K-Means Clustering

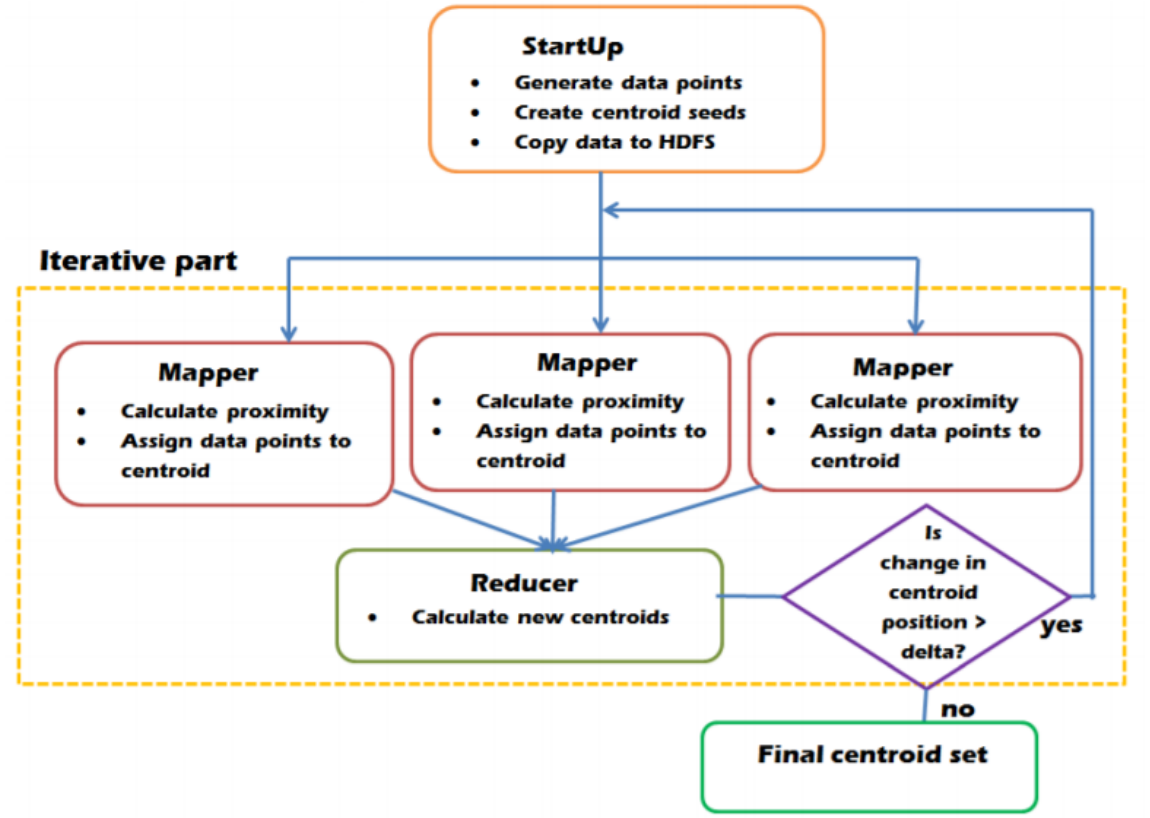
Using Map-Reduce

Team Members:

Rahul Sathyajit
Karthick Raja
Akshar Vashist

Contents

Structure of the Algorithm



Algorithm for Startup

Require:

- A set of d-dimensional objects X= $\{x_1, x_2, \ldots, x_n\}$
- k-number of clusters where $k < n$
- initial set of centroids $C = \{c_1, c_2, \ldots, c_k\}$
- δ convergence delta

Output: A new set of centroids, number of iterations, final clusters and the time taken to converge

1: $current\_centroids \Leftarrow C$
2: $initialise\ numIter, timetoConverge, finalClusters$
3: $startTime \Leftarrow currentTime()$
4: $C' \Leftarrow$ call $SimpleAssignment$
5: $new\_centroids \Leftarrow C'$
6: $numIter \Leftarrow 1$
7: **while** $change(new\_centroids, current\_centroids) > \delta$ **do**
8: $current\_centroids \Leftarrow new\_centroids$
9: $C' \Leftarrow$ call $SimpleAssignment$
10: $numIter \Leftarrow numIter + 1$
11: $current\_centroids \Leftarrow C'$
12: **end while**
13: $endTime \Leftarrow currentTime()$
14: $timetoConverge \Leftarrow (endTime - startTime)$
15: $perform\ outlierRemoval$
16: $finalClusters \Leftarrow$ perform $finalClustering$
17: $writeStatistics$
18: **return** $current\_centroids$, numIter, $finalClusters$, $timetoConverge$

Algorithm for Simple Assignment

Require:

- A set of d-dimensional objects of $\{x_1, x_2, \ldots, x_n\}$
- initial set of centroids $C = \{c_1, c_2, \ldots, c_k\}$

Output: A dictionary list consisting of each centroid and the objects assigned to them. This list is passed as an argument to the CentroidCalculator program.

$M_1 \Leftarrow \{x_1, x_2, \ldots, x_n\}$
$current\_centroids \Leftarrow C$
$distance\,(p, q) = \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2}$ where p_i (or q_i) is the coordinate of p (or q) in dimension i $v \Leftarrow \{\}$
for all $x_i \in M_1$ such that $1 \leq i \leq n$ **do**
    $bestCentroid \Leftarrow null$
    $minDist \Leftarrow \infty$
    for all c in $current\_centroids$ **do**
        $dist \Leftarrow distance(x_i, c)$
        if $bestCentroid = null \;||\; dist < minDist$ **then**
            $bestCentroid \Leftarrow c$
            $minDist \Leftarrow dist$
        end if
    end for
    $v[bestCentroid] \Leftarrow (x_i)$
    i +=1
end for
call $CentroidCalculator(v)$

Algorithm for centroid calculator

Require:
Input: List of centroids and their assigned objects
Output: List of new centroids

```
1: v = {}
2: v = outputlist from SimpleAssignment
3: newCentroidList ⇐ null
4: for all centroid ∈ v do
5:     newCentroid, sumofObjects, numofObjects ⇐ null
6:     for all object ∈ v[centroid] do
7:         sumofObjects += object
8:         numofObjects += 1
9:     end for
10:    newCentroid ⇐ (sumofObjects ÷ numofObjects)
11:    newCentroidList << (newCentroid)
12: end for
13: return newCentroidList
```

Algorithm for Mapper

Require:

- A subset of d-dimensional objects of $\{x_1, x_2, \ldots, x_n\}$ in each mapper
- initial set of centroids $C = \{c_1, c_2, \ldots, c_k\}$

Output: A list of centroids and objects assigned to each centroid separated by tab. This list is written locally one line per data point and read by the Reducer program.

1: $M_1 \Leftarrow \{x_1, x_2, \ldots, x_m\}$
2: $current\_centroids \Leftarrow C$
3: $distance\,(p, q) = \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2}$ where p_i (or q_i) is the coordinate of p (or q) in dimension i
4: **for all** $x_i \in M_1$ such that $1 \leq i \leq m$ **do**
5: $bestCentroid \Leftarrow null$
6: $minDist \Leftarrow \infty$
7: **for all** $c \in current\_centroids$ **do**
8: $dist \Leftarrow distance(x_i, c)$
9: **if** $bestCentroid = null \;||\; dist < minDist$ **then**
10: $minDist \Leftarrow dist$
11: $bestCentroid \Leftarrow c$
12: **end if**
13: **end for**
14: $outputlist << (bestCentroid, x_i)$
15: i +=1
16: **end for**
17: **return** *outputlist*

Algorithm for Reducer

Require:

Input: (key, value) where key = bestCentroid and value = objects assigned to the centroids by the mapper.

Output: (key, value) where key = oldcentroid and value = newBestCentroid which is the new centroid value calculated for that bestCentroid.

```
1: outputlist ⇐ outputlists from mappers
2: v ⇐ {}
3: newCentroidList ⇐ null
4: for all y ∈ outputlist do
5:     centroid ⇐ y.key
6:     object ⇐ y.value
7:     v[centroid] ⇐ object
8: end for
9: for all centroid ∈ v do
10:    newCentroid, sumofObjects, numofObjects ⇐ null
11:    for all object ∈ v[centroid] do
12:        sumofObjects += object
13:        numofObjects += 1
14:    end for
15:    newCentroid ⇐ (sumofObjects ÷ numofObjects)
16:    newCentroidList << (newCentroid)
17: end for
18: return newCentroidList
```

Reference:

D. Gillick, A. Faria, and J. DeNero. Mapreduce: Distributed computing for machine learning, 2006

W. Zhao, H. Ma, and Q. He. Parallel k-means clustering based on mapreduce. Cloud Computing, pages 674–679, 2009.

K-means Clustering Using Hadoop MapReduce, Grace Nila Ramamoorthy