NEURAL INFORMATION PROCESSING SYSTEMS

NeurIPS 2026

Mini-batch kernel k-means

Ben Jourdan
University of Edinburgh

Gregory Schwartzman
JAIST

THE UNIVERSITY of EDINBURGH

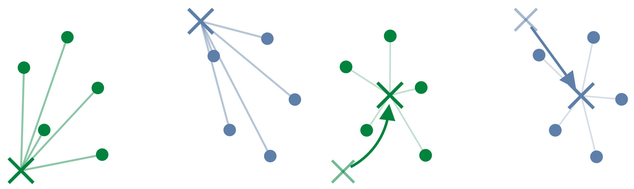
JAIST JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY 1990

Motivation

Naive mini-batch kernel k-means is prohibitively slow.

Lloyd's algorithm in feature space

1. Assign **2. Update**



Choose the nearest centre — Move to the cluster mean

$$c_A = \sum_{y \in A} w_y \phi(y)$$

$$w_y \geq 0, \quad \sum_{y \in A} w_y = 1$$

Distances via the kernel trick:

$$\|\phi(x) - c_A\|^2 = \langle \phi(x), \phi(x) \rangle - 2\langle \phi(x), c_A \rangle + \langle c_A, c_A \rangle$$

$$\langle c_A, c_A \rangle = \sum_{y \in A} \sum_{z \in A} w_y w_z \langle \phi(y), \phi(z) \rangle$$

Dense centre representations ✗

Each centre is a weighted sum of potentially all n points.

- Computing $\langle c_A, c_A \rangle$ can cost $O(n^2)$.
- Computing $\langle \phi(x), c_A \rangle$ can cost $O(n)$.

Minibatch algorithms should be sublinear in n:

Euclidean mini-batch k-means: $O(bkd)$ per iteration

Naive kernel mini-batch: $O(kn^2 + bkn)$ per iteration

b: batch size; k: clusters; d: input dimension, hides kernel cost

Our method: Truncated Mini-batch kernel k-means

1. Recursive update rule

For any cluster c, let B_i be its assigned batch in iteration i.

Adaptive centre update

$$c_{i+1} = (1 - \alpha_i) c_i + \alpha_i \mu_i$$

Learning rate: Schwartzman (ICLR 2023).

$$\alpha_i = \sqrt{|B_i|/b}$$

$$\mu_i = \frac{1}{|B_i|} \sum_{x \in B_i} \phi(x)$$

$$\langle \phi(x), c_{i+1} \rangle = (1 - \alpha_i) \langle \phi(x), c_i \rangle + \alpha_i \langle \phi(x), \mu_i \rangle$$

$$\langle c_{i+1}, c_{i+1} \rangle = (1 - \alpha_i)^2 \langle c_i, c_i \rangle + 2\alpha_i (1 - \alpha_i) \langle c_i, \mu_i \rangle + \alpha_i^2 \langle \mu_i, \mu_i \rangle$$

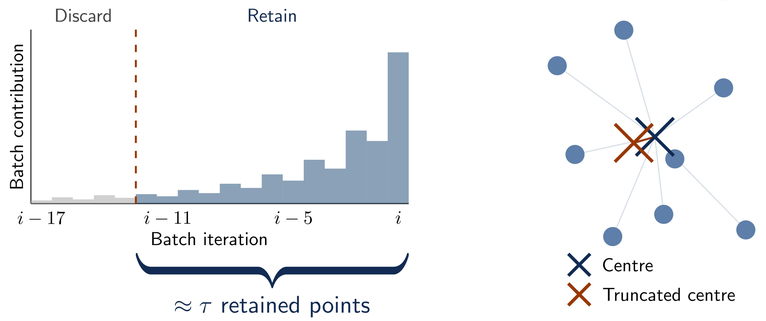
Dynamic programming: $O(n(b + k))$ per iteration

2. Sparse centre linear combinations

Unroll the recursive update for t iterations:

$$c_{i+1} = \underbrace{\left[\prod_{s=i-t+1}^{i} (1 - \alpha_s) \right] c_{i-t+1}}_{\text{old history: discarded}} + \underbrace{\sum_{r=i-t+1}^{i} \alpha_r \left[\prod_{s=r+1}^{i} (1 - \alpha_s) \right] \mu_r}_{\text{recent contributions}}$$

Each update shrinks old contributions by $1 - \alpha_i$: exponential forgetting.



Sparse centre representations!

3. Per iteration running time

Setting $\tau = \widetilde{\Theta}(b)$:

$$\underbrace{O(k(\tau + b)^2)}_{\text{centre self-products}} + \underbrace{O(kb(\tau + b))}_{\text{batch–centre products}} = \widetilde{O}(kb^2)$$

Only polylogarithmic dependence on n.

Theory and experiments

Algorithmic guarantees

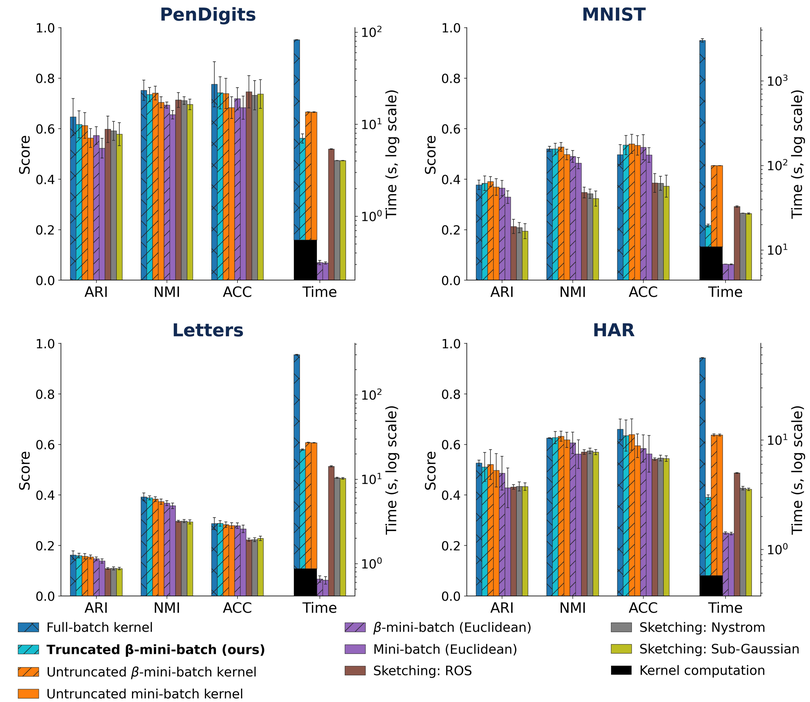
Fast iterations	Early termination*	Clustering quality
$\widetilde{O}(kb^2)$	$O(1/\epsilon)$	$O(\log k)$
time per iteration	iterations, with high probability	expected approximation with k-means++ initialisation

Parameter choices $b = \widetilde{\Theta}(k/\epsilon^2), \quad \tau = \widetilde{\Theta}(b)$

*Early termination: stop when the improvement in batch cost is below ϵ.

Experimental results

10–100× faster than full-batch kernel k-means, with comparable quality.
Up to 10× faster than untruncated mini-batch kernel k-means.



Gaussian kernel; $b = 1024$, $\tau = 200$; 200 iterations.
β: Schwartzman learning rate; mean ± std over 10 runs.