Fast K-Means with Accurate Bounds

James Newling & François Fleuret

Idiap Research Institute
Computer Vision and Learning Group
& EPFL

June 20th, 2016

Given data $(x_i)_{i=1}^N \in (\mathcal{R}^d)^N$, find centers $(c_k)_{k=1}^K \in (\mathcal{R}^d)^K$ minimising

$$\sum_{i=1}^{N} \min_{k=1:K} \|x_i - c_k\|^2.$$

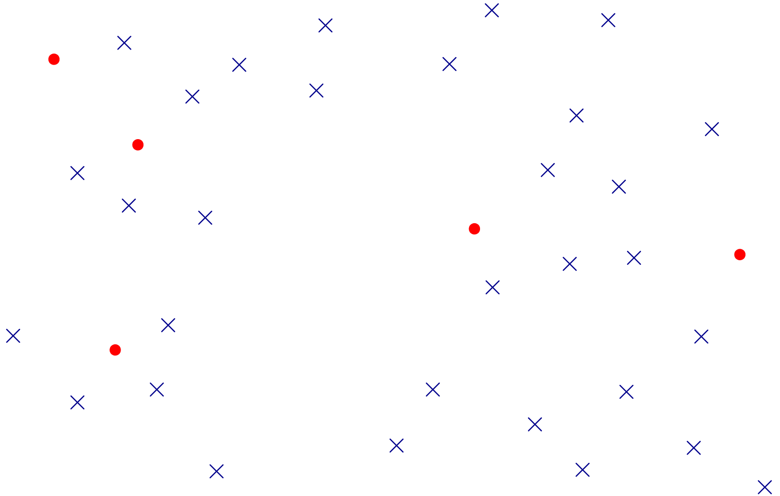
NP-hard, so heuristic algorithms such as Lloyd's are used

Lloyd's algorithm run for T iterations requires $dKNT$ FLOPs

We are interested in making it faster

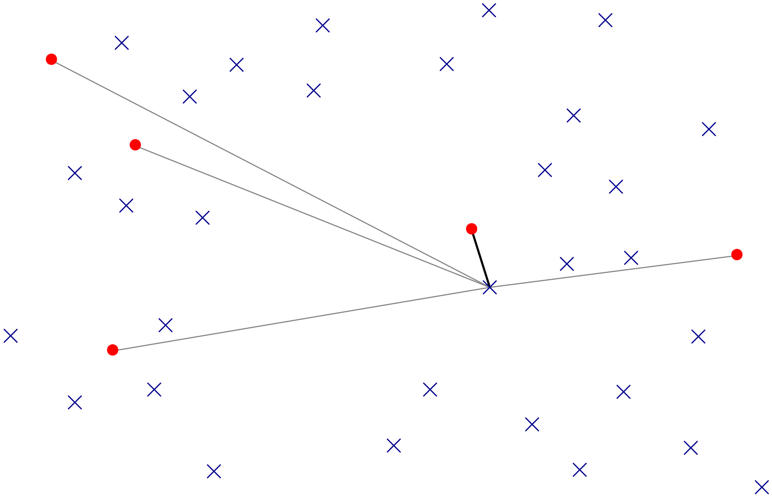
Lloyd's Algorithm

× : data ● : centers

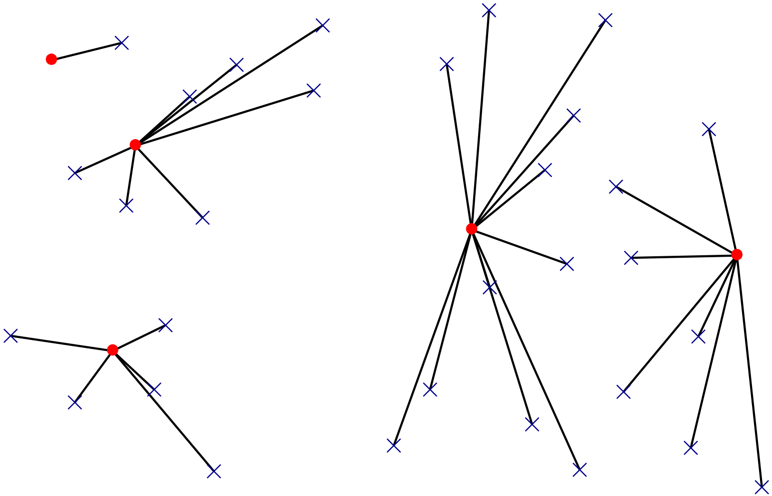


Lloyd's Algorithm

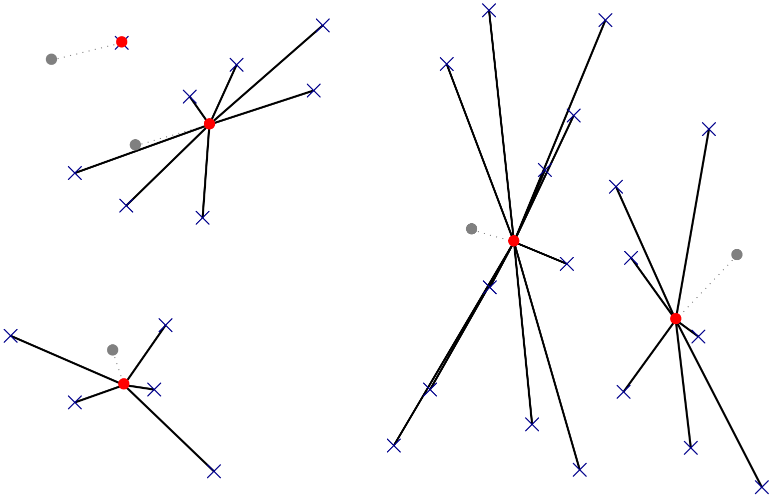
Assignment of datapoint at iteration 1



Lloyd's Algorithm
All assignments at iteration 1

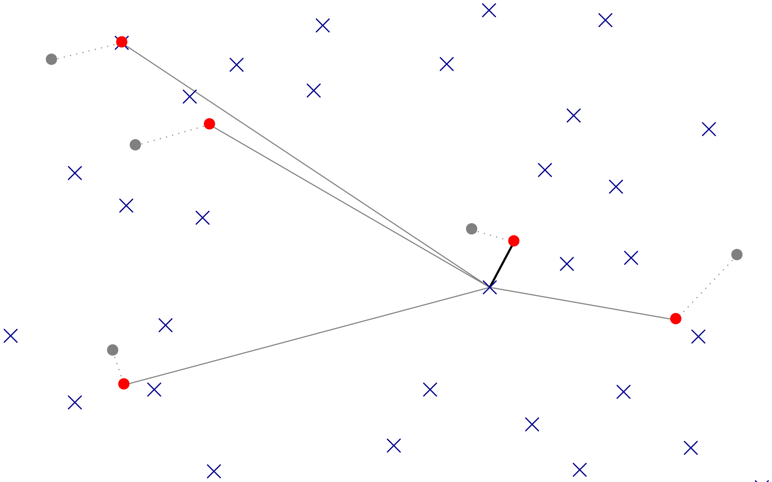


Lloyd's Algorithm
Updates at iteration 1

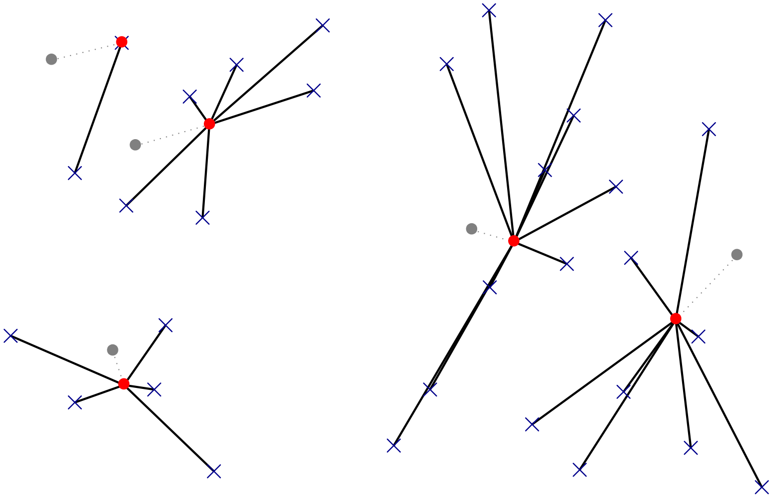


Lloyd’s Algorithm

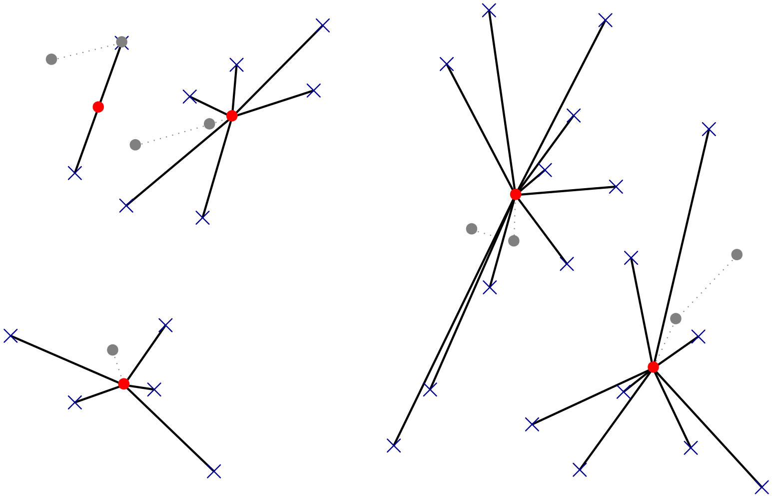
Assignment of datapoint at iteration 2



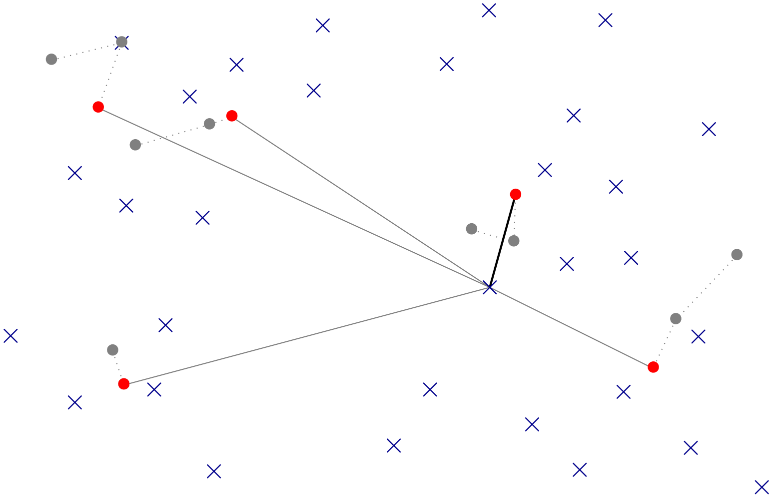
Lloyd's Algorithm
All assignments at iteration 2



Lloyd's Algorithm
Updates at iteration 2

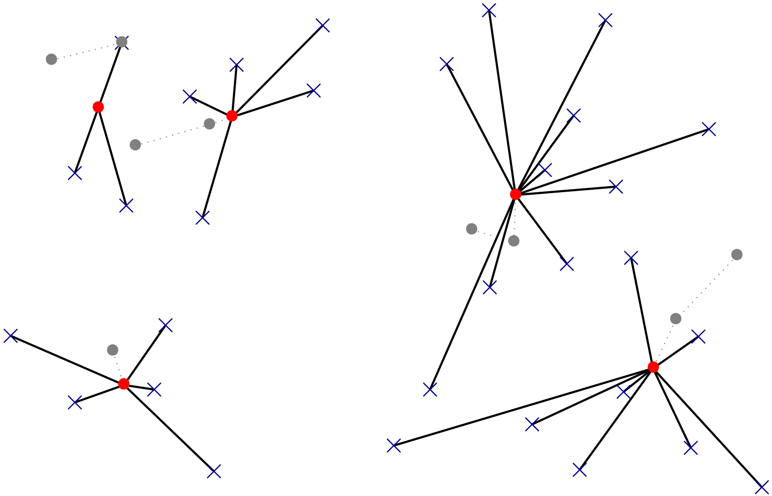


Lloyd's Algorithm
Assignment of datapoint at iteration 3

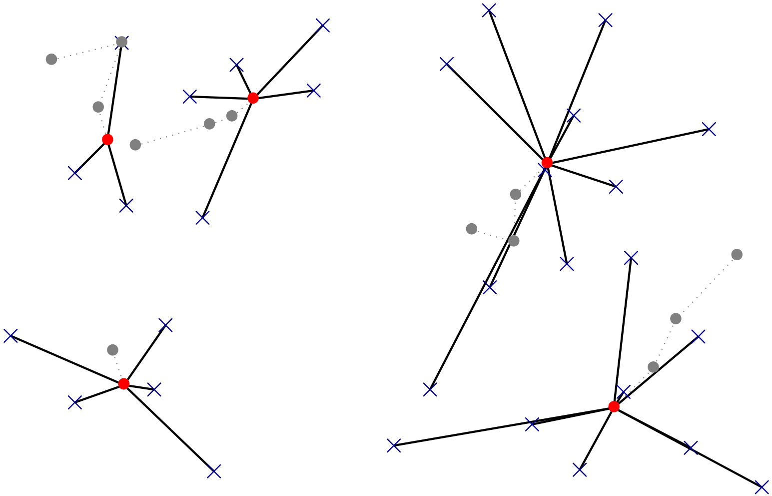


Lloyd's Algorithm

All assignments at iteration 3

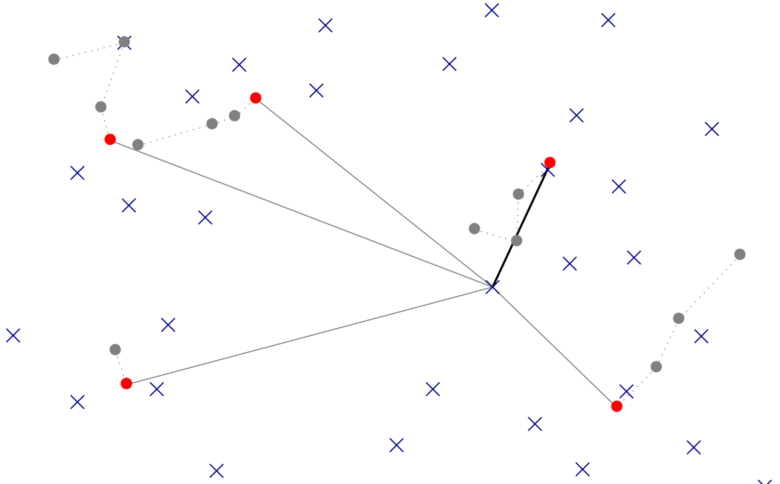


Lloyd's Algorithm
Updates at iteration 3



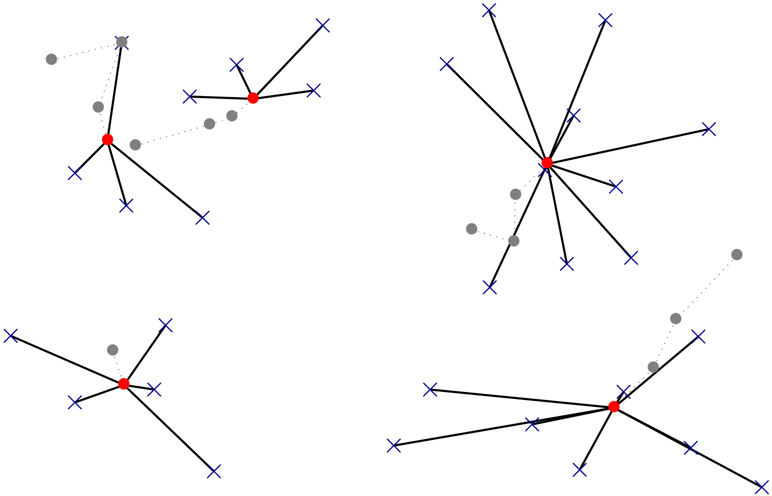
Lloyd's Algorithm

Assignment of datapoint at iteration 4

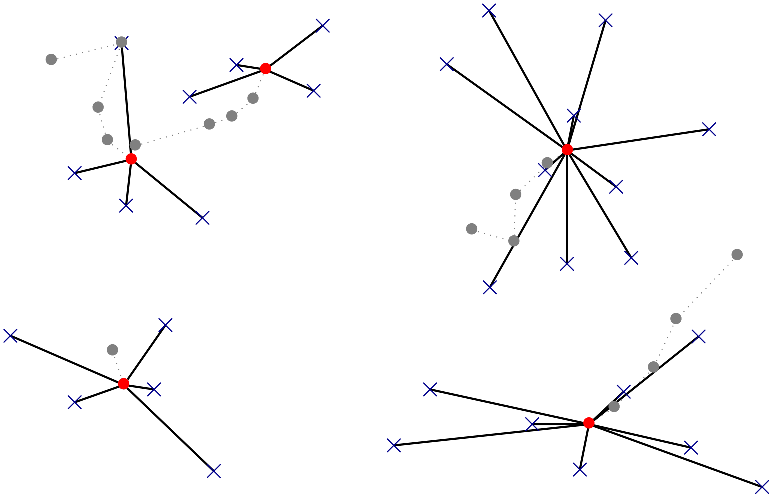


Lloyd’s Algorithm

All assignments at iteration 4



Lloyd's Algorithm
Updates at iteration 4



Two approaches :

(1) approximate it

(2) be more efficient – get exactly the same output as Lloyd's algorithm without all data-center distances

- Pelleg et al. (1999)
- Kanungo et al. (2002)
- Δ Hamerly (2010)
- Δ Elkan (2003) best high-*d*
- Δ Yinyang (2015) best mid-*d*
- Δ Annular (2013) best low-*d*

Two approaches :

(1) ~~approximate it~~ only *exact* for next 13 minutes

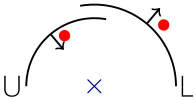
(2) be more efficient – get exactly the same output as Lloyd's algorithm without all data-center distances

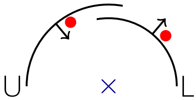
- Pelleg et al. (1999)
- Kanungo et al. (2002)
- Δ Hamerly (2010)
- Δ Elkan (2003) best high-d
- Δ Yinyang (2015) best mid-d
- Δ Annular (2013) best low-d

Elkan uses the triangle inequality in two distinct ways

(1) center-center distances to bound data-center distances

(2) directly maintain bounds on data-center distances

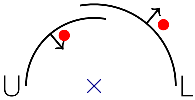


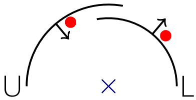


Elkan uses the triangle inequality in two distinct ways

(1) center-center distances to bound data-center distances

(2) directly maintain bounds on data-center distances

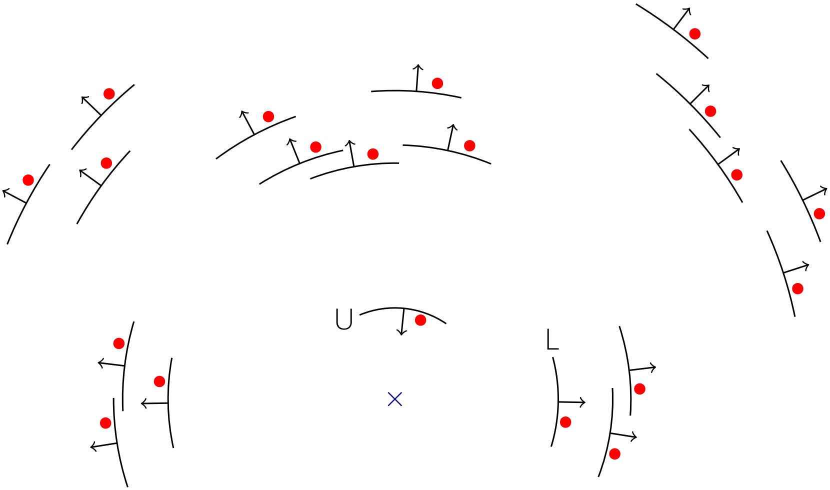




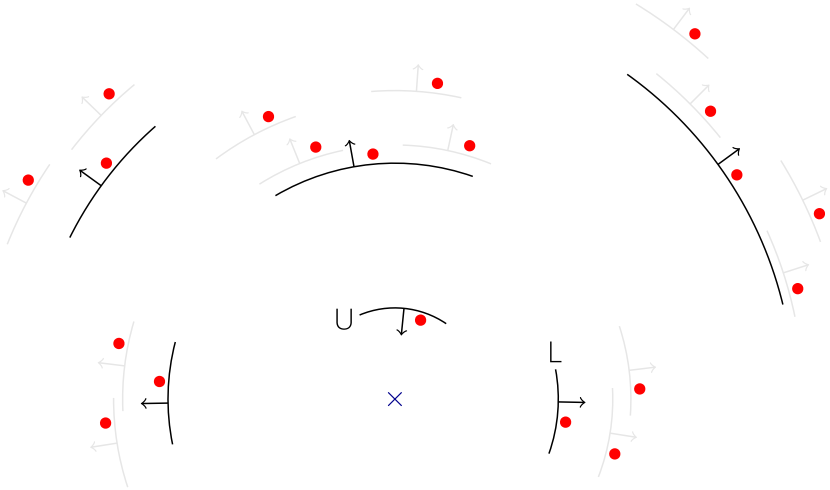
(A) *We show that (1) + (2) is slower than just (2). Simplifying helps!*

Using The Triangle Inequality

Elkan $K - 1$ lower bounds

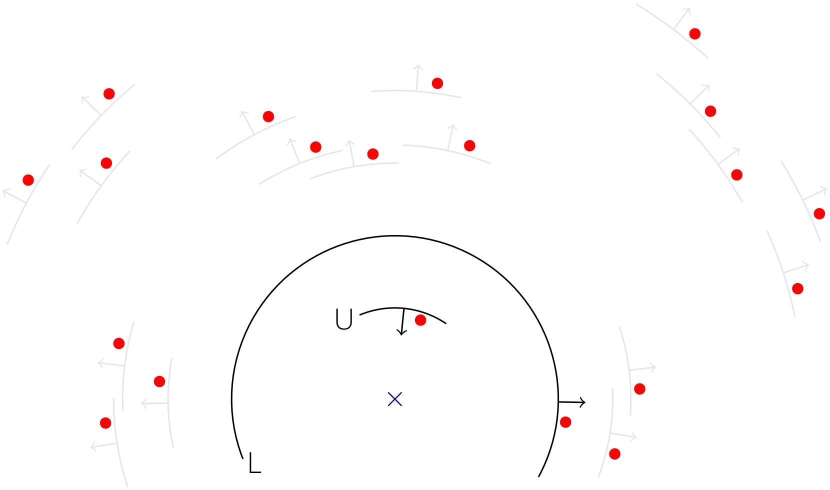


Using The Triangle Inequality
Yinyang group lower bounds

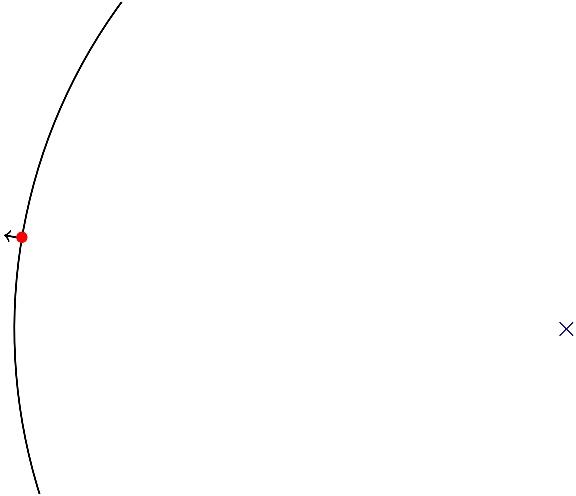


Using The Triangle Inequality

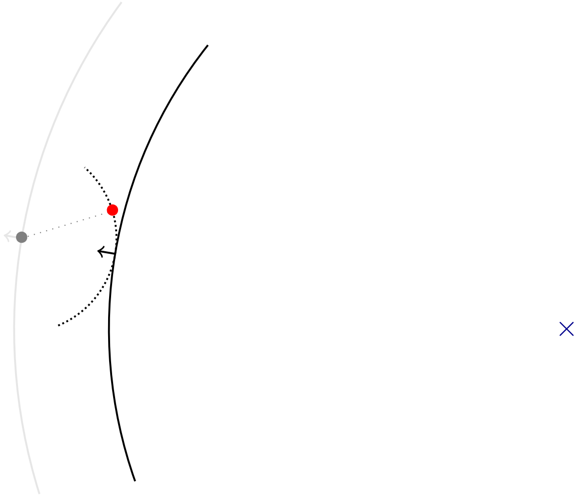
Hamerly 1 lower bound



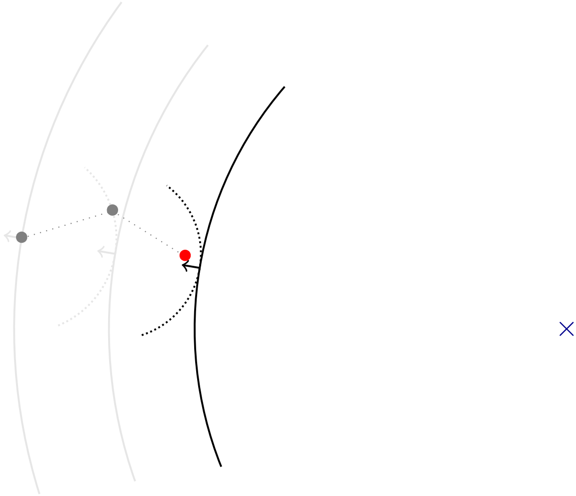
Lower bound updating



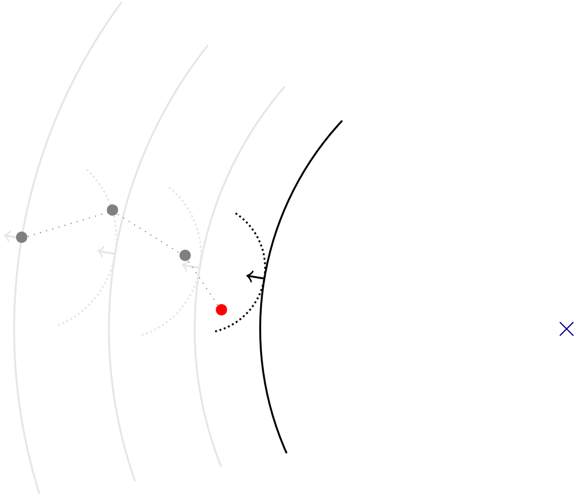
Lower bound updating



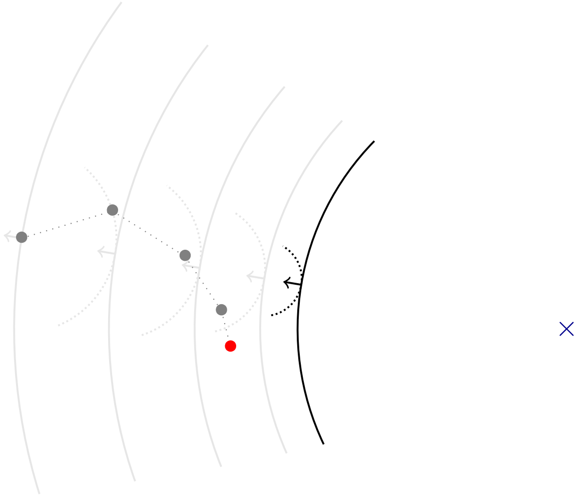
Lower bound updating



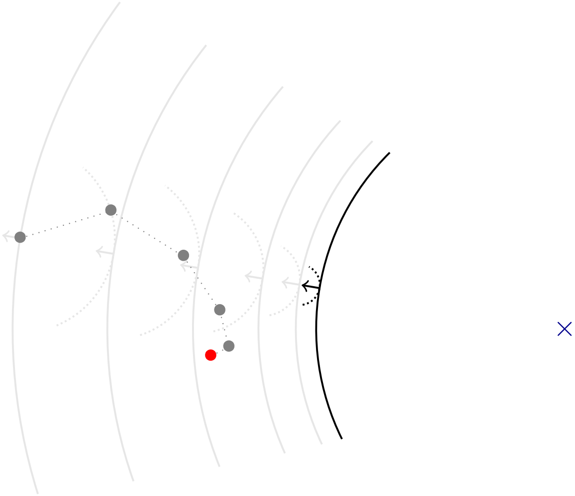
Lower bound updating



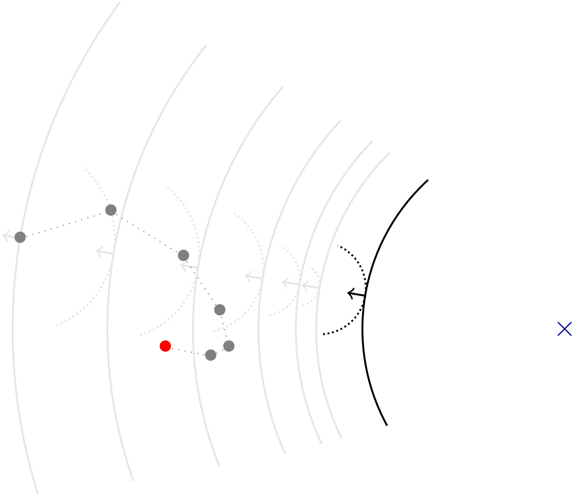
Lower bound updating



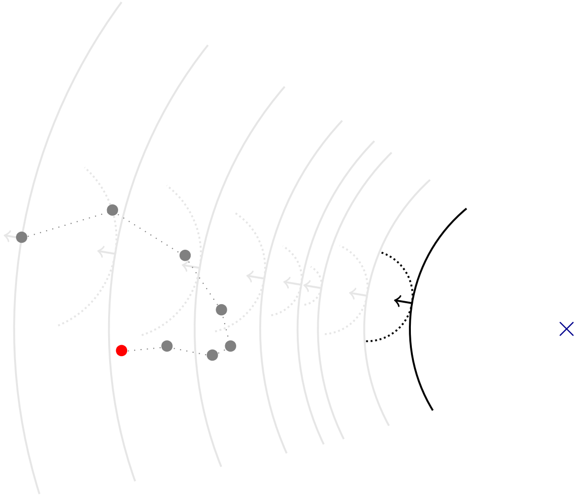
Lower bound updating



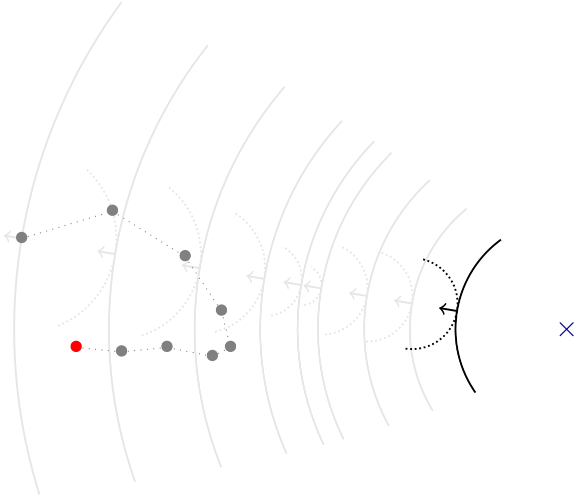
Lower bound updating



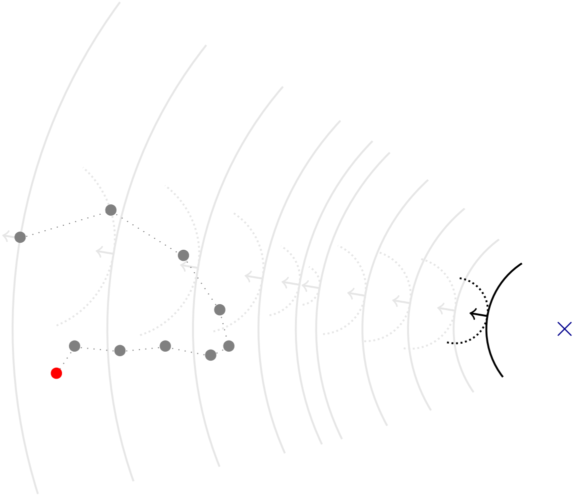
Lower bound updating



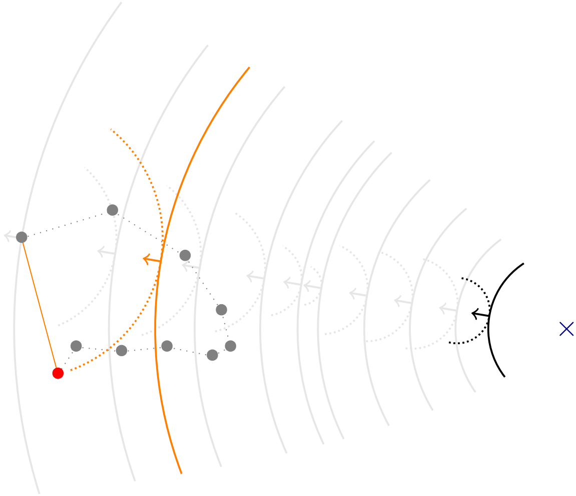
Lower bound updating



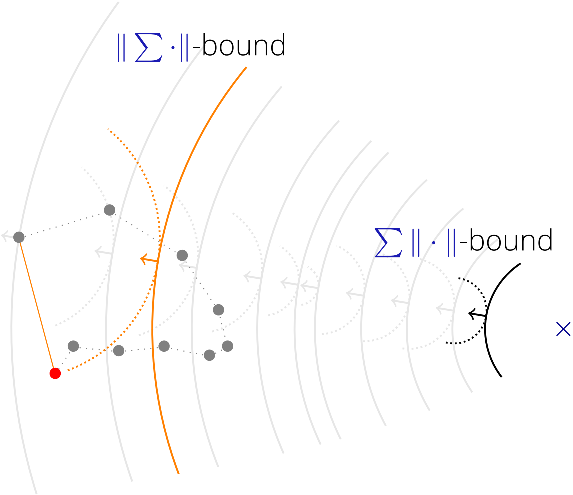
Lower bound updating



Lower bound updating



Lower bound updating



$\| \sum \cdot \|$-bounds

All upper and lower bounds in Elkan, Hamerly, Yinyang, Annular are $\sum \| \cdot \|$-bounds, and can be replaced by tighter $\| \sum \cdot \|$-bounds.

There is a cost to $\| \sum \cdot \|$-bounds, additional memory is required:

- Store historical centers from all rounds
- Store the round in which bounds are made tight

This memory overhead can be controlled by periodically clearing the history, requiring a $\sum \| \cdot \|$-bound update

$\|\sum \cdot\|$-bounds

All upper and lower bounds in Elkan, Hamerly, Yinyang, Annular are $\sum \|\cdot\|$-bounds, and can be replaced by tighter $\|\sum \cdot\|$-bounds.

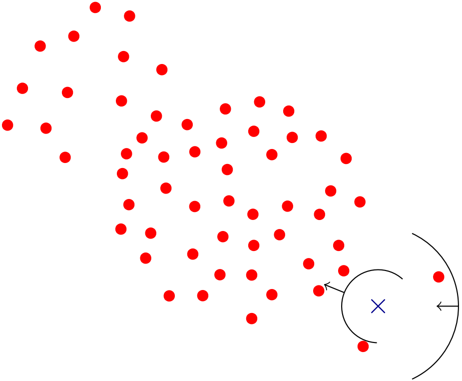
There is a cost to $\|\sum \cdot\|$-bounds, additional memory is required:

- Store historical centers from all rounds
- Store the round in which bounds are made tight

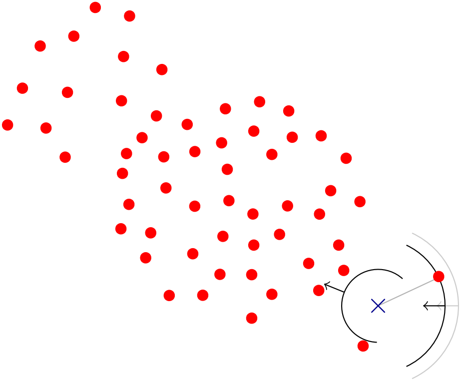
This memory overhead can be controlled by periodically clearing the history, requiring a $\sum \|\cdot\|$-bound update

(B) *We show that $\|\sum \cdot\|$-bounding generally improves algorithms.*

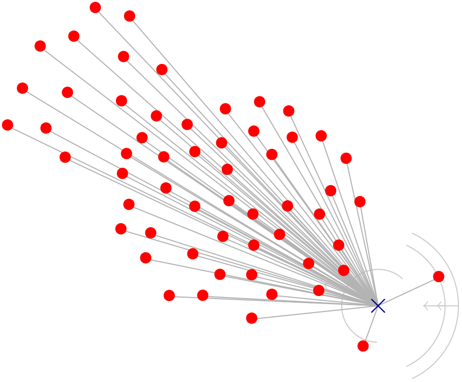
Hamerly (2010) bound test, failure 1



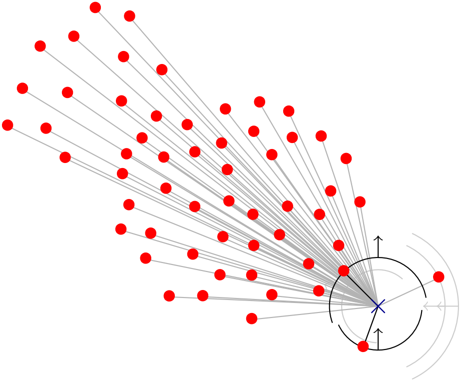
Hamerly (2010) bound test, failure 2



Hamerly (2010) compute all distances

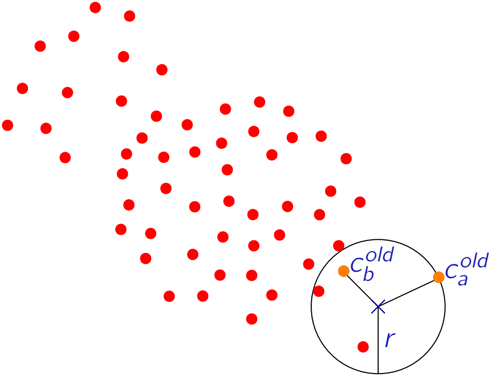


Hamerly (2010) reset bounds



Eliminating distance calculations

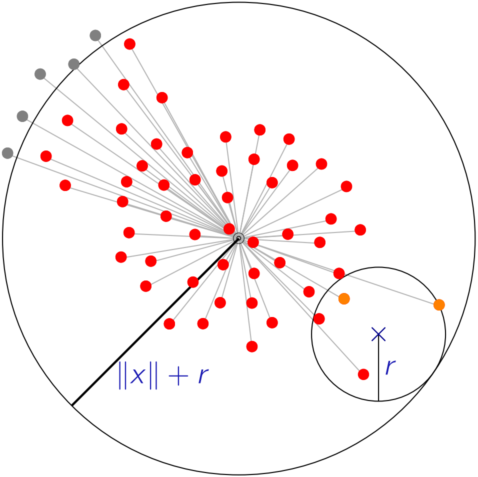
$$c \notin \mathcal{B}(x, r) \Rightarrow c \notin \{c_a^{new}, c_b^{new}\}$$



$$r = \max_{c \in \{c_a^{old}, c_b^{old}\}} \|x - c\|$$

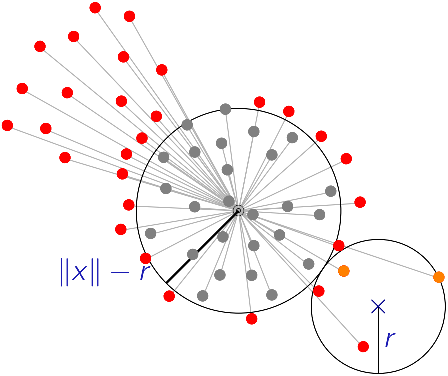
Annular (2013) elimination zone

$\|c\| > \|x\| + r \Rightarrow c \notin \mathcal{B}(x, r)$ (● : centers eliminated)



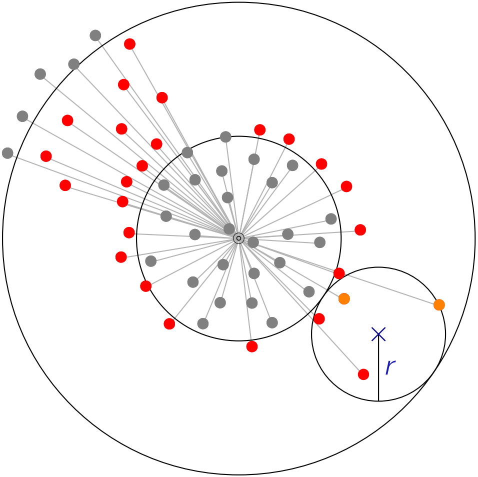
Annular (2013) elimination zone

$\|c\| < \|x\| - r \Rightarrow c \notin \mathcal{B}(x, r)$ (• : centers eliminated)



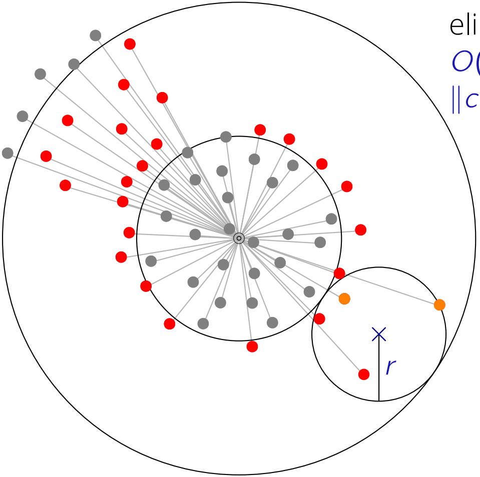
Annular (2013) elimination zone

$\big|\, \|c\| - \|x\| \,\big| < r \Rightarrow c \notin \mathcal{B}(x, r)$ ($\bullet$: centers eliminated)



Annular (2013) elimination zone

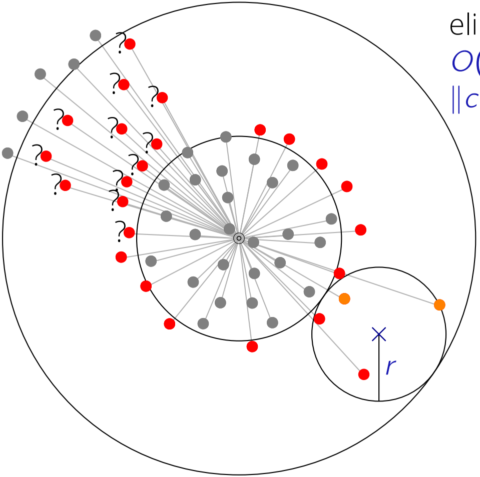
$\big|\, \|c\| - \|x\| \,\big| < r \Rightarrow c \notin \mathcal{B}(x, r)$ ($\bullet$: centers eliminated)



elimination
$O(\log N)$ if
$\|c\|$ sorted

Annular (2013) elimination zone

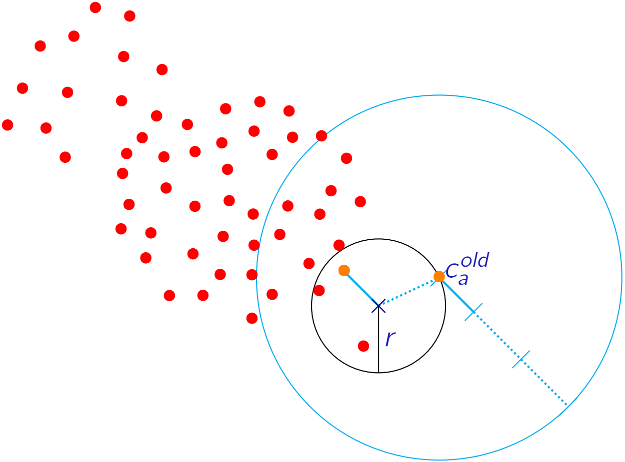
$\big|\, \|c\| - \|x\| \,\big| < r \Rightarrow c \notin \mathcal{B}(x, r)$ ($\bullet$: centers eliminated)



elimination
$O(\log N)$ if
$\|c\|$ sorted

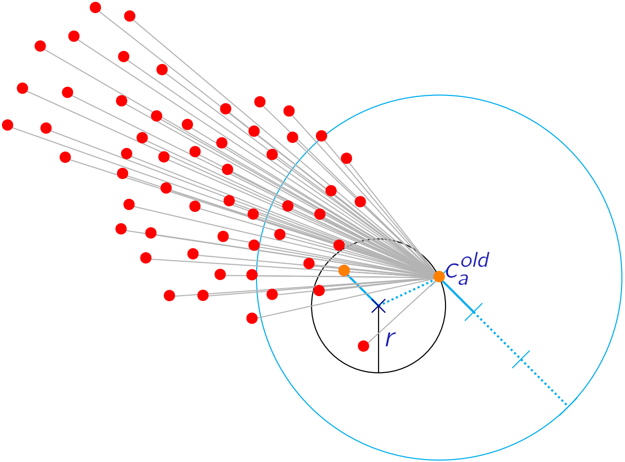
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



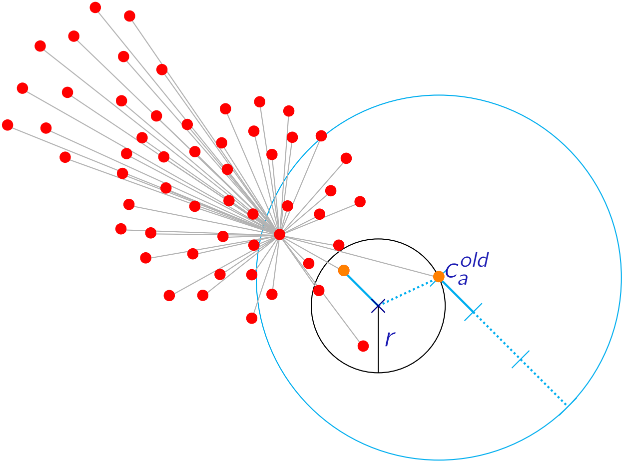
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



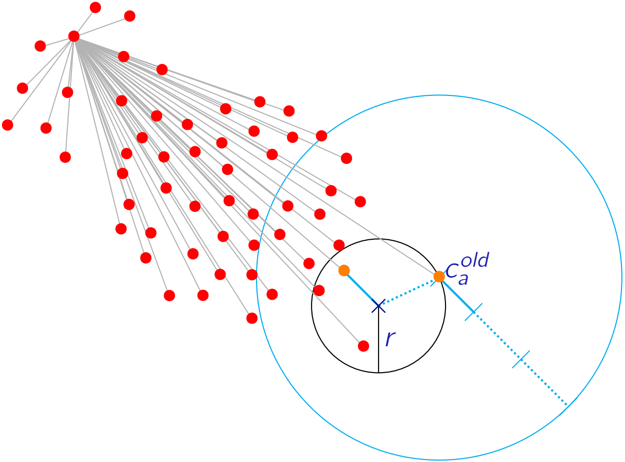
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



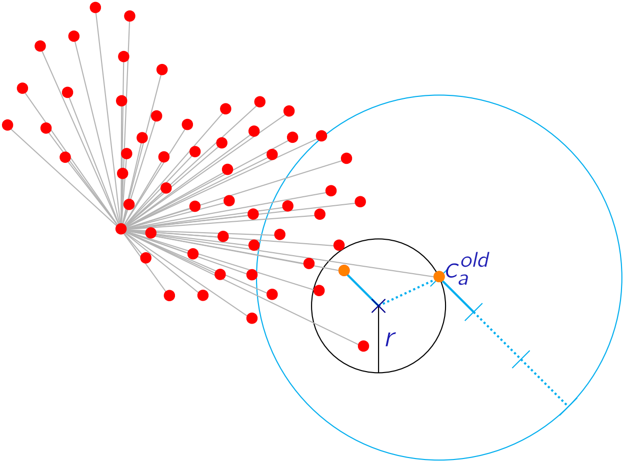
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



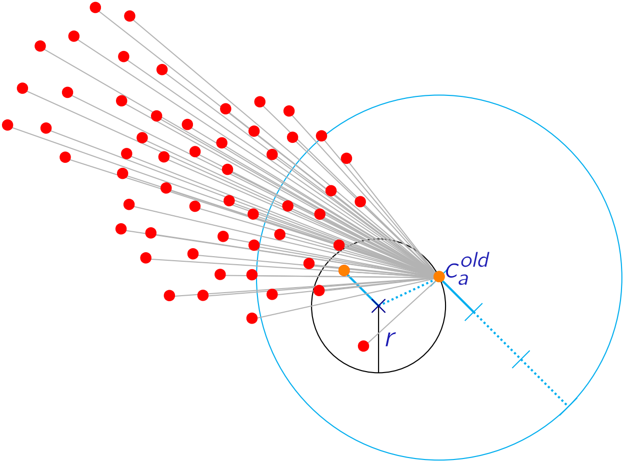
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



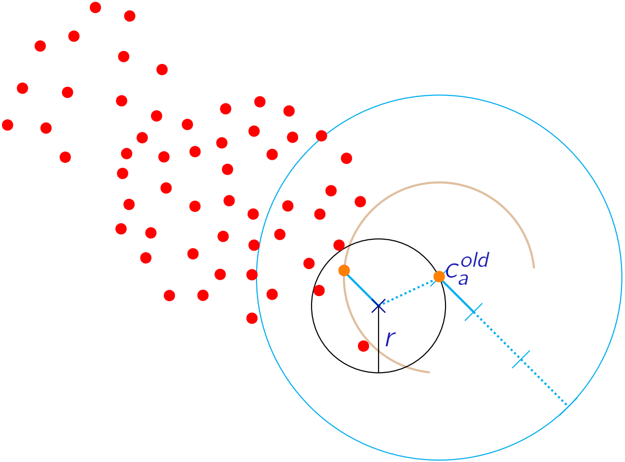
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



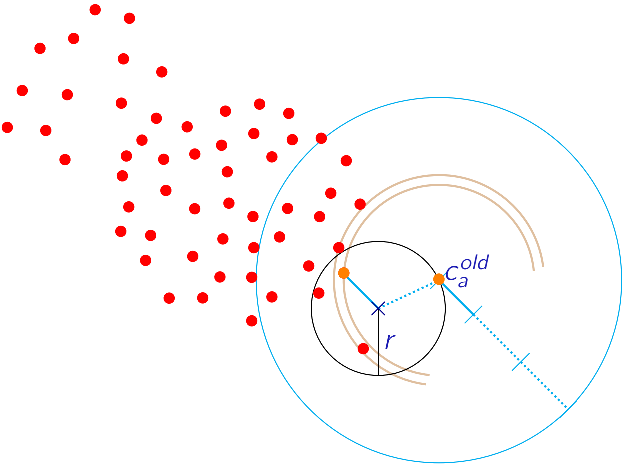
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



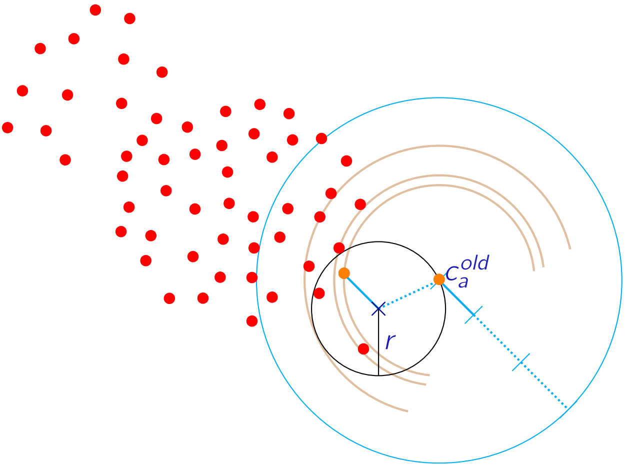
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



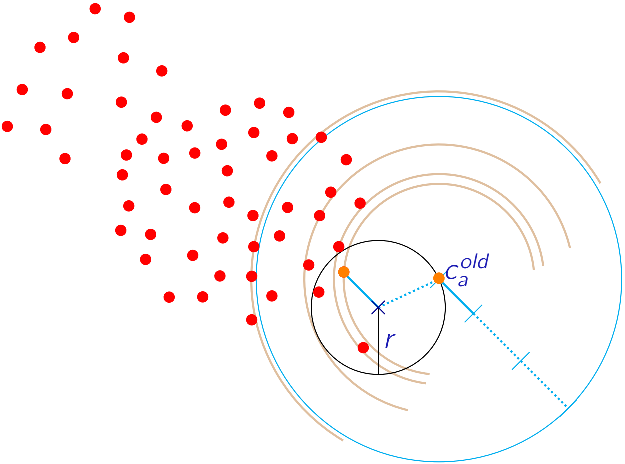
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



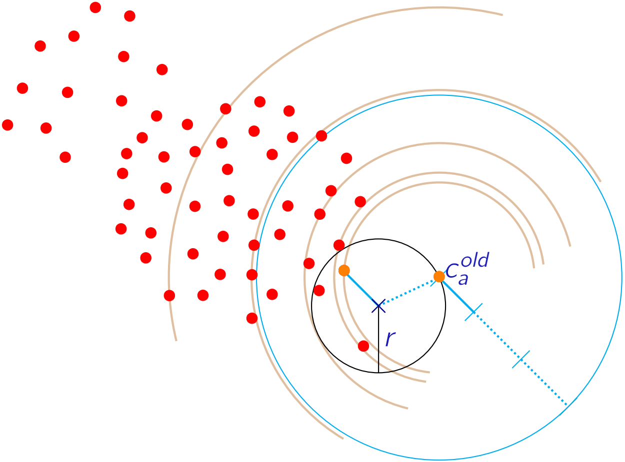
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



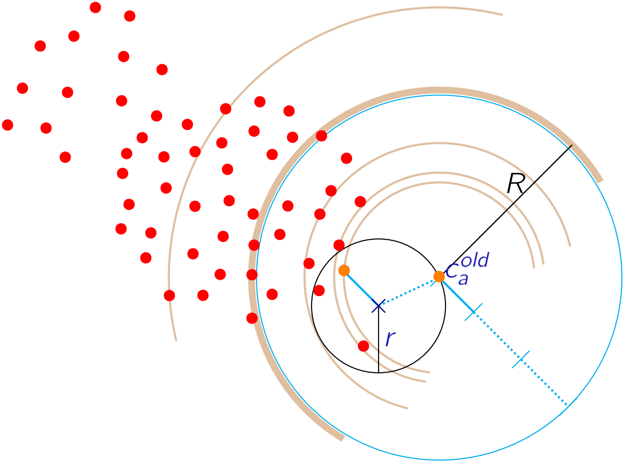
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



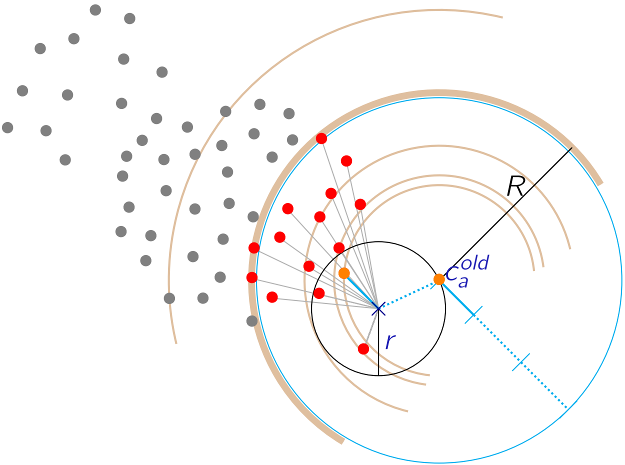
Exponion (ours) elimination zone

$$\|c - c_a^{old}\| > 2\|x - c_a^{old}\| + \|x - c_b^{old}\| \Rightarrow c \notin \mathcal{B}(x, r)$$



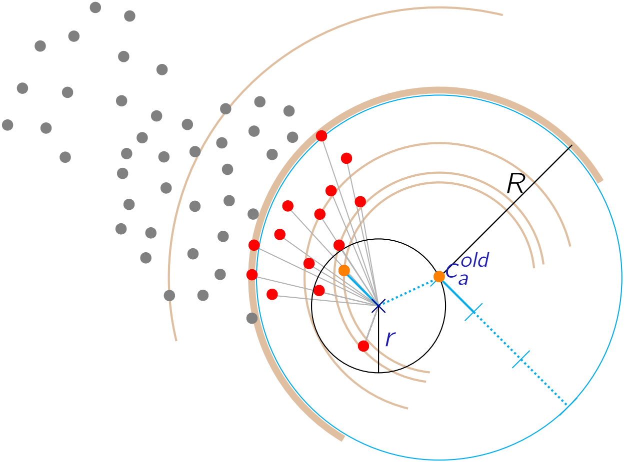
Exponion (ours) elimination zone

$\|c - c_a^{old}\| > R \Rightarrow c \notin \mathcal{B}(x, r)$ (● : centers eliminated)



Exponion (ours) elimination zone

(C) *We find that Exponion is generally faster than Annular*



Experiments and Results

22 datasets ($d : 2 \to 784$, $N : 60k \to 2.6m$) and $K \in \{100, 1000\}$

4 public code bases (mlpack, BaylorML, PowerGraph, VLFeat) +

+ all from scratch

22 datasets ($d : 2 \to 784$, $N : 60k \to 2.6m$) and $K \in \{100, 1000\}$
4 public code bases (mlpack, BaylorML, PowerGraph, VLFeat) +
+ all from scratch

(A) *Simplification accelerates,*
- Elkan in $16/18$ high-d experiments, mean speed-up 15%
- Yinyang in $43/44$ all-d experiments, mean speed-up 60%

22 datasets ($d: 2 \to 784$, $N: 60k \to 2.6m$) and $K \in \{100, 1000\}$
4 public code bases (mlpack, BaylorML, PowerGraph, VLFeat) +
+ all from scratch

(A) *Simplification accelerates,*
- Elkan in $16/18$ high-d experiments, mean speed-up 15%
- Yinyang in $43/44$ all-d experiments, mean speed-up 60%

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

Experiments and Results

22 datasets ($d : 2 \to 784$, $N : 60k \to 2.6m$) and $K \in \{100, 1000\}$
4 public code bases (mlpack, BaylorML, PowerGraph, VLFeat) +
+ all from scratch

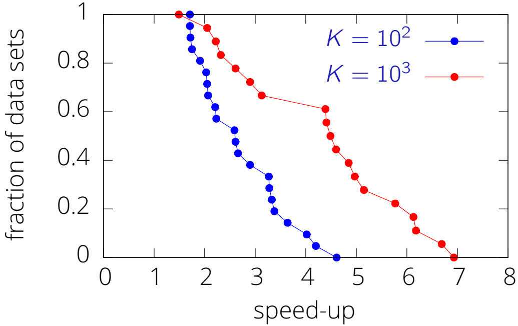
(A) *Simplification accelerates,*
- Elkan in $16/18$ high-d experiments, mean speed-up 15%
- Yinyang in $43/44$ all-d experiments, mean speed-up 60%

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

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

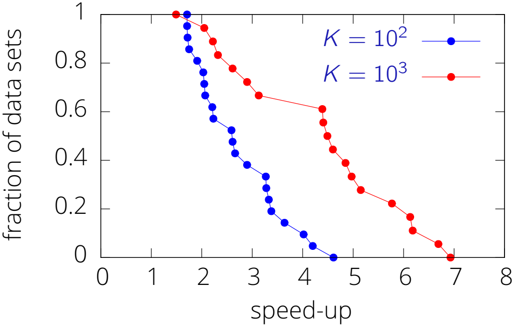
Conclusion

Speed-up: run-times of any of the other 4 implementations of any algorithm relative to our fastest implementations of our algorithms



Conclusion

Speed-up: run-times of any of the other 4 implementations of any algorithm relative to our fastest implementations of our algorithms



Our multi-threaded & easy-to-use code is available under an open source licence

The end

james.newling@idiap.ch