BanditPAM: Almost Linear Time *k*-medoids Clustering via Multi-Armed Bandits


Mo Tiwari


Martin Zhang


James Mayclin


Sebastian Thrun


Chris Piech


Ilan Shomorony

Introduction: the *k*-medoids problem

- The *k*-medoids problem is a clustering problem, like *k*-means, in which the objective is to minimize the distance from each point to its nearest cluster center
- Unlike *k*-means, *k*-medoids requires cluster centers to be actual datapoints:

$$L(\mathcal{M}) = \sum_{i=1}^{n} \min_{m \in \mathcal{M}} d(m, x_i)$$

- *k*-medoids has several advantages over *k*-means:
  - interpretability of the cluster centers (the medoids)
  - support for arbitrary distance metrics

Prior Approaches

- Prior state-of-the-art approaches (PAM) work in a two-step process:

1. BUILD: Initialize the medoids one-by-one in a greedy fashion ($\wedge$ denotes *min*):

$$\text{BUILD:} \quad \arg\min_{x \in \mathcal{X} \setminus \mathcal{M}_l} \frac{1}{n} \sum_{j=1}^{n} \left[\left(d(x, x_j) - \min_{m' \in \mathcal{M}_l} d(m', x_j) \right) \wedge 0 \right]$$

2. SWAP: Consider all (medoid, non-medoid) pairs and swap these pairs greedily; iterate until convergence:

$$\text{SWAP:} \quad \arg\min_{(m,x) \in \mathcal{M} \times (\mathcal{X} \setminus \mathcal{M})} \frac{1}{n} \sum_{j=1}^{n} \left[\left(d(x, x_j) - \min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j) \right) \wedge 0 \right]$$

- Each step is $O(n^2)$ since we compute the loss over the entire dataset

Contributions

- We propose BanditPAM, a new PAM-like algorithm based on multi-armed bandits, which:
  - returns the same results as PAM with high probability
  - reduces the complexity from $O(n^2)$ to $O(n \log n)$
- We also release high-performance Python and C++ packages that enable *k*-medoids on very large datasets

NEURAL INFORMATION PROCESSING SYSTEMS

Algorithm: BanditPAM

- BanditPAM each optimization problem as a multi-armed bandit (MAB) problem
- In the BUILD step, "arms" correspond to candidate medoids
- In the SWAP step, "arms" correspond to (medoid, non-medoid) pairs.

Algorithm 1 `Adaptive-Search` ($\mathcal{S}_{\text{tar}}, \mathcal{S}_{\text{ref}}, g_x(\cdot), B, \delta, \sigma_x$)

$\mathcal{S}_{\text{solution}} \leftarrow \mathcal{S}_{\text{tar}}$ ▷ Set of potential solutions to Problem (8)
$n_{\text{used\_ref}} \leftarrow 0$ ▷ Number of reference points evaluated
For all $x \in \mathcal{S}_{\text{tar}}$, set $\hat{\mu}_x \leftarrow 0, C_x \leftarrow \infty$ ▷ Initial mean and confidence interval for each arm
while $n_{\text{used\_ref}} < |\mathcal{S}_{\text{ref}}|$ and $|\mathcal{S}_{\text{solution}}| > 1$ **do**
  Draw a batch samples of size B with replacement from reference $\mathcal{S}_{\text{ref\_batch}} \subset \mathcal{S}_{\text{ref}}$
  for all $x \in \mathcal{S}_{\text{solution}}$ **do**
    $\hat{\mu}_x \leftarrow \frac{n_{\text{used\_ref}} \hat{\mu}_x + \sum_{y \in \mathcal{S}_{\text{ref\_batch}}} g_x(y)}{n_{\text{used\_ref}} + B}$ ▷ Update running mean
    $C_x \leftarrow \sigma_x \sqrt{\frac{\log(\frac{1}{\delta})}{n_{\text{used\_ref}} + B}}$ ▷ Update confidence interval
  $\mathcal{S}_{\text{solution}} \leftarrow \{x : \hat{\mu}_x - C_x \leq \min_y(\hat{\mu}_y + C_y)\}$ ▷ Remove points that can no longer be solution
  $n_{\text{used\_ref}} \leftarrow n_{\text{used\_ref}} + B$
if $|\mathcal{S}_{\text{solution}}| = 1$ **then**
  return $x^* \in \mathcal{S}_{\text{solution}}$
else
  Compute μ_x exactly for all $x \in \mathcal{S}_{\text{solution}}$
  return $x^* = \arg\min_{x \in \mathcal{S}_{\text{solution}}} \mu_x$

$$\text{BUILD:} \ \mathcal{S}_{\text{tar}} = \mathcal{X} \setminus \mathcal{M}_l, \ \mathcal{S}_{\text{ref}} = \mathcal{X}, \ g_x(x_j) = \left(d(x, x_j) - \min_{m' \in \mathcal{M}_l} d(m', x_j) \right) \wedge 0$$

$$\text{SWAP:} \ \mathcal{S}_{\text{tar}} = \mathcal{M} \times (\mathcal{X} \setminus \mathcal{M}), \ \mathcal{S}_{\text{ref}} = \mathcal{X},$$
$$g_x(x_j) = \left(d(x, x_j) - \min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j) \right) \wedge 0$$

- δ is a hyperparameter that governs the probability of error
- each σ_x is estimated from the data

Theoretical Analysis

Theorem 1. *If BanditPAM is run on a dataset $\mathcal{X}$ with $\delta = n^{-3}$, then it returns the same set of k medoids as PAM with probability $1 - o(1)$. Furthermore, the total number of distance computations M_{total} required satisfies*

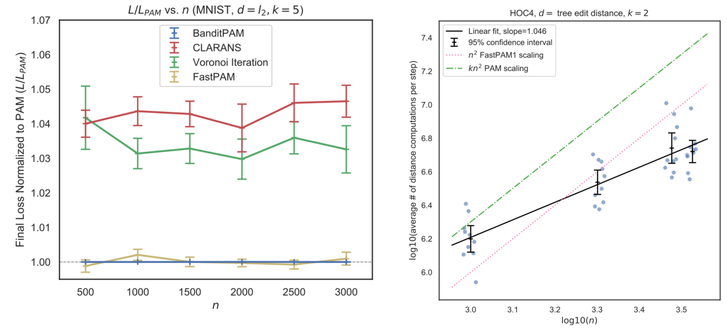
$$E[M_{\text{total}}] = O(n \log n).$$

Funding Information

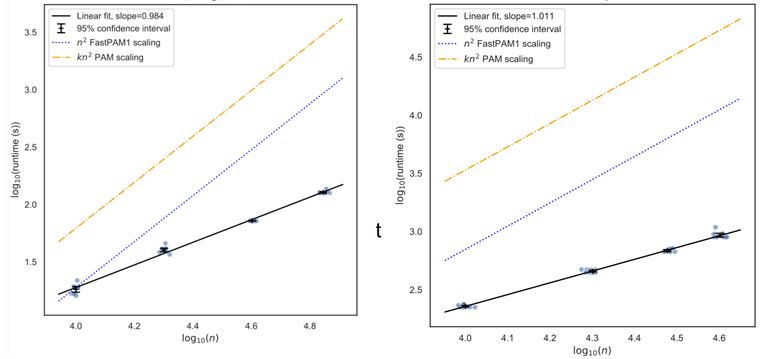
This research was funded in part by JPMorgan Chase & Co. Any views or opinions expressed herein are solely those of the authors listed, and may differ from the views and opinions expressed by JPMorgan Chase \& Co. or its affiliates. This material is not a product of the Research Department of J.P. Morgan Securities LLC. This material should not be construed as an individual recommendation for any particular client and is not intended as a recommendation of particular securities, financial instruments or strategies for a particular client. This material does not constitute a solicitation or offer in any jurisdiction.

M.Z. was also supported by NIH grant R01 MH115676.

Experimental Results



(Left) BanditPAM matches the state-of-the-art clustering quality of PAM in all experiments. (Right) On a dataset of Abstract Syntax Trees with tree edit distance, BanditPAM scales almost linearly in dataset size.



BanditPAM demonstrates almost linear scaling on the MNIST handwritten digit dataset with L_2 distance (left) and on a single-cell RNA sequencing dataset with L_1 distance (right).

Python Package

```
import numpy as np, banditpam

# Create toy data from Gaussian Mixture Model
mus = np.array([[0,0], [-5,5]])
X = np.vstack([np.random.randn(10, 2) + mu for mu in mus])

# Run BanditPAM
kmed = banditpam.KMedoids(n_medoids = 2, algorithm = "BanditPAM")
kmed.fit(X, 'L2')
print(kmed.final_medoids)
```