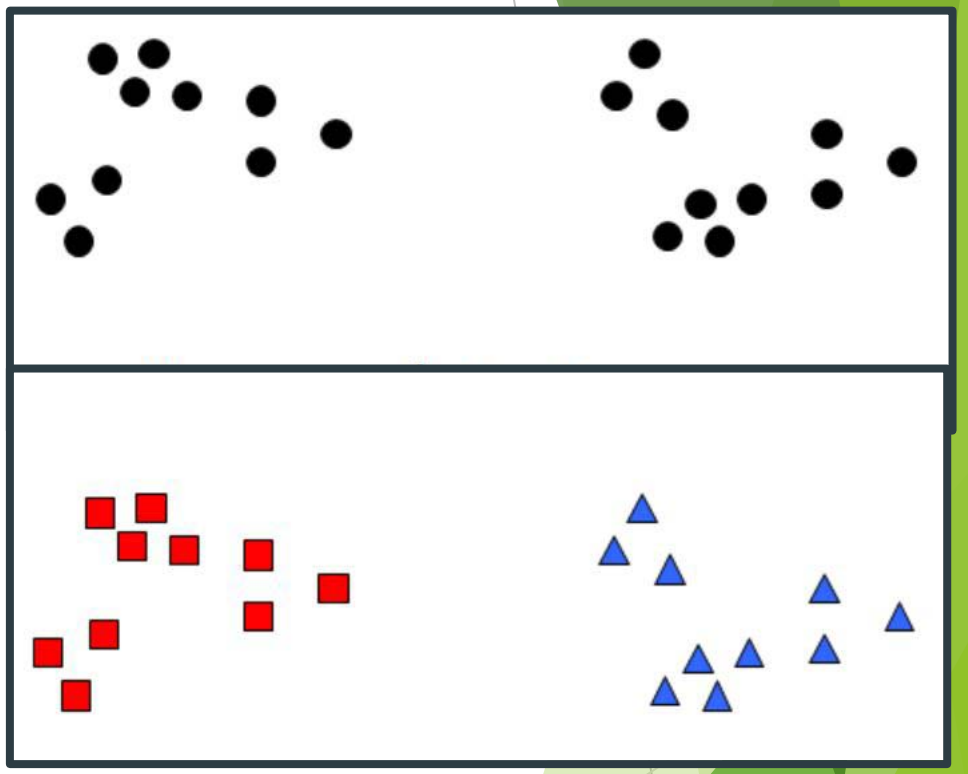
Parallel clustering: k-Means

Soumi Maiti

April 20, 2017

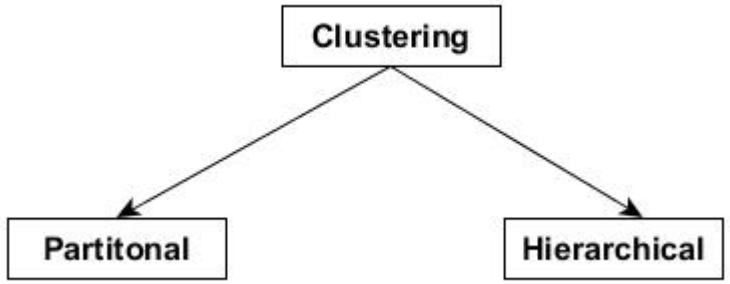
Clustering

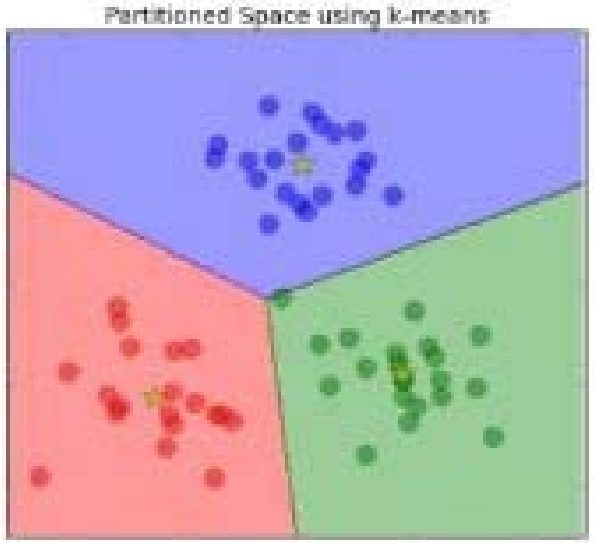
- Grouping input data into subsets called "***clusters***"
- Within each cluster elements are ***similar***
- Unsupervised learning
- Computationally expensive
  - Many of the algorithms are iterative or recursive procedure

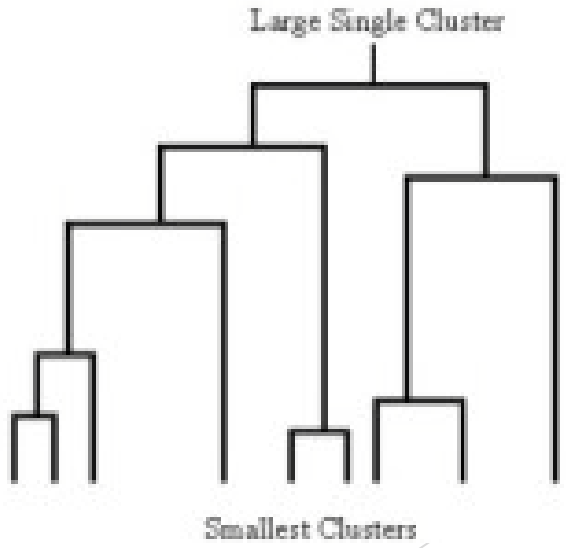


Different types of clustering

- Partitional
  - **K-Means**
- Hierarchical
  - Single-Linkage

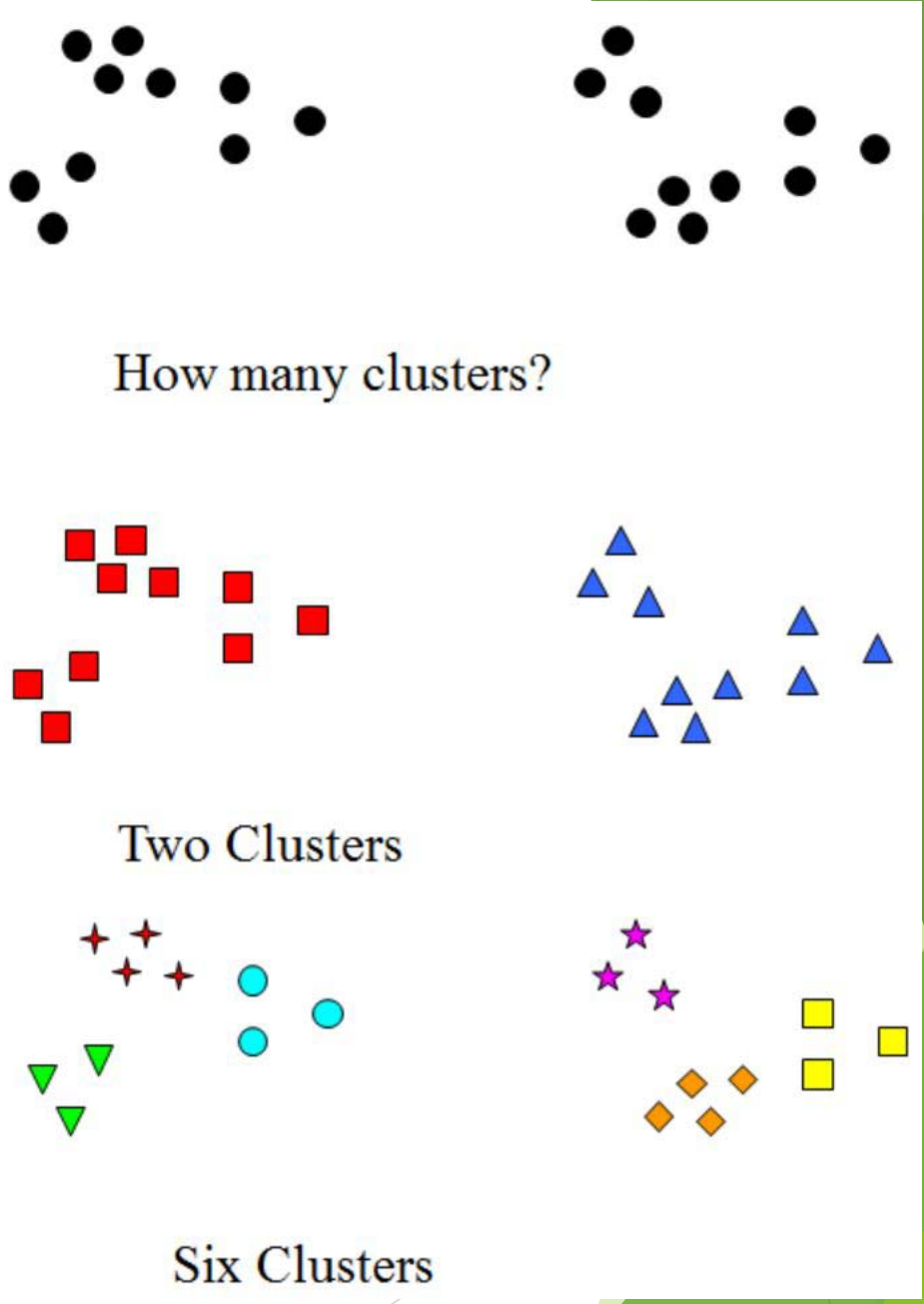






K-Means clustering

- Simplest and most efficient clustering algorithm
- Most widely used
- Computationally expensive
- K= number of clusters
  - User defined parameter



Formalizing K-Means

- Given a set S of N points,
- Form K subsets $\{C_1, C_2, C_3, \ldots C_K\}$ s.t.,
- Each point v_{ij} is closest to it's cluster mean $\overline{X_i}$

  where $1 \leq i \leq K$, $1 \leq j \leq |C_i|$
- Mean $\overline{X_i} = \frac{1}{|C_i|} \sum_{v \in C_i} v$

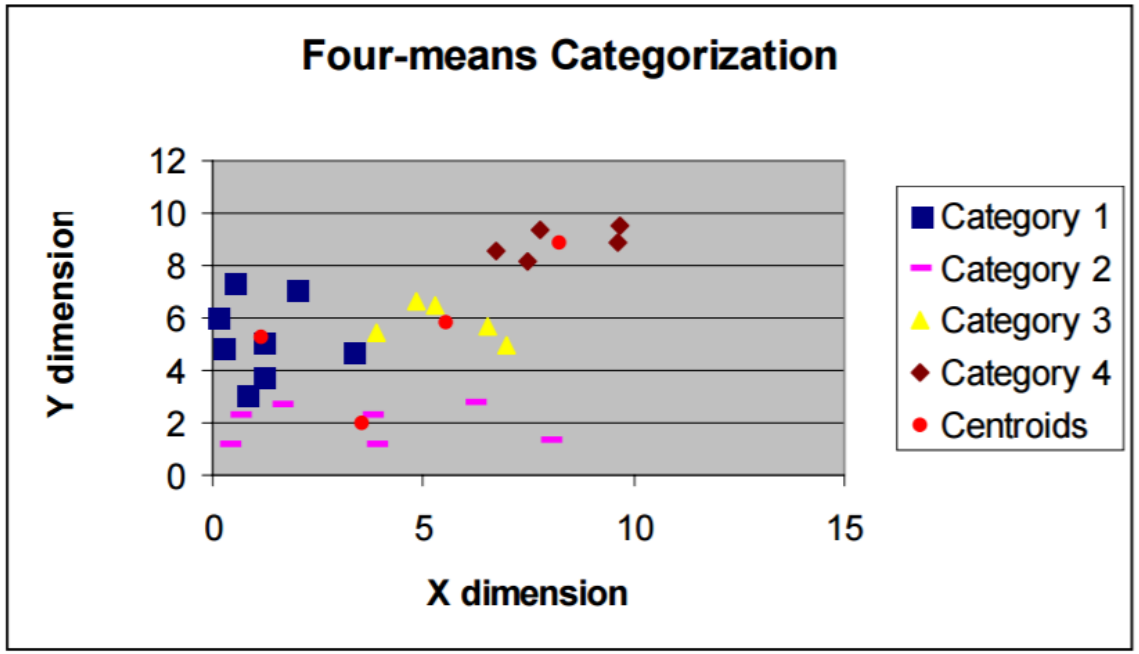


Figure 1. Two-dimensional data are clustered into 4 categories by running K-means algorithm on the data. Each color represents each category. The centroids are the representatives of each category.

- Achieves by minimizes the ***square-error*** (E),

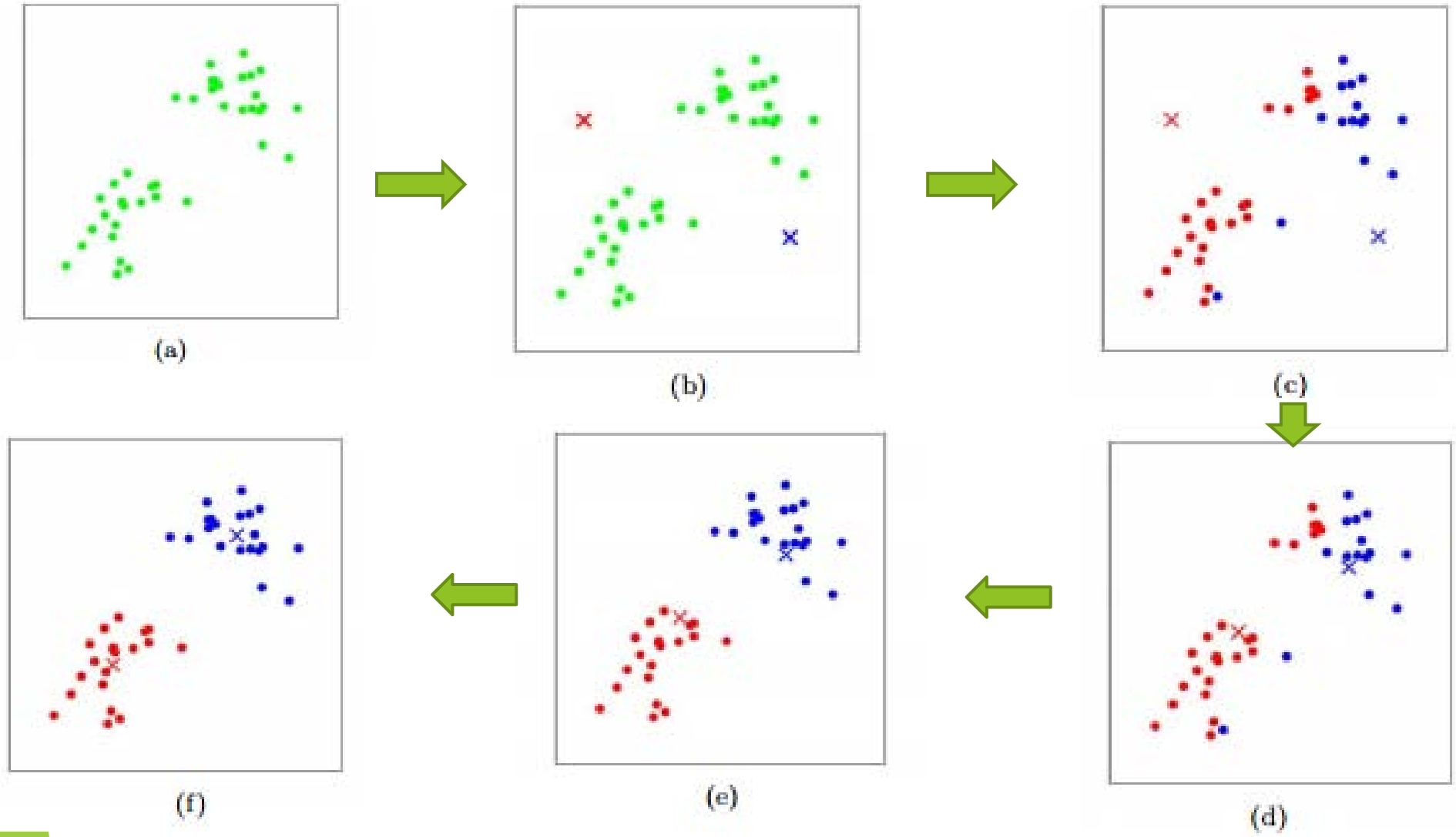
$$E = \sum_{i=1}^{K} \sum_{v \in C_i} \left| \overline{X_i} - v \right|^2$$

Serial K-means Algorithm

- Randomly select K-subsets
- While Square-Error(E) is not stable:
  - Compute mean $\overline{X_i}$, 1≤ i ≤ K
  - Compute distance d(i,j), 1≤ i ≤ K, 1≤ j ≤ N for each point to each mean.
  - Choose closest members as the new member of K clusters
- end

- Time complexity = O(KN) per iterations
  - where K is the number of desired clusters
  - If you have d dimensional data, T = O(KNd)
- Total time complexity = $O(R_s\ KN)$
  - R_s is the number of iterations

Visualization 2-Means clustering



Expensive calculations

- Most intensive calculation → calculation of distances
- In each iteration,
  - total of (**NK**) distances are computed
    - where N is the number of data points
    - K is the number of clusters
- Distance computations between one object with the centers is irrelevant to the distance computations between other objects with the corresponding centers.
- Distance computations between different objects with centers can be parallel executed

Parallel Algorithm -1

- Uses master-slave architecture
  - One master
  - K slaves
    - Each slave computes one cluster
- Message passing interface
  - Uses broadcast to reduce time
- Speedup of O(K/2)
- Kantabutra, Sanpawat, and Alva L. Couch. "Parallel K-means clustering algorithm on NOWs." *NECTEC Technical journal* 1.6 (2000): 243-247.

Master Process

- Divides set S into K equal subsets
- Send each subset to each of the K slaves
- Receive K resulting subsets from K slaves
- Complexity
  - $K(T_{startup} + N/K\ T_{data})$

Slave Process

- Receive a subset P from master process
- While Error E is not stable:
  - Compute a mean $\overline{X}$
  - Broadcast $\overline{X}$ to every other slaves
  - Compute distance d(i,j), $1 \le i \le K, 1 \le j \le |P|$
  - Choose new vector members of the new K subsets
    - according to their closest distance to $\overline{Xi}$, $1 \le i \le K$
  - Broadcast K subsets computed in previous step to every other slaves
  - Update P by collecting vectors that belong to $\overline{X}$ from other slaves
- end
- Send the subset P to master process

Example

- K= 2, N=6, data = [40,102,42, 35, 99, 85]

Master

S1 = [40, 102, 42]
S2 = [35, 99, 85]

S1

S2

Slave 1

P = 40,102,42

$\overline{X1}$ = 61

Broadcast x1:61

dist	X1=61	X2=73
40	21	33
102	61	29
42	19	31

Slave 2

P = 35, 99, 85

$\overline{X2}$ = 73

Broadcast x2:73

dist	X1=61	X2=73
35	26	38
99	38	26
85	24	12

Master

Slave 1

Choose groups:
1: {40, 42} 2:{102}

Broadcast 40:1, 42:1 102:2

P = collect all members of 1 = 40, 42, 35
Repeat loop again

$\overline{X1}$ = 39

Broadcast x1:39

dist	X1=39	X2=95
40	1	55
42	3	53
35	4	60

1: {40, 42, 35}

Slave 2

Choose groups:
1: {35} 2:{99, 85}

Broadcast 35:1, 99:2 85:2

P = 99, 85, 102
Repeat loop again

$\overline{X2}$ = 95

Broadcast x2:95

dist	X1=39	X2=95
99	60	4
85	46	10
102	63	7

2: {99, 85, 102}

- Let R_p = the number of iterations

- Total time = Communication Time + Computation Time
- Communication Time = $O(N + R_p\ |P|)$
- Computation Time = $O(2R_p\ K|P|)$
- Total time = $O(N + R_p|P|) + O(2R_p K|P|) = O(2R_p K|P|)$

- Speedup = Execution time of single processor / Execution time of K processors

$$= \frac{O(R_S\ K\ N)}{O(2R_p\ K\ |P|)} = \frac{O(R_S\ K^2\ P)}{O(2R_p\ K\ |P|)} = O(\frac{K}{2})$$

[as, |p| = N/K]

Parallel Algorithm 2

- Master-slave procedure
  - Variable number of slave process
- Splits the dataset among processes
  - Each processor works on part of the data
- Reduces the communication cost
  - Only communicates means between slaves

- Joshi, Manasi N. "Parallel k-means algorithm on distributed memory multiprocessors." *Computer* 9 (2003).

Algorithm

- Master calculates the initial centroids
- Send k centroids to all slaves
- Each processor
  - Computes distance of a subset of data points
  - Assign points to closest centroid
  - Compute local error
- Performs reduction for global centroids and global error
  - Uses MPIAllReduce()

Example

- K= 2, N=6, data = [40,102,42, 35, 99, 85]
- X1 = 61, X2 = 73
- 3 processor

Process 1	Process 2	Process 3
D = 40,102 X1 = 61, X2 = 73 1: {40} 2:{102} S1 = 40, N1=1 S2 = 102 N2=1	D = 42, 35 X1 = 61, X2 = 73 1: {42, 35} 2:{} S1 = 77, N1=2	D= 99, 85 X1 = 61, X2 = 73 1: {} 2:{99,85} S2 = 184, N2=2

All Reduce

S1 = 117 N1=3
S2 = 286 N2=3

X1 = S1/N1 = 39 X2 = 95 1: {40} 2:{102}	X1 = S1/N1 = 39 X2 = 95 1: {42, 35} 2:{}	X1 = S1/N1 = 39 X2 = 95 1: {} 2:{99,85}

Process 1	Process 2	Process 3
S1 = 40, N1=1 S2 = 102 N2=1	S1 = 77, N1=2	S2 = 184, N2=2
All Reduce S1 = 117 N1=3 S2 = 286 N2=3		
Converged	Converged	Converged

Analysis

- Assume we have P processors
- Also assume n is divisible by P
- The process identified by ‘id’
  - Process data $\{X_i, i = (id) *(n/P),\ldots,(id + 1)*(n/P)\}$.
  - n/P data points
- Each of the P processes can carry out the “distance calculations” in parallel
- With n/P data points
- Extra overhead is reduce for K centroids

Analysis- Speedup

Sequential: $(3nkd + nk + nd + kd)* \mu * T$

-- *3nkd* for distance calculation

-- *nk* finding the closest centroid

-- *nd* ($m'_l = m'_l + X_i$; $n'_l = n'_l + 1$;)

Parallel: $\frac{(3nkd * \mu * T)}{P} + d*k* \mu *Tp^{reduce}$

- Speedup is approximately P