Performance Comparisons of the K-Means Algorithm

Mark A. Ward

Center for Data Science, NYU

May 19, 2015

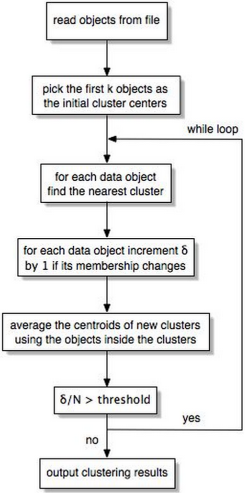
Outline

Outline

The K-Means Algorithm

- Algorithm that aims to partition n observations into k clusters where each observation belongs to the cluster nearest mean or centroid
- Vector quantization in signal processing
- Cluster analysis in data mining, how do you choose k?
- Feature learning in semi-supervised or unsupervised learning
- Clustering is NP-hard, but k-means is a heuristic that converges quickly to a local optimum
- Typical running time $O(nkdi)$, repeat for T trials
- For n points in $[0,1]^d$ with independent gaussian perturbations with 0 mean and σ^2 variance, upper bound expected running time by $O(n^{34}k^{34}d^8\log^4(n)/\sigma^6)$. [Arthur 2009]

The K-Means Algorithm

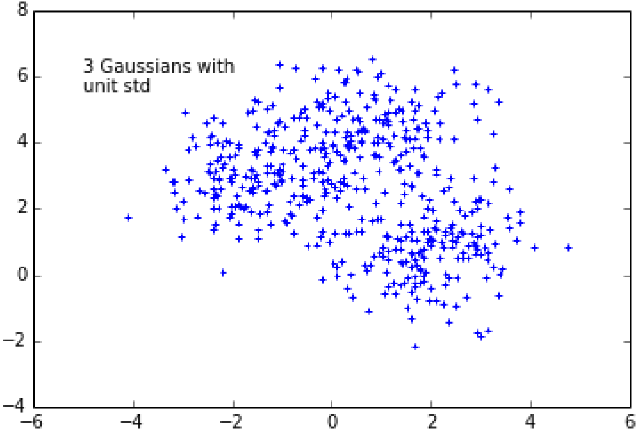


N: *number of data objects*
K: *number of clusters*

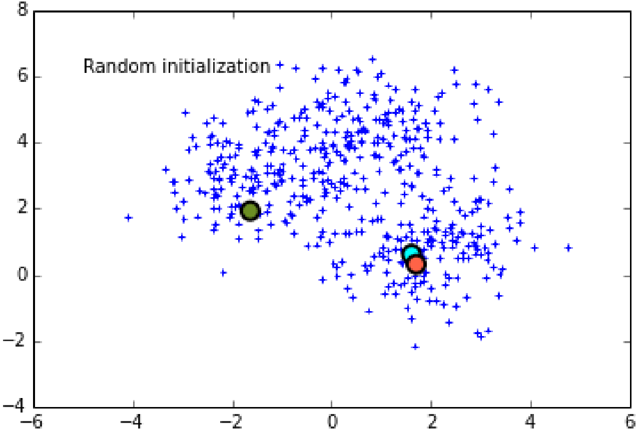
objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
while δ/N > threshold
    δ ← 0
    for i ← 0 to N-1
        for j ← 0 to K-1
            distance ← | objects[i] - clusters[j] |
            if distance < dmin
                dmin ← distance
                n ← j
        if membership[i] ≠ n
            δ ← δ + 1
            membership[i] ← n
        new_clusters[n] ← new_clusters[n] + objects[i]
        new_cluster_size[n] ← new_cluster_size[n] + 1
    for j ← 0 to K-1
        clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
        new_clusters[j][*] ← 0
        new_cluster_size[j] ← 0
```

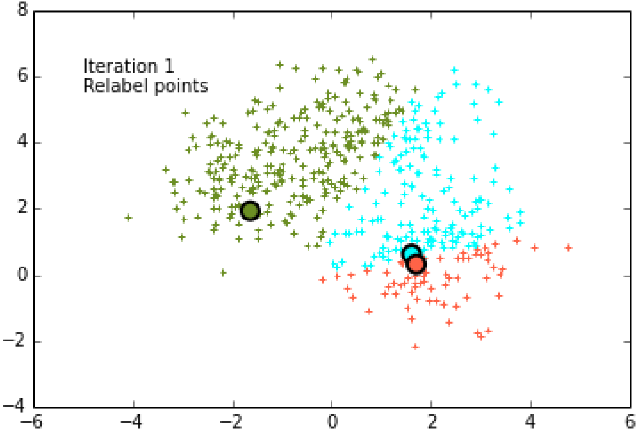
Examples



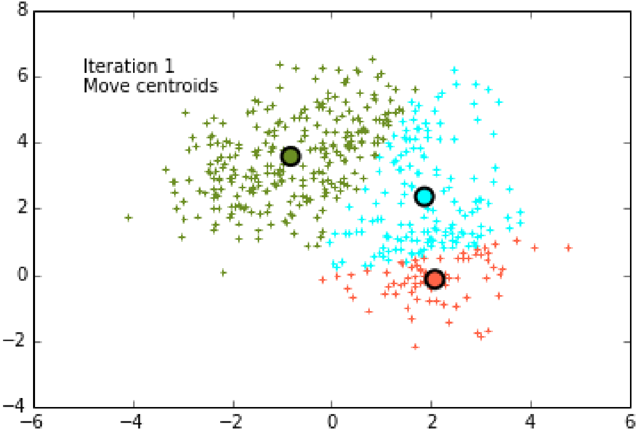
Examples



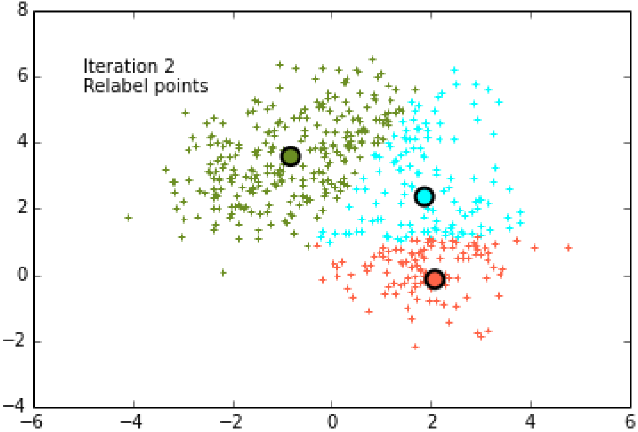
Examples



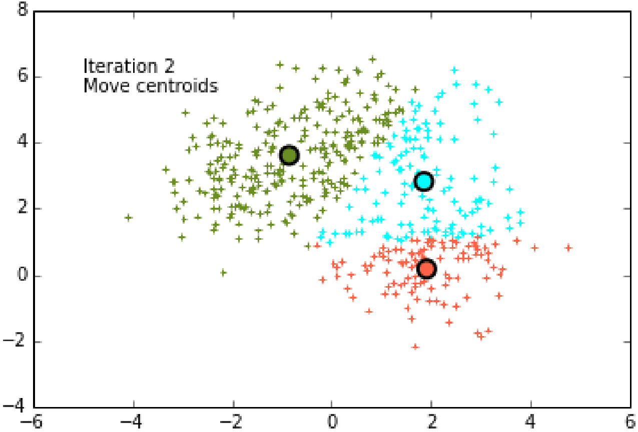
Examples



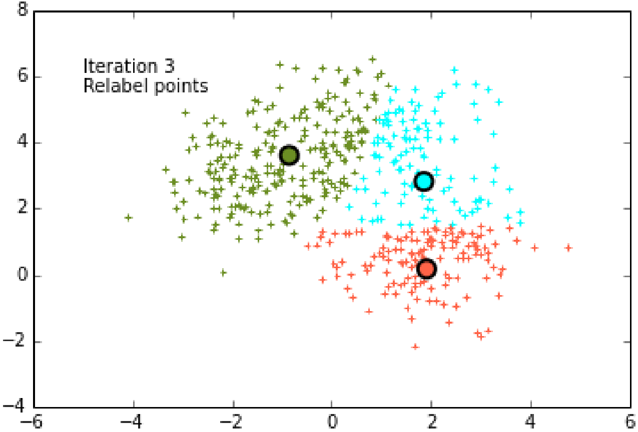
Examples



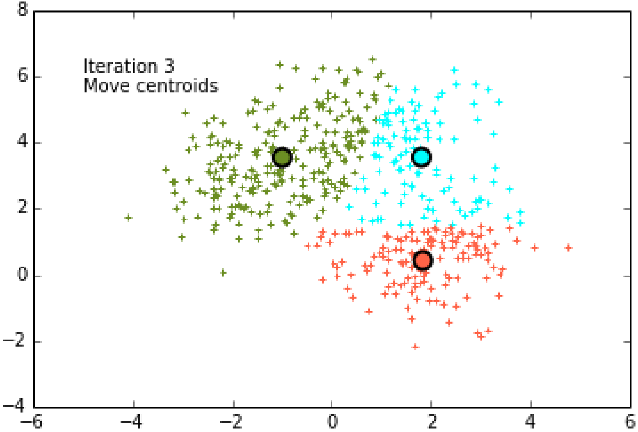
Examples



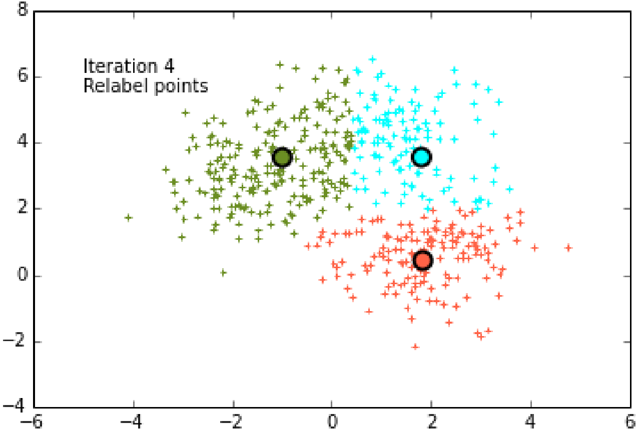
Examples



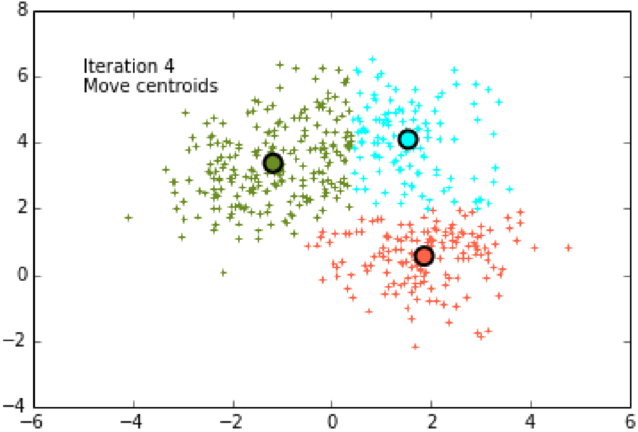
Examples



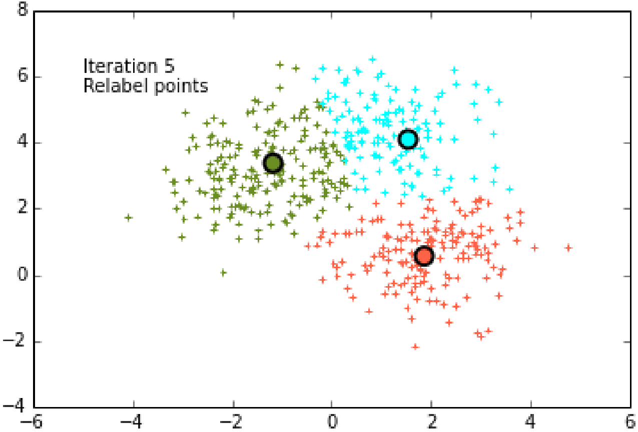
Examples



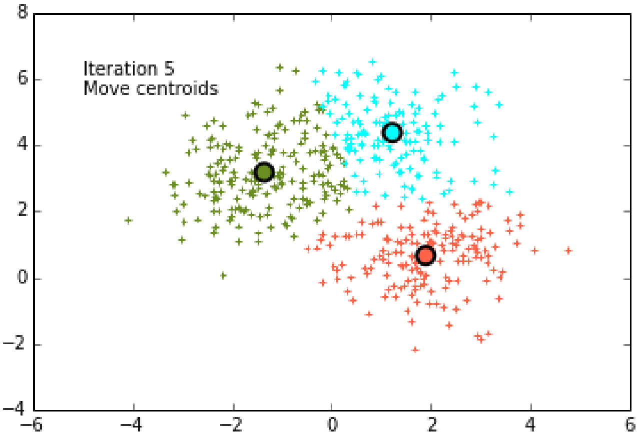
Examples



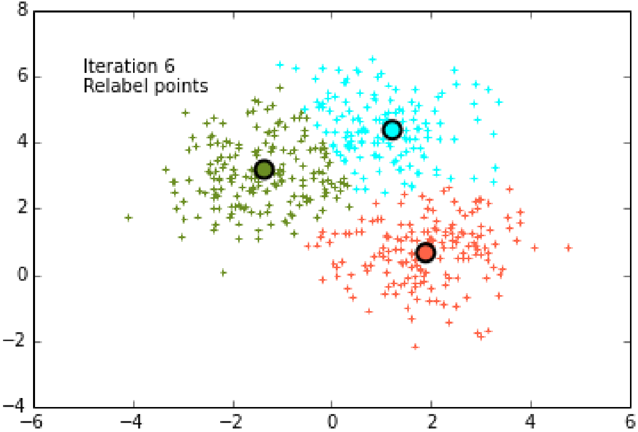
Examples



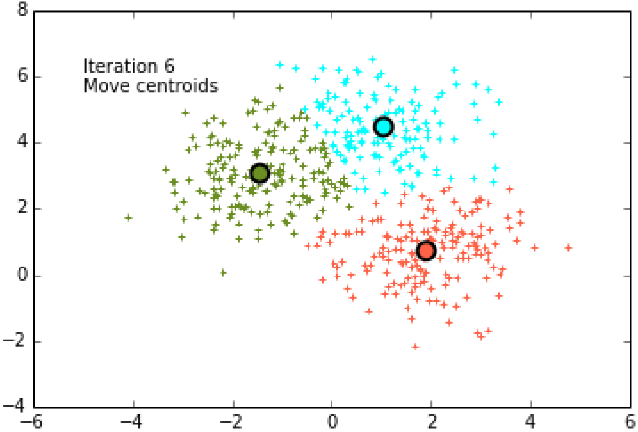
Examples



Examples



Examples



Examples

229,931 colors

Examples

256 colors

Examples

128 colors

Examples

64 colors

Examples

32 colors

Examples

16 colors

Examples

10 colors

Examples

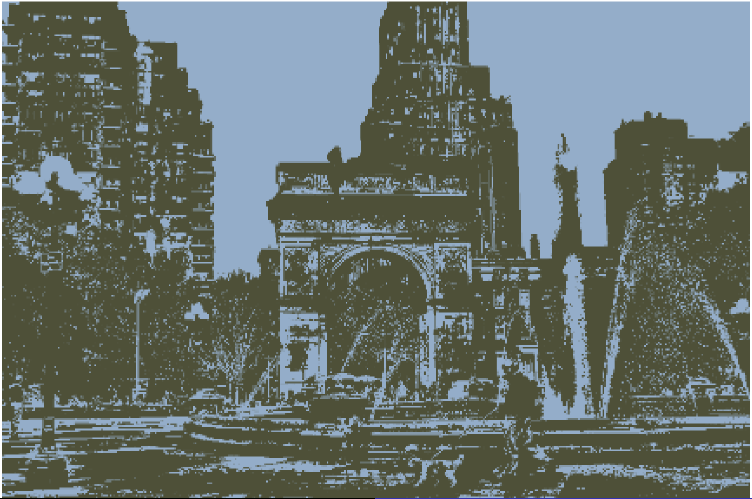
5 colors

Examples

3 colors

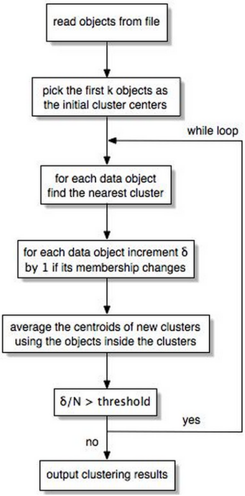
Examples

2 colors



Outline

MPI Communication Overview

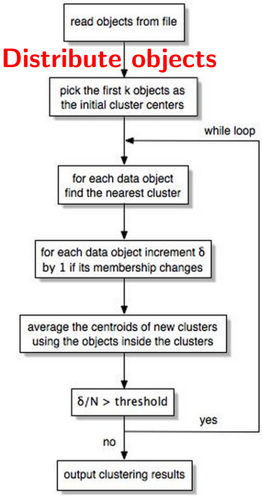


N: *number of data objects*
K: *number of clusters*

objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
while δ/N > threshold
    δ ← 0
    for i ← 0 to N-1
        for j ← 0 to K-1
            distance ← | objects[i] - clusters[j] |
            if distance < dmin
                dmin ← distance
                n ← j
        if membership[i] ≠ n
            δ ← δ + 1
            membership[i] ← n
        new_clusters[n] ← new_clusters[n] + objects[i]
        new_cluster_size[n] ← new_cluster_size[n] + 1
    for j ← 0 to K-1
        clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
        new_clusters[j][*] ← 0
        new_cluster_size[j] ← 0
```

MPI Communication Overview

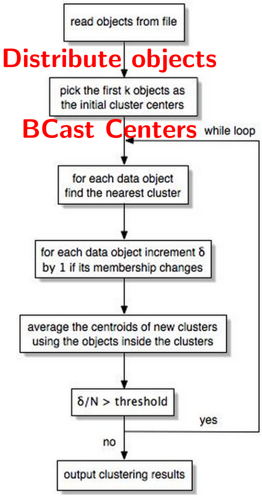


N: *number of data objects*
K: *number of clusters*

objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
while δ/N > threshold
    δ ← 0
    for i ← 0 to N-1
        for j ← 0 to K-1
            distance ← | objects[i] - clusters[j] |
            if distance < dmin
                dmin ← distance
                n ← j
        if membership[i] ≠ n
            δ ← δ + 1
            membership[i] ← n
        new_clusters[n] ← new_clusters[n] + objects[i]
        new_cluster_size[n] ← new_cluster_size[n] + 1
    for j ← 0 to K-1
        clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
        new_clusters[j][*] ← 0
        new_cluster_size[j] ← 0
```

MPI Communication Overview

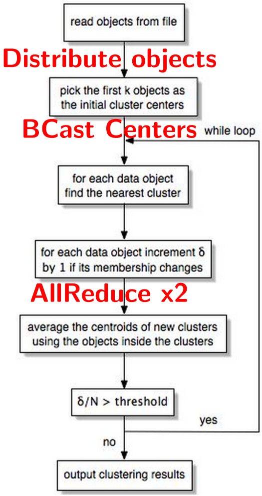


N: *number of data objects*
K: *number of clusters*

objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
while δ/N > threshold
  δ ← 0
  for i ← 0 to N-1
    for j ← 0 to K-1
      distance ← | objects[i] - clusters[j] |
      if distance < dmin
        dmin ← distance
        n ← j
    if membership[i] ≠ n
      δ ← δ + 1
      membership[i] ← n
    new_clusters[n] ← new_clusters[n] + objects[i]
    new_cluster_size[n] ← new_cluster_size[n] + 1
  for j ← 0 to K-1
    clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
    new_clusters[j][*] ← 0
    new_cluster_size[j] ← 0
```

MPI Communication Overview

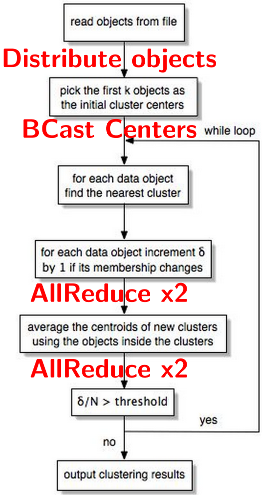


N: *number of data objects*
K: *number of clusters*

objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
while δ/N > threshold
    δ ← 0
    for i ← 0 to N-1
        for j ← 0 to K-1
            distance ← | objects[i] - clusters[j] |
            if distance < dmin
                dmin ← distance
                n ← j
        if membership[i] ≠ n
            δ ← δ + 1
            membership[i] ← n
        new_clusters[n] ← new_clusters[n] + objects[i]
        new_cluster_size[n] ← new_cluster_size[n] + 1
    for j ← 0 to K-1
        clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
        new_clusters[j][*] ← 0
        new_cluster_size[j] ← 0
```

MPI Communication Overview

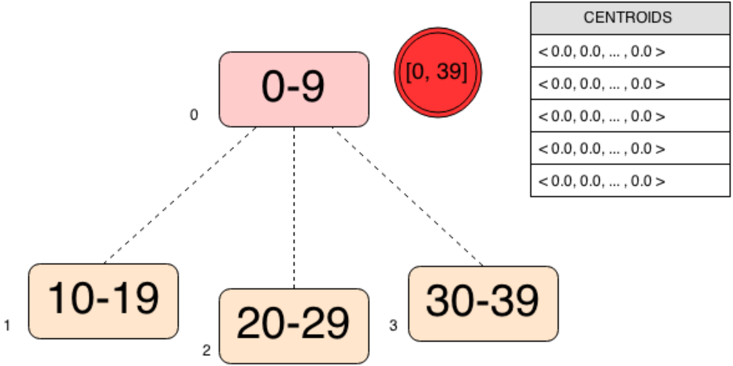


N: *number of data objects*
K: *number of clusters*

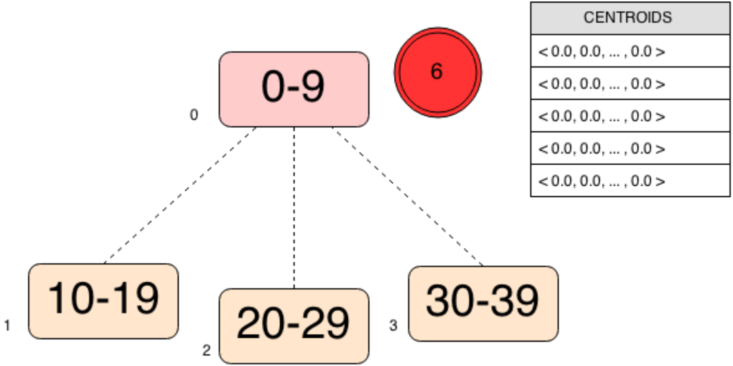
objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
while δ/N > threshold
    δ ← 0
    for i ← 0 to N-1
        for j ← 0 to K-1
            distance ← | objects[i] - clusters[j] |
            if distance < dmin
                dmin ← distance
                n ← j
        if membership[i] ≠ n
            δ ← δ + 1
            membership[i] ← n
        new_clusters[n] ← new_clusters[n] + objects[i]
        new_cluster_size[n] ← new_cluster_size[n] + 1
    for j ← 0 to K-1
        clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
        new_clusters[j][*] ← 0
        new_cluster_size[j] ← 0
```

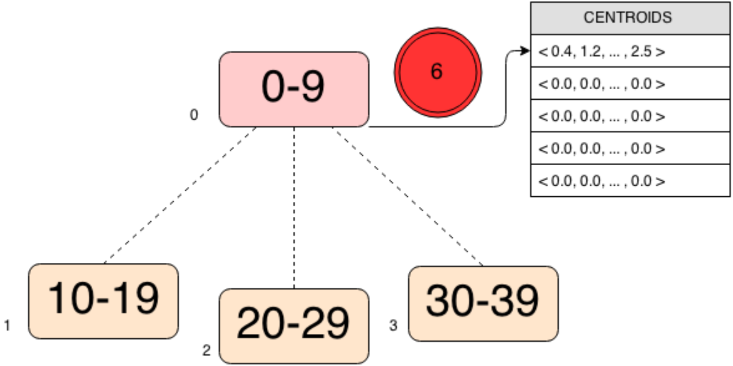
Initialization



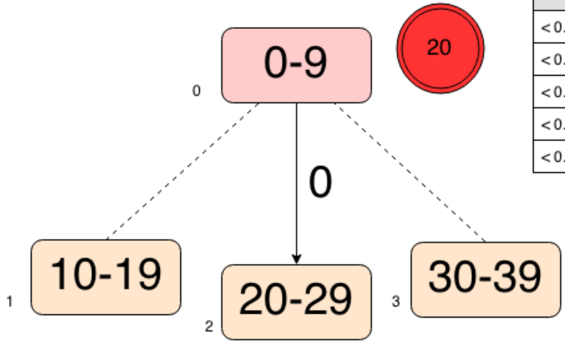
Initialization



Initialization

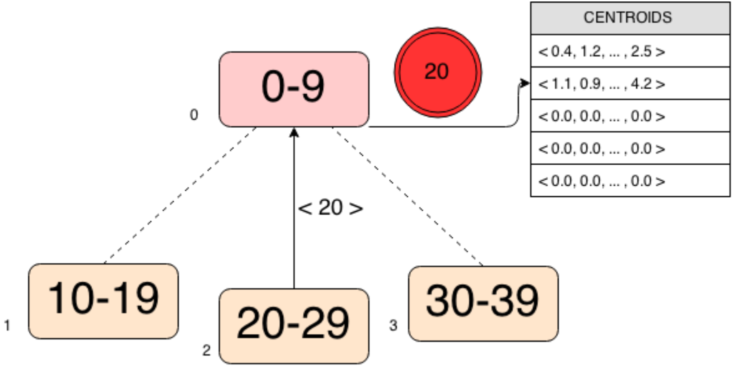


Initialization

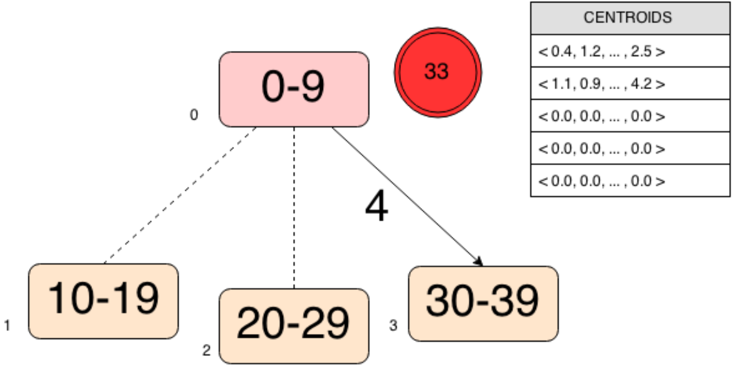


CENTROIDS
< 0.4, 1.2, ... , 2.5 >
< 0.0, 0.0, ... , 0.0 >
< 0.0, 0.0, ... , 0.0 >
< 0.0, 0.0, ... , 0.0 >
< 0.0, 0.0, ... , 0.0 >

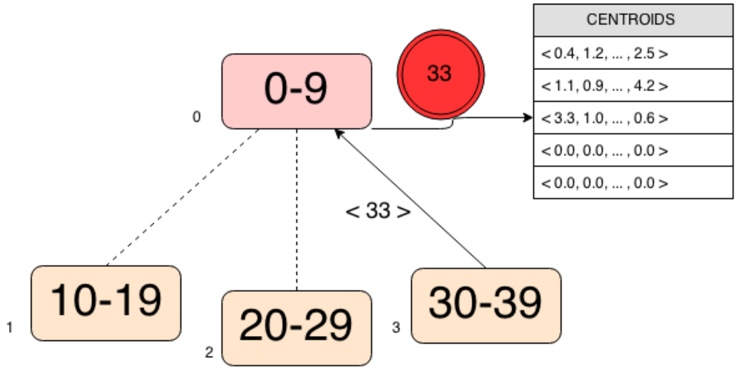
Initialization



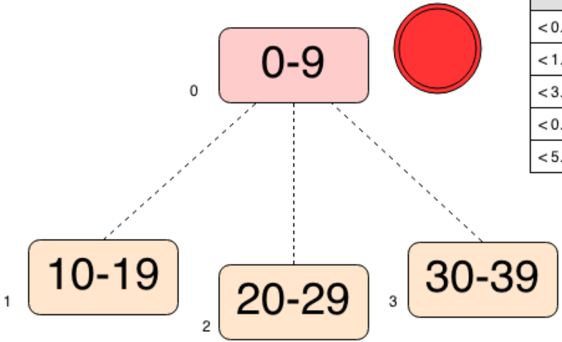
Initialization



Initialization

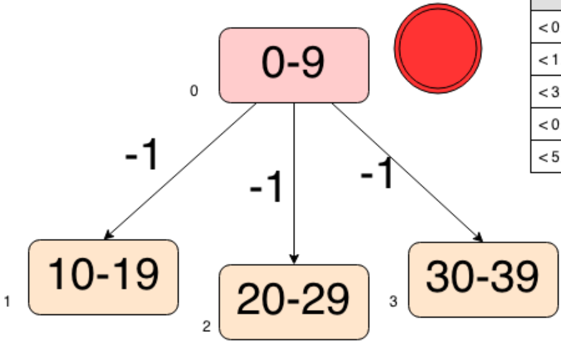


Initialization



CENTROIDS
< 0.4, 1.2, ... , 2.5 >
< 1.1, 0.9, ... , 4.2 >
< 3.3, 1.0, ... , 0.6 >
< 0.7, 2.8, ... , 0.4 >
< 5.6, 0.8, ... , 0.9 >

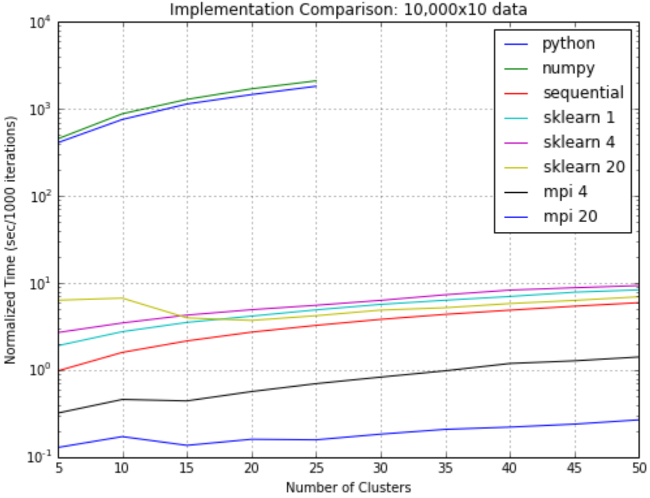
Initialization



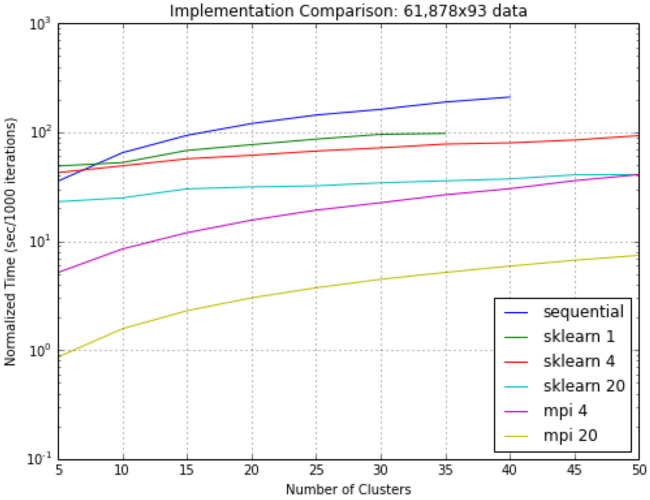
CENTROIDS
< 0.4, 1.2, ... , 2.5 >
< 1.1, 0.9, ... , 4.2 >
< 3.3, 1.0, ... , 0.6 >
< 0.7, 2.8, ... , 0.4 >
< 5.6, 0.8, ... , 0.9 >

Outline

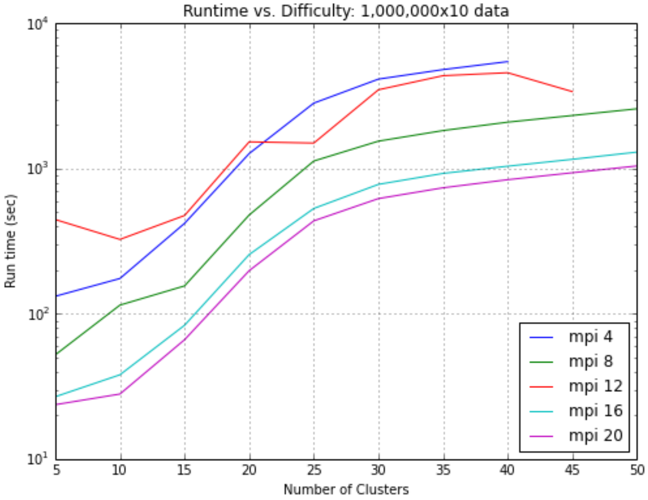
Multiple Implementations



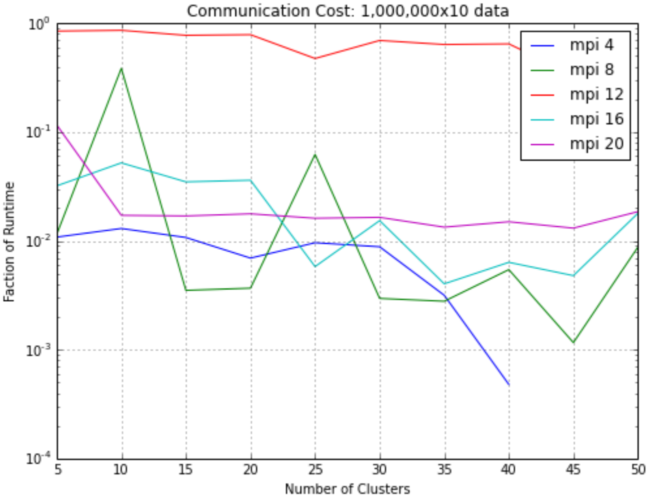
Larger Dataset



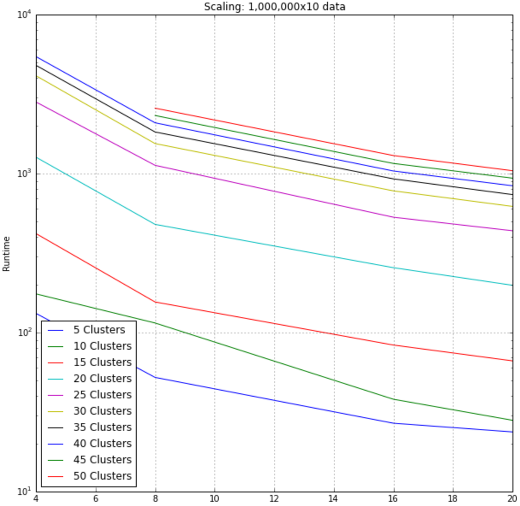
Affect of the Number of Clusters



Communication Cost



Strong Scaling



Outline

Comments

- The best method for parallelization will depend on your data
- There are much better methods for initializing the centroids to converge faster, kmeans++
- Reading plain text files with MPI is annoying
- Every now and then communication would hang (locally) for a long time before
- Saw some errors/warnings when running on Mercer for larger data that I didn't see previously

huhh??

```
Computation time: 10412.692955s
Communication time: 68.016616s
--------------------------------------------------------------------------
_orterun noticed that process rank 6 with PID 10686 on node compute-15-9 exited on signal 11 (Segmentation fault).
--------------------------------------------------------------------------
--------------------------------------------------------------------------
WARNING: a request was made to bind a process. While the system
supports binding the process itself, at least one node does NOT
support binding memory to the process location.

  Node:  compute-15-9

This usually is due to not having the required NUMA support installed
on the node. In some Linux distributions, the required support is
contained in the libnumactl and libnumactl-devel packages.
This is a warning only; your job will continue, though performance may be degraded.
--------------------------------------------------------------------------
--------------------------------------------------------------------------
The library attempted to open the following supporting CUDA libraries,
but each of them failed.  CUDA-aware support is disabled.
libcuda.so.1: cannot open shared object file: No such file or directory
/usr/lib64/libcuda.so.1: cannot open shared object file: No such file or directory
If you are not interested in CUDA-aware support, then run with
--mca mpi_cuda_support 0 to suppress this message.  If you are interested
in CUDA-aware support, then try setting LD_LIBRARY_PATH to the location
of libcuda.so.1 to get passed this issue.
--------------------------------------------------------------------------
(END)
```

Further Reading I

Arthur, David and Manthey, Bodo and Roglin, H
k-Means has polynomial smoothed complexity
Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on. (2009) 405-414

MacQueen, James and others
Some methods for classification and analysis of multivariate observations
Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. (1967) 281-297

Arthur, David and Vassilvitskii, Sergei
k-means++: The advantages of careful seeding
Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. (2007) 1027-1035