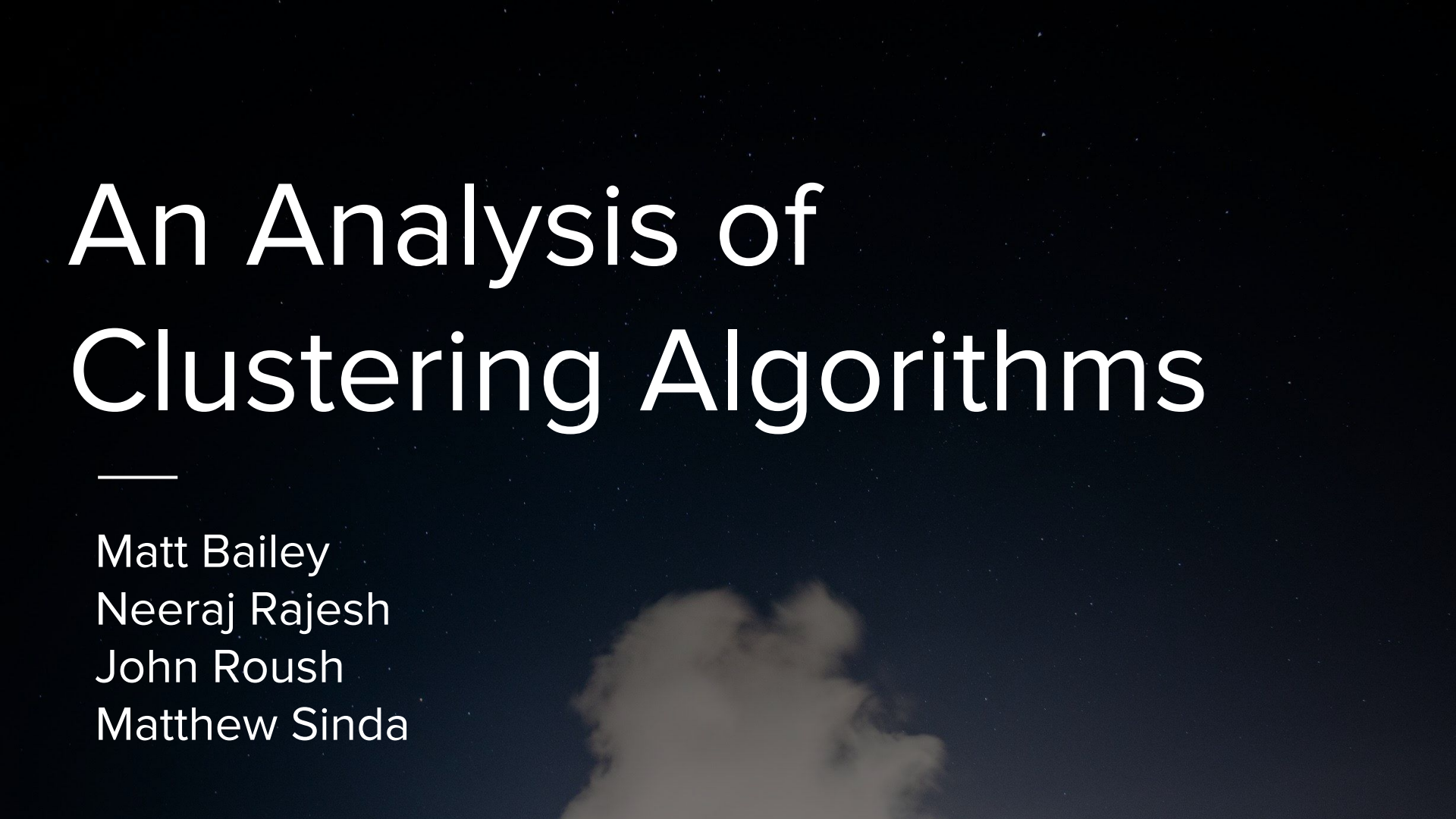
An Analysis of Clustering Algorithms

Matt Bailey
Neeraj Rajesh
John Roush
Matthew Sinda



Agenda

- Introduction
- Objective
- Clustering Algorithms
  - Overview
  - Algorithm Implementation
- Testing Framework
- Results
- Conclusion

Introduction

- Data has grown at an explosive rate
  - 2007 saw 295 exabytes of digital data
  - 2013 saw 1,200 exabytes of digital data
    - Average of 35 petabytes per day
- Presents a challenge not only in efficient storage, but perhaps more importantly, how to efficiently process the data

Introduction

Data Mining – helps solve the problem of analyzing the wealth of data

Data mining results from a natural evolution of data storage and analysis, and it comprises of combination of statistical and technological methods for learning about data and then making predictions based on what was learned[1].

1. Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Elsevier, Amsterdam, third edition, 2012.

Objective

- Can we arbitrarily pick a clustering algorithm and use it on a data set?
- Our project looks at four clustering algorithms to see how they compare to one another
  - Which algorithms work with which types of data set
  - When to use each algorithm
- Characteristics that will be compared are:
  - Overall Performance
    - CPU usage/run time
    - Memory usage
  - Accuracy

Clustering Algorithm Background

- Clustering involves grouping data objects based on some similarity metric
  - Numeric attributes and numeric similarity measure
  - For consistency, Euclidean distance is used as the similarity measure

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

  - Other measures could be used

Clustering Algorithm Background

- 3 types of clustering algorithms
  - Distance-Based
    - Multiple centers - closest wins
  - Density-Based
    - Similar objects form densely-packed “neighborhoods”
  - Probabilistic-Based
    - Every object belongs to every cluster with some probability - highest value wins

K-Means Algorithm

- Centroid-based algorithm:
  - Assigns data to the nearest cluster
  - Calculates new centers from the mean of each cluster
  - Iterates until centers stop changing
- Trade-offs:
  - Very simple to implement
  - Good performance characteristics: 0(*n*) runtime and memory usage
  - Performance is sensitive to initial choice of cluster centers
    - Choosing the number and location of initial centers is difficult
  - Sensitive to outliers
  - Works best for compact, convex clusters

K-Medoids Algorithm

- Also centroid-based
  - Distance-based
  - Center is an object that is most representative for each cluster
- Algorithm Implementation
  - K-Medoids implemented from the PAM (Partitioning Around Medoids) algorithm
    - Tries to find one object per cluster that most accurately represents all of the data objects in a cluster
    - Greedy Method - chooses the object with the lowest cost as determined by the similarity measure
    - Efficiency is determined by desired accuracy
      - Algorithm must check each object against all other objects
      - Number of objects to check is defined by the programmer
    - Actual implementation runs in $O(n^2)$

DBSCAN Algorithm

- Density-based algorithm
  - Clusters together all data objects which are *density-connected*
  - Density is defined over a fixed-radius "***ε**-neighborhood*"
- Trade-offs
  - Can determine number of clusters automatically
  - Handles non-convex or non-compact clusters
  - Calculates distance between pairs of objects rather than object-to-cluster
    - Potentially requires $O(n^2)$ runtime
    - Can be $O(n \log n)$, but very complex to implement
  - Density and neighborhood radius must be given a priori
    - Fails if clusters do not have consistent density

Fuzzy C-Means Algorithm

- Probabilistic Model-Based
  - C-Means or Fuzzy Cluster
    - Assign data points to a cluster based on some degree of membership
    - Points can belong to a cluster to some degree but also to other clusters to a lesser degree
- Algorithm Implementation
  - C-Means or Fuzzy Cluster implemented from the EM (Expectation-Maximization) algorithm
    - Starts with randomly-selected objects as potential centers and then iteratively works through new centers in two steps
      - Expectation - Calculates distance to each of the potential centers, and their degree of membership
      - Maximization - Recalculates the new centers based off optimal centers
    - Runs until there is no more changes
      - Potential for algorithm never to converge due to floating point inaccuracies inherent in all CPUs
      - Actual implementation stopped when the change between values was sufficiently small enough
        - Determined to be 0.000001
    - Run time is O(n)

Implementation

- All clustering algorithms implemented in Java
  - Easy to use, familiar
  - Performance is comparable to C/C++ (if used carefully!)
  - Challenging to get accurate performance measurements
- Pseudo-random data generator, also in Java
  - Generates clusters in various shapes and configurations
  - Random background noise or other complications
  - Provides “ground-truth” information for evaluating correctness

Evaluating Performance

- Run-time performance:
  - Measured with the JVM internal clock (microsecond resolution)
  - Averaging over many iterations to smooth out GC pauses, etc.
- Memory usage:
  - Difficult by design to monitor true memory usage of the JVM
  - Cheated by using Java Instrumentation Agent and reflection
    - Measurements are approximate
    - Requires unusual coding style

Evaluating Cluster Quality

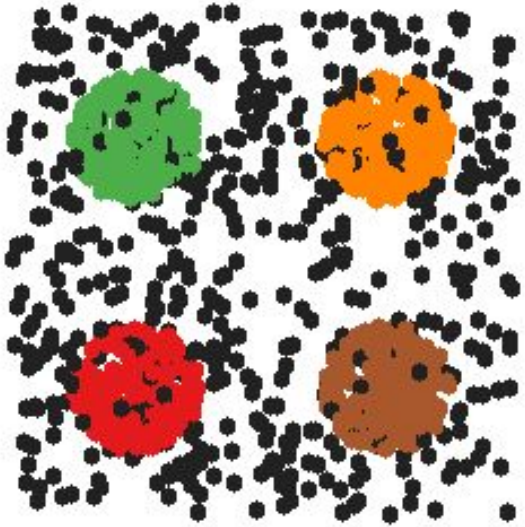
- Intrinsic Methods - compare to ground truth
  - **BCubed Precision**: fraction of objects in a cluster that truly belong together
  - **BCubed Recall**: fraction of objects belonging together that end up clustered together
- Extrinsic Methods - absolute quality metrics
  - **Compactness**: average distance between objects in the same cluster
  - **Separation**: average distance between objects and the nearest foreign cluster
  - **Silhouette coefficient**: the difference between Separation and Compactness

Results

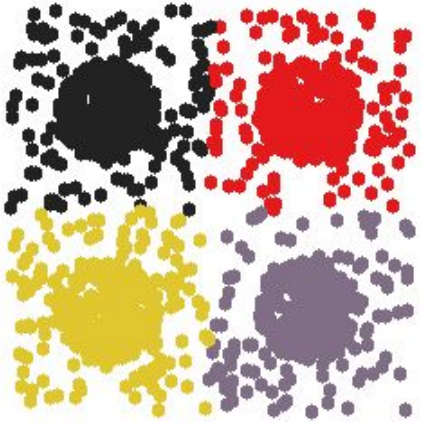
- Given the data sets that we ran the algorithms against, it appears that:
  - DBSCAN was the most accurate; K-Medoids being the least accurate
  - Based on Clusters
    - K-Means used the least memory
    - DBSCAN ran the quickest
  - Based on Number of Attributes
    - C-Means had a spike but overall, the lowest run time
    - DBSCAN used the least memory
  - Based on Number of Tuples
    - K-Means ran the quickest
    - K-Means used the least Memory

Results

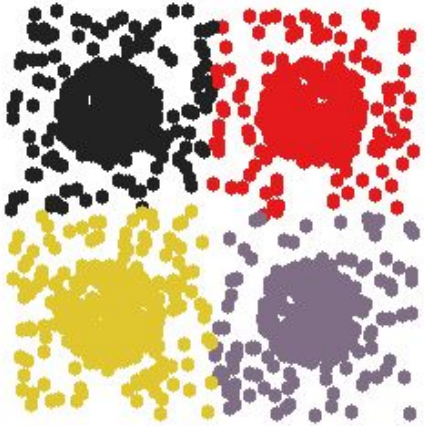
Clusters - GeneratedData



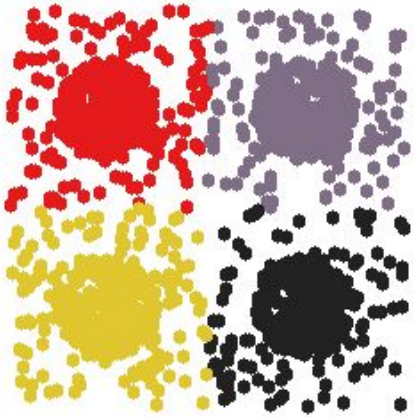
Clusters - KMeans



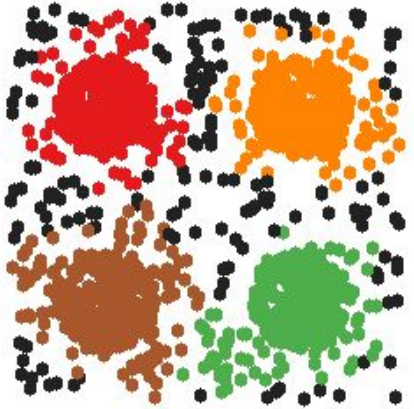
Clusters - kMedoids



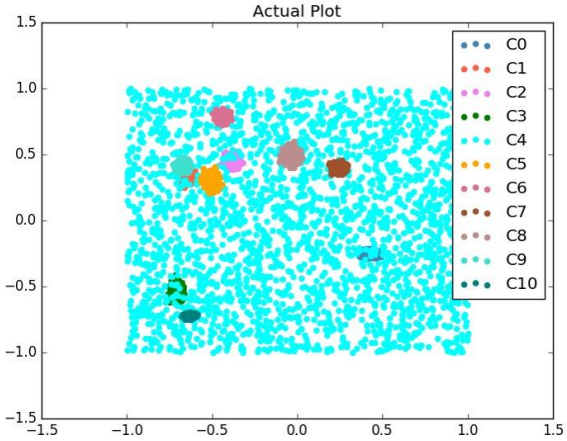
Clusters - CMeans

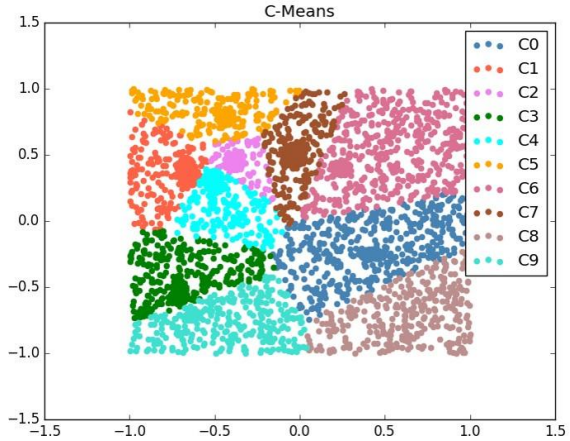


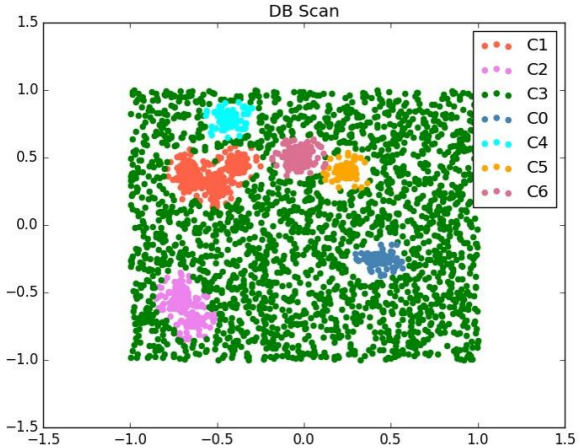
Clusters - DBSCAN

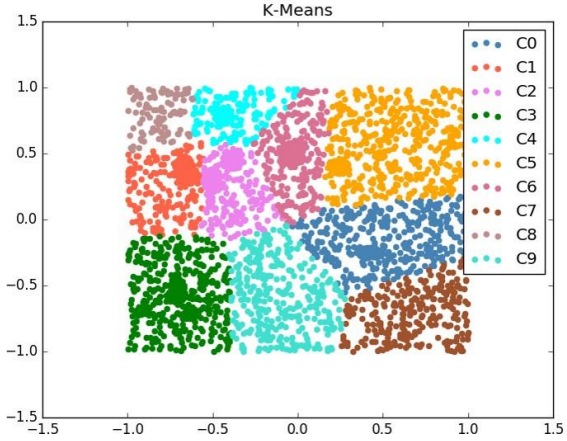


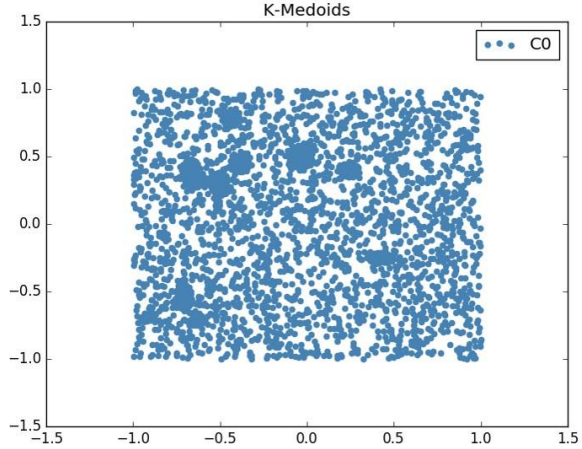
Results


Actual Plot
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10

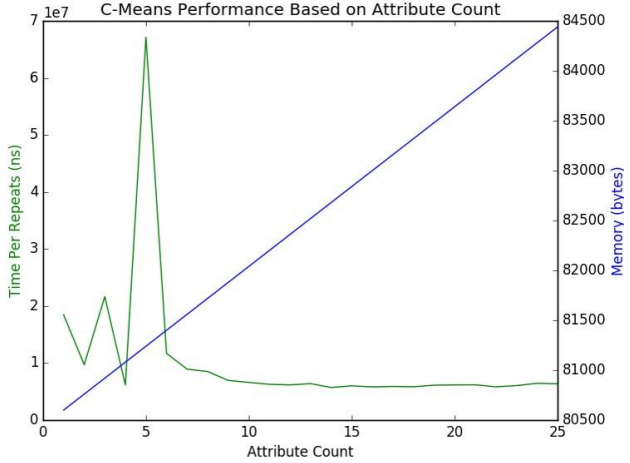

C-Means
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9

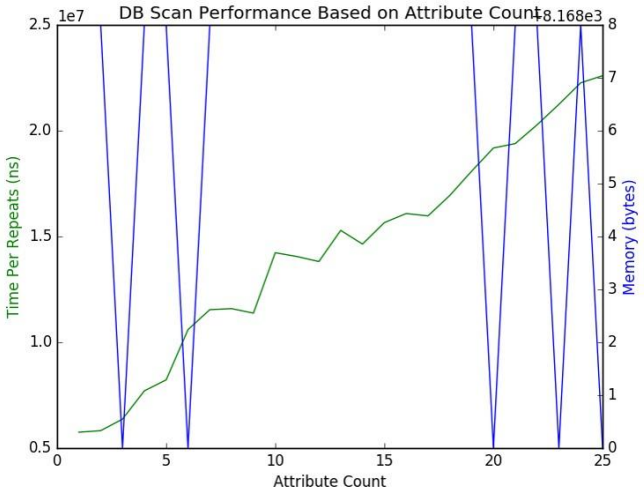

DB Scan
C1
C2
C3
C0
C4
C5
C6

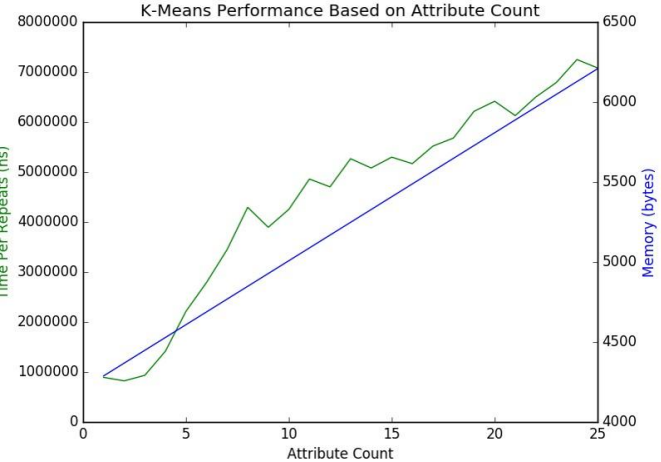

K-Means
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9

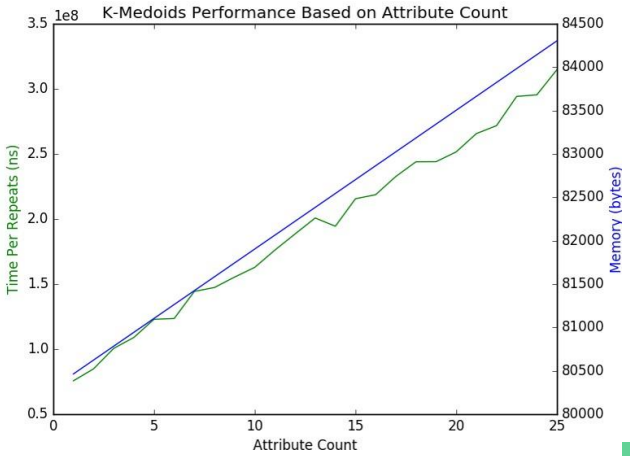

K-Medoids
C0

Results

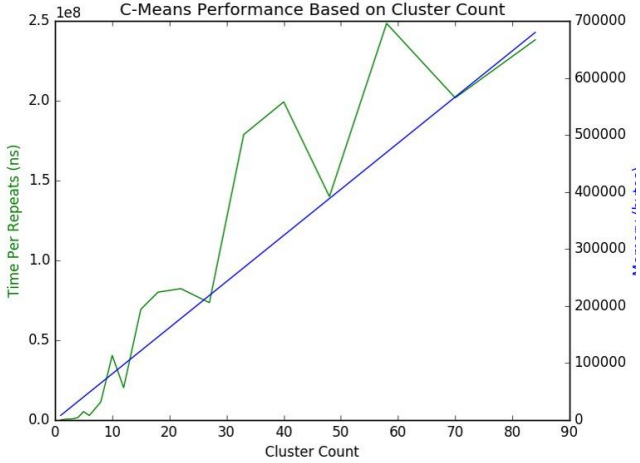

C-Means Performance Based on Attribute Count
1e7
Time Per Repeats (ns)
Memory (bytes)
Attribute Count


DB Scan Performance Based on Attribute Count
1e7
1e3
Time Per Repeats (ns)
Memory (bytes)
Attribute Count

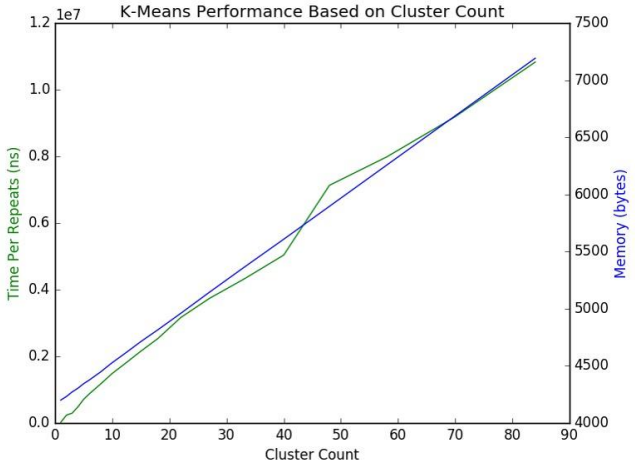

K-Means Performance Based on Attribute Count
Time Per Repeats (ns)
Memory (bytes)
Attribute Count

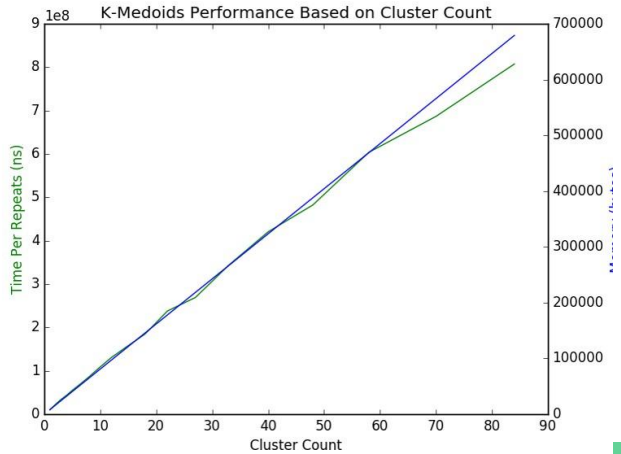

K-Medoids Performance Based on Attribute Count
1e8
Time Per Repeats (ns)
Memory (bytes)
Attribute Count

Results

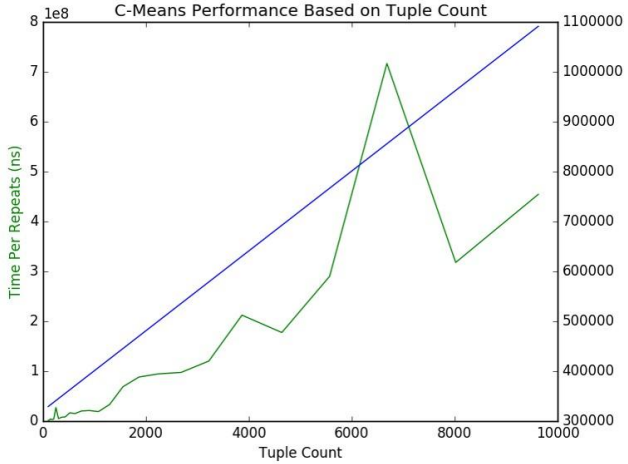

C-Means Performance Based on Cluster Count
1e8
Time Per Repeats (ns)
Cluster Count

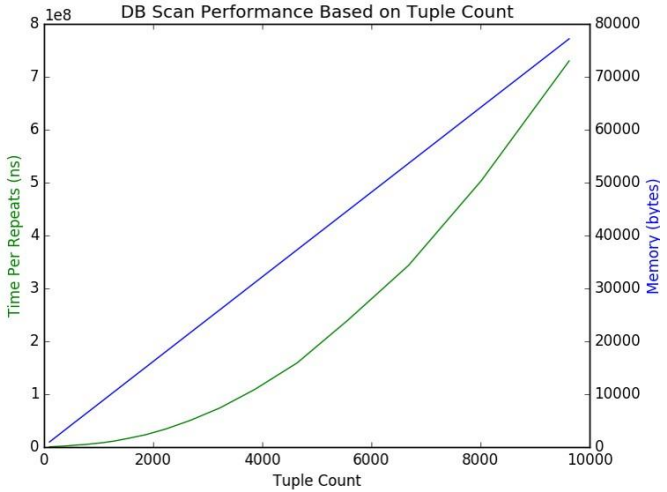

DB Scan Performance Based on Cluster Count
1e7
+8.168e3
Time Per Repeats (ns)
Memory (bytes)
Cluster Count

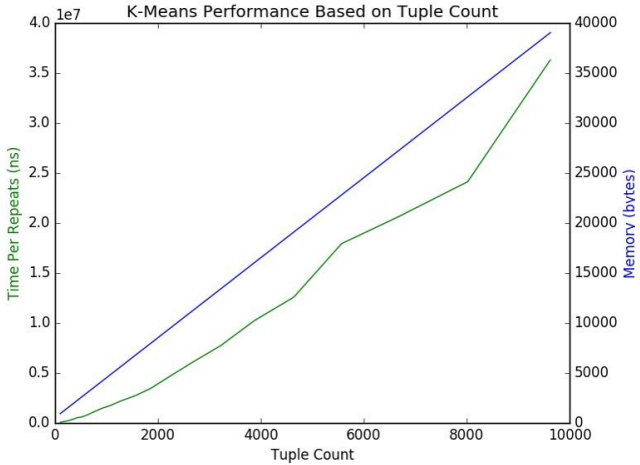

K-Means Performance Based on Cluster Count
1e7
Time Per Repeats (ns)
Memory (bytes)
Cluster Count

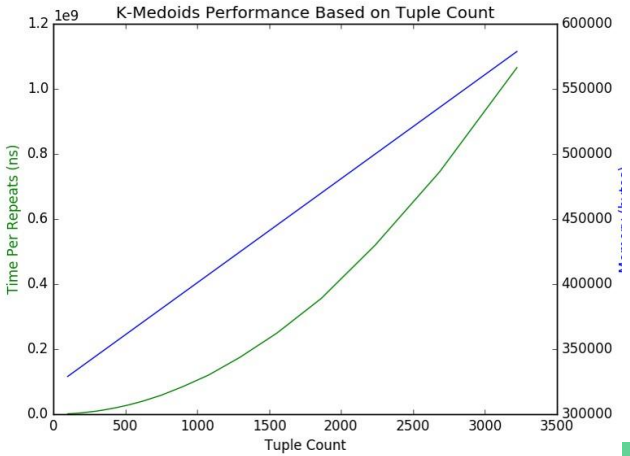

K-Medoids Performance Based on Cluster Count
1e8
Time Per Repeats (ns)
Cluster Count

Results


C-Means Performance Based on Tuple Count
1e8
Time Per Repeats (ns)
Tuple Count
0
1
2
3
4
5
6
7
8
0
2000
4000
6000
8000
10000
300000
400000
500000
600000
700000
800000
900000
1000000
1100000


DB Scan Performance Based on Tuple Count
1e8
Time Per Repeats (ns)
Memory (bytes)
Tuple Count
0
1
2
3
4
5
6
7
8
0
2000
4000
6000
8000
10000
0
10000
20000
30000
40000
50000
60000
70000
80000


K-Means Performance Based on Tuple Count
1e7
Time Per Repeats (ns)
Memory (bytes)
Tuple Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
2000
4000
6000
8000
10000
0
5000
10000
15000
20000
25000
30000
35000
40000


K-Medoids Performance Based on Tuple Count
1e9
Time Per Repeats (ns)
Tuple Count
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
500
1000
1500
2000
2500
3000
3500
300000
350000
400000
450000
500000
550000
600000

Conclusion

- Due to differences in algorithm properties, it’s difficult to say which is best
  - “Best” based off
    - Different data sets
    - Number of attributes
  - Easy to create a data set that an algorithm will NOT be good at

References

•Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Elsevier, Amsterdam, third edition, 2012.

•Martin Hilbert and Priscila Lopez. The World's Technological Capacity to Store, Communicate, and Compute Information. Science, 332(6025):60-65, April 2011.

•H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. Com mun. ACM, 57(7):86-94, July 2014.

•Juha Lehikoinen and Ville Koistinen. In big data we trust? Interactions, 21(5):38-41, September 2014.

Questions?