Determining the k in k-means with MapReduce

Thibault Debatty, Pietro Michiardi, Wim Mees & Olivier Thonnard
Clustering & k-means

• Clustering

• K-means
  - 1982 (a great year!)
  - But still largely used
  - Drawbacks (amongst others):
    • Local minimum
    • K is a parameter!
Clustering & k-means

• Determine k:
  – VERY difficult
  – Using cluster evaluation metrics:
    Dunn's index, Elbow, Silhouette, “jump method” (based on information theory), “Gap statistic”, ...

\[ O(k^2) \]
G-means

- G-means

- K-means: points in each cluster are spherically distributed around the center
G-means

- G-means

- K-means: points in each cluster are spherically distributed around the center

  normality test & recursion
G-means

Dataset

Determining the k in k-means with MapReduce
G-means

1. Pick 2 centers
G-means

2. k-means

Determining the k in k-means with MapReduce
G-means

3. Project

Determining the k in k-means with MapReduce
3. Project

Determining the k in k-means with MapReduce
G-means

4. Normality test

Normal? No => iterate
G-means

5. Recursion
MapReduce G-means

- Challenges:
  1. Reduce I/O operations
  2. Reduce number of jobs
  3. Maximize parallelism
  4. Limit memory usage
MapReduce G-means

- Challenges:
  1. Reduce I/O operations
  2. Reduce number of jobs
  3. Maximize parallelism
  4. Limit memory usage
2. Reduce number of jobs

PickInitialCenters
while Not ClusteringCompleted do
    KMeans
    KMeansAndFindNewCenters
    TestClusters
end while
MapReduce G-means

3. Maximize parallelism
4. Limit memory usage

TestClusters

Map(key, point)
  Find cluster
  Find vector
  Project point on vector
  Emit(cluster, projection)
end procedure

Reduce(cluster, projections)
  Build a vector
  ADtest(vector)
  if normal then
    Mark cluster
  end if
end procedure
3. Maximize parallelism

4. Limit memory usage (risk of crash)
Determining the k in k-means with MapReduce

MapReduce G-means

TestFewClusters

Map(key, point)
  Find cluster
  Find vector
  Project point on vector
  Add projection to list
end procedure

Close()
  For each list do
    Build a vector
    A2 = ADtest(vector)
    Emit(cluster, A2)
  End for each
end procedure

TestClusters

Map(key, point)
  Find cluster
  Find vector
  Project point on vector
  Emit(cluster, projection)
end procedure

Reduce(cluster, projections)
  Build a vector
  ADtest(vector)
  if normal then
    Mark cluster
  end if
end procedure

In memory combiner
MapReduce G-means

TestFewClusters

Map(key, point)
Find cluster
Find vector
Project point on vector
Emit(cluster, projection)
end procedure

Reduce(cluster, projections)
Build a vector
ADtest(vector)
if normal then
Mark cluster
end if
end procedure

TestClusters

#clusters > #reducers
&
Estimated required memory < Java heap
MapReduce G-means

TestFewClusters

Map
Find
Project
Add
end procedure

Close()
For each
Build a vector
A2 = ADtest(vector)
Emit(cluster, A2)
End for each
end procedure

TestClusters

Reduce(cluster, projections)
Build a vector
ADtest(vector)
if normal then
Mark cluster
end if
end procedure

#clusters > #reducers
&
Estimated required memory < Java heap

Experimentally: 64 Bytes / point
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>MR multi-k-means</th>
<th>MR G-means</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- all possible values of $k$ in a single job
## Comparison

<table>
<thead>
<tr>
<th></th>
<th><strong>MR multi-k-means</strong></th>
<th><strong>MR G-means</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speed</strong></td>
<td>$O(nk^2)$ computations</td>
<td>$O(nk)$ computations</td>
</tr>
<tr>
<td></td>
<td>But:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● more iterations</td>
<td>● more iterations</td>
</tr>
<tr>
<td></td>
<td>● more dataset reads</td>
<td>● $\log_2(k)$</td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td>New centers added if and where needed</td>
<td>But: tends to overestimate k!</td>
</tr>
</tbody>
</table>

Determining the k in k-means with MapReduce
Experimental results: Speed

- Hadoop
- Synthetic dataset
- 10M points in $\mathbb{R}^{10}$
- Euclidean distance
- 8 machines
## Experimental results: Quality

### Within Cluster Sum of Square
(less is better)

<table>
<thead>
<tr>
<th></th>
<th>100</th>
<th>200</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>k</strong></td>
<td>150</td>
<td>279</td>
<td>639</td>
</tr>
<tr>
<td><strong>k_{found}</strong></td>
<td>150</td>
<td>279</td>
<td>639</td>
</tr>
<tr>
<td><strong>MR G-means</strong></td>
<td><strong>3.34</strong></td>
<td><strong>3.33</strong></td>
<td><strong>3.23</strong></td>
</tr>
<tr>
<td><strong>multi-k-means</strong></td>
<td><strong>3.71</strong></td>
<td><strong>3.60</strong></td>
<td><strong>3.39</strong></td>
</tr>
</tbody>
</table>

- Hadoop
- Synthetic dataset
- 10M points in $\mathbb{R}^{10}$
- Euclidean distance
- 8 machines

---

Determining the $k$ in $k$-means with MapReduce
Conclusions & future work...

- MapReduce algorithm to determine $k$
- Running time proportional to $k$
- Future:
  - Overestimation of $k$
  - Test on real data
  - Test scalability
  - Reduce I/O (using Spark)
  - Consider skewed data
  - Consider impact of machine failure
Thank you!