Aalto University
School of Science
and Technology

Yinyang K-Means: A Drop-In Replacement for the Classic K-Means with Consistent Speedup

Yufei Ding et al.

Presented by: Ehsan Amid

in ICML 2015

Motivation

K-means becomes slow when n, k, or d is large

The idea is to develop an algorithm such that

- yields consistent speedup over classic K-means by avoiding unnecessary distance calculations
- produces exact results (results should be exactly the same)
- provides user-control of overheads

Main Idea: using *triangular inequality* to maintain upper bounds and lower bounds for each point

Optimizing the *Assignment* Step

Triangular Inequality: In a metric space, we have

$$|d(x, b) - d(b, c)| \leq d(x, c) \leq d(x, b) + d(b, c)$$

In particular, b and c represent centers of the same cluster in two consecutive iterations

Global Filtering

Notation:

- $b(x)$ (for "best of x") denotes the cluster the point x is assigned to
- C represents the set of cluster centers
- prime is used to show the variables in the next iteration
- $\delta(c) = d(c, c')$ is the shift of the cluster center due to the center update
- $ub(x) \geq d(x, b(x))$ upper bound
- $lb(x) \leq d(x, c), \forall c \in C - b(x)$ global lower bound

Global Filtering

Lemma 1: A point x in the cluster $b = b(x)$ does not change its cluster after a center update if

$$lb(x) - \max_{c \in C} \delta(c) \geq ub(x) + \delta(b)$$

Global Filtering

Lemma 1: A point x in the cluster $b = b(x)$ does not change its cluster after a center update if

$$lb(x) - \max_{c \in C} \delta(c) \geq ub(x) + \delta(b)$$

Proof. Note that the r.h.s is a new upper bound on $d(x, b')$ and the l.h.s is a new lower bound on $d(x, c) \, \forall c \in C - b(x)$

$$d(x, c') \geq d(x, c) - d(c, c') = d(x, c) - \delta(c) \geq d(x, c) - \max_{c \in C} \delta(c)$$

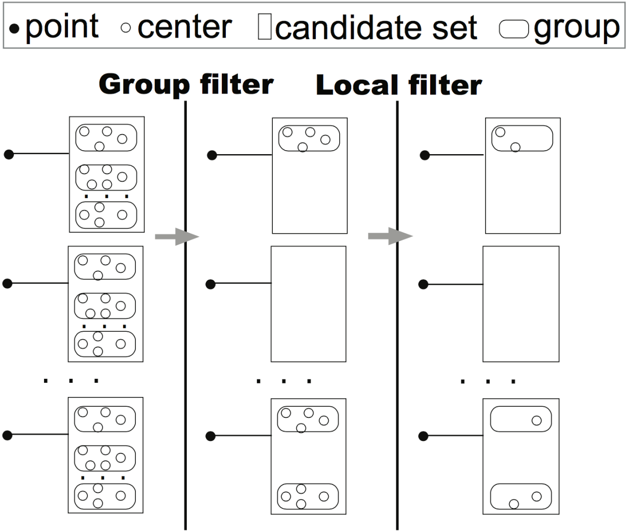
$$d(x, b') \leq d(x, b) + \delta(b) \leq ub(x) + \delta(b)$$

Group Filtering

Global filtering becomes inefficient when there are *big movers*
Solution: instead of a single global lower bound,

1. group the clusters into t groups $G = \{G_1, G_2, \ldots, G_t\}$
2. maintain a lower bound for each group
   $lb(x, G_i) \leq d(x, c), \forall c \in G_i - b(x)$

Group Filtering



Local Filtering

Lemma 2: A center $c' \in G'_i$ cannot be the closest center to a point x if there is a center $p' \neq c'$ (p' does not have to be a part of G'_i) such that

$$d(x, p') < lb(x, G_i) - \delta(c)$$

Local Filtering

Lemma 2: A center $c' \in G_i'$ cannot be the closest center to a point x if there is a center $p' \neq c'$ (p' does not have to be a part of G_i') such that

$$d(x, p') < lb(x, G_i) - \delta(c)$$

Proof. Triangular inequality

$$d(x, c') \geq d(x, c) - d(c, c') \geq lb(x, G_i) - \delta(c) > d(x, p')$$

In practice, using the <u>second closest</u> center as p' gives better results

Optimizing the *Center Update* Step

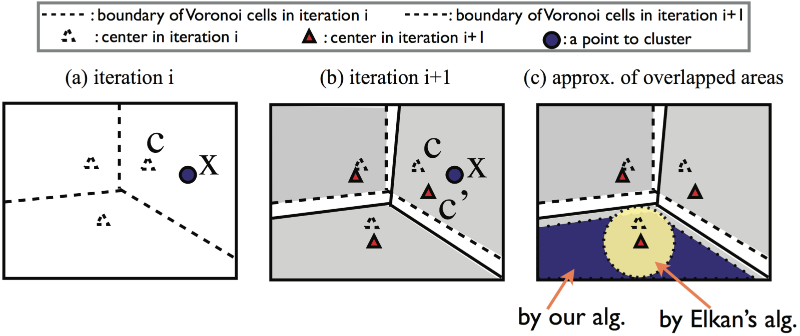
$$c' = (c\,|V| - \sum_{y \in V - OV} y + \sum_{y' \in V' - OV} y')/|V'|$$

where $OV = V \cap V'$

Algorithm

See the paper :)

Comparison



: boundary of Voronoi cells in iteration i
: boundary of Voronoi cells in iteration i+1
: center in iteration i
: center in iteration i+1
: a point to cluster
(a) iteration i
(b) iteration i+1
(c) approx. of overlapped areas
C
X
C'
by our alg.
by Elkan's alg.

Results

Table 1. Cost Comparison (n: # points; k: # clusters; t: # lower bounds per point; α: fraction of points passing through the non-local filter; Drake's algorithm has no local filter)

Algorithm	space cost	time cost		
		lower bounds maintenance	non-local filtering	local filtering
Yinyang K-means	$O(n*t)$	$O(n*t)$	$O(n) \sim O(n*t)$	$O(n*\alpha*k)$
Elkan's (Elkan, 2003)	$O(n*k)$	$O(n*k)$	$O(k^2*d+n)$	$O(n*\alpha*k)$
Drake's (Drake & Hamerly, 2012)	$O(n*t)$	$O(n*t) \sim O(n*k+n*t*\log t)$	$O(n*t)$	–

Results

Table 2. Time and speedup on an Ivybridge machine (16GB memory, 8-core i7-3770K processor)
("-" indicates that the algorithm fails to run for out of memory)

Data Set	n	d	k	No. iter	Assignment: Standard time/iter (ms)	Assignment: Speedup (X) over Standard: Elkan	Drake	Yinyang K-means $t=1$	Yinyang K-means elastic	Overall Speedup (X) of Yinyang over: Standard	Elkan	Drake
I. Kegg Network	6.5E4	28	4	50	2.7	1.29	1.97	2.08	2.08	1.14	1.09	1.07
			16	52	9.9	1.62	2.13	2.48	2.48	1.61	1.36	1.12
			64	68	28.0	1.78	2.21	2.55	3.37	2.61	1.98	1.56
			256	59	89.6	1.89	1.63	2.23	4.98	4.86	3.60	3.98
II. Gassensor	1.4E4	129	4	16	3.1	4.60	4.34	4.68	4.68	1.13	1.07	1.11
			16	54	5.4	2.84	2.01	2.70	2.70	1.41	1.07	1.27
			64	66	20.3	5.08	3.08	3.17	5.49	3.29	1.82	2.28
			256	55	84.3	6.48	2.06	3.01	10.28	5.40	1.85	4.72
III. Road Network	4.3E5	4	4	24	10.1	**0.72**	1.23	1.36	1.36	1.18	1.24	1.17
			64	154	80.0	**0.85**	3.42	4.10	3.85	3.63	3.82	1.12
			1,024	161	1647.3	1.25	2.14	4.08	8.45	13.59	12.71	5.21
			10,000	74	16256.1	-	1.88	2.80	9.63	12.57	-	6.84
IV. US Census Data	2.5E6	68	4	6	182.0	1.88	1.94	2.08	2.08	1.10	1.04	1.04
			64	56	2176.4	3.57	4.56	4.85	8.47	5.40	2.43	2.14
			1,024	154	37603.9	**0.23**	2.96	3.56	24.89	23.45	89.53	6.33
			10,000	152	432976	-	- (1.64)	2.90	3.05	5.70	-	- (2.15)
V. Caltech101	1E6	128	4	55	111.0	2.44	2.88	3.02	3.02	1.83	1.41	1.04
			64	314	1432.6	5.52	5.07	5.64	10.21	8.65	1.79	1.26
			1,024	369	22816.8	5.56	3.62	3.38	21.99	22.33	6.41	5.71
			10,000	129	316850	-	- (3.25)	3.12	20.24	22.23	-	- (6.74)
VI. NotreDame	4E5	128	4	145	46.8	2.85	3.38	3.69	3.69	2.40	1.65	1.05
			64	232	585.8	5.27	4.57	4.29	6.81	6.16	1.88	1.76
			1,024	149	9334.1	5.66	2.82	2.28	10.44	10.69	3.25	4.19
			10,000	47	126815	-	2.35	2.32	10.81	11.53	-	5.27
VII. Tiny	1E6	384	4	103	277.0	6.67	7.58	8.20	8.20	3.24	1.90	1.21
			64	837	4113.4	14.23	7.39	6.32	15.26	13.89	1.93	1.93
			1,024	488	64078.8	16.02	4.37	2.94	23.64	23.21	2.78	5.14
			10,000	146	781537	-	- (3.45)	2.35	15.51	16.13	-	- (5.96)
VIII. Ukbench	1E6	128	4	62	113.7	2.63	2.86	3.17	3.17	1.94	1.46	1.10
			64	506	1431.1	5.75	7.36	6.61	13.21	10.85	3.12	1.72
			1,024	517	22787.4	5.95	4.28	3.42	23.41	24.26	6.85	5.18
			10,000	208	316299	-	- (3.92)	3.09	28.50	32.18	-	- (6.32)
					average	4.33	3.39	3.51	9.87	**9.36**	**6.12**	**3.08**

Results

Table 3. Overall speedup over standard K-means on a Core2 machine (4GB mem, 4-core Core2 CPU)
(*: not a meaningful setting for the small data size; -: out of memory)

Data Set		I	II	III	IV	V	VI	VIII
k=4	Yinyang	1.35	1.10	1.09	1.13	1.97	2.60	2.05
	Elkan	1.09	1.08	**0.90**	1.06	1.30	1.44	1.32
	Drake	1.26	1.05	1.05	1.08	1.79	2.44	1.82
k=64	Yinyang	2.34	2.91	2.79	5.23	8.12	5.75	10.39
	Elkan	1.33	2.29	**0.97**	2.25	3.52	3.23	3.43
	Drake	2.04	1.67	2.31	4.42	3.17	2.96	3.28
k=1024	Yinyang	*	*	8.98	20.41	22.64	10.64	27.34
	Elkan	*	*	1.20	–	–	3.52	–
	Drake	*	*	2.18	– (3.18)	3.26	2.48	3.79
k=10,000	Yinyang	*	*	14.74	6.39	17.87	8.20	28.11
	Elkan	*	*	–	–	–	–	–
	Drake	*	*	– (2.01)	– (1.68)	– (2.58)	– (1.73)	– (3.02)

Results

Table 4. Unnecessary distance calculations detected by the non-local filters of Yinyang K-means and two prior algorithms ($k = 64$)

Data Set	I	II	III	IV	V	VI	VII	VIII	average
Yinyang	69.3%	71.0%	88.5%	85.1%	81.7%	77.9%	82.0%	86.2%	80.2%
Elkan	31.1%	32.5%	32.6%	9.2%	0.5 %	2.0E-6	1.6E-6	1.4%	13.4%
Drake	64.2%	58.6%	69.3%	71.7%	73.6 %	68.1%	62.9%	77.7%	68.3%

Results

Table 5. Yinyang K-means accelerates the center update step by many times (*: not a meaningful setting for the small data size)

Data Set	I	II	III	IV	V	VI	VII	VIII
k=4	2.4	3.2	2.1	1.8	4.1	4.3	9.0	4.0
k=64	9.6	20.1	8.4	8.0	11.0	9.8	15.4	11.9
k=1024	*	*	107.0	59.9	97.0	43.1	106.6	114.3
k=10,000	*	*	62.5	79.7	66.2	27.5	50.8	100.4

Conclusions