Determining the k in k-means with MapReduce

Thibault Debatty, Pietro Michiardi,
Wim Mees & Olivier Thonnard

Clustering & k-means

- Clustering
- K-means
  [Stuart P. Lloyd. Least squares quantization in pcm. IEEE Transactions on Information Theory, 28:129–137, 1982.]
  - 1982 (a great year!)
  - But still largely used
  - Drawbacks (amongst others):
    - Local minimum
    - K is a parameter!

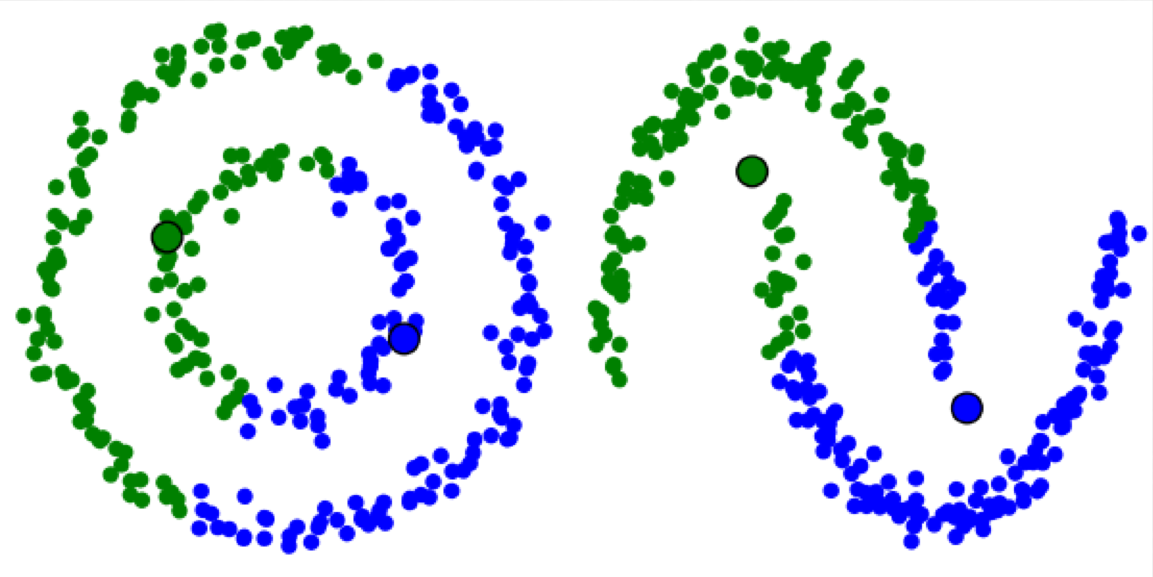
Clustering & k-means

- Determine k:
  - VERY difficult
    [Anil K Jain. Data Clustering : 50 Years Beyond K-Means. Pattern Recognition Letters, 2009]
  - Using cluster evaluation metrics:
    Dunn's index, Elbow, Silhouette, “jump method” (based on information theory), “Gap statistic”,...

→ $O(k^2)$

G-means

- G-means
  [Greg Hamerly and Charles Elkan. Learning the k in k-means. In Neural Information Processing Systems. MIT Press, 2003]
- K-means : points in each cluster are spherically distributed around the center



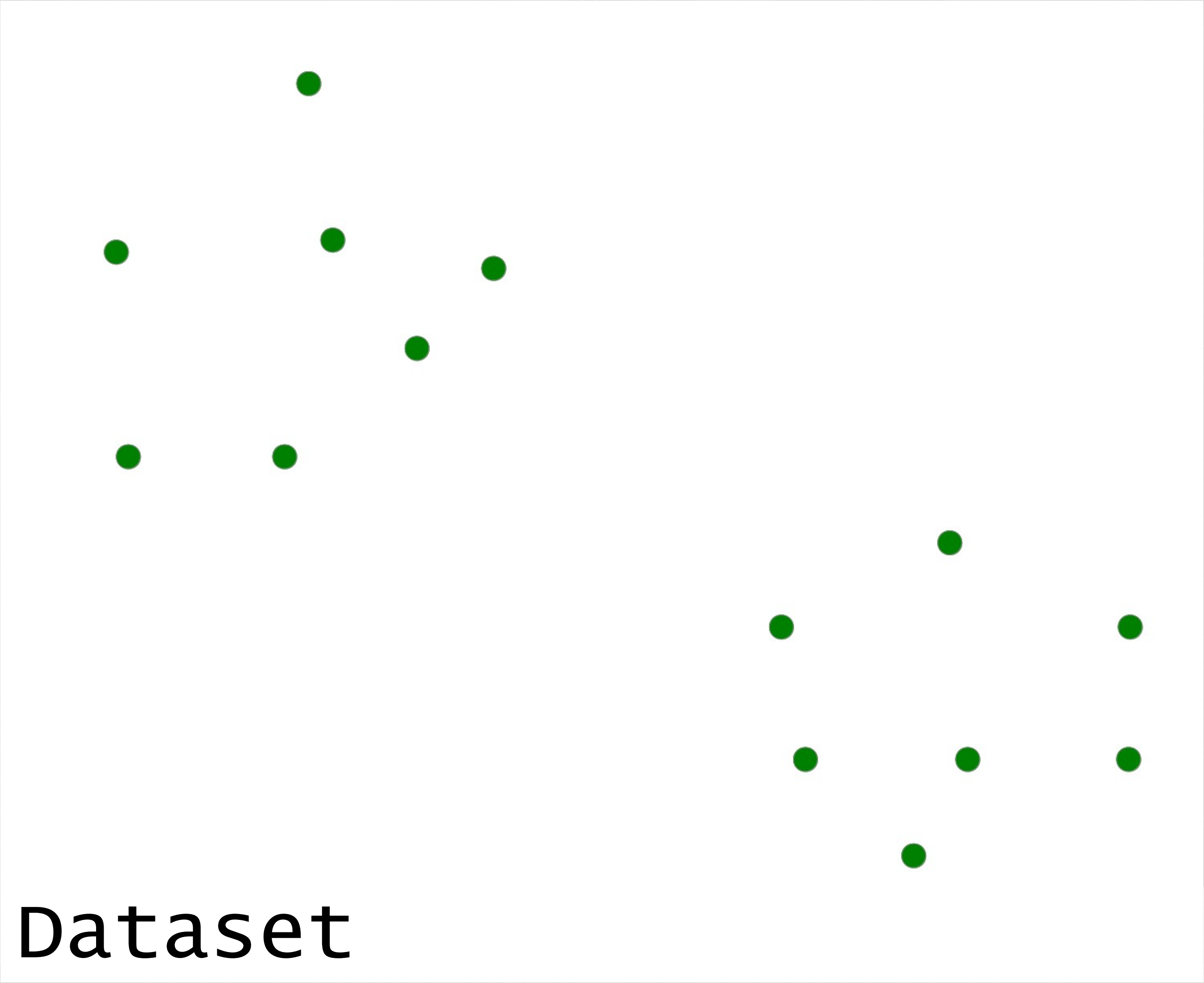
Source: scikit-learn

G-means

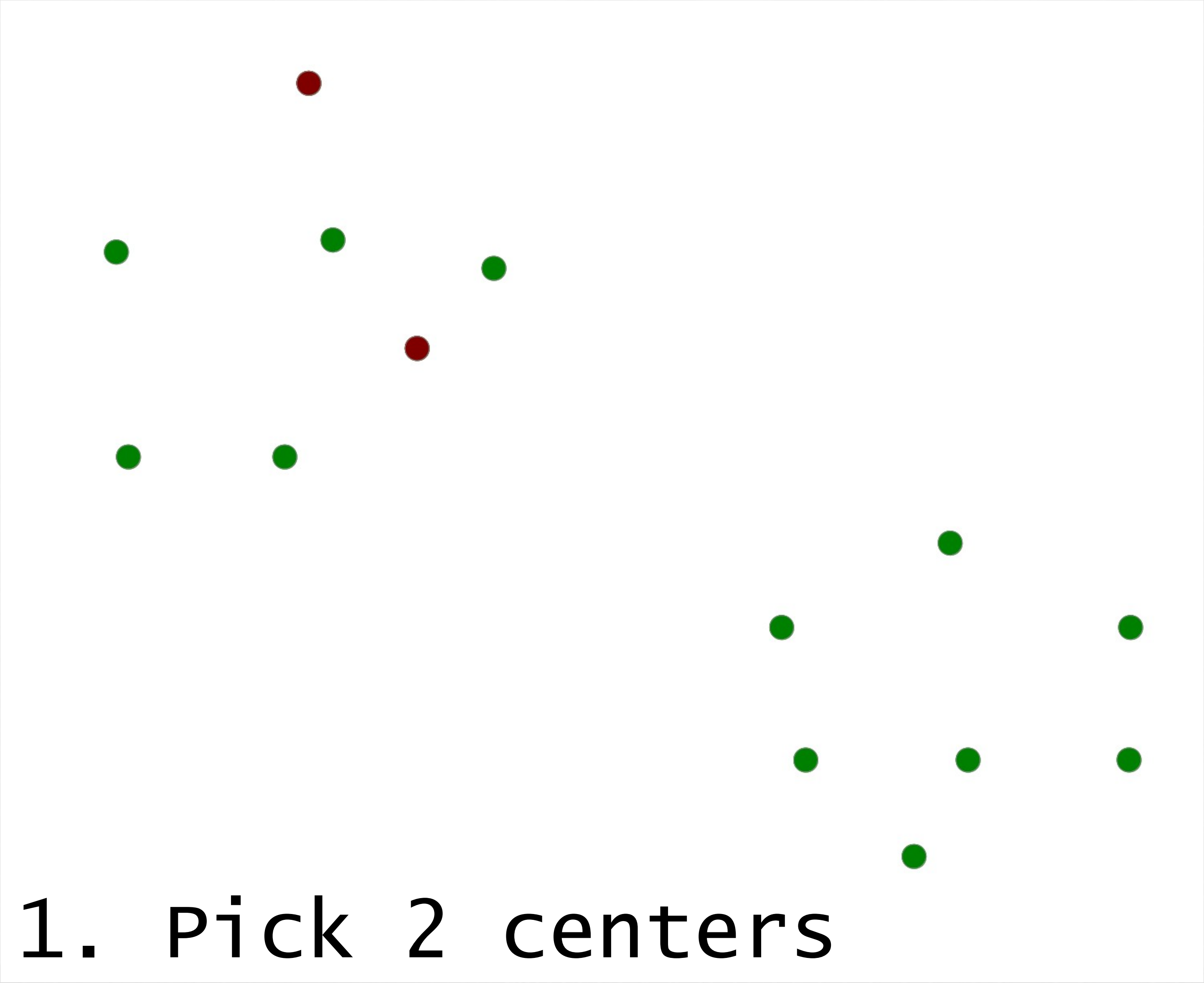
- G-means
  [Greg Hamerly and Charles Elkan. Learning the k in k-means. In Neural Information Processing Systems. MIT Press, 2003]
- K-means : points in each cluster are spherically distributed around the center

→ normality test & recursion

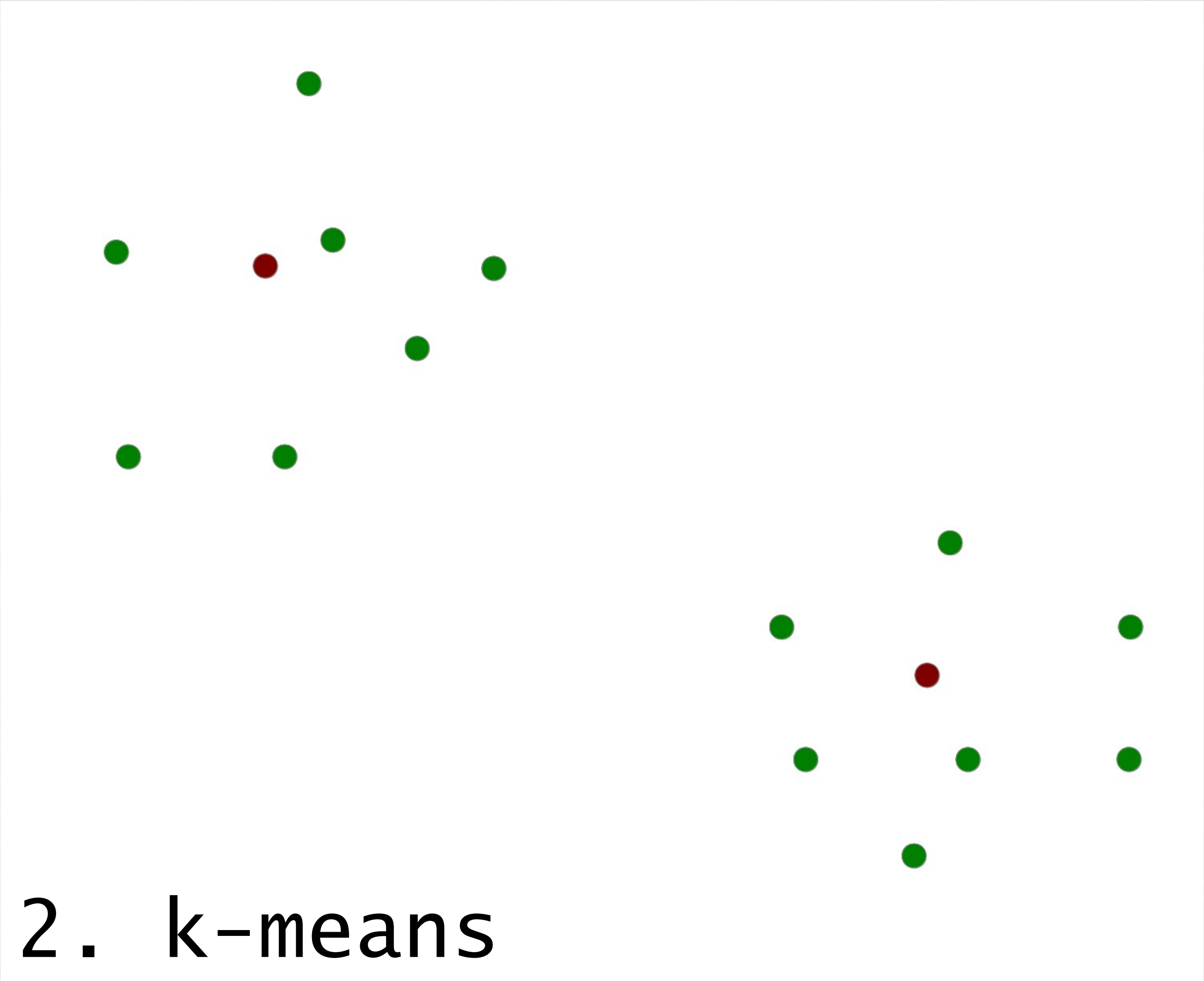
G-means



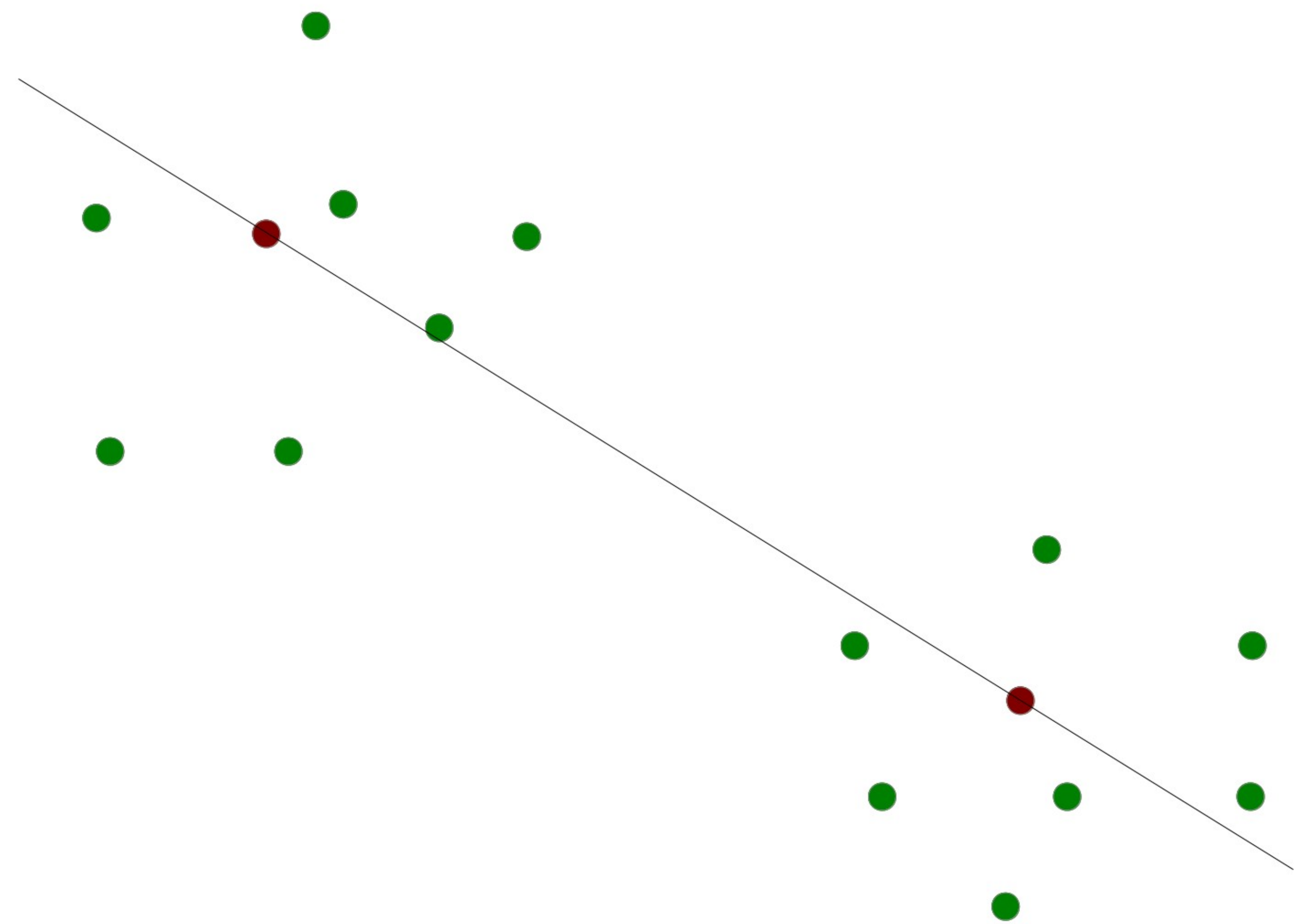
G-means



G-means

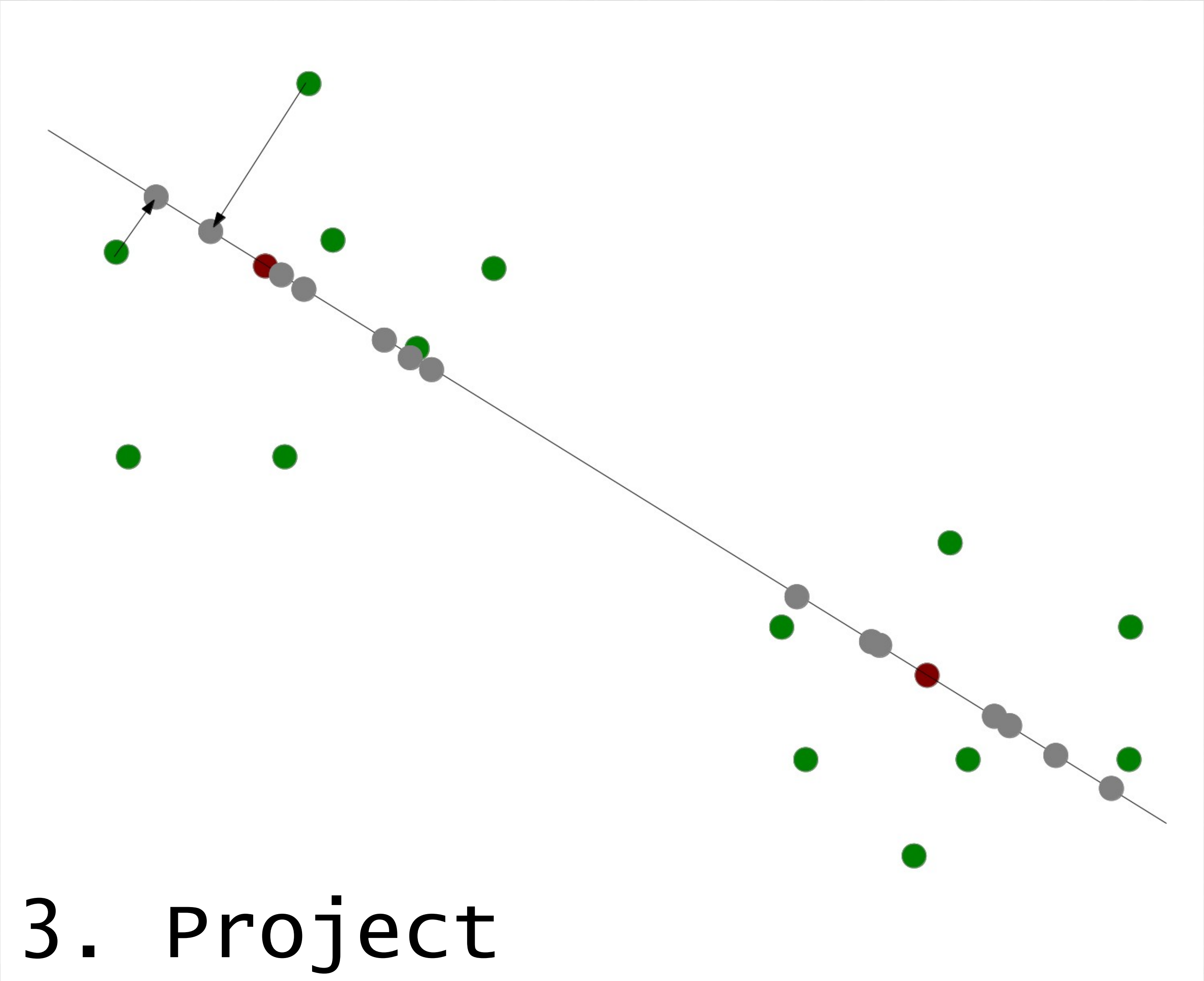


G-means

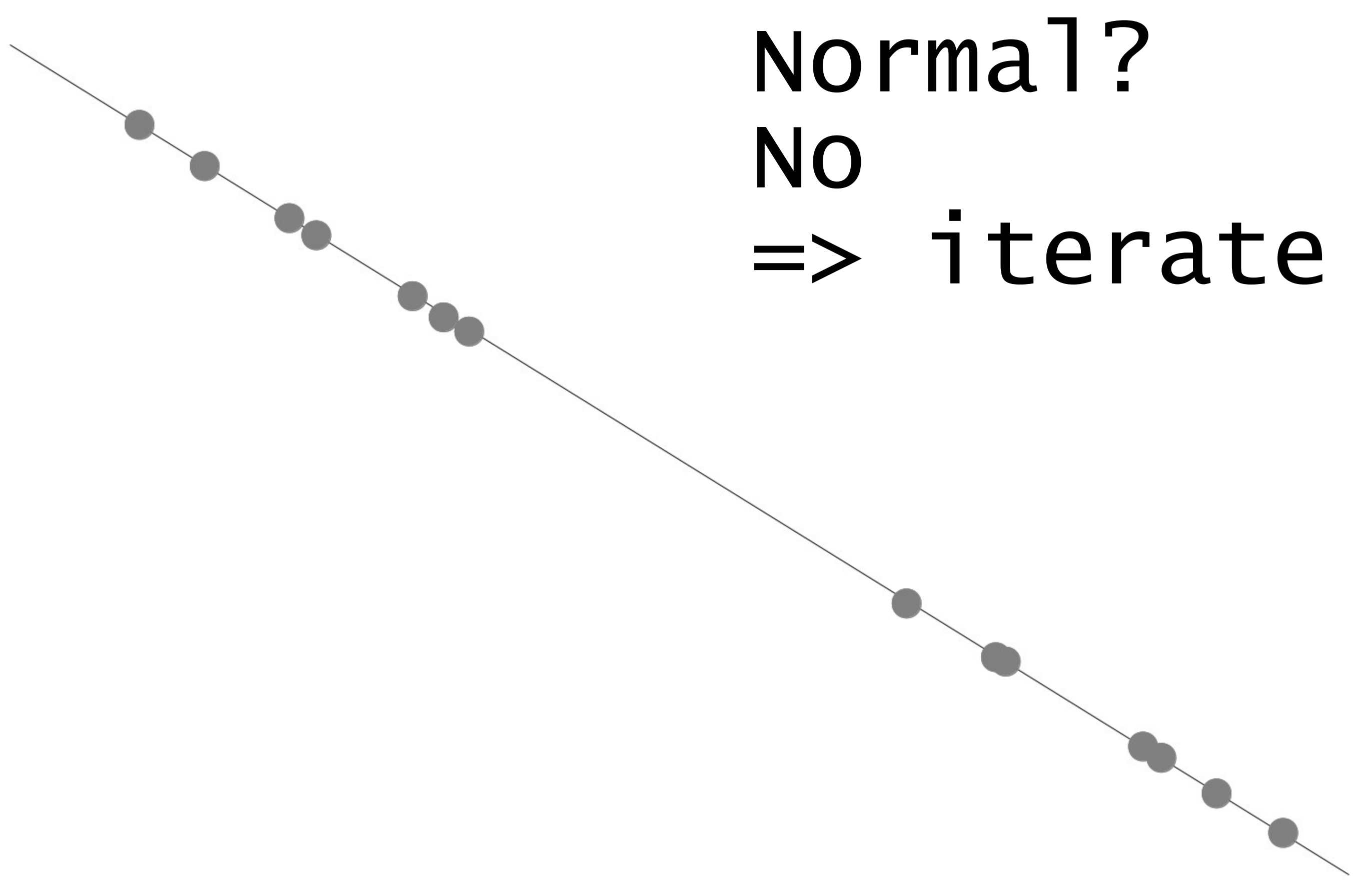


3. Project

G-means

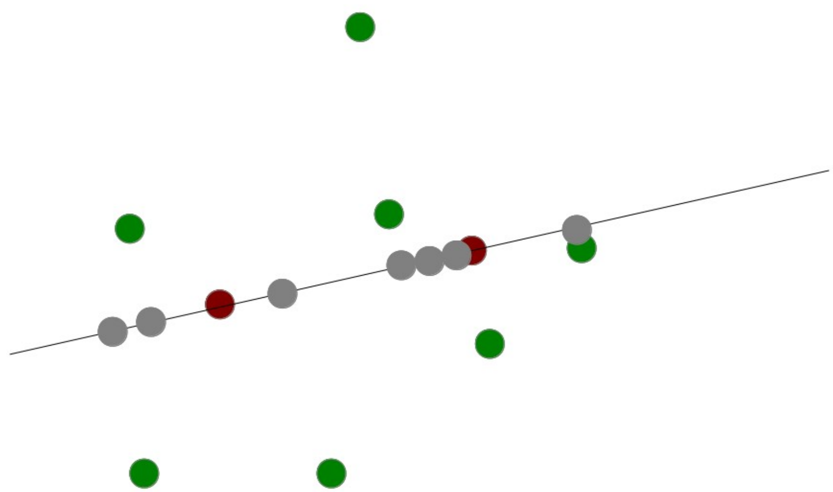


G-means



4. Normality test

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

MapReduce G-means

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
```

MapReduce G-means

3. Maximize parallelism

4. Limit memory usage (risk of crash)

```
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
```

Bottleneck

MapReduce G-means

Test**Few**Clusters

```
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
```

In memory combiner

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
```

MapReduce G-means

TestFewClusters

```
Ma
   F
   Fi
   Pro
   Ad
end

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

#clusters > #reducers

&

Estimated required memory < Java heap

TestClusters

```
Reduce(cluster, proj  ons)
   Build a vector
   ADtest(vector)
   if normal then
      Mark cluster
   end if
end procedure
```

MapReduce G-means

TestFewClusters

```
Map
    F
    Fi
    Pro
    Ad
end

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

TestClusters

```
educe(cluster, proj    ons)
    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

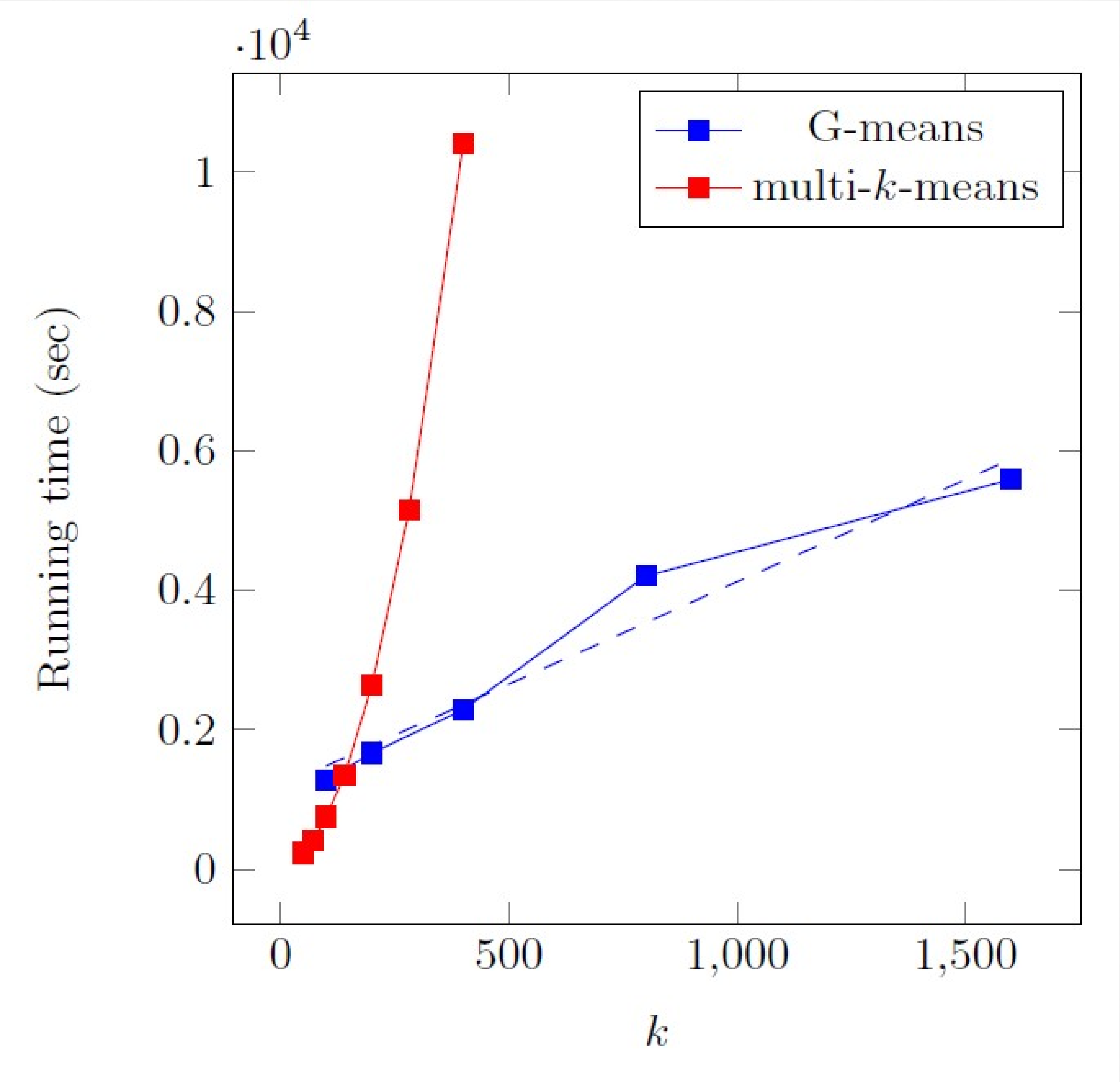
	MR multi-k-means	MR G-means
Speed		
Quality		

all possible values of k in a single job

Comparison

	MR multi-k-means	**MR G-means**
Speed	$O(nk^2)$ computations	$O(nk)$ computations
		But: • more iterations • more dataset reads • $\log_2(k)$
Quality		New centers added if and where needed
		But: tends to overestimate k!

Experimental results : Speed



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

Experimental results : Quality

k	**100**	**200**	**400**
k_{found}	150	279	639
Within Cluster Sum of Square			
(less is better)			
MR G-means	3.34	3.33	3.23
multi-k-means	3.71	3.6	3.39
(with same k)			

x ~1.5

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

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!