Fast K-Means with Accurate Bounds

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

K-Means

Given data points $\mathcal{X} = (x(i))_{i=1}^{N} \in (\mathbb{R}^d)^N$, find centers $\mathcal{C} = (c(k))_{k=1}^{K} \in (\mathbb{R}^d)^K$ to minimise

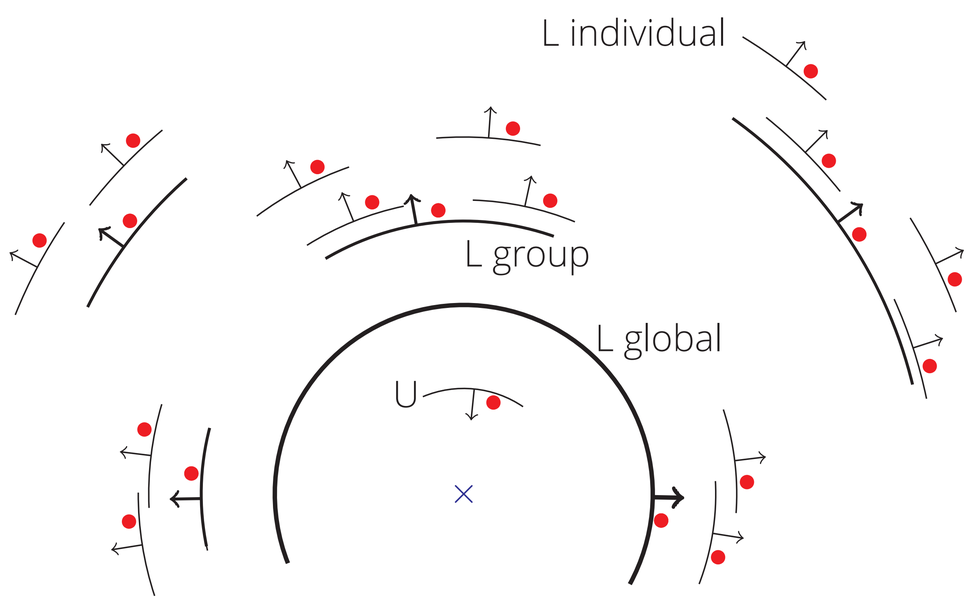
$$\sum_{i=1}^{N} \min_{k=1:K} \|x(i) - c(k)\|^2.$$

The problem is in general NP-hard and heuristic algorithms are used. The most popular is Lloyd's algorithm, which alternates between assignment and update steps,

$$a(i) \leftarrow \operatorname*{argmin}_{k=1:K} \|x(i) - c(k)\| \qquad c(k) \leftarrow \frac{\sum_{i:a(i)=k} x(i)}{\|i : a(i) = k\|}.$$

Previous Triangle Inequality Bounding Algorithms

By maintaining bounds on distances, the nearest center can be determined without computing all K distances. Let U be an upper bound on the distance to the previous nearest center, and L a lower bound on one or several other data-center distances.



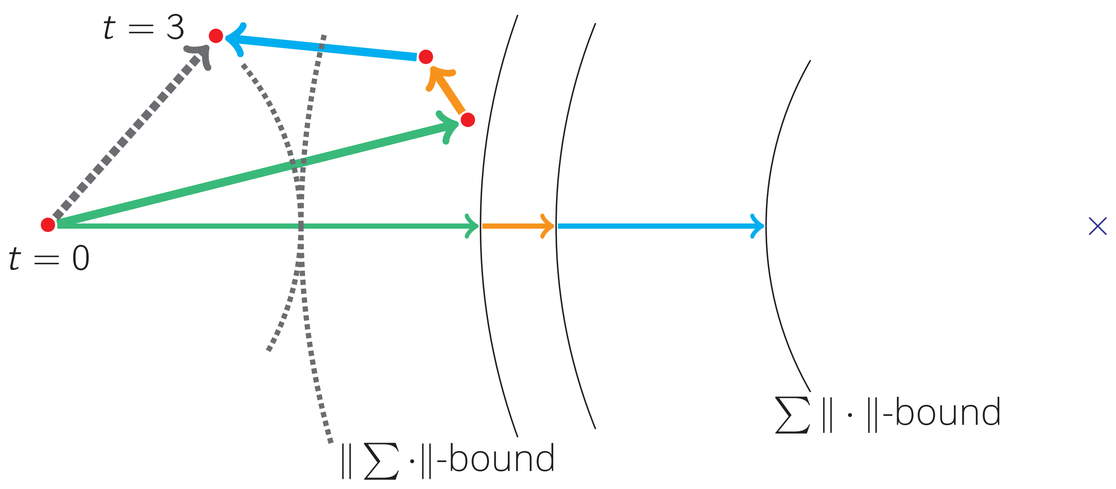
U $\leq$ L $\Rightarrow$ a center bound by L is not nearest $\Rightarrow$ don't compute distance

Dimension (d)	Low (< 10)	Moderate (< 100)	High (≥ 100)
Algorithm	Annular	Yinyang	Elkan
Year	2013	2015	2003
Lower Bounds	global	group	individual

State-of-the-art exact K-Means algorithms use the triangle inequality, with the optimal number of bounds depending on the dimension.

Updating Bounds : A New Approach

The standard approach is to adjust bounds by distances moved by centers. We store historical center posititions, allowing the use of the norm of total displacement. This approach, which we call $\|\sum \cdot\|$-bounding, is applicable to all exisiting algorithms.

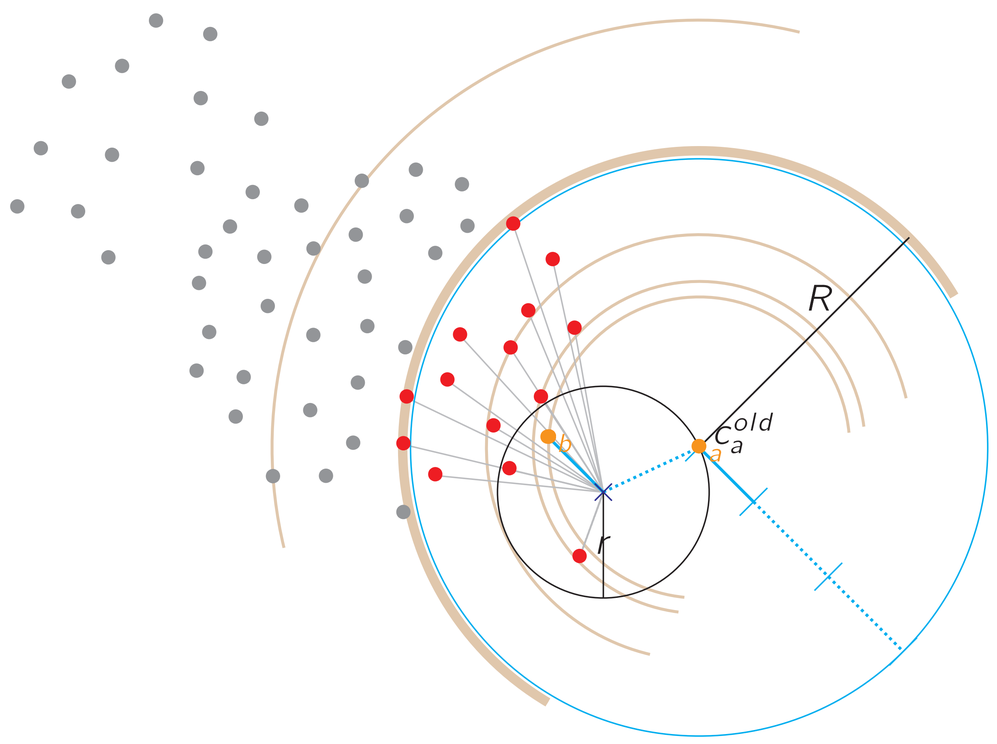


The Exponion Algorithm

An extension to Hamerly's algorithm, and related to Annular. With Hamerly's algorithm, when the lower bound on all non-nearest centers falls below the distance to the current nearest, all distances are computed. The Exponion Algorithm eliminates some of these using,

$$\|c(k) - c(a(i))\| > 2\|x(i) - c(a(i))\| + \|x(i) - c(b(i))\| \Rightarrow \quad k \notin \{a_{new}(i), b_{new}(i)\} \tag{1}$$

where $a(i)$ and $b(i)$ are old nearest and second nearest center indices and $a_{new}(i)$ and $b_{new}(i)$ their updates, which we wish to determine.



Given sorted distances from $c(a(i))$, determining which centers are eliminated by (1) is $O(\log N)$. To prevent the need to maintain K sorted vectors, we use a partial sort where only the order between $O(\log K)$ concentric shells is maintained. The radius of the smallest shell containing the hypersphere from (1) is then used,

$$\|c(k) - c(a(i))\| > R \Rightarrow k \notin \{a_{new}(i), b_{new}(i)\} \qquad (c(k) : \bullet)$$

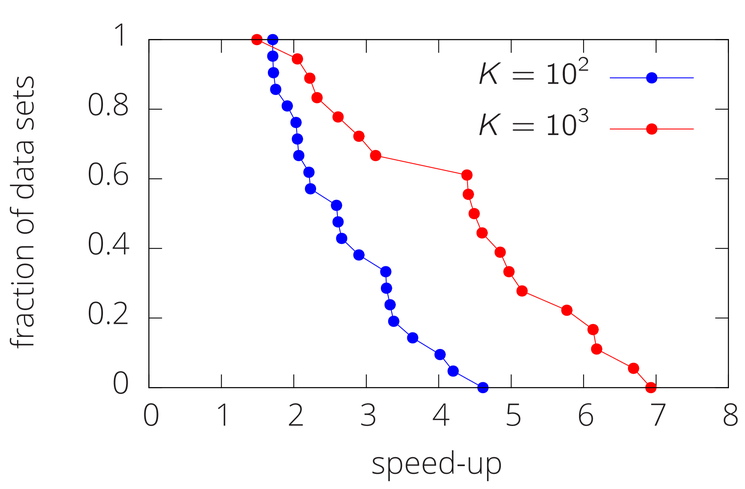
Results and Conclusions

22 datasets ($d : 2 \rightarrow 784$, $N : 60k \rightarrow 2.6m$) and $K \in \{100, 1000\}$, 4 public code bases (mlpack, BaylorML, PowerGraph, VLFeat), as well as our own implementations of all algorithms.

(A) *Existing algorithms can be accelerated by simplification*
- Elkan (remove center-center distances) with mean speed-up of 15% ($d \geq 20$), and Yinyang (remove final filter) with mean speed-up 60%.

(B) *Replacing $\sum\|\cdot\|$-bounding by $\|\sum\cdot\|$-bounding helps*
- speed-up in 15/20 experiments, mean speed-up of 12% ($d \geq 20$)

(C) *Exponion is generally faster than Annular*
- Speed-up in 18/22 experiments, a mean speed-up of 35% ($d < 20$)



eakmeans

Speed-up of the fastest of our implementations of our algorithms relative to fastest of any of the other 4 implementations of any algorithm. Link to source code on right.

Acknowledgements

This work was sponsored by HASLERSTIFTUNG