theorem for Consistent Clustering for Bounded Input

Suppose an offline α-approximation algorithm for k-means runs in $T(n)$ time for n points. Then there exists an algorithm that is $(1+\epsilon)\alpha^2$-competitive and $\tilde{O}_\epsilon(m)$-consistent for the online k-means problem with initial set P_0 and stream $p_1, \ldots, p_m$ (where $P_i = P_0 \cup \{p_1, \ldots, p_i\}$), such that the optimal cost after adding all points is at most twice the initial cost.

The algorithm processes points in batches; for each batch, it

- Starts with k centers C_0.
- Deletes maximum possible ℓ centers from C_0 while maintaining solution quality.
- Adds new points if total centers $\le k$.
- If centers reach k, swaps $\tilde{O}_\epsilon(\ell)$ centers from C_0 (forgetting the intermediate steps).
- Run subroutine MakeRobust to prepare for next batch.

Detail of the Subroutine MakeRobust

We use a similar strategy as in [FLNS21], where the key idea is to use "robust centers". A "robust center" is a center such that it is *approximately locally optimal* – any other center that is close enough to it cannot significantly improve the approximation ratio. It enables our algorithm to build cluster centers that remain effective even as data evolves over time, without requiring knowledge of future changes.

Visualization of the Algorithm

We run the subroutine MakeRobust after Step 4, which is introduced in an independent block.

Comparison with [FLNS21]: The algorithm is based on the framework of [FLNS21]. The original framework only works for k-Median and achieves only an $O(1)$-approximation ratio. Our work

- conducts new analysis techniques to bypass the integrality gap of linear programming in [FLNS21], achieving $(1+\epsilon)$-approximate ratio.
- generalizes the result from k-Median to general (k, z)-clustering, including k-Means.

Experiments

Datasets

- SKIN [BD09]: 3-dimension, 245057 points;
- SHUTTLE [Cat]: 7-dimension, 58000 points;
- COVERTYPE [Bla98]: 54-dimension, 581012 points.

Algorithms:

- Baseline1 (LV17): Algorithm in [LV17].
- Baseline2 (naive): Directly running k-Means++ on the consistent coreset instead of running any additional algorithm.
- Our algorithm (ours-faithful): plugging k-Means++ to our framework.
- A heuristic implementation of our algorithm (ours-heuristic): when a new point is added, it replaces only one existing center if doing so maximizes cost reduction.

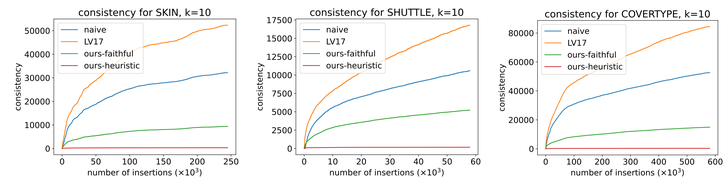


Figure 1. The consistency curve over the insertions of points, for all datasets and $k = 10$.

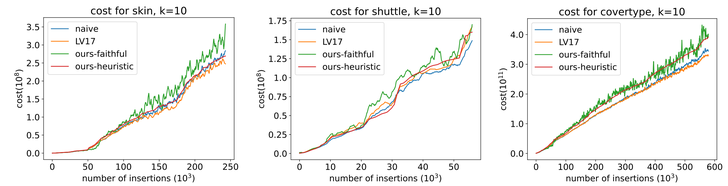


Figure 2. The cost curve over the insertions of points, for all datasets and $k = 10$. We plot the curve after applying a moving average with a window size equal to 1% of the dataset size.