Technology for X-Informatics
Kmeans

June 19 2013

Geoffrey Fox
gcf@indiana.edu
http://www.infomall.org/

Associate Dean for Research, School of Informatics and Computing
Indiana University Bloomington
2013

Big Data Ecosystem in One Sentence

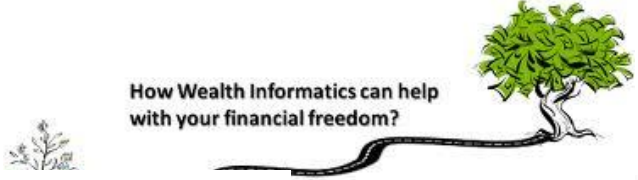
Use Clouds running Data Analytics Collaboratively processing Big Data to solve problems in X-Informatics (or e-X)

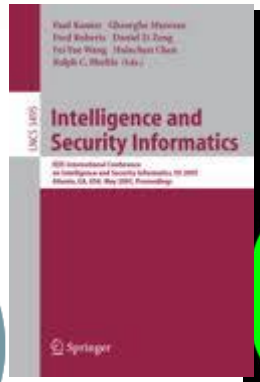
X = Astronomy, Biology, Biomedicine, Business, Chemistry, Climate, Crisis, Earth Science, Energy, Environment, Finance, Health, Intelligence, Lifestyle, Marketing, Medicine, Pathology, Policy, Radar, Security, Sensor, Social, Sustainability, Wealth and Wellness with more fields (physics) defined implicitly

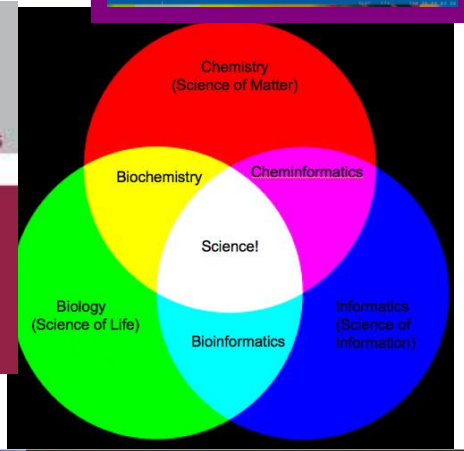
Spans Industry and Science (research)

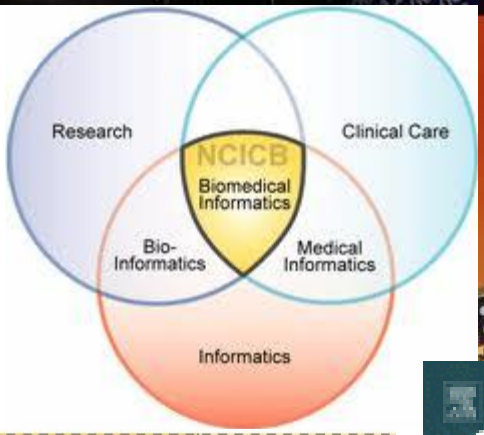
Education: Data Science see recent New York Times articles

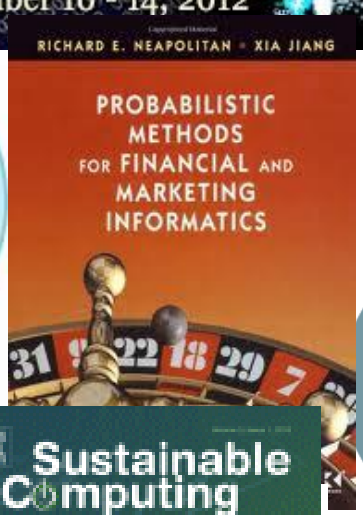
http://datascience101.wordpress.com/2013/04/13/new-york-times-data-science-articles/

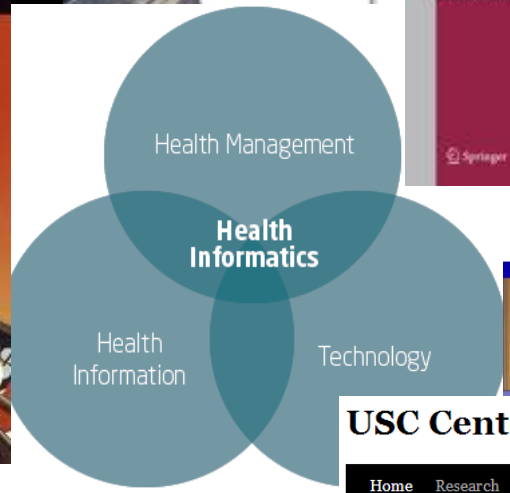


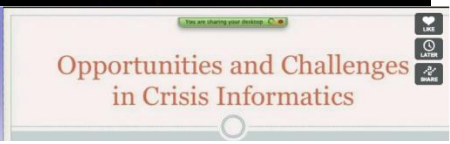


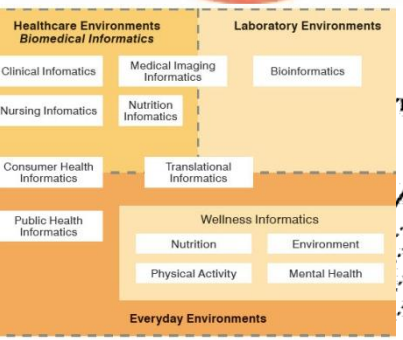


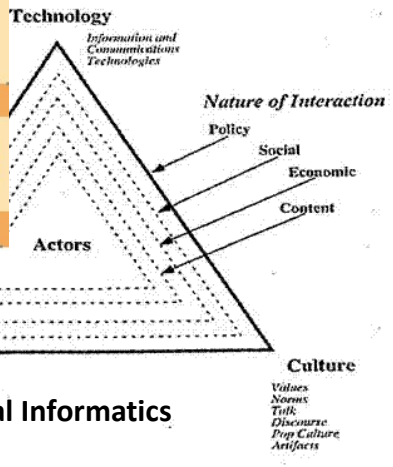


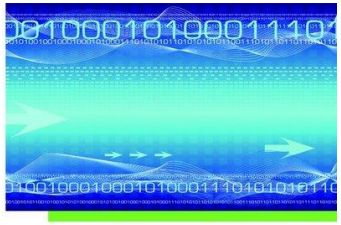








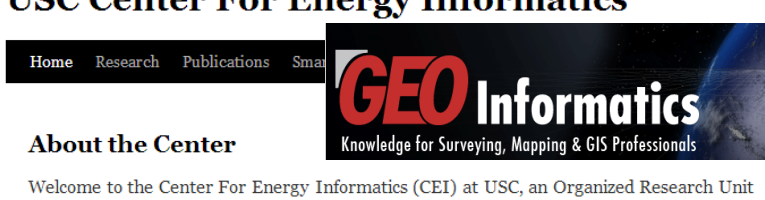


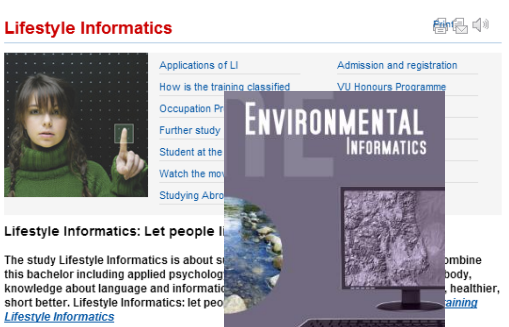


Noelia Penelope Greer (Ed.)

Business Informatics

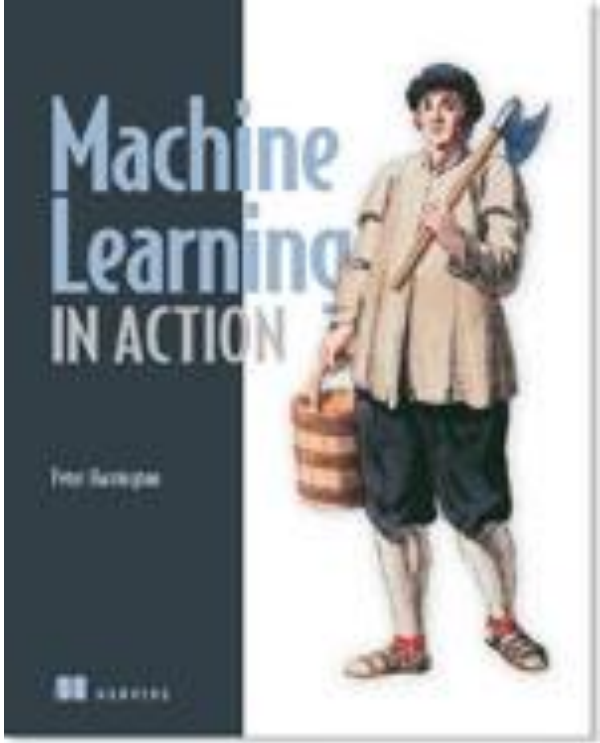
Information technology, Management,





Kmeans in Python

Kmeans Resources



- **Machine Learning in Action**

  Peter Harrington

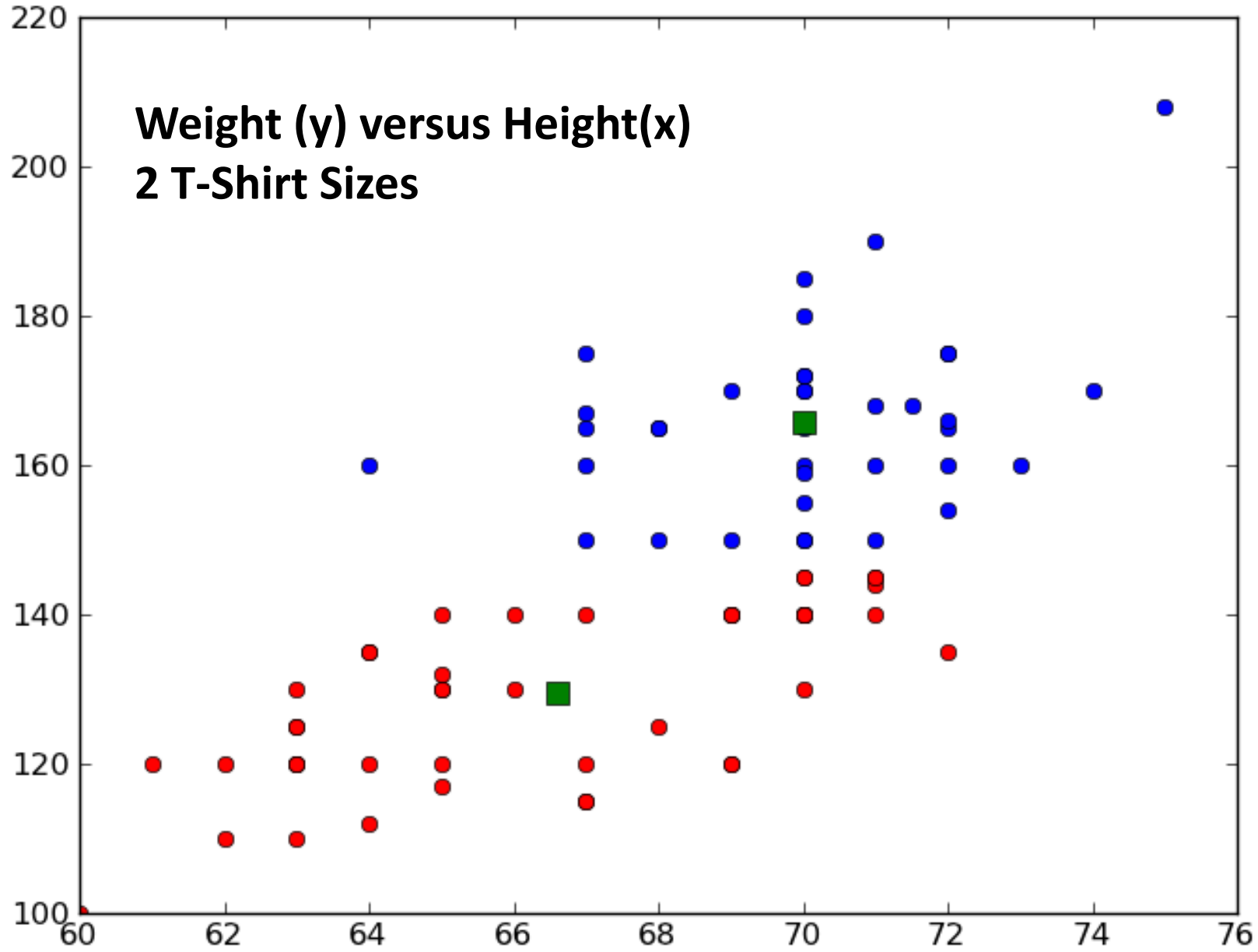
  April, 2012 | 384 pages
  ISBN: 9781617290183
- http://www.manning.com/pharrington/MLiA_SourceCode.zip
- Chapter 10 discusses Kmeans but we will NOT use this software but rather the built in support of Kmeans in SciPy
- This SciPy software will be modified for some of experimentation reported here

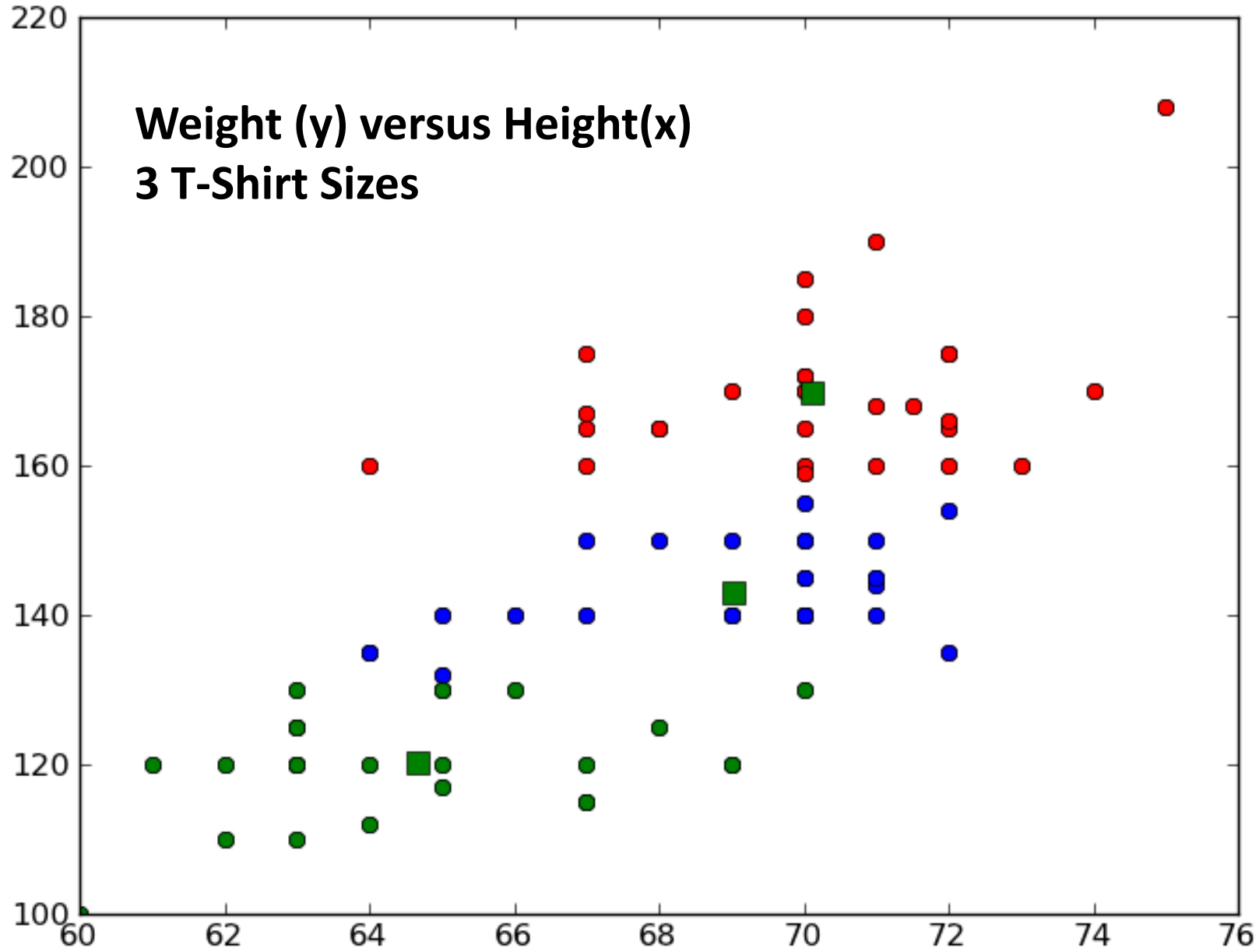
Resources Used

- SciPy Software https://github.com/scipy/scipy/blob/master/scipy/cluster/vq.py has both kmeans code and a "utility" vq to associate points with clusters
- http://docs.scipy.org/doc/scipy/reference/cluster.vq.html describes software
- File KmeansExtra.py has drivers to invoke SciPy code to calculate 6 different scenarios based on "fake" data corresponding to 4 actual clusters of three different sizes "small" "large" "very large"
- File ParallelKmeans.py has modified Kmeans code invoking same vq Utility. Modification supports MapReduce Parallelism (see later) and different goodness measure. Gives 8 plots (adds K=6 and 8) and will be used in lecture
- xmean.py uses Python built in Kmeans to read data from CSV file
- sample.csv Sample height/weight data from http://www.kosbie.net/cmu/fall-10/15-110/
  - http://www.kosbie.net/cmu/fall-10/15-110/handouts/notes-clustering/tshirts-GHIJ-nooutliers.csv

Simple T-Shirt Example

- Cluster 85 T-shirt records in sample.csv into a number of categories set by number of clusters set in xmean.py file
  - 2 clusters is default
- We try 2 3 4 5 Clusters
- Note Kmeans allows more clusters but plot only works up to 5


Weight (y) versus Height(x)
2 T-Shirt Sizes
220
200
180
160
140
120
100
60
62
64
66
68
70
72
74
76


Weight (y) versus Height(x)
3 T-Shirt Sizes
220
200
180
160
140
120
100
60
62
64
66
68
70
72
74
76

Clusters Generated

- There are 4 clusters each with 250 points generated as Gaussians (remember discussion of Physics Higgs particle) with given centers and radii (standard deviations)
- Center (0, 0) Radius 0.375
- Center (3, 3) Radius 0.55
- Center (0, 3) Radius 0.6
- Center (3, 0) Radius 0.25
- Note largest clusters are those with y=3 (on top)
- These are “large” clusters
- “small” clusters have radii = 0.25 these retaining ratio given above
- “very large” clusters have radii = 1.5 these

Kmeans

- Have a set N of Points – find a specified number K of clusters
- Choose a “random” starting point e.g. randomly choose K points to be centers
- Iterate until converged
  - Associate points with cluster whose center they are nearest
  - Calculate new center positions as centroid (average) of points associated with center
- Note Python has a Kmeans code which we use
- As source available we modify to illustrate parallel computing and change goodness criterion

Sequential version of Kmeans

- Overall iteration over starting positions
  - Initialize Centroids
  - Iterate until converged
    - Find association of points and centers
    - Calculate distortion (Point-Center distance)
    - Find centroids as mean of point vectors
    - Delete zero size clusters
    - Check convergence
- Return best solution based on Goodness criterion which for SciPy version is average distortion
- Later tweak algorithm to put in MapReduce form

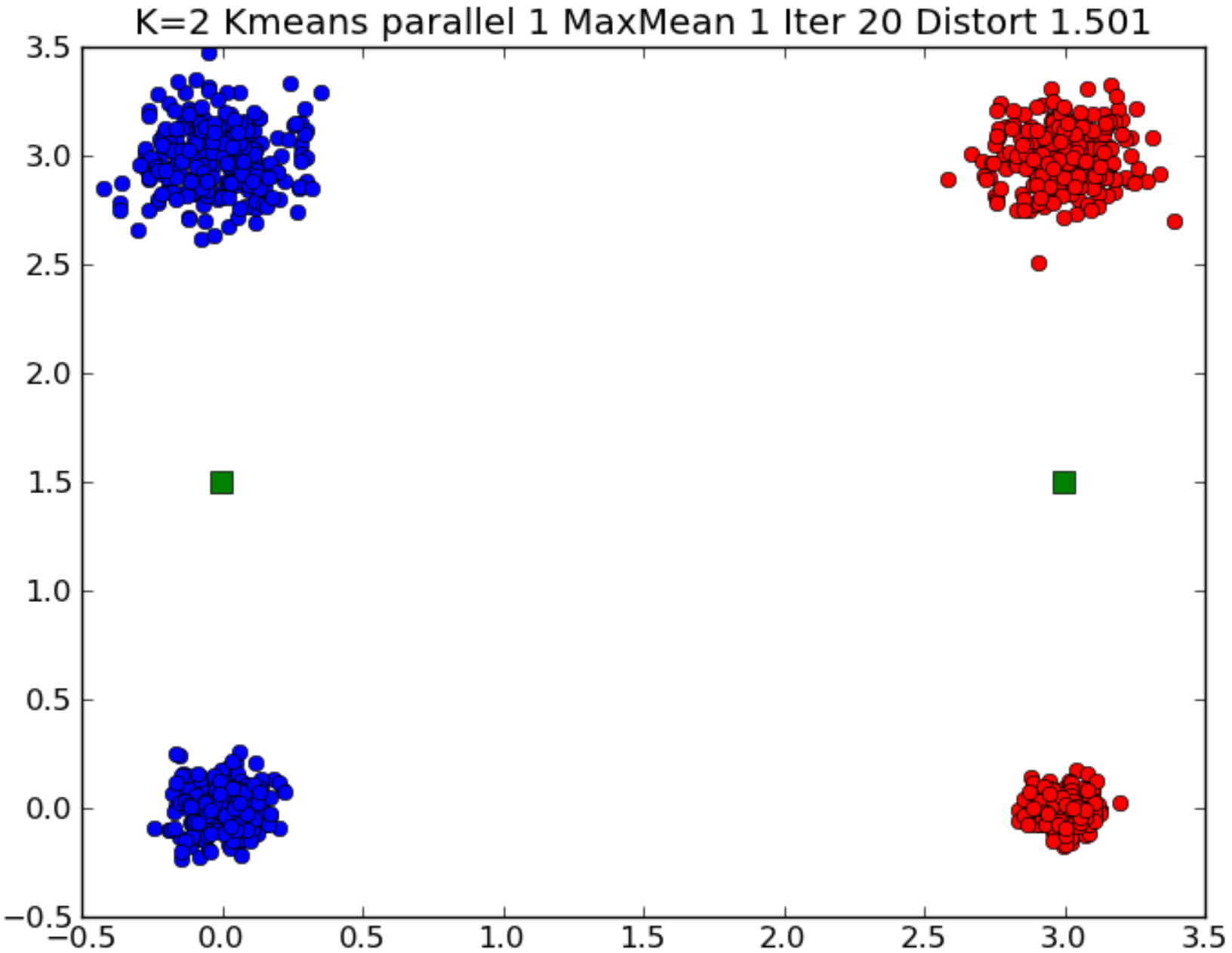
Analysis of 4 Artificial Clusters

Kmeans on 4 "Artificial" Clusters

- Fixed Center positions at
  - (0,0) radius 0.375
  - (3,3) radius 0.55
  - (0,3) radius 0.6
  - (3,0) radius 0.25
- These are "large clusters"
- "Small Clusters" have radius multiplied by 0.25
- "Very Large Clusters" radius multiplied by 1.5
- Each clusters has 250 points generated in normal (Gaussian) distribution with standard deviation 1
- We used Kmeans with 2, 4 and just for fun 6 or 8 clusters (i.e. more than "real number" of clusters

Comments

- As random data and random starting points, get different answers each time
- Can fix seeds for reproducible results
- Generally get “correct” answers
- Reduce number of iterations (20 default in Python) of independent runs to increase “interesting” results with non optimal solutions



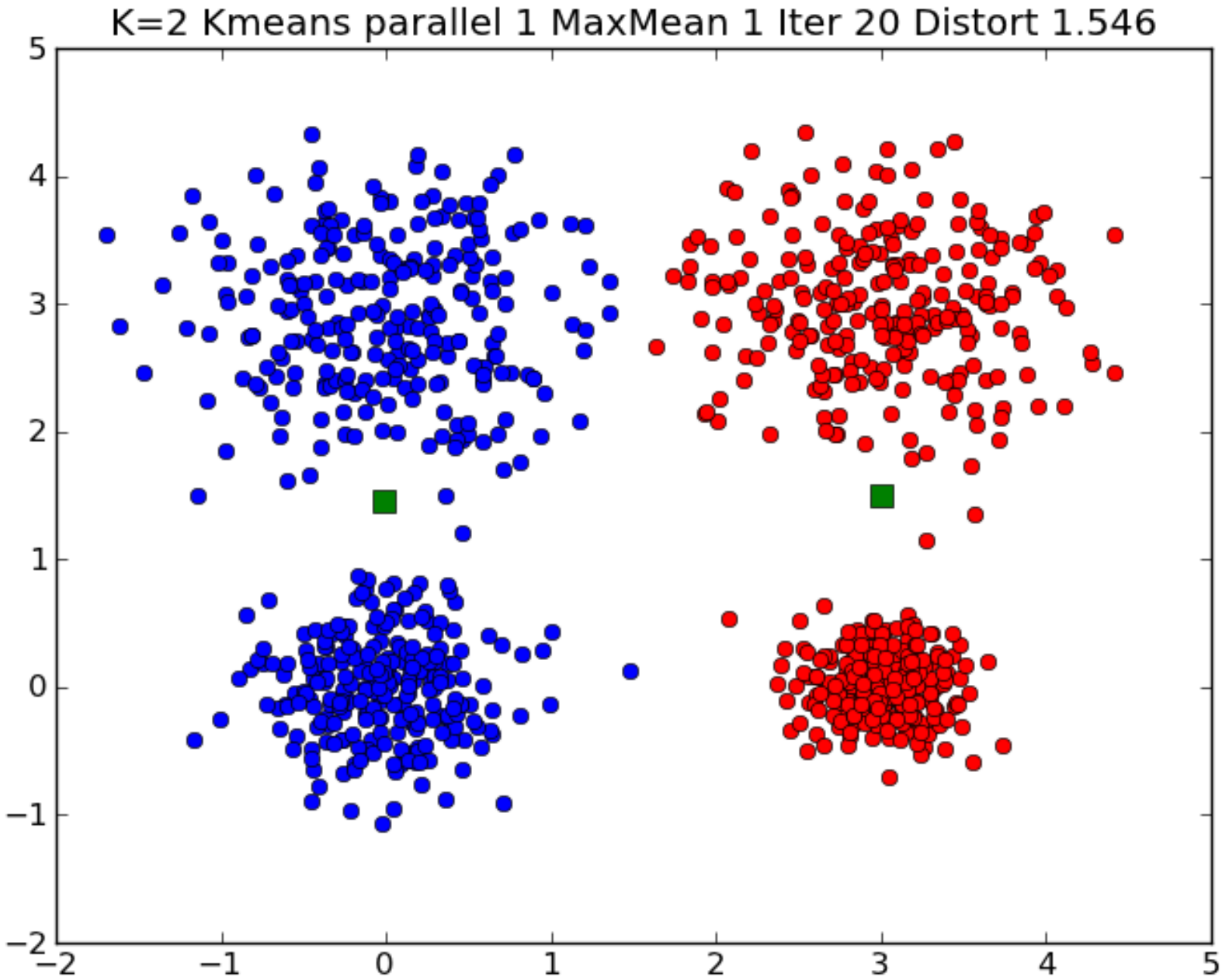
Typical K=2 Small Clustering Note distortion ~1.5

In Graph Title

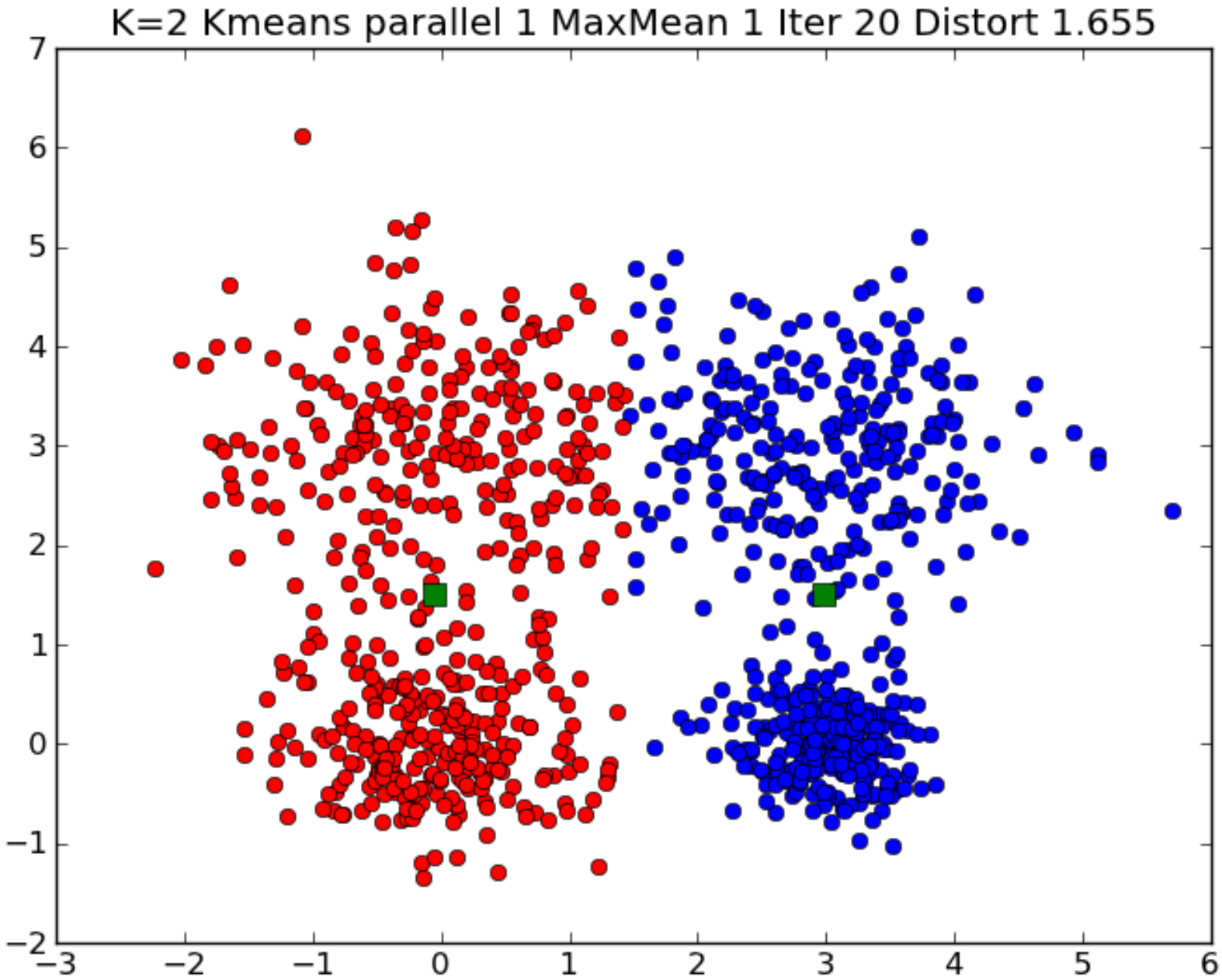
- K is number of clusters looked for K = 2…8
- Parallelism is level of (pseudo) parallelism. Used for illustrating MapReduce
- MaxMean = 1 Judge quality by mean distortion
  - Distortion is distance from point to its center
- MaxMean = 2 Judge quality by max distortion
- Iter is number of totally independent Kmeans iterations to run; return highest quality
- Minimizing distortion measure controlled by MaxMean is quality measure

kmeans_gcf(data, Numclusters, NumIterations, Thresh, Parallelism, MaxMean)

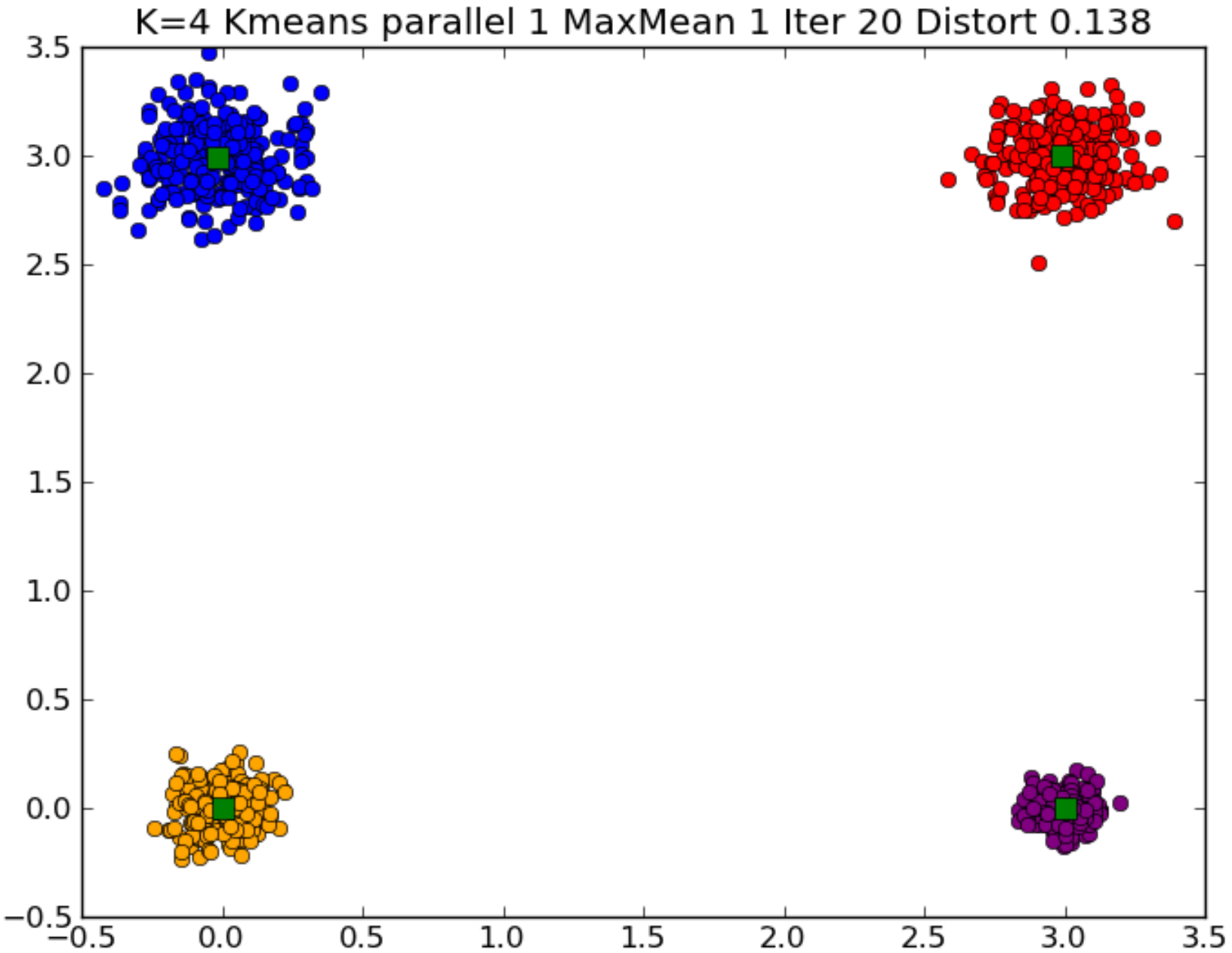
- **data** is points to be clustered
- **Numclusters** (2 4 6 8 here) is number of clusters to find
- **NumIterations** is number of independent iterations to perform returning solution with minimum “distortion” as described earlier
- **Thresh** controls each iteration of Kmeans; stop when change in distortion < Thresh. Note this can still be a non optimal solution
- **Parallelism** = 1 here. Later lecture runs a simple MapReduce parallel version with Parallelism = 2
- **MaxMean** = 1 is usual Python Kmeans. MaxMean = 2 is alternative quality measure described earlier



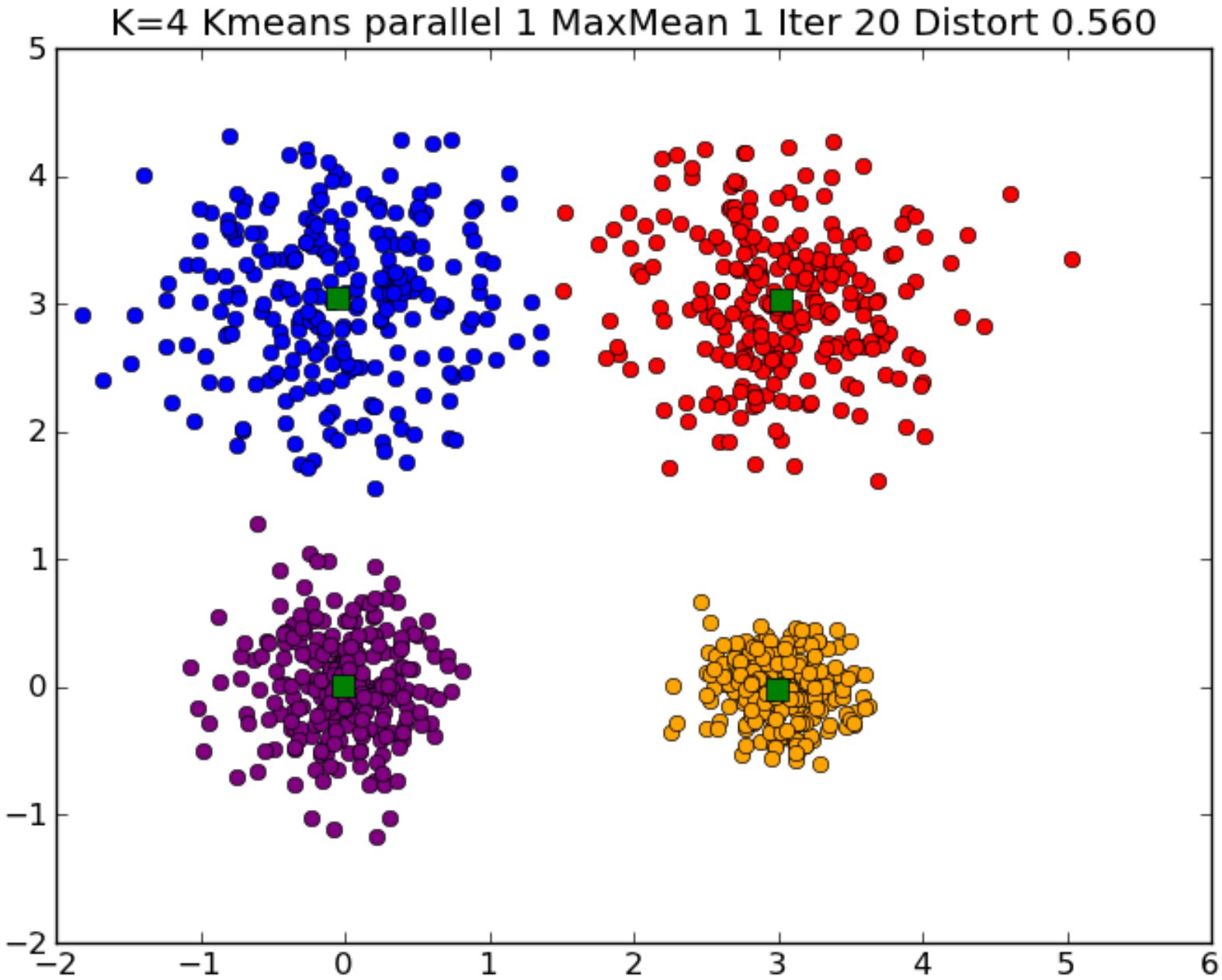
Typical K=2 Large Clustering



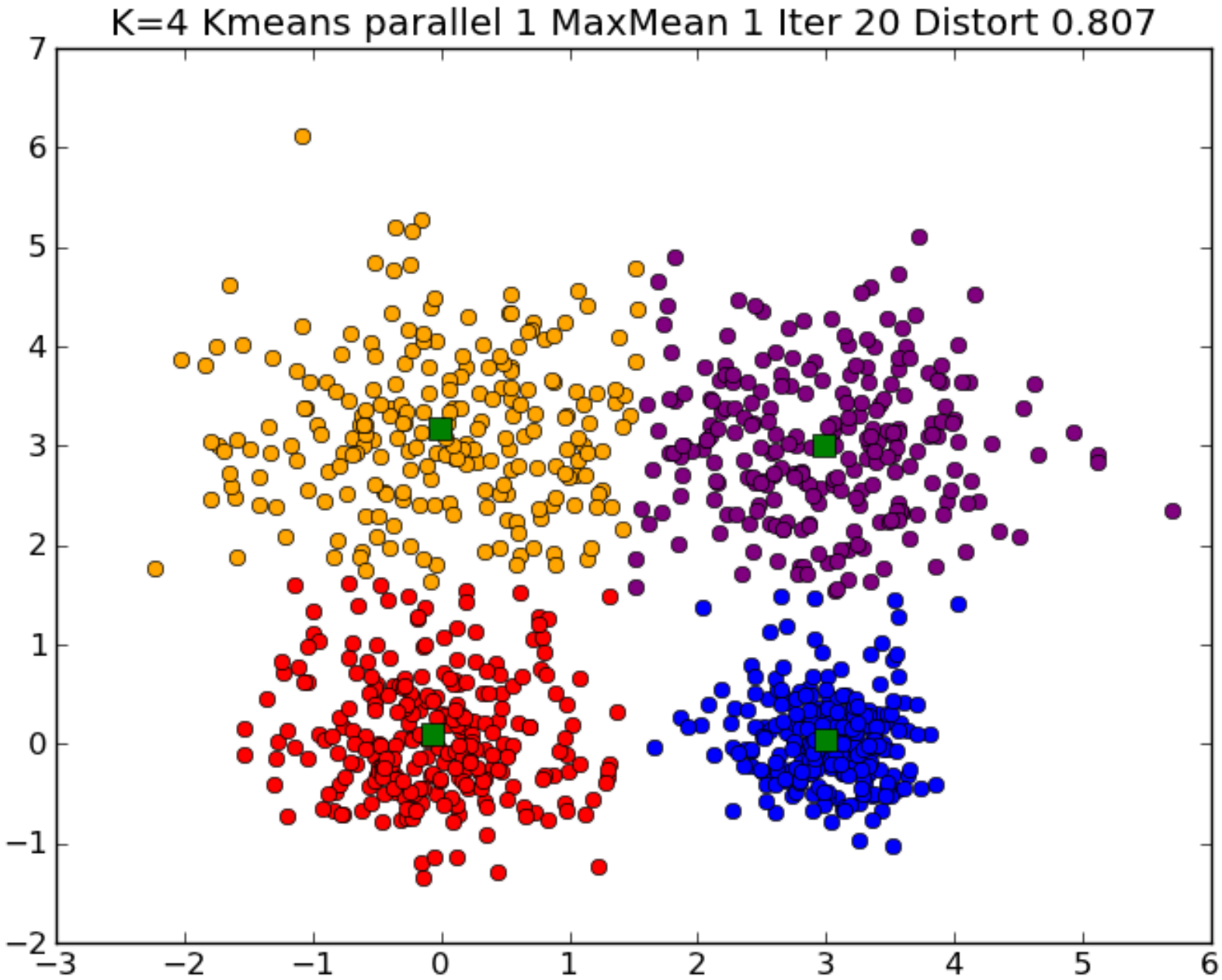
Typical K=2 Very Large Clustering



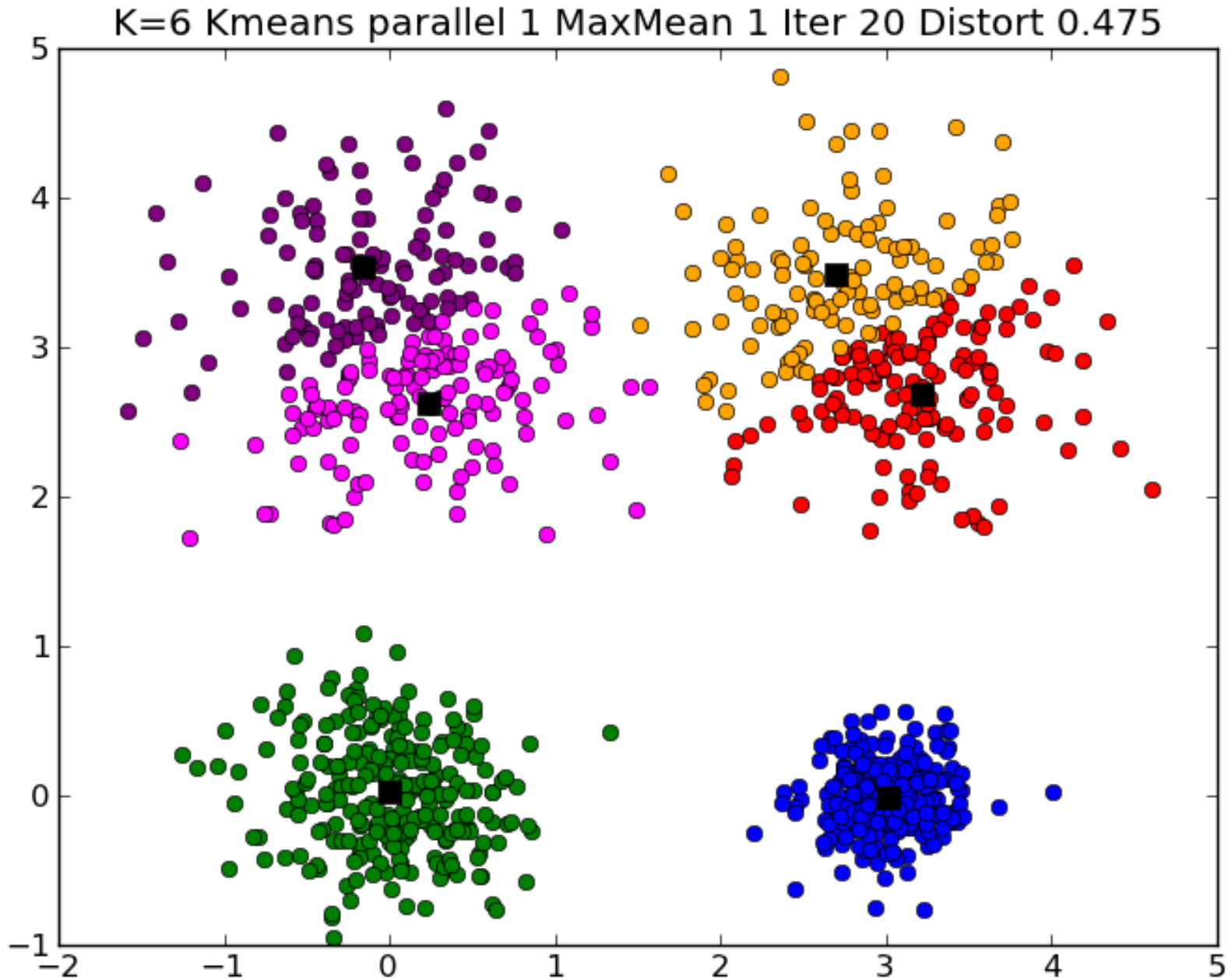
Typical K=4 Small Clustering



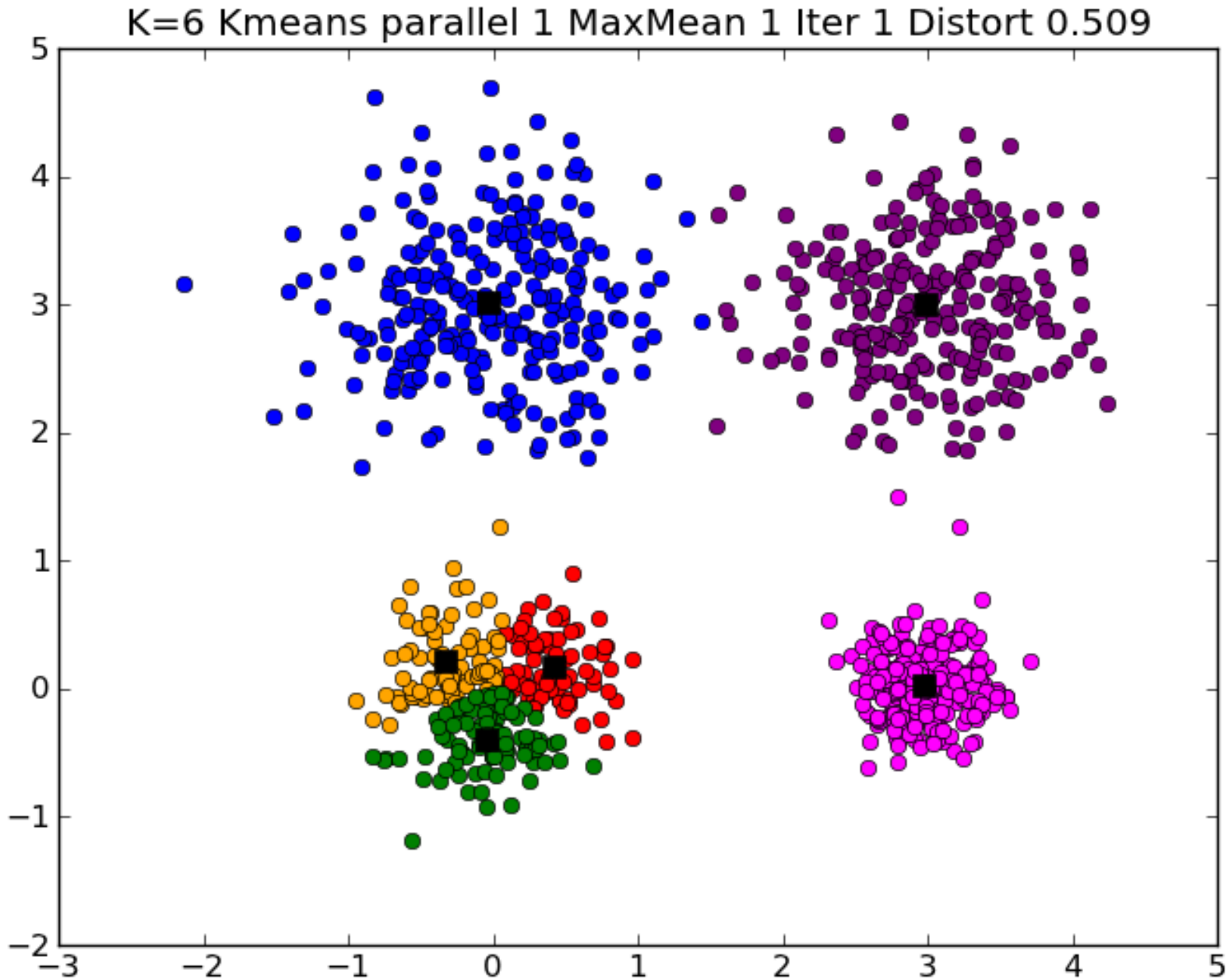
Typical K=4 Large Clustering



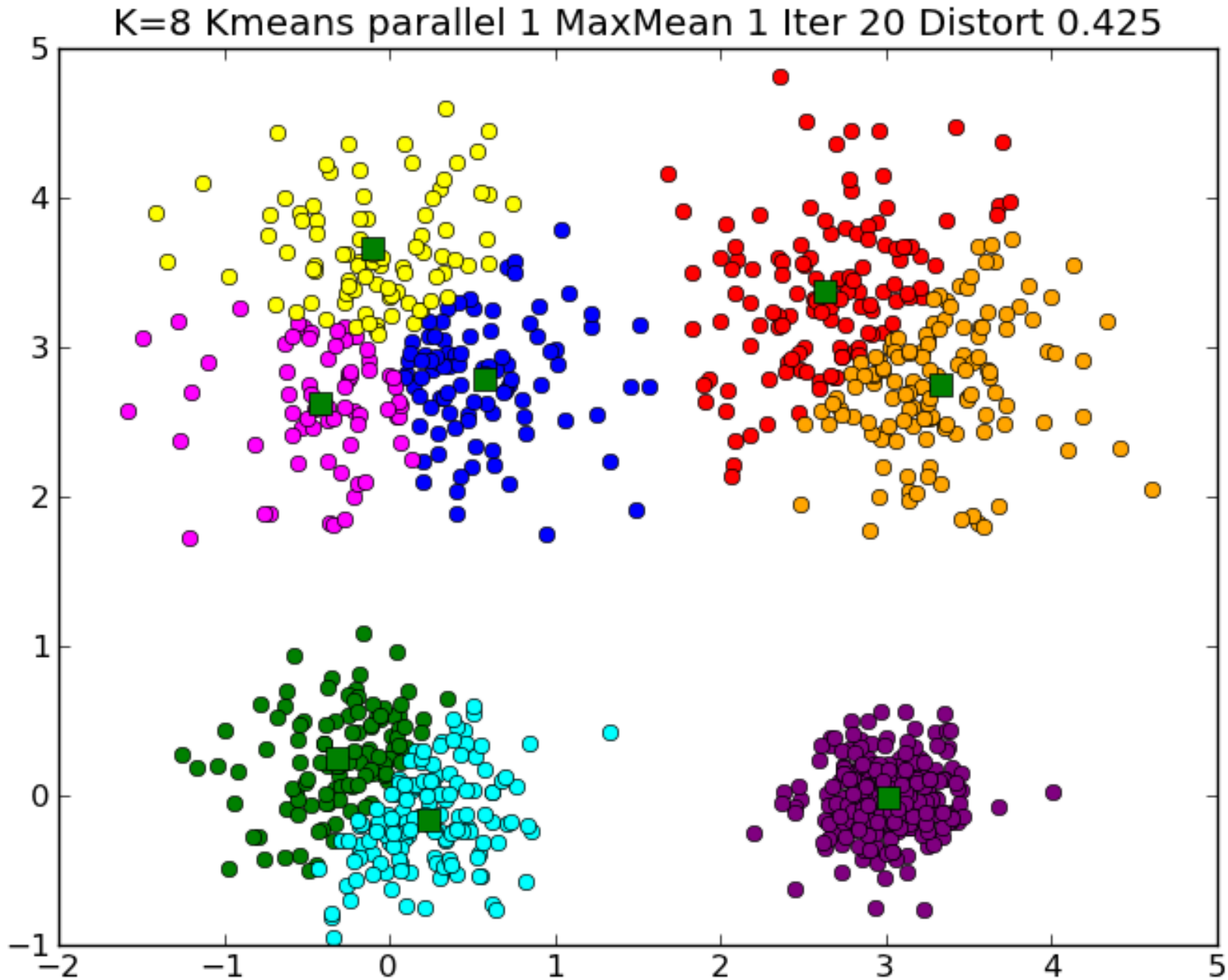
Typical K=4 Very Large Clustering



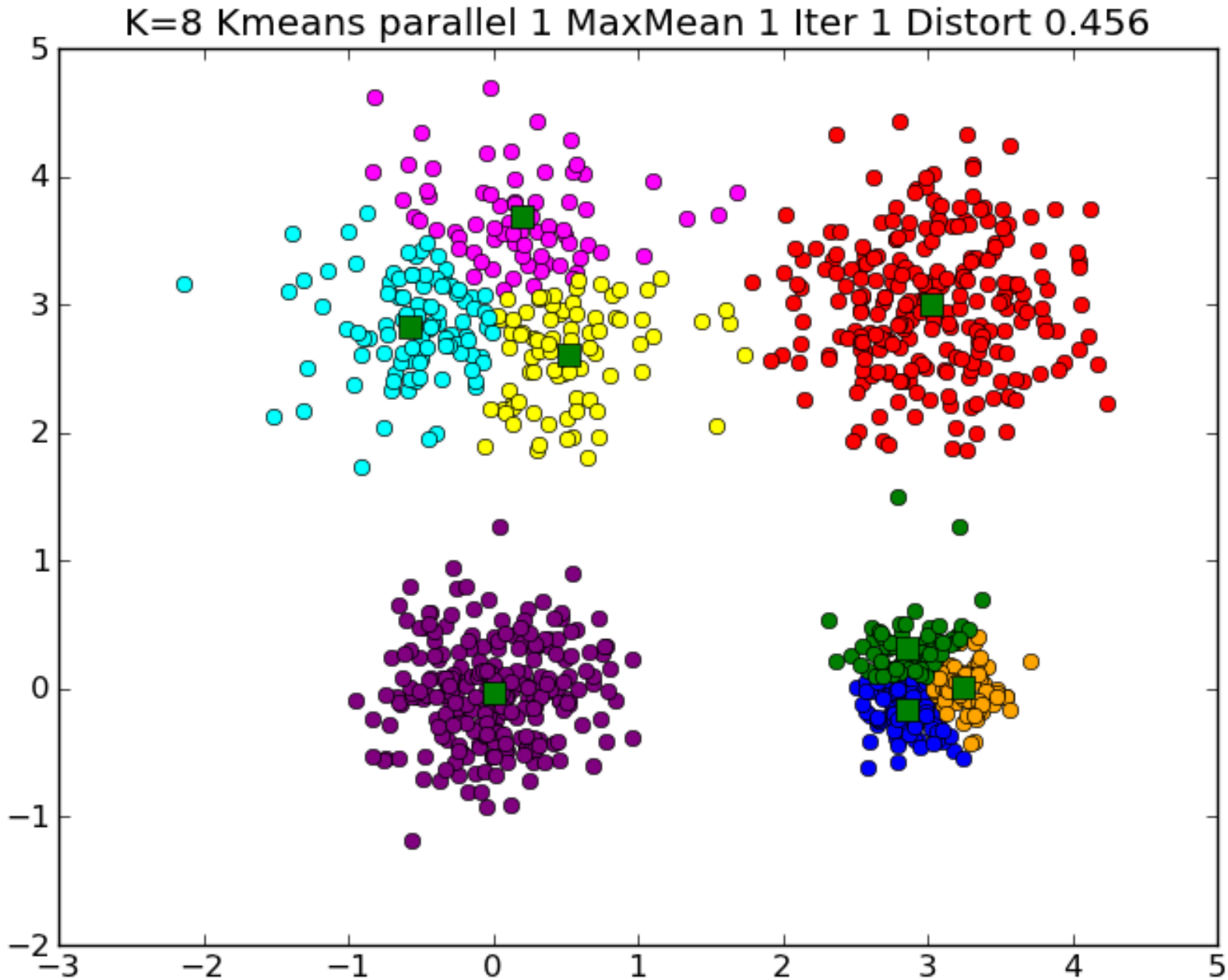
Typical K=6 Large Clustering



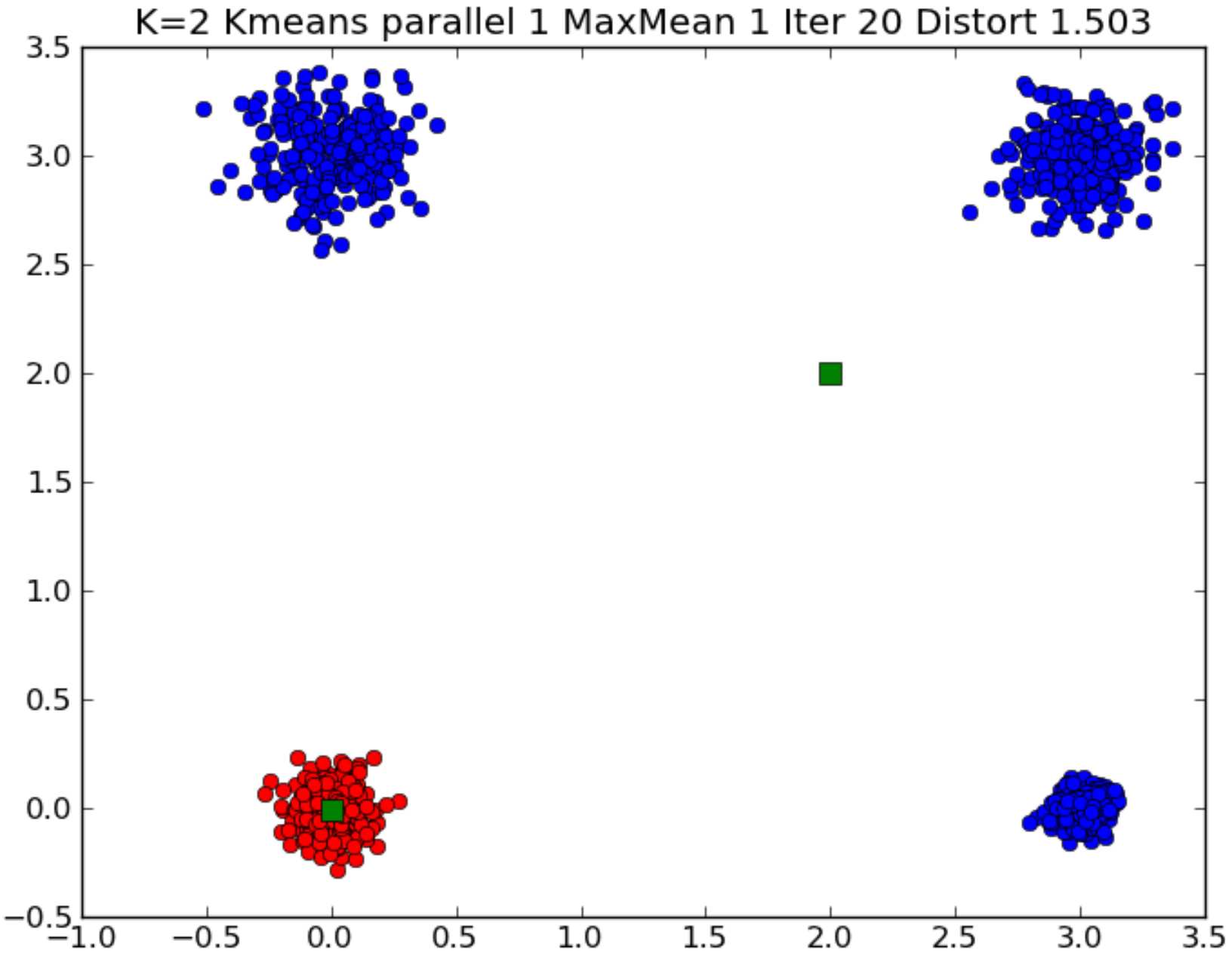
Different K=6 Large Clustering



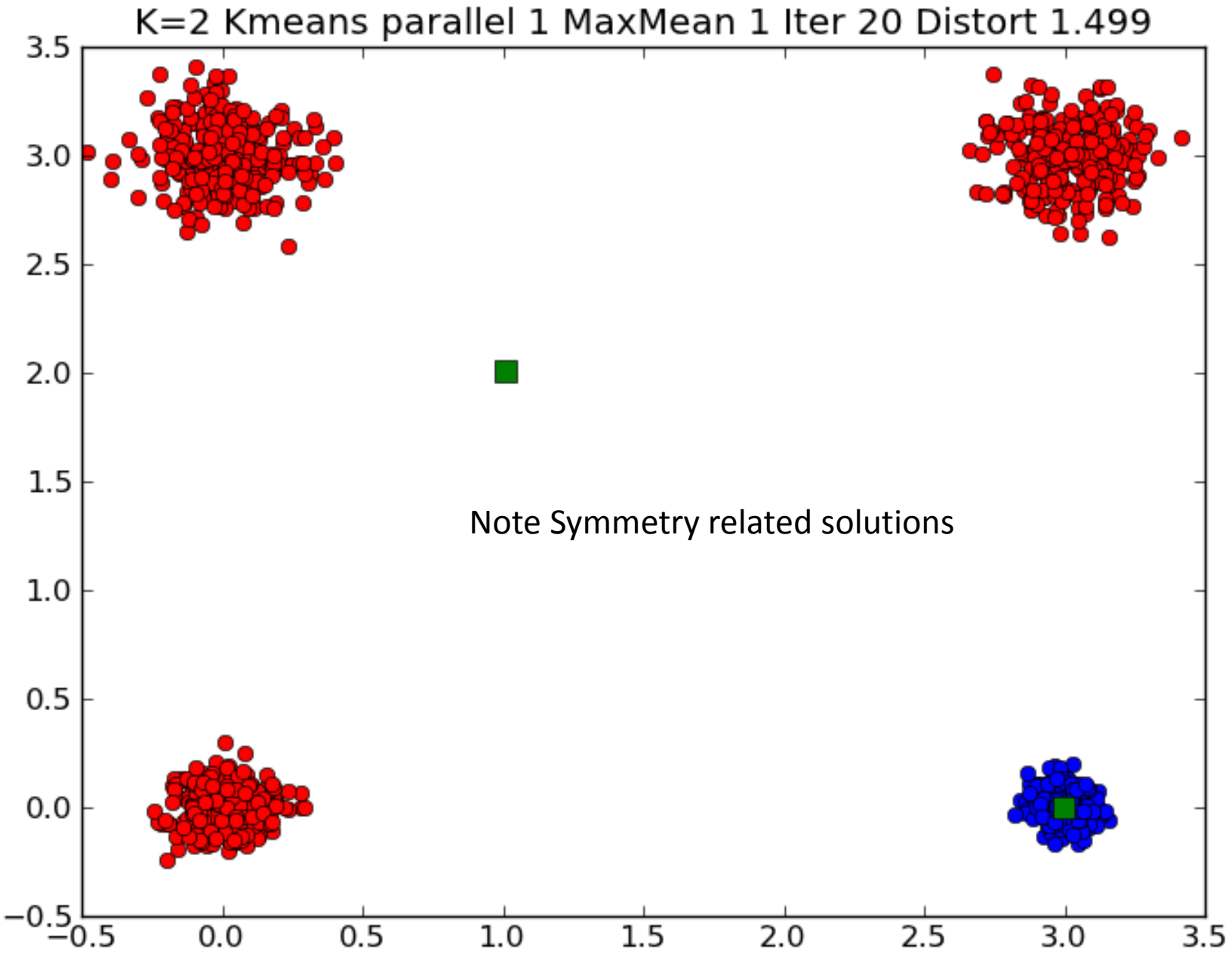
Typical K=8 Large Clustering



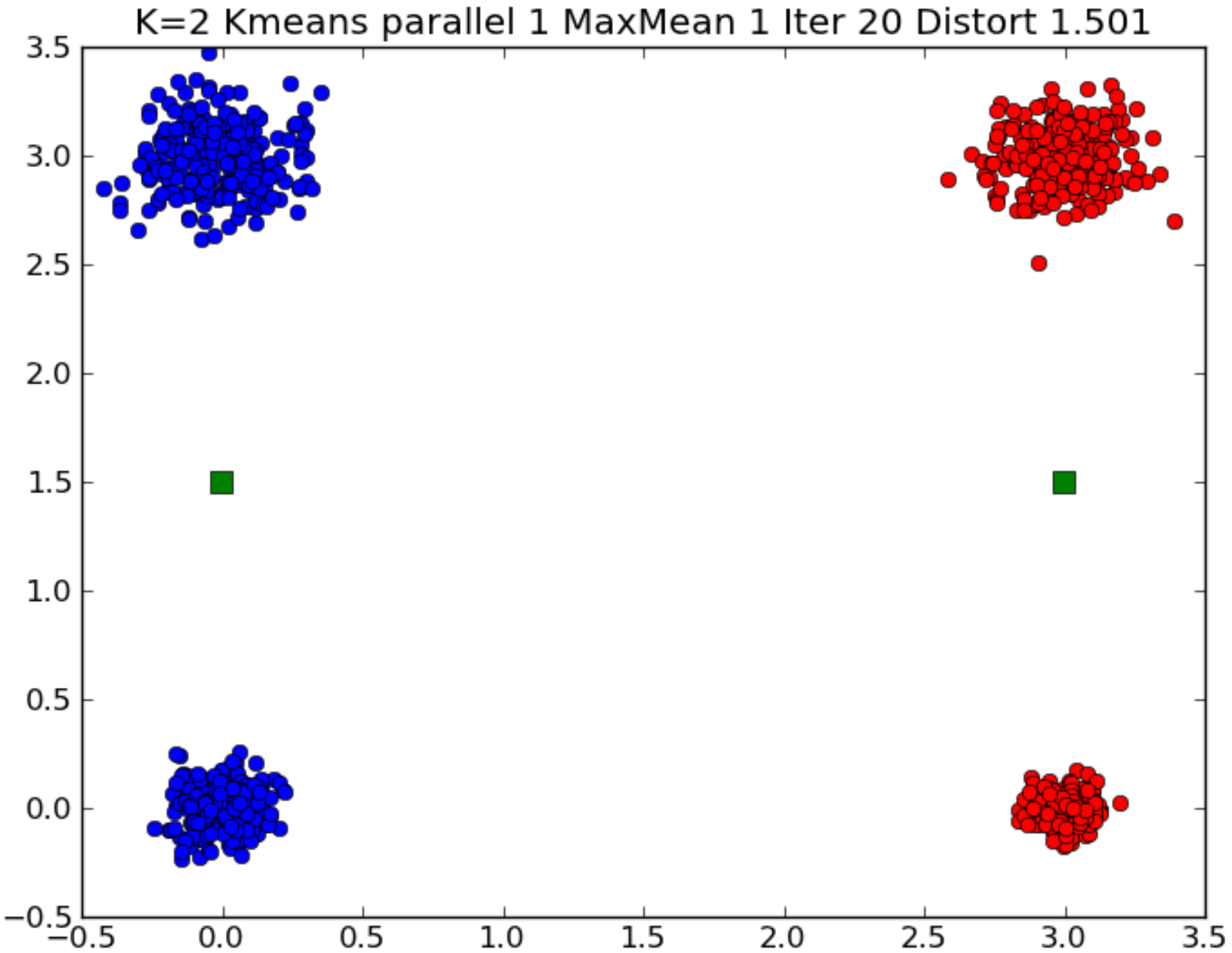
Another K=8 Large Clustering



A different solution of K=2 Small Clustering



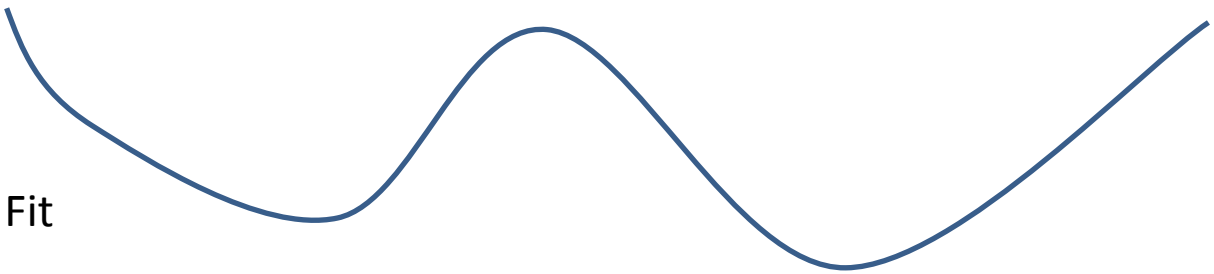
A second look at a different solution of K=2 Small Clustering



Typical K=2 Small Clustering

Note 4 Small Cluster Case is Ambiguous

- There are two solutions for K =2 which have similar measures of goodness
  - 2 Clusters each of 2 real clusters
  - 1 Cluster is 1 real cluster, the other is three real clusters
- Current measure of goodness is “Average distance between point and its cluster”. This roughly 1.5 for both solutions
- Details of starting point determines which solution you find
  - Also sensitive to “random clusters” i.e. solutions are so near that they can change with choice of Gaussianly distributed points

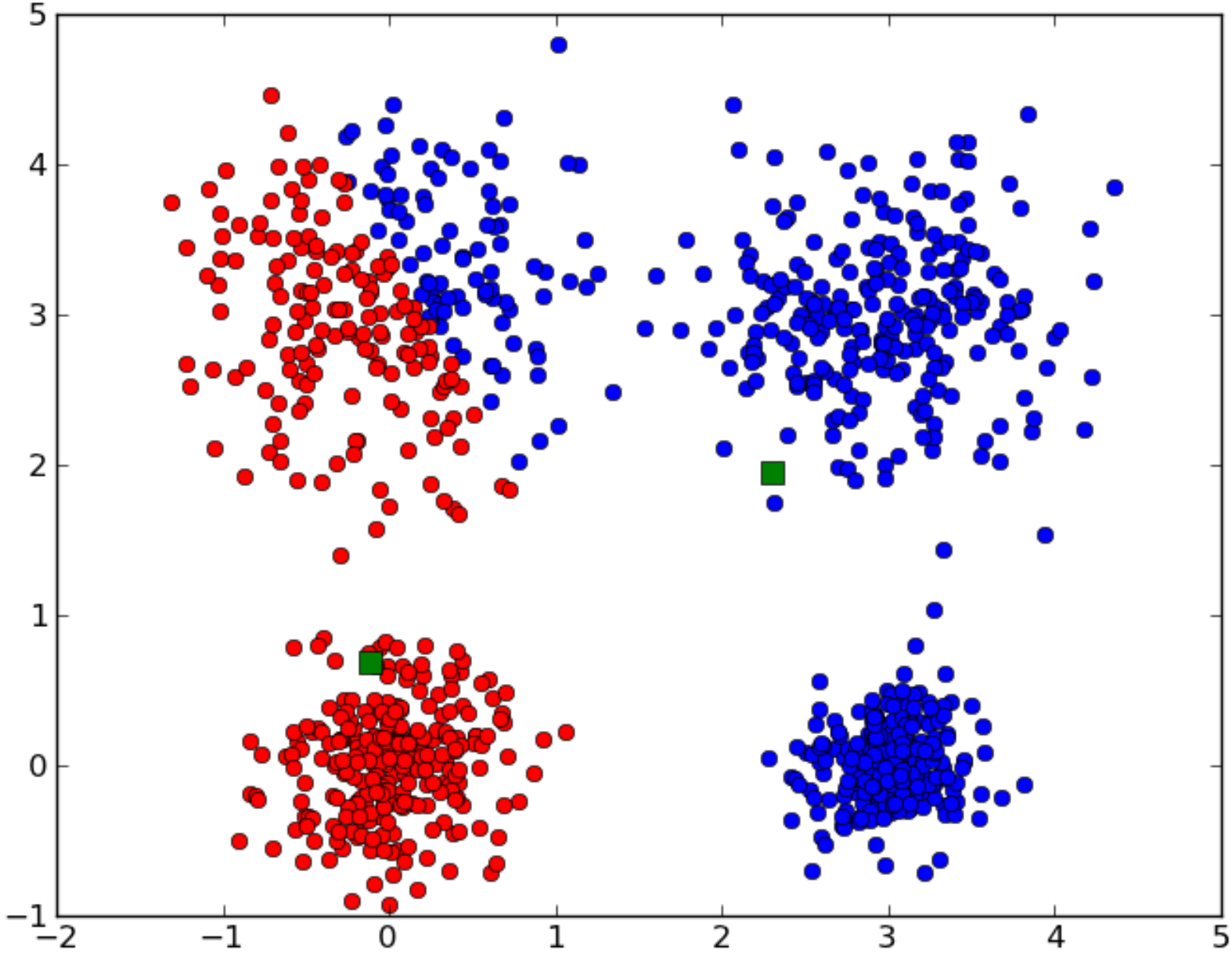


Ambiguous Solutions

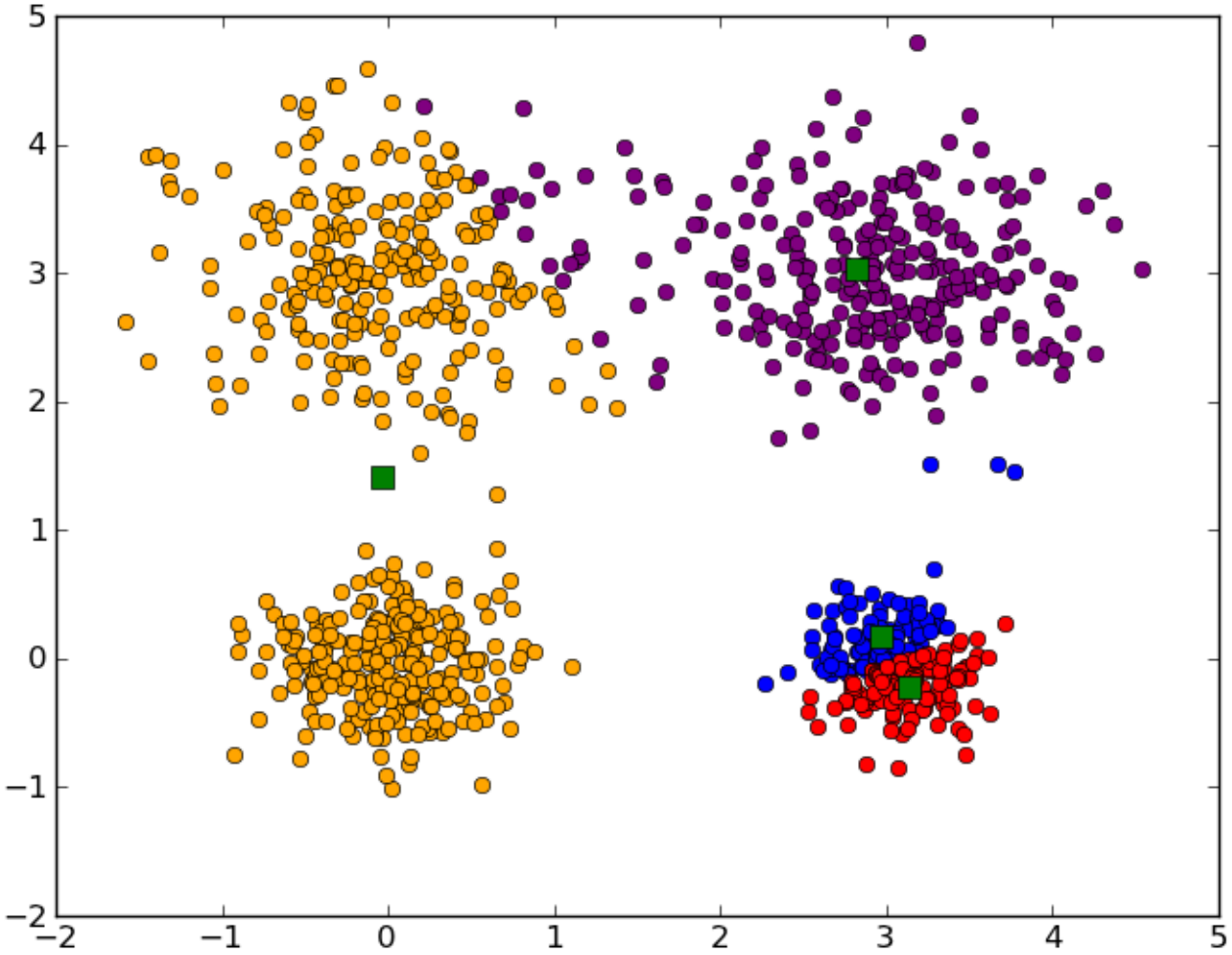
- The Python code runs Kmeans 20 times and chooses best solution out of this 20.
- The 4 small cluster case has such similar goodness values for two cases that results vary from run to run
- The 2 real cluster case has goodness ~ 3 (distance between generated centers)/2 = 1.5
- In other solution a typical case is centers at (0,0) and (2,2) with goodness = $(2\sqrt{5} + \sqrt{2} + 0)/4 = 1.47$
  - Exact for very small size clusters
- Changing goodness to Maximum not mean distance of points to centers will lead to unique solution with each cluster having two real clusters
  - Max values is 1.5 which is much less than $\sqrt{5} = 2.24$

Note the Hill between Solutions

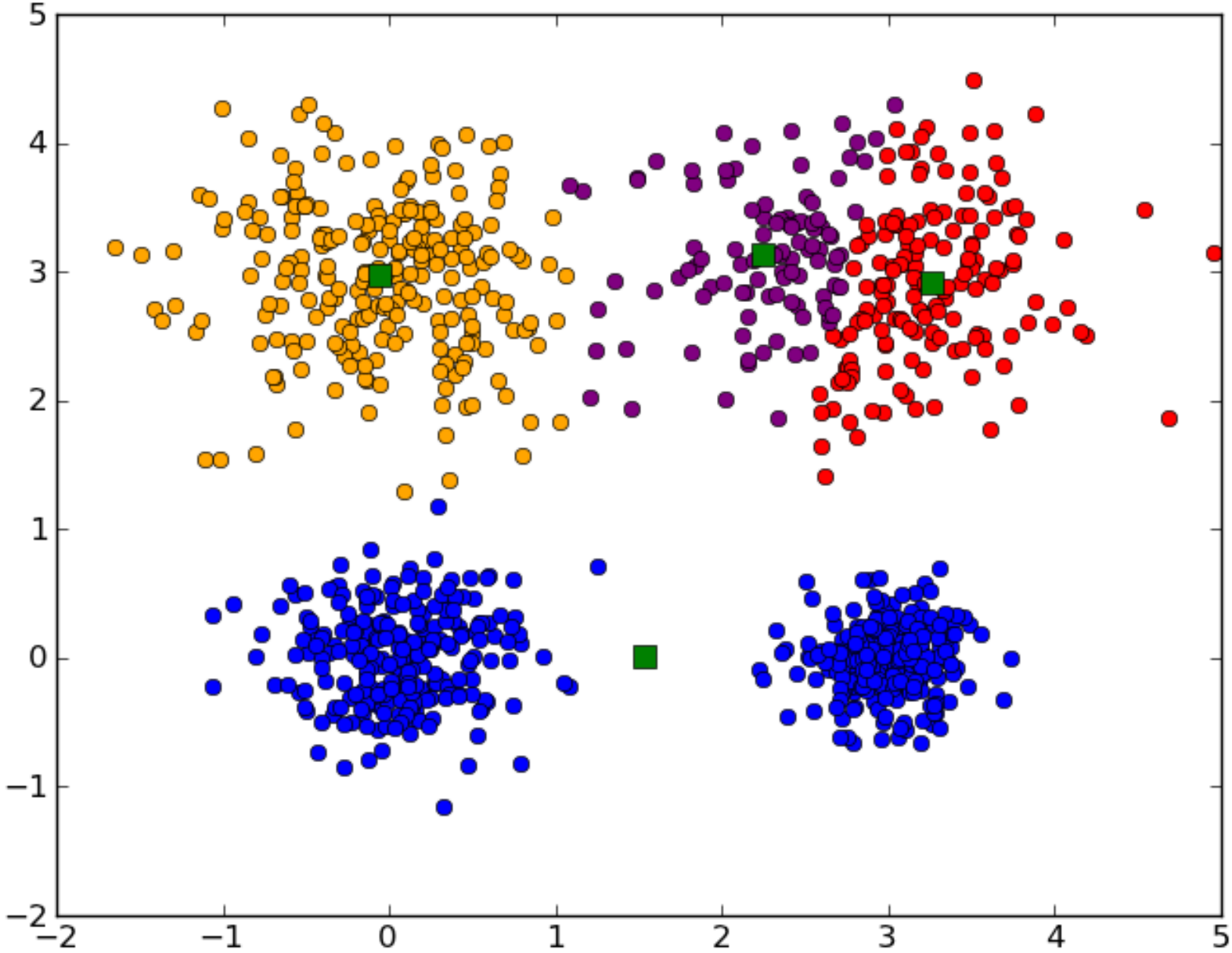
- There are 2 symmetry related solutions with 2 real clusters per Kmeans solution
- There are 4 symmetry related solutions with 3+1 real clusters per Kmeans solution
- These 6 solutions are separated by hills and are local minima
  - They are stable on small perturbations
  - Need big change to get from one to another
- Very common feature seen in change of leadership in both democratic and non democratic societies
  - Perhaps in play in battle IoS v Android v Blackberry v Windows 8 in smart phones
- Blackberry messed up enough to let Android/IoS take over
- To change state, it is insufficient to be better
  - You need to be so much better that can jump hill



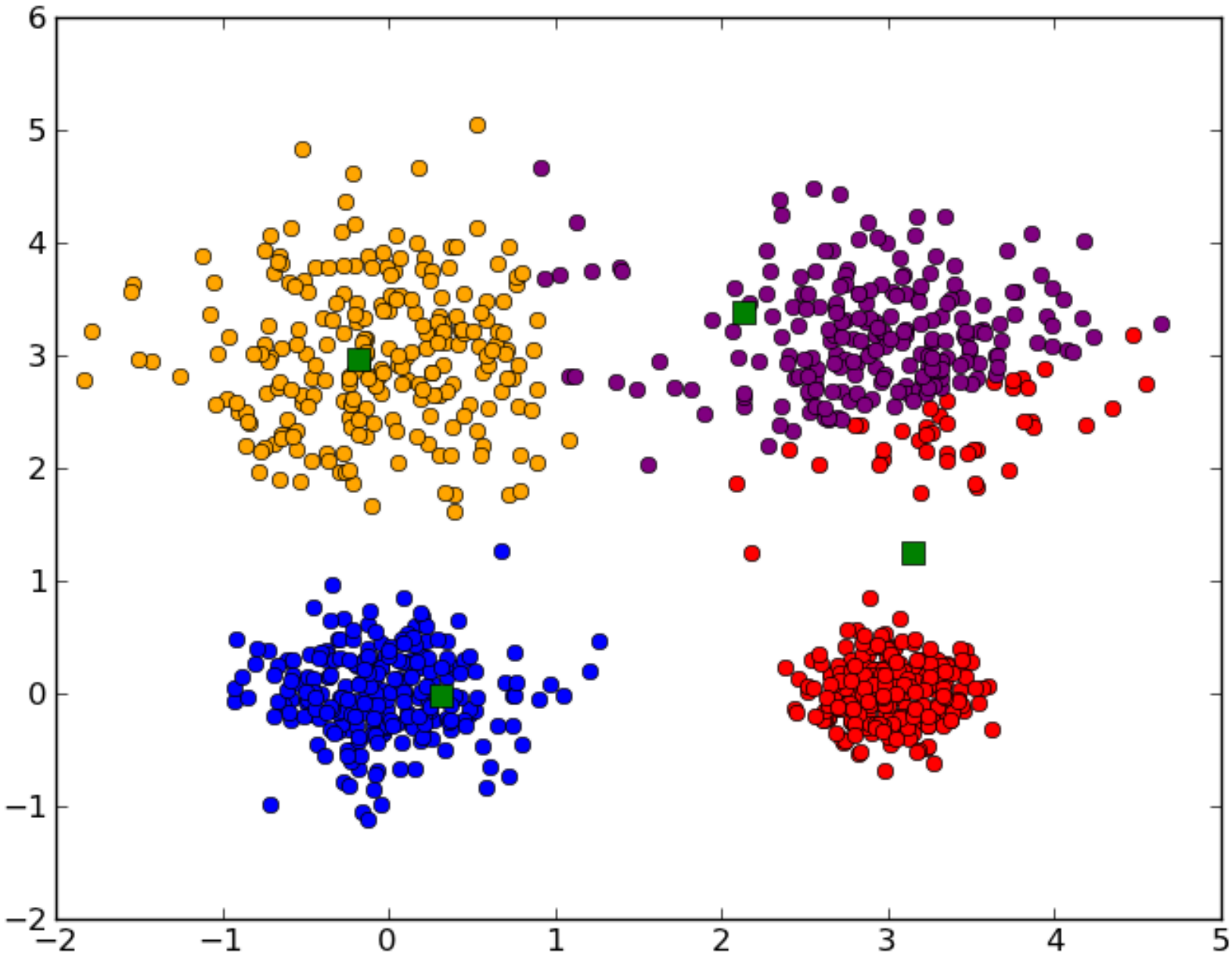
Messy K=2 large Clustering



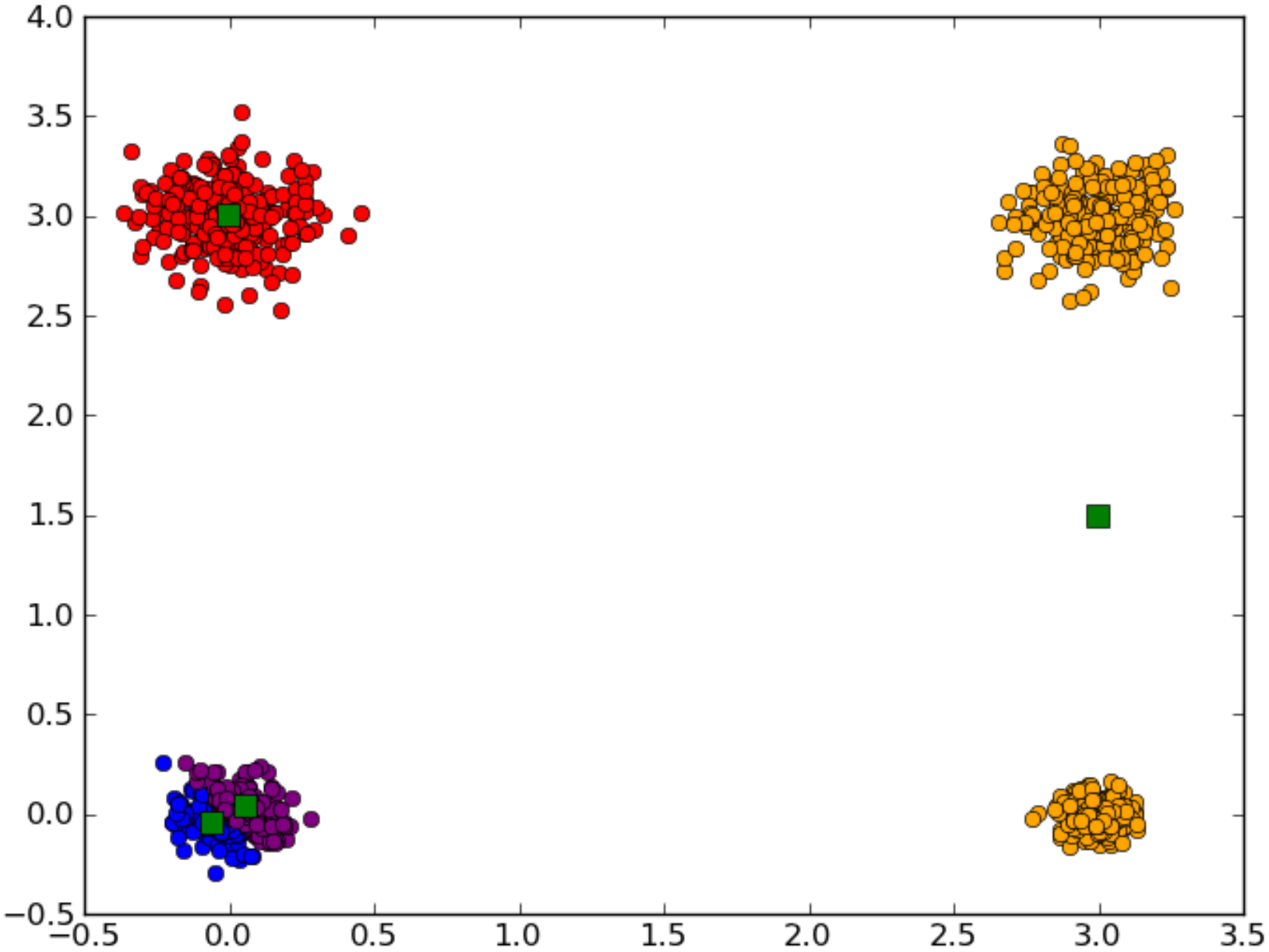
Messy K=4 large Clustering



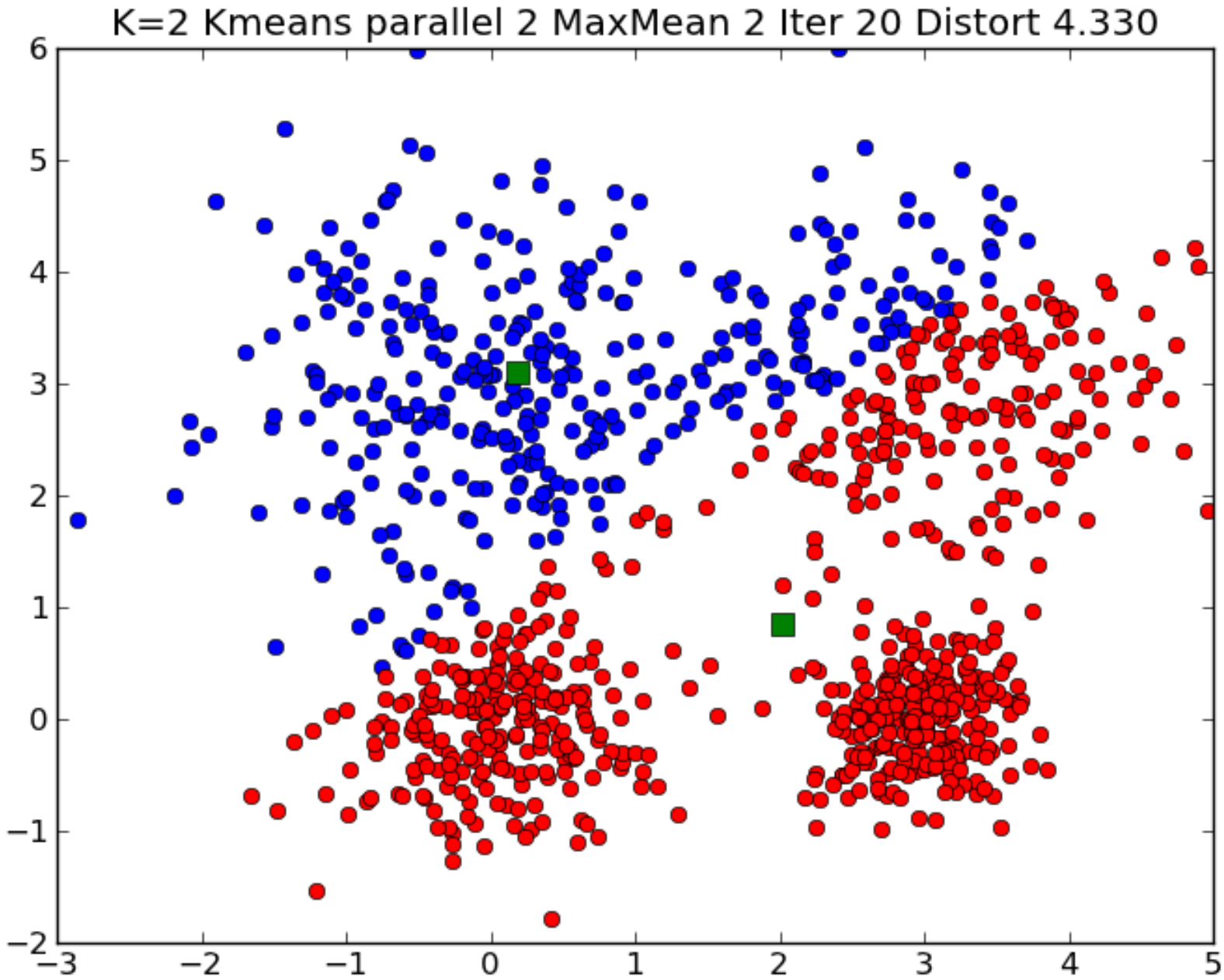
Messy K=4 large Clustering



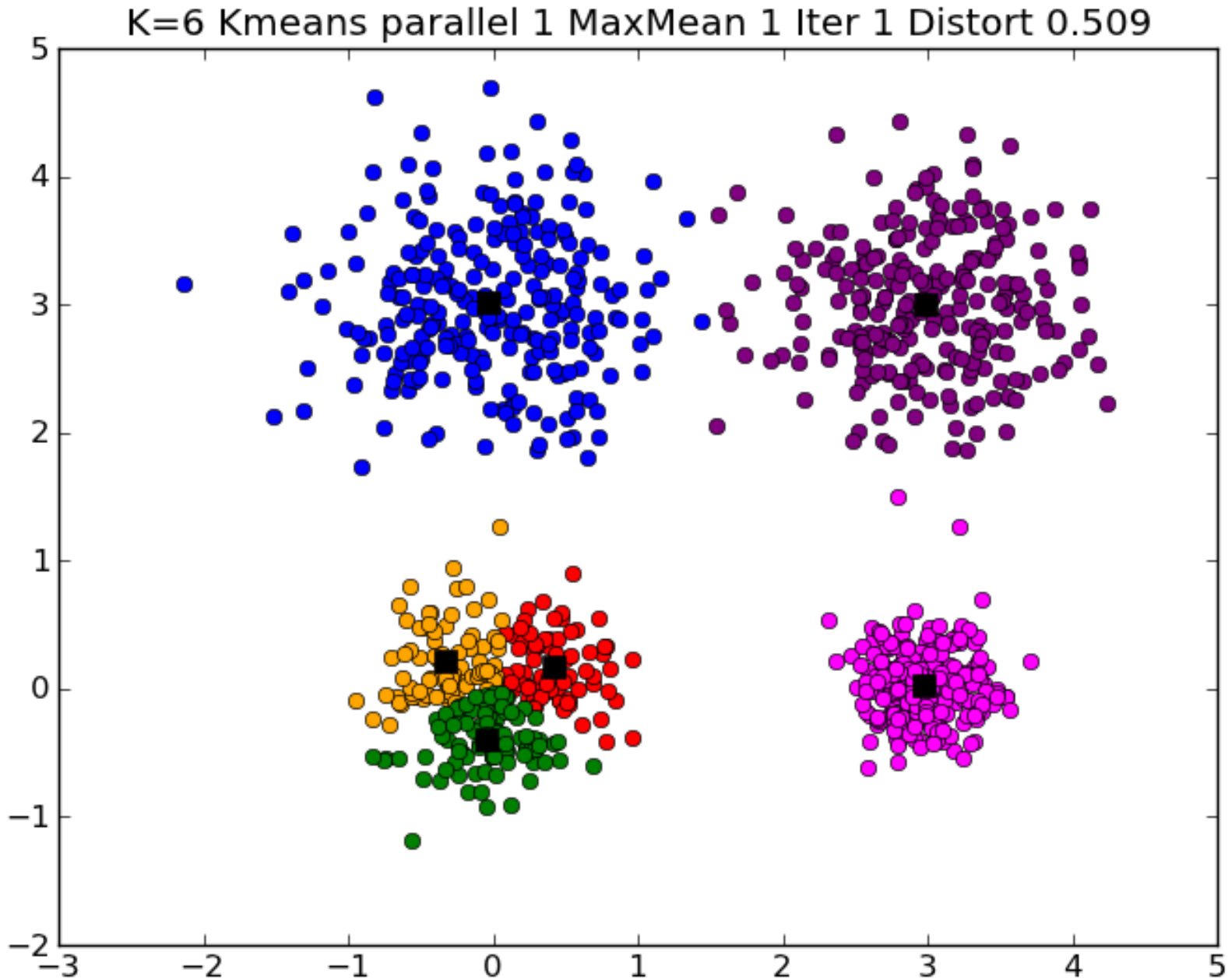
Messy K=4 large Clustering



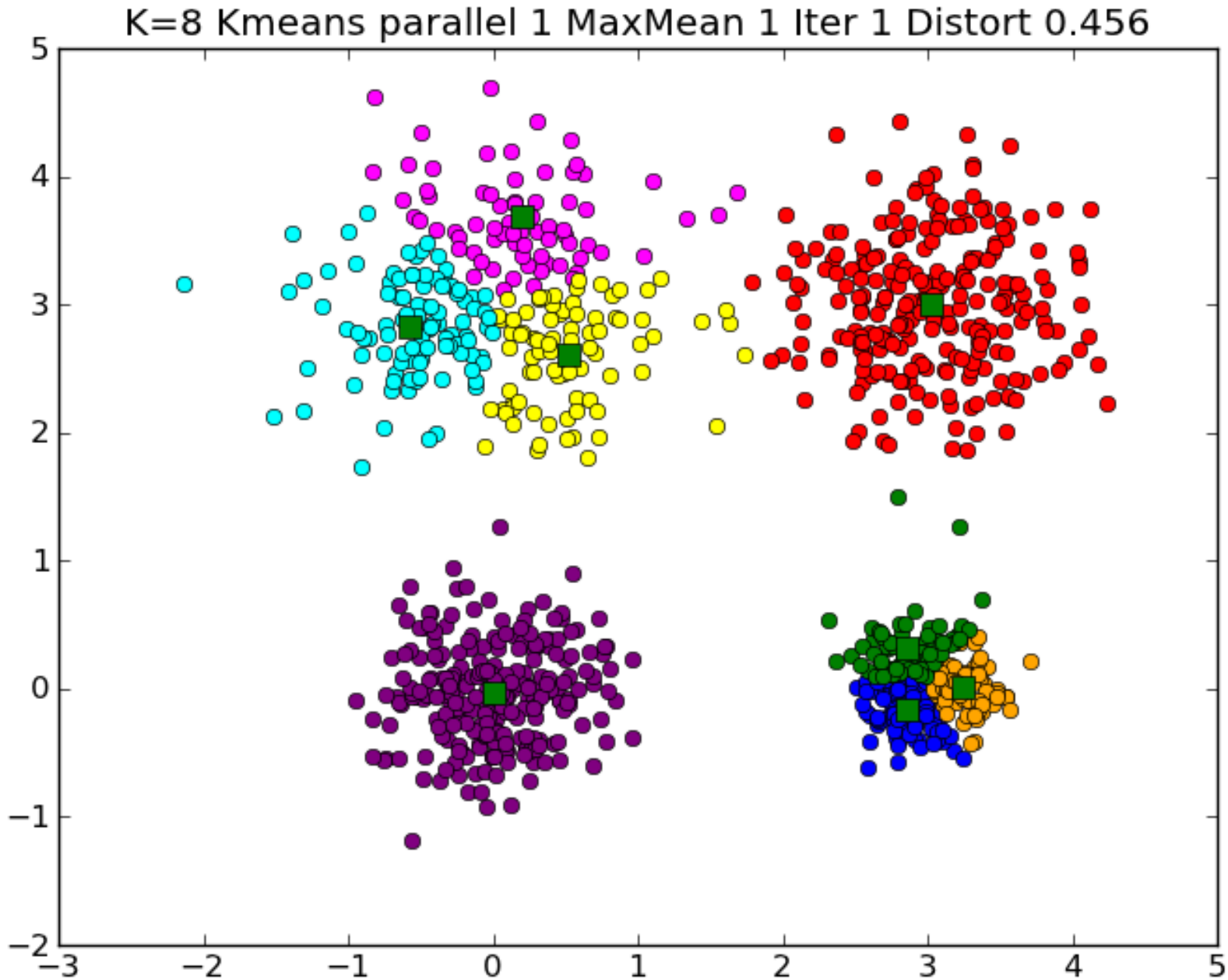
Messy K=4 Small Clustering



Messy K=2 Very large Clustering



Different K=6 Large Clustering



Another K=8 Large Clustering