K-medoids for K-means seeding

James Newling & François Fleuret
{james.newling,francois.fleuret}@idiap.ch

Abstract

We present a scheme for K-means seeding, which results in a 3% geometric mean reduction in K-means loss as compared to vanilla K-means++ seeding, on 16 publicly available datasets. It is based on the CLARANS K-medoids algorithm of Ng and Han (1994).

K-medoids

Given samples $\mathcal{X} = \{x(1), \dots, x(N)\}$, function $dist : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ and monotonic energy function $\psi : \mathbb{R}^+ \to \mathbb{R}^+$, find $\mathcal{C} \subset \{1, \dots, N\}$ where $|\mathcal{C}| = K$ to minimize

$$E = \sum_{i=1}^{N} \min_{i' \in \mathcal{C}} \psi(dist(x(i), x(i'))).$$

It has applications in clustering sequences, graph vertices, sparse and dense vectors, etc. Two popular algorithms are,

- MEDLLOYD ((Hastie et al. (2001), Park and Jun (2009)): like Lloyd's algorithm, but centroids are replaced by medoids
- CLARANS (Ng and Han (1994)): random swaps between centers and non-centers are proposed, and only accepted if $E(\mathcal{C})$ decreases.

K-means and K-means seeding

The K-means task is to find K centers, $\{C(1), \dots, C(K)\}$, *not* necessarily elements of $\{x(1), \dots, x(N)\}$, to minimize

$$E = \sum_{i=1}^{N} \min_{k \in \{1,\dots,K\}} \|x(i) - C(k)\|_2^2. \qquad (1)$$

In the popular LLOYD algorithm centers are initialized or *seeded* as a subset of $\mathcal{X}$. Good seeding is critical to avoid poor local minima. Most seeding algorithms attempt to minimize initial energy (K-means++, Bradley-Fayad, etc.). Minimizing seeding energy is the special case of K-medoids with

$$dist(x(i), x(i')) = \|x(i) - x(i')\|_2 \quad \text{and} \quad \psi(v) = v^2.$$

This motivates the use of other popular and well established K-medoids algorithms for K-means seeding.

Acknowledgements

This work was sponsored by HASLERSTIFTUNG

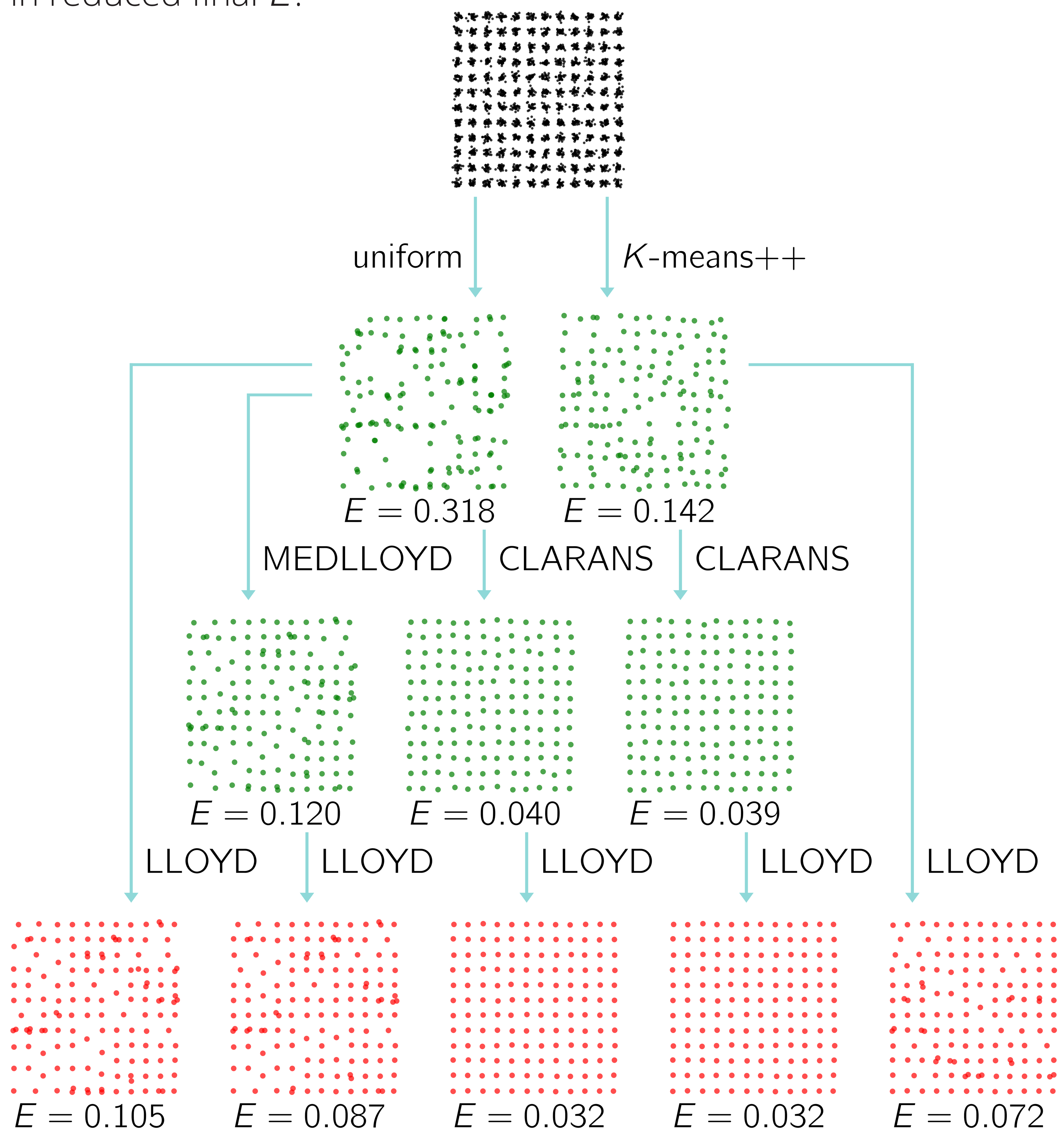
The CLARANS K-medoids algorithm

The algorithm iteratively proposes swapping a medoid $x(i_-)$ with a non-medoid $x(i_+)$. Only energy reducing swaps are implemented.

1: Initialize center indices $\mathcal{C} \subset \{1, \dots, N\}$, where $|\mathcal{C}| = K$
2: $E \leftarrow \sum_{i=1}^{N} \min_{i' \in \mathcal{C}} \psi(dist(x(i), x(i')))$
3: **while** stopping criterion false **do**
4: sample $i_- \in \mathcal{C}$ and $i_+ \in \{1, \dots, N\} \setminus \mathcal{C}$
5: $E^+ \leftarrow \sum_{i=1}^{N} \min_{i' \in \mathcal{C} \setminus \{i_-\} \cup \{i_+\}} \psi(dist(x(i), x(i')))$
6: **if** $E^+ < E$ **then**
7: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{i_-\} \cup \{i_+\}$
8: $E \leftarrow E^+$
9: **end if**
10: **end while**

Five routes to K-means local minima

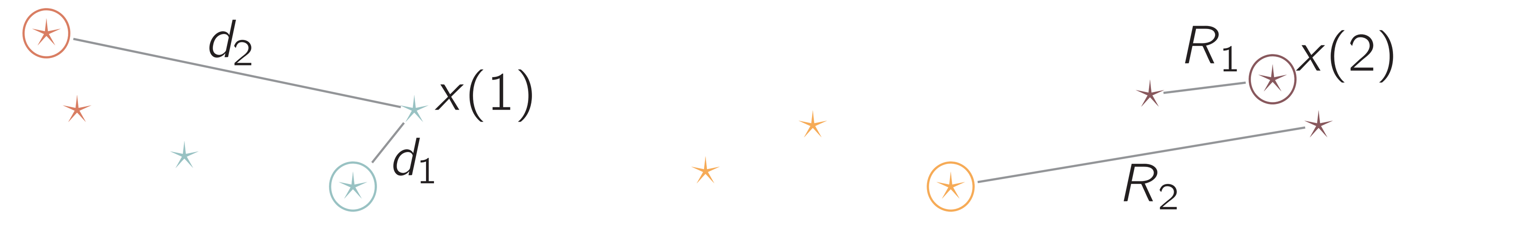
(Below) Clustering with $K = 12^2$ centers on a 2-d grid, $N = 25K$ samples. First row: the generated samples. Second row: uniform and K-means++ seedings. Third row: K-medoids refinements. Fourth row: final LLOYD clusterings. CLARANS refinement results in reduced final E.



Accelerating the CLARANS algorithm

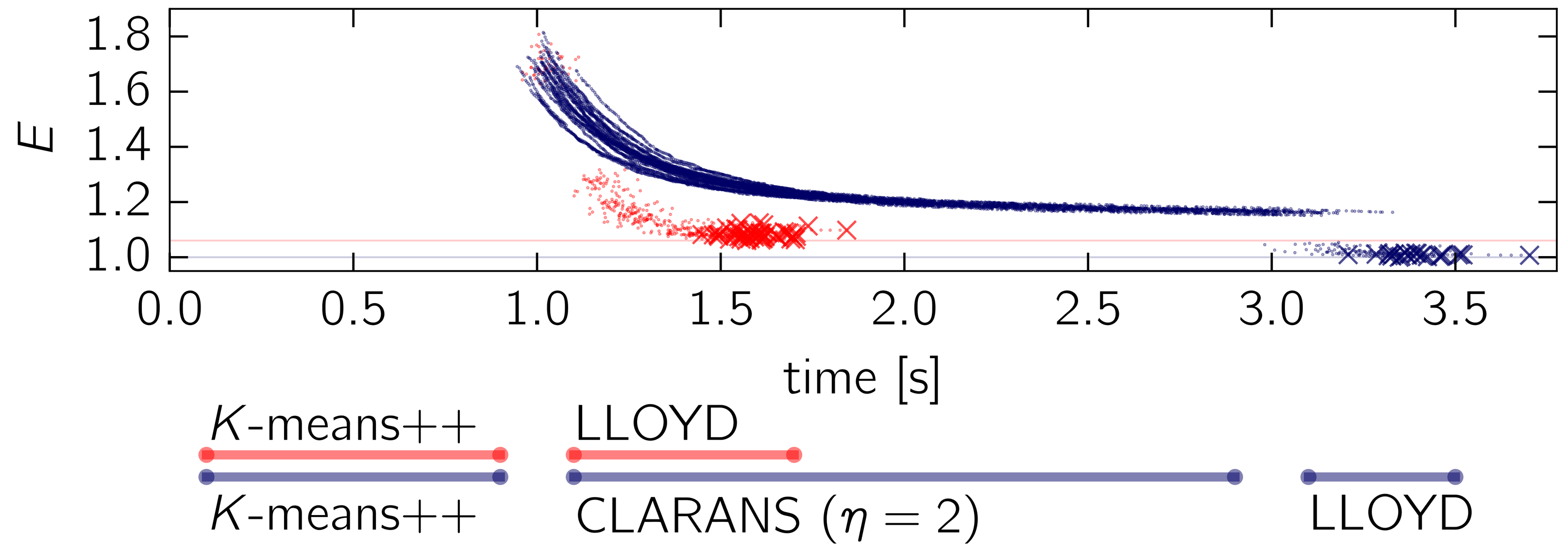
There are many more *evaluations* (line 5) than *implementations*. Assuming balanced clusters, and that *dist* satisfies the triangle inequality, we present a technique where evaluation is $O(N/K)$, and implementation is $O(N)$. It requires recording,

- for non-centers (such as $x(1)$ below), distance to nearest (d_1) and second nearest (d_2) centers (as in Ng and Han),
- for centers (such as $x(2)$ below), maximum over cluster of d_1 and d_2 (R_1 and R_2 respectively), and distances to all centers.

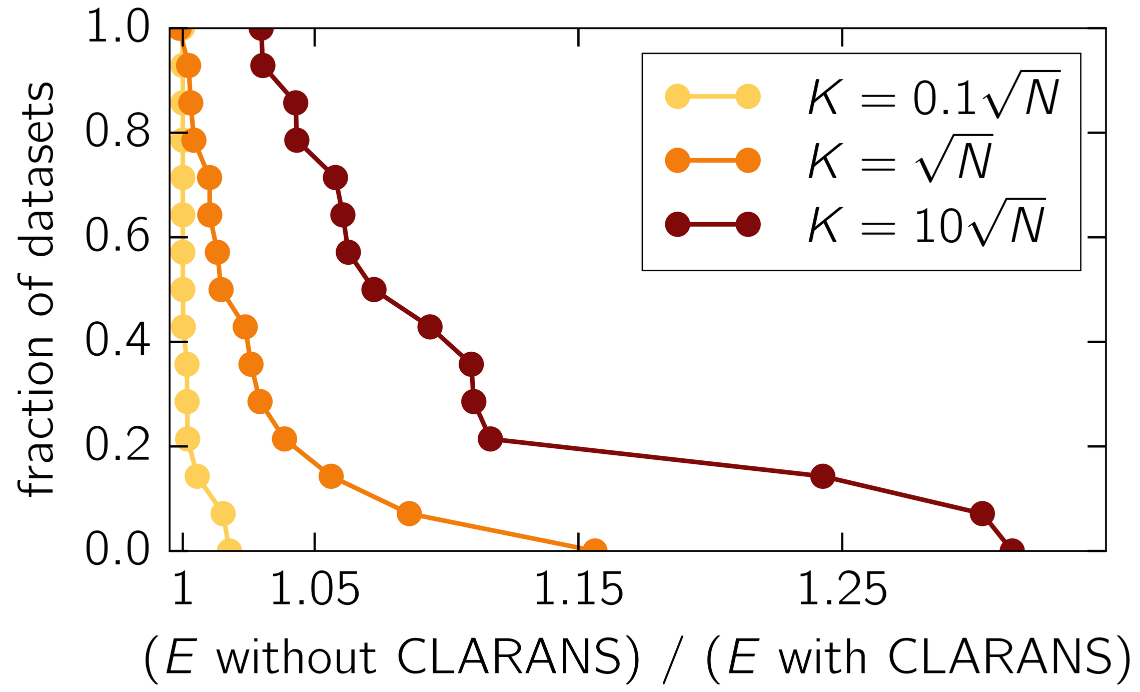


Results

(Below) An experiment with an RNA dataset, $N = 16 \times 10^4$, $d = 8$ and $K = 4 \times 10^2$. With 50 runs seeded with K-means++ (red), and several runs with CLARANS inbetween K-means and K-means++ (blue). The best run without CLARANS has 6% higher E.



(Below) Summary for 16 datasets. Each point is an experiment with same setup as RNA, horizontal position is reduction in E with CLARANS. Dimensions range in $d : 2 \to 90$, $N : 1484 \to 488565$. At $K\sqrt{N}$, mean reduction is 3.2% vs K-means++ and 1.2% vs greedy K-means++ (not shown).



(On github)
Sequences, sparse vectors, various $dist$, ϕ, etc.