K-Means Clustering

- Clustering is a form of unsupervised learning whereby a set of observations (i.e., data points) is partitioned into natural groupings or clusters of patterns in such a way that the measure of similarity between any pair of observations assigned to each cluster minimizes a specified cost function

- Let $\{\boldsymbol{x}_i\}_{i=1}^{N}$ denote a set of multidimensional observations that is to be partitioned into a proposed set of *K* clusters, where *K* is smaller than the number of observations, *N.*

Examples of Clustering Applications

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- Urban planning: Identifying groups of houses according to their house type, value, and geographical location

Examples of Clustering Applications

- Seismology: Observed earthquake epicenters should be clustered along continent faults
- Often used as an exploratory data analysis tool
- In one-dimension, a good way to quantize real-valued variables into K non-uniform buckets
- Used on acoustic data in speech understanding to convert waveforms into one of K categories (known as Vector Quantization)
- Also used for choosing color palettes on old fashioned graphical display devices!

What Is a Good Clustering?

- A good clustering method will produce clusters with
  - High <u>intra-class</u> similarity
  - Low <u>inter-class</u> similarity
- Precise definition of clustering quality is difficult
  - Application-dependent
  - Ultimately subjective

Requirements for Clustering in Data Mining

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal domain knowledge required to determine input parameters
- Ability to deal with noise and outliers
- Insensitivity to order of input records
- Robustness w.r.t. high dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

Similarity and Dissimilarity Between Objects

- Distance measure between instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

  Minkowski (L_p) (Euclidean for p = 2)

$$L_p = d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left[\sum_{m=1}^{l}\left(x_{im} - x_{jm}\right)^p\right]^{1/p}$$

  City-block distance
$$L_1 = d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left[\sum_{m=1}^{l}\left|x_{im} - x_{jm}\right|\right]$$

- Euclidean distance (p = 2):

$$L_2 = d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left[\sum_{m=1}^{l}\left(x_{im} - x_{jm}\right)^2\right]^{1/2}$$

- Properties of a metric $d(\mathbf{x}_i, \mathbf{x}_j)$:

  1) $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$　　2) $d(\mathbf{x}_i, \mathbf{x}_i) = 0$

  3) $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$　　*4)* $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k) + d(\mathbf{x}_k, \mathbf{x}_j)$

Major Clustering Approaches

- Partitioning: Construct various partitions and then evaluate them by some criterion
- Hierarchical: Create a hierarchical decomposition of the set of objects using some criterion
- Model-based: Hypothesize a model for each cluster and find best fit of models to data
- Density-based: Guided by connectivity and density functions

Partitioning Algorithms

- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of *K* clusters
- Given a *K*, find a partition of *K clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *K-means* and *K-medoids* algorithms
  - *K-means* (MacQueen, 1967): Each cluster is represented by the center of the cluster
  - *K-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw, 1987): Each cluster is represented by one of the objects in the cluster

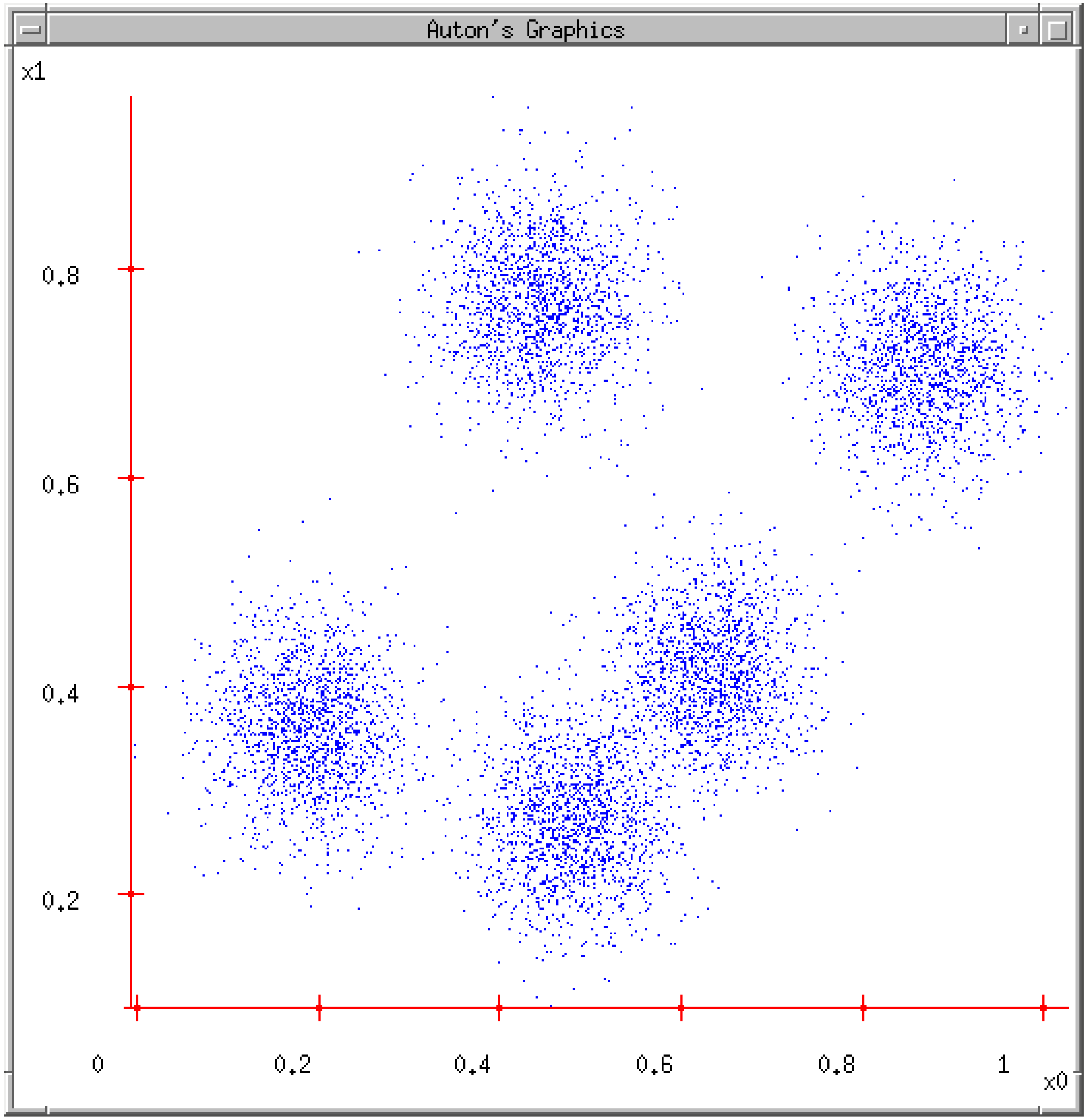
K-Means Clustering

- Given *K*, the *K-means* algorithm consists of four steps:
  - Select initial centroids at random.
  - Assign each object to the cluster with the nearest centroid.
  - Compute each centroid as the mean of the objects assigned to it.
  - Repeat previous 2 steps until no change.
- Optimal partition achieved via minimizing the sum of squared distance to its "representative object" in each cluster

$$E = \sum_{k=1}^{K} \sum_{\mathbf{x} \in \omega_k} d(\mathbf{x}, \boldsymbol{\mu}_k)$$

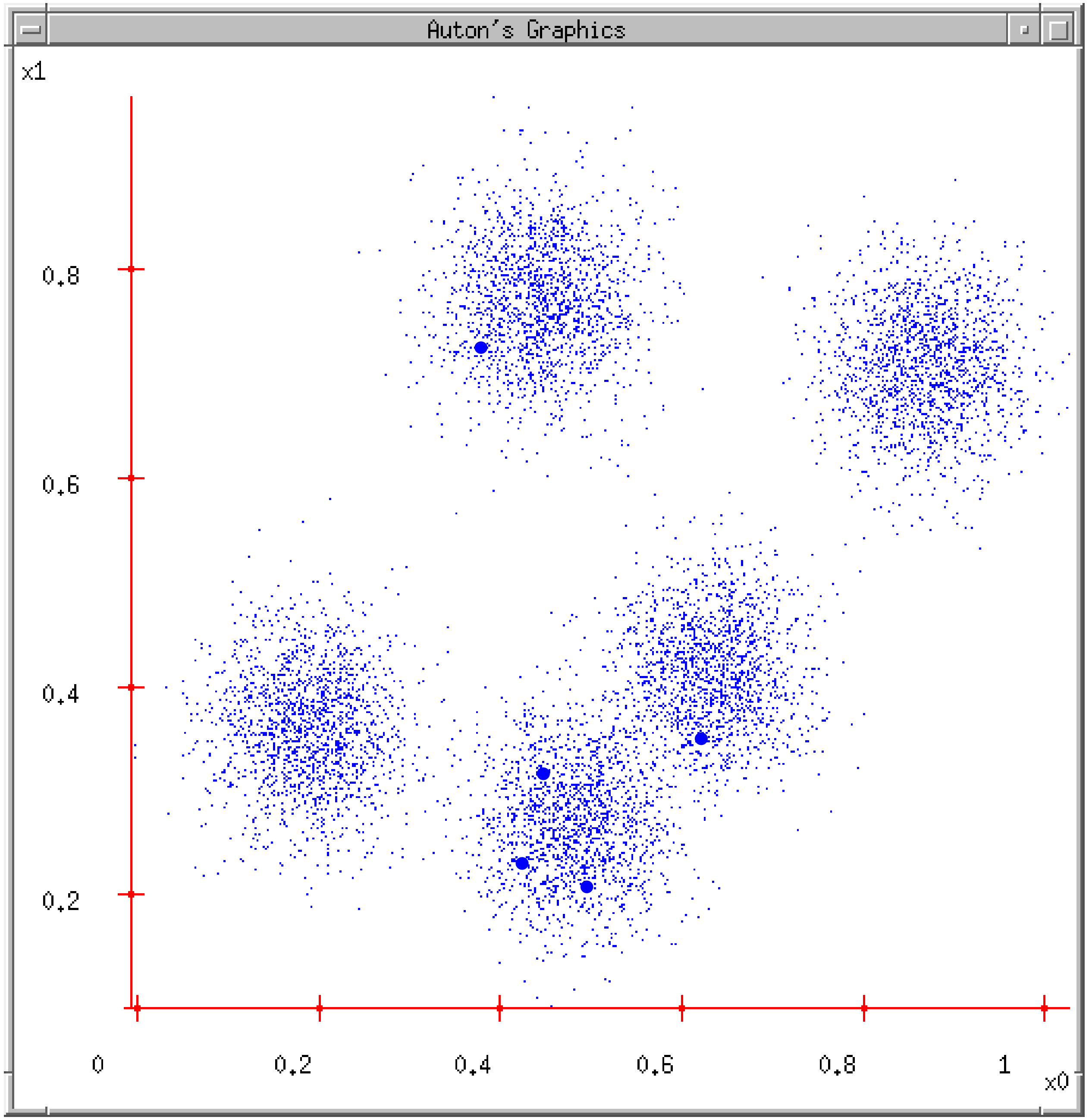
K-means

1. Ask user how many clusters they'd like. *(e.g. K=5)*



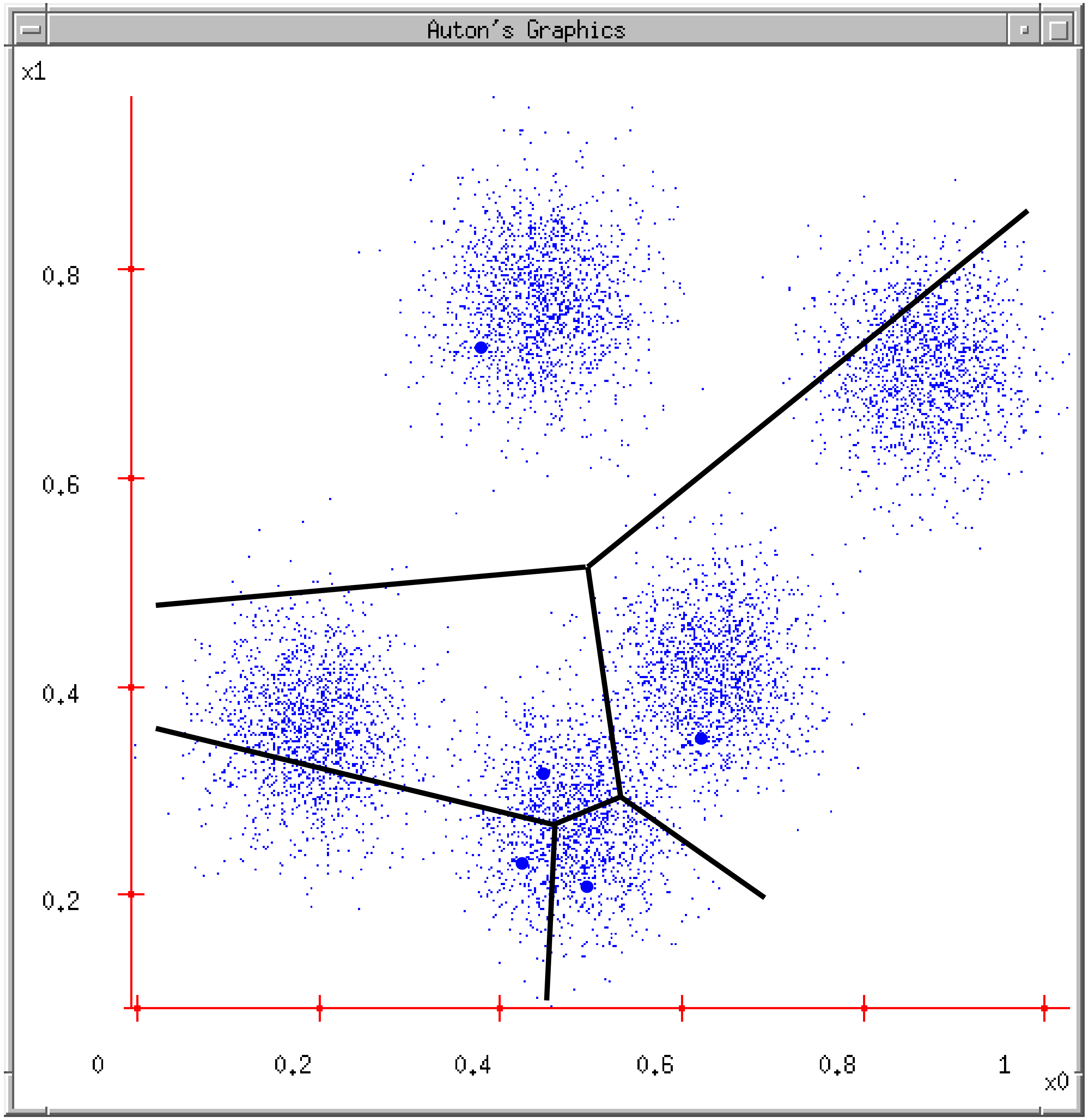
K-means

1. Ask user how many clusters they'd like. *(e.g. K=5)*
2. Randomly guess K cluster Center locations



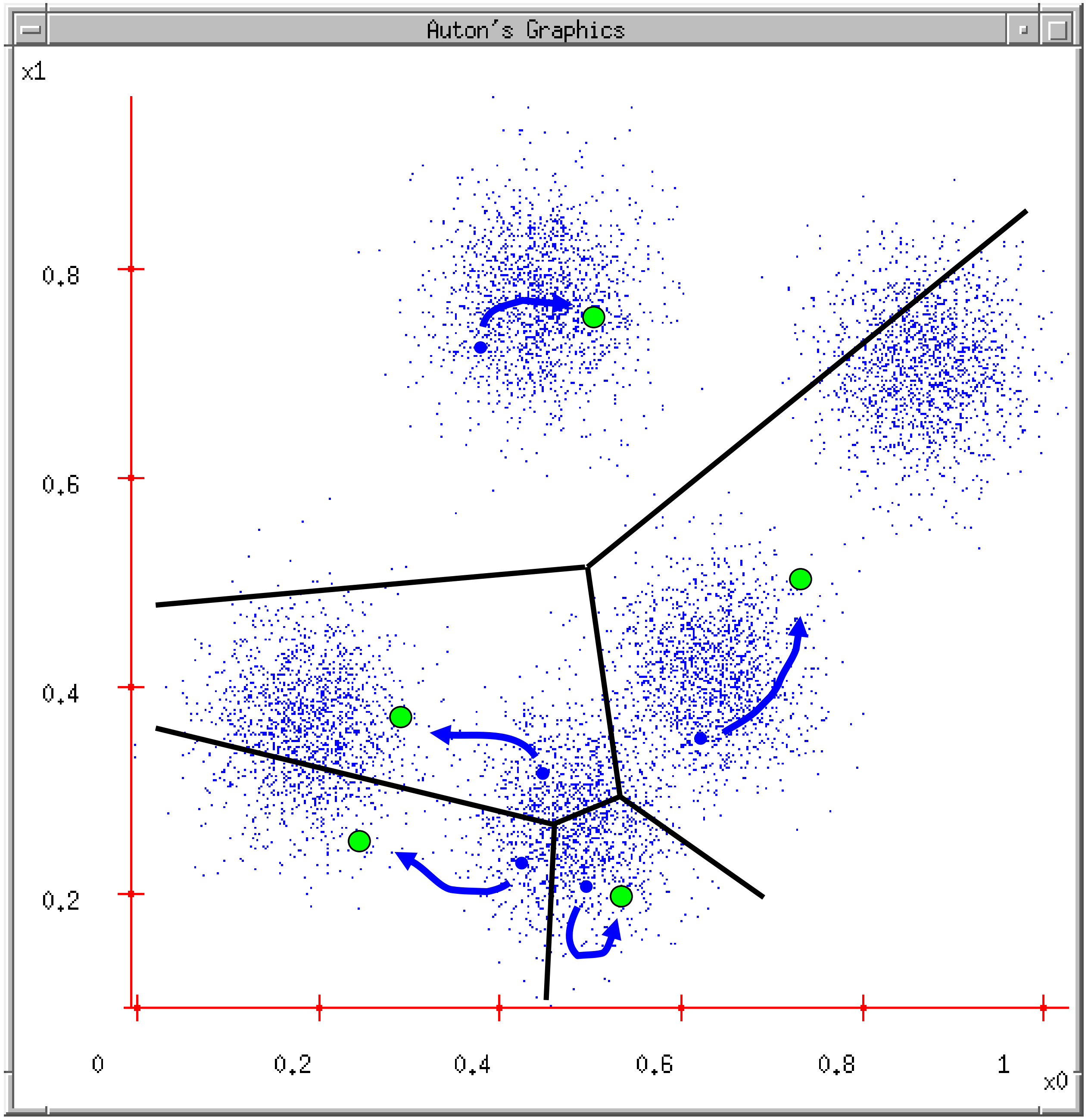
K-means

1. Ask user how many clusters they'd like. *(e.g. K=5)*
2. Randomly guess K cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



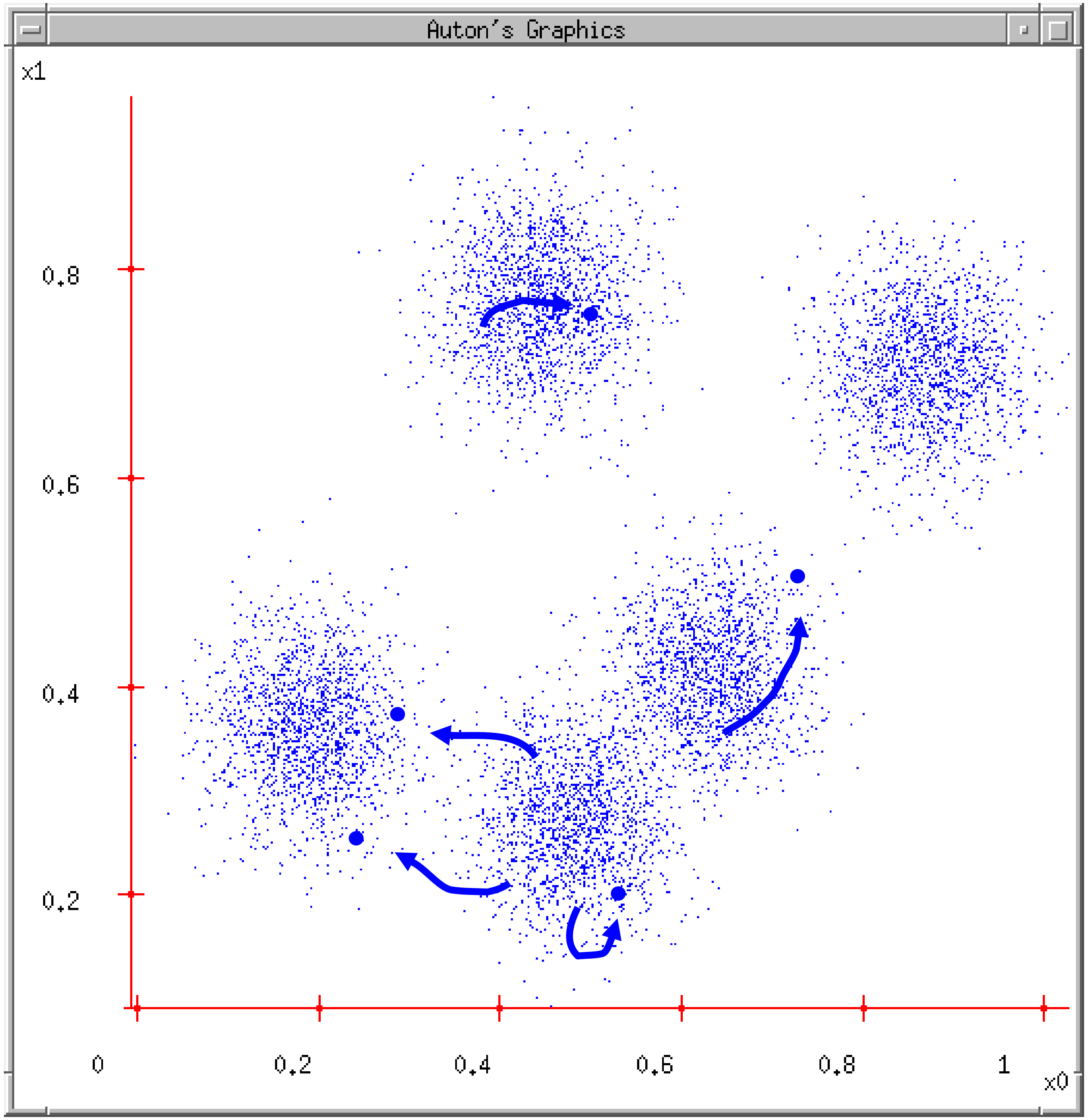
K-means

1. Ask user how many clusters they'd like. *(e.g. K=5)*
2. Randomly guess K cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



K-means

1. Ask user how many clusters they'd like. *(e.g. K=5)*
2. Randomly guess K cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

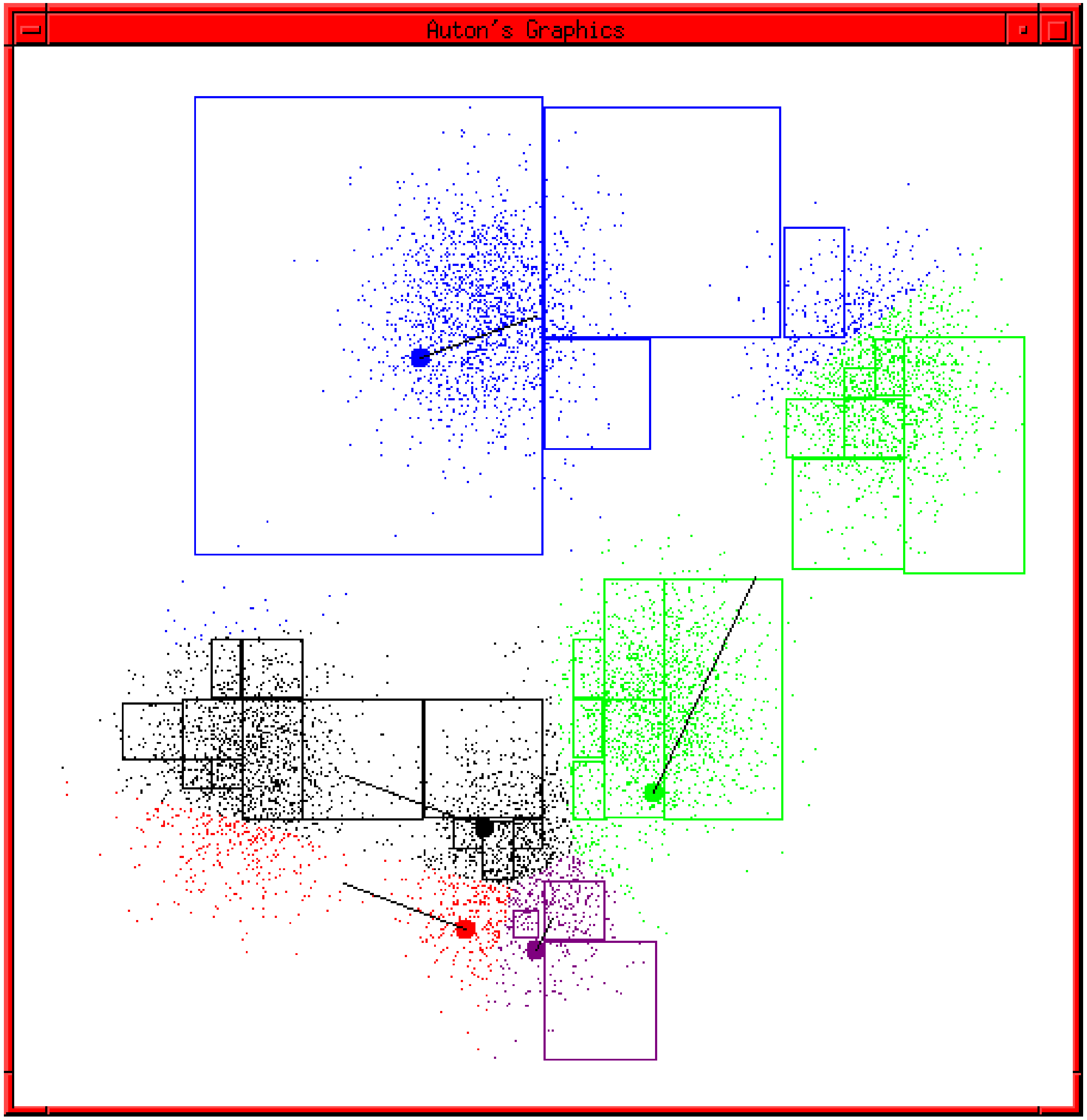


K-means Start

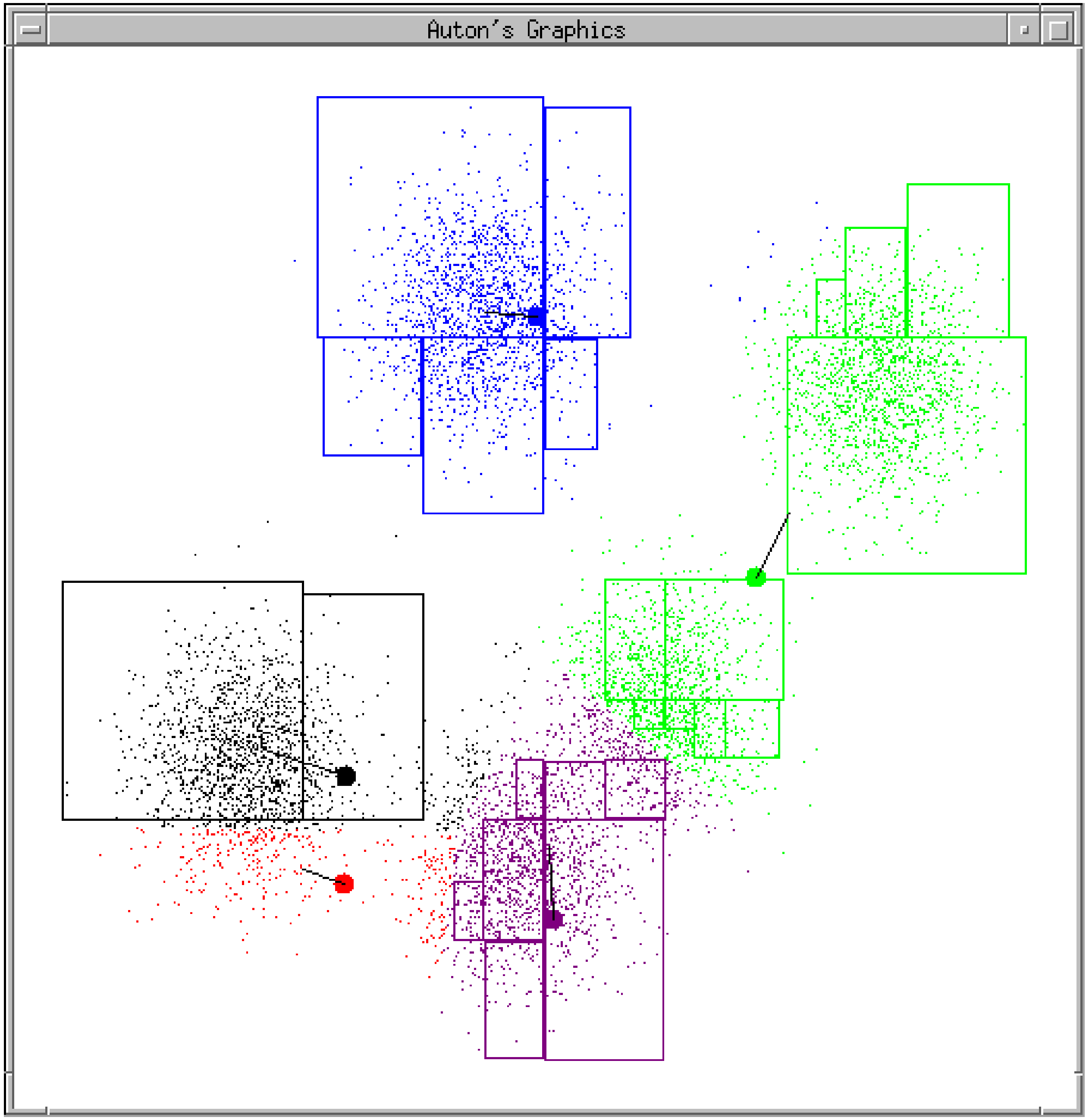
Advance apologies: in Black and White this example will deteriorate

Example generated by Dan Pelleg's super-duper fast K-means system:

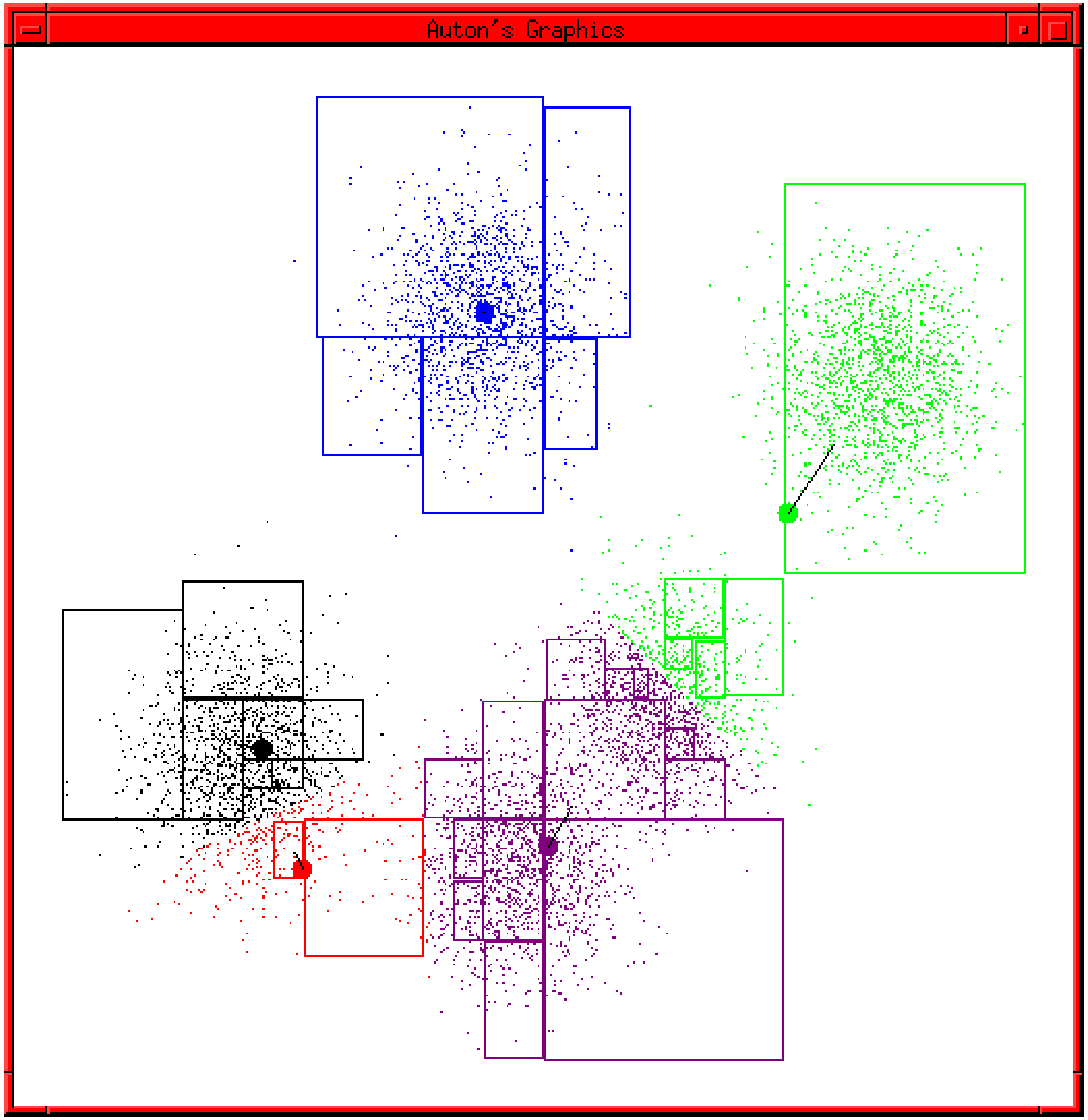
> *Dan Pelleg and Andrew Moore. Accelerating Exact k-means Algorithms with Geometric Reasoning. Proc. Conference on Knowledge Discovery in Databases 1999, (KDD99) (available on* www.autonlab.org/pap.html*)*



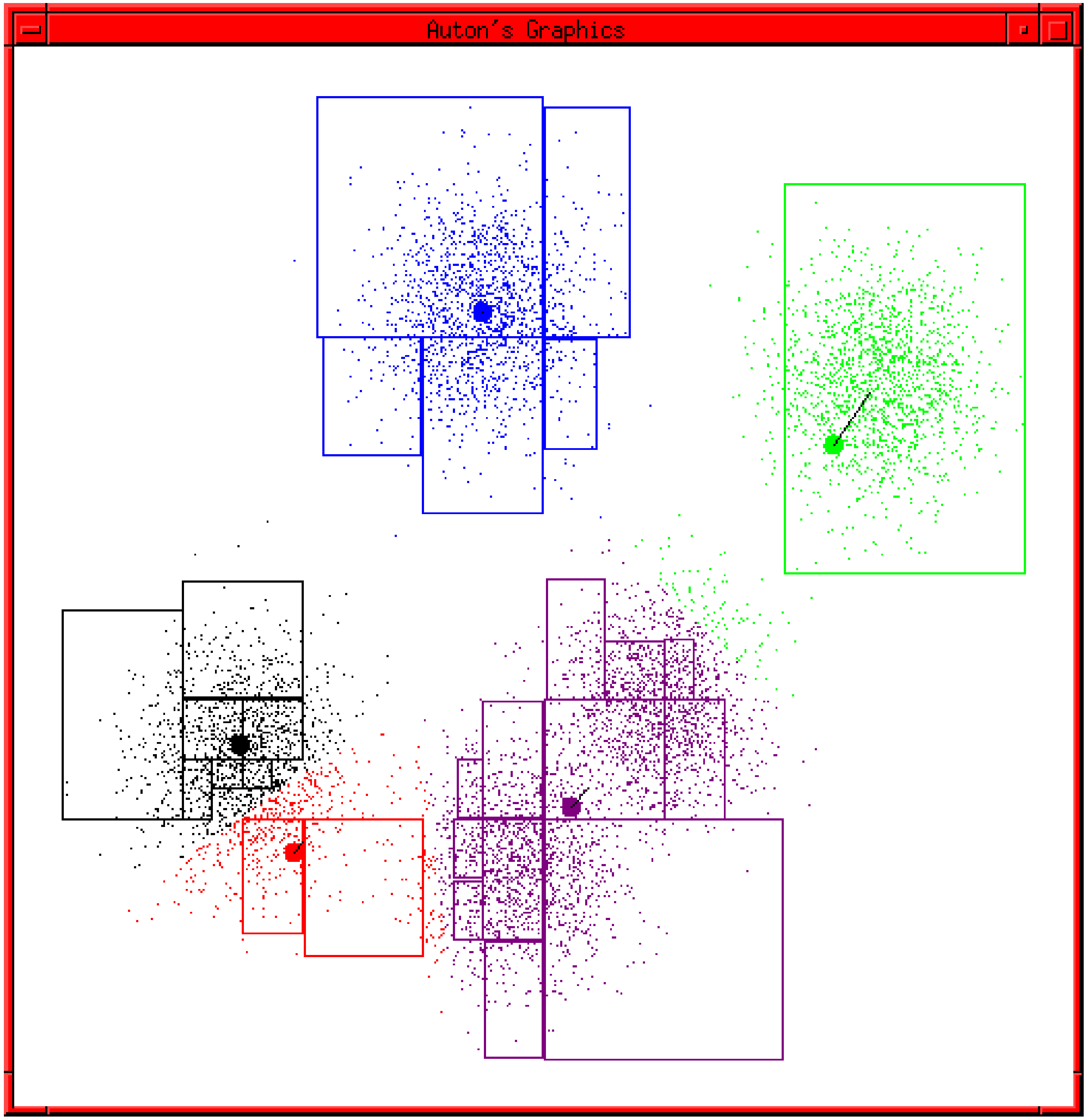
K-means continues …



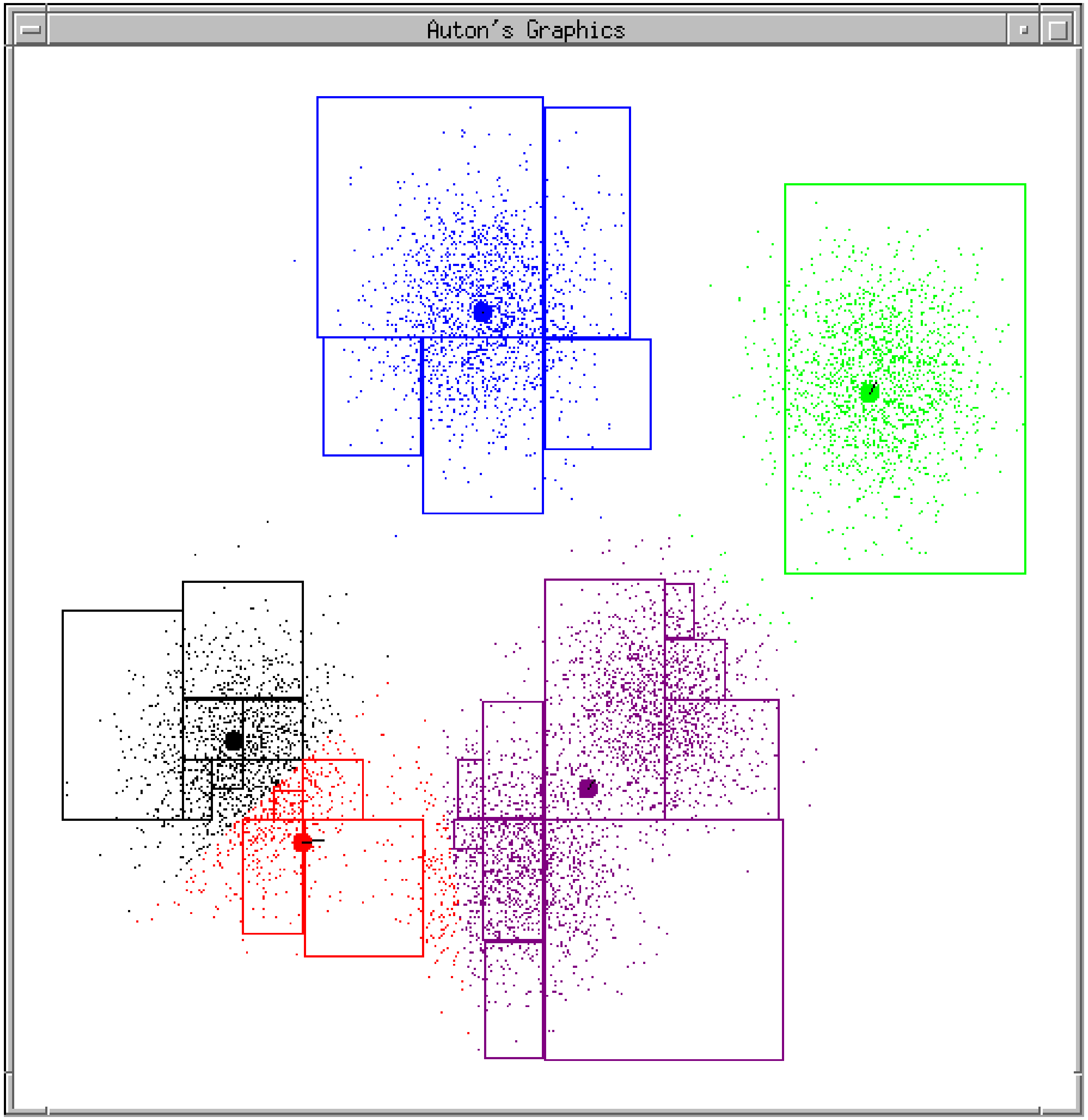
K-means continues …



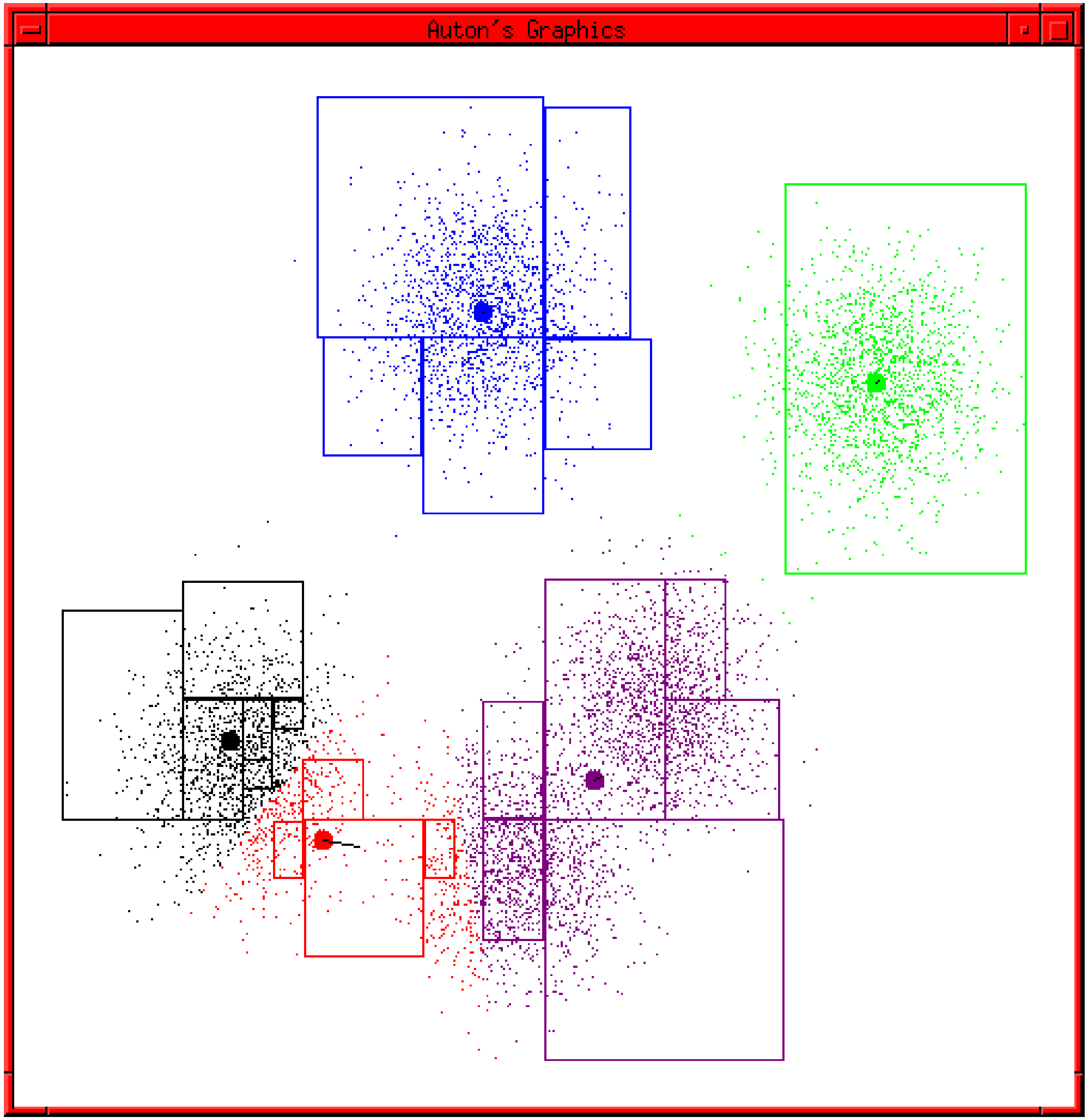
K-means continues …



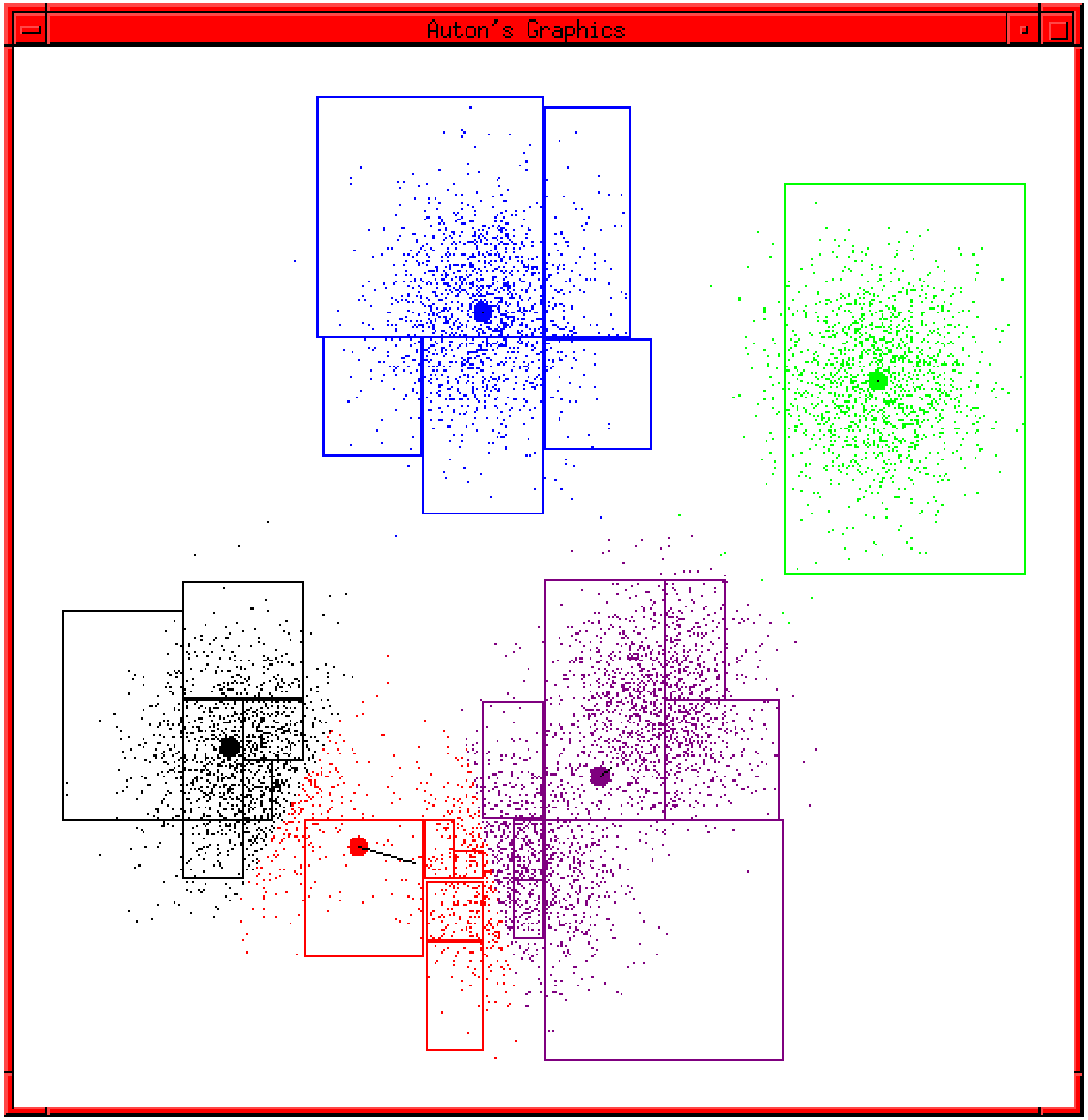
K-means continues …



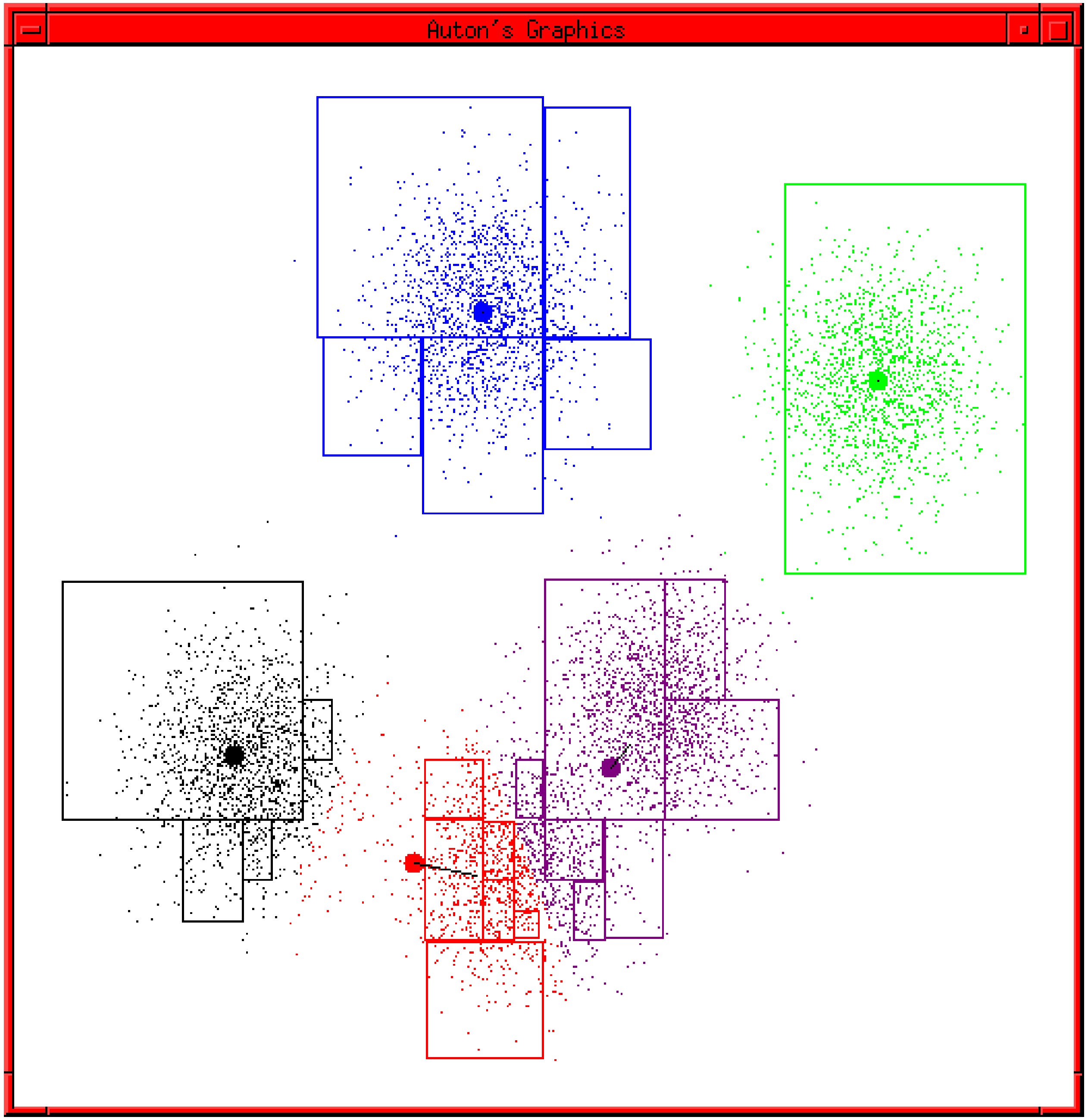
K-means continues …



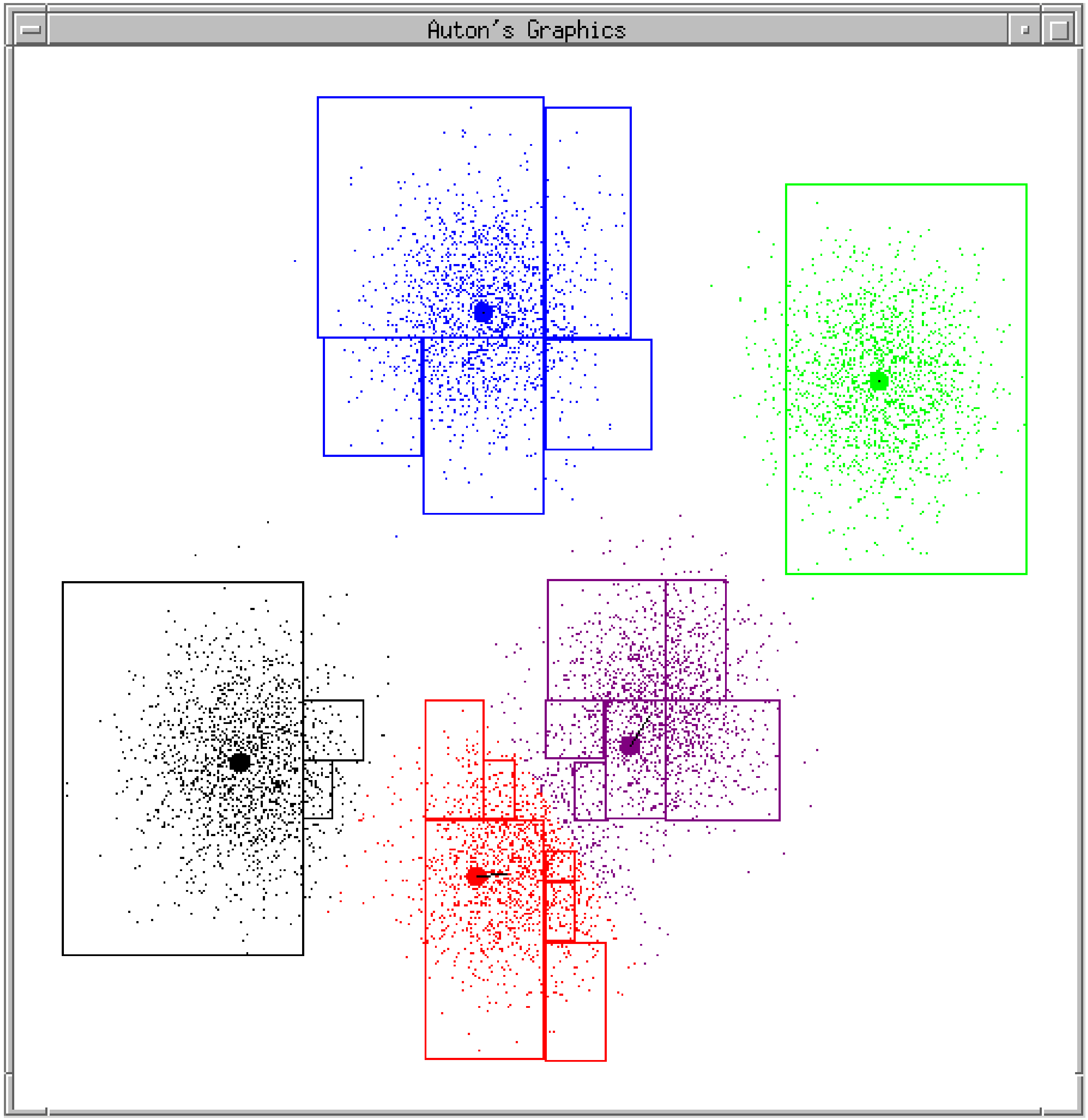
K-means continues …



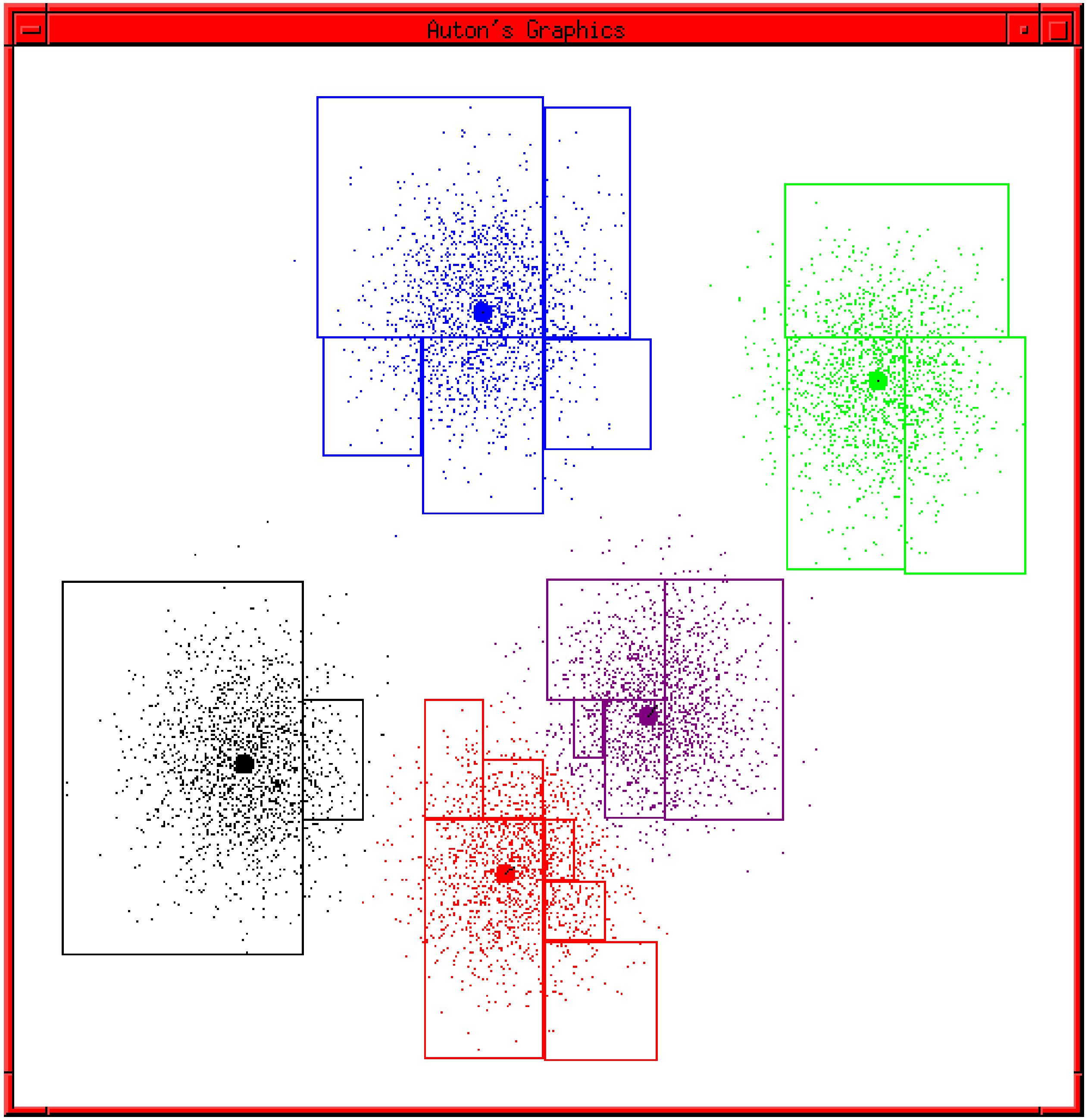
K-means continues …



K-means continues …



K-means terminates



– If n is the known number of patterns and K the desired number of clusters, the K-means algorithm is (K samples randomly chosen from the dataset as initial cluster centers):

```
Begin
    initialize $n$, $K$, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$(randomly selected)
        do classify $n$ samples according to nearest $\boldsymbol{\mu}_i$
            recompute $\boldsymbol{\mu}_i$
        until no change in $\boldsymbol{\mu}_i$
    return $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$
End
```

Exercise 2 p.594 (Textbook)

Example:

The 25 samples shown in the table were drawn sequentially from the following mixture with $\mu_1 = -2$ and $\mu_2 = 2$.

$$p(x|\mu_1,\mu_2) = \underbrace{\frac{1}{3\sqrt{2\pi}}\exp\left[-\frac{1}{2}(x-\mu_1)^2\right]}_{\omega_1} + \underbrace{\frac{2}{3\sqrt{2\pi}}\exp\left[-\frac{1}{2}(x-\mu_2)^2\right]}_{\omega_2},$$

k	x_k	ω_1	ω_2
1	0.608		×
2	-1.590	×	
3	0.235		×
4	3.949		×
5	-2.249	×	
6	2.704		×
7	-2.473	×	
8	0.672		×

k	x_k	ω_1	ω_2
9	0.262		×
10	1.072		×
11	-1.773	×	
12	0.537		×
13	3.240		×
14	2.400		×
15	-2.499	×	
16	2.608		×

k	x_k	ω_1	ω_2
17	-3.458	×	
18	0.257		×
19	2.569		×
20	1.415		×
21	1.410		×
22	-2.653	×	
23	1.396		×
24	3.286		×
25	-0.712	×	

- Considering the example in the previous figure

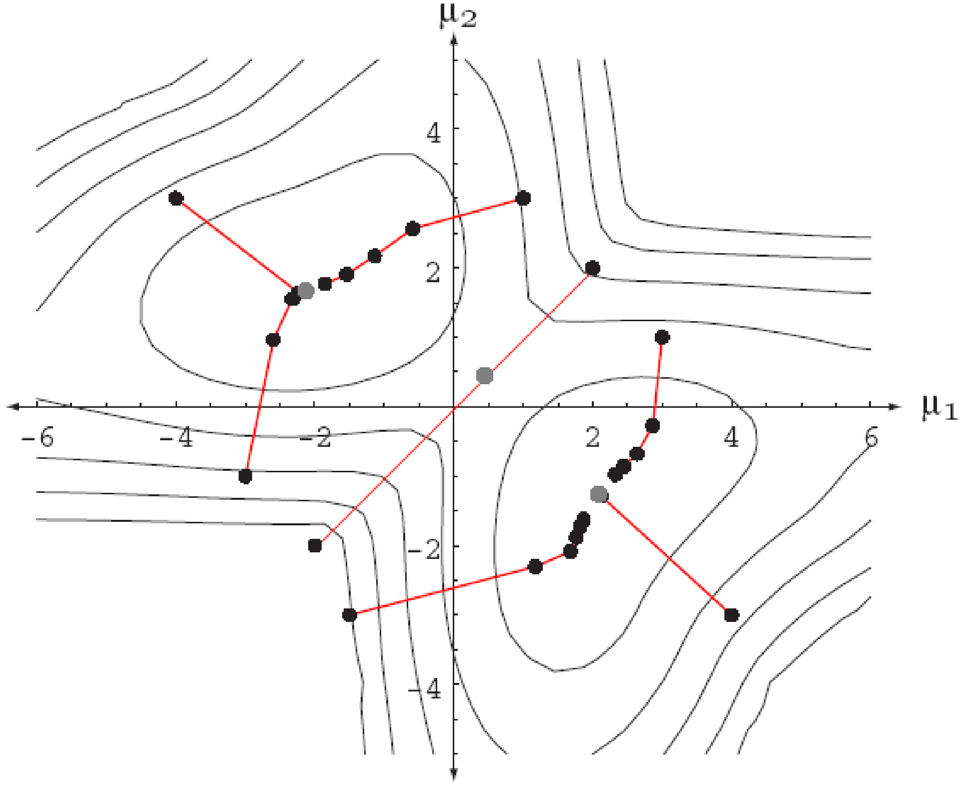
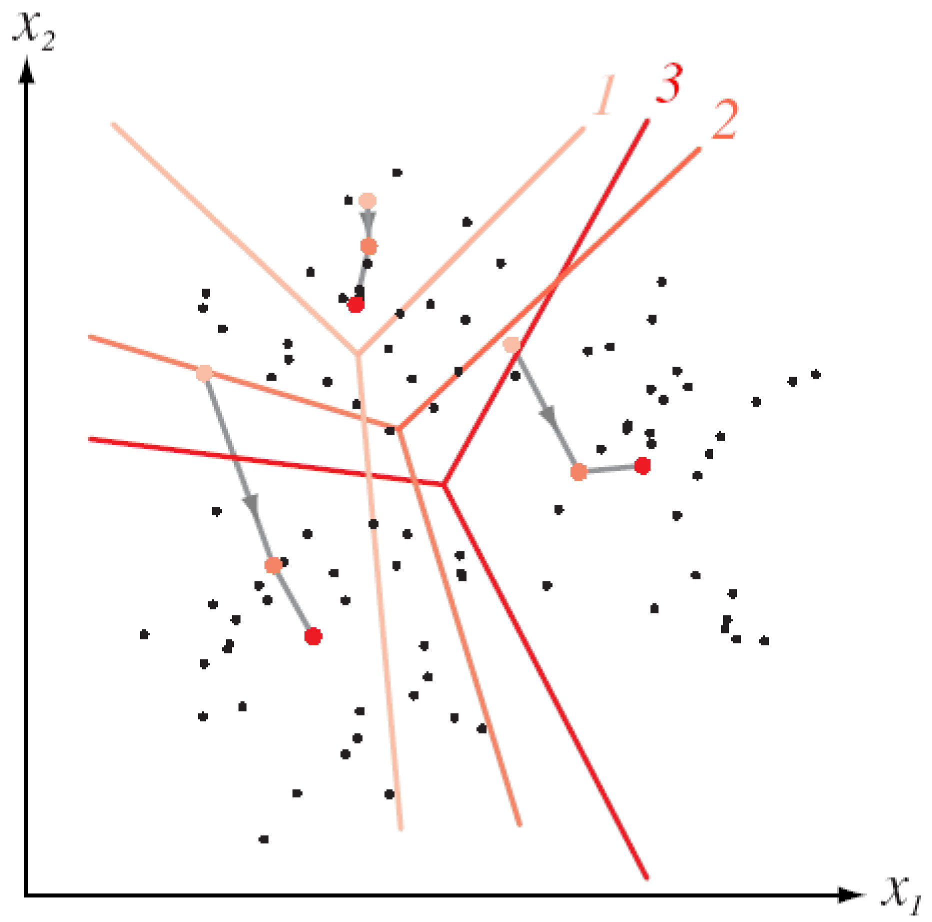


Figure 10.1: The *K*-means clustering procedure is a form of stochastic hill climbing in the log-likelihood function. The contours represent equal log-likelihood values for the one-dimensional data in Example 1. The dots indicate parameter values after different iterations of the *K*-means algorithm. Six of the starting points shown lead to local maxima, whereas two (i.e., $\mu_1(0) = \mu_2(0)$) *lead to a saddle point near* $\boldsymbol{\mu} = \mathbf{0}$.

- Figure 10.1 shows the sequence of values for $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_2$ obtained for several different starting points. Since interchanging $\hat{\boldsymbol{\mu}}_1$and $\hat{\boldsymbol{\mu}}_2$ merely interchanges the labels assigned to the data, the trajectories are symmetric about the line $\hat{\boldsymbol{\mu}}_1 = \hat{\boldsymbol{\mu}}_2$. The trajectories lead either to the point $\hat{\boldsymbol{\mu}}_1 = -2.176, \hat{\boldsymbol{\mu}}_2 = 1.684$ or to its symmetric image. This is close to the solution found by the maximum-likelihood method (viz., $\hat{\boldsymbol{\mu}}_1 = -2.130$ and $\hat{\boldsymbol{\mu}}_2 = 1.688$), and the trajectories show a general resemblance to those shown in Example 1.



K=3

The three initial cluster centers, chosen randomly from the training points.

FIGURE 10.3. Trajectories for the means of the K-means clustering procedure applied to two-dimensional data. The final Voronoi tesselation (for classification) is also shown— the means correspond to the "centers" of the Voronoi cells. In this case, convergence is obtained in three iterations.

Comments on the *K-Means* Method

- Strengths
  - *Relatively efficient*: $O(tKn)$, where n is # objects, K is #clusters, and t is #iterations. Normally, K, $t << n$.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as *simulated annealing* and *genetic algorithms*
- Weaknesses
  - Applicable only when *mean* is defined (what about categorical data?)
  - Need to specify K, the *number* of clusters, in advance
  - Trouble with noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Fuzzy *K*-means clustering

- In every iteration of the classical K-means procedure, each data point is assumed to be in exactly one cluster
- We can relax this condition and assume that each sample $\mathbf{x}_\mathrm{j}$ has some graded or "fuzzy" cluster membership $\mu_i(\mathbf{x}_j)$ in cluster ω_i, where $0 \leq \mu_i(\mathbf{x}_j) \leq 1$.
- At root, these "memberships" are equivalent to the probabilities $\hat{P}(\omega_i \mid \mathbf{x}_j, \hat{\boldsymbol{\theta}})$
- In the resulting fuzzy K-means clustering algorithm we seek a minimum of a global cost function

$$L = \sum_{i=1}^{c} \sum_{j=1}^{n} [\hat{P}(\omega_i|\mathbf{x}_j, \hat{\theta})]^b ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2,$$

where b is a free parameter chosen to adjust the "blending" of different clusters. If b is set to 0, this criterion function is merely a sum-of-squared errors criterion.

If $b > 1$, criterion allows each pattern to belong to multiple clusters. The probabilities of cluster membership for each point are normalized as

$$\sum_{i=1}^{c} \hat{P}(\omega_i|\mathbf{x}_j) = 1, \qquad j = 1, \ldots, n. \qquad (25)$$

At the solution, i.e., the minimum of L, we have

$$\partial L/\partial \boldsymbol{\mu}_i = 0 \quad \text{and} \quad \partial L/\partial \hat{P}_j = 0,$$

Then we have

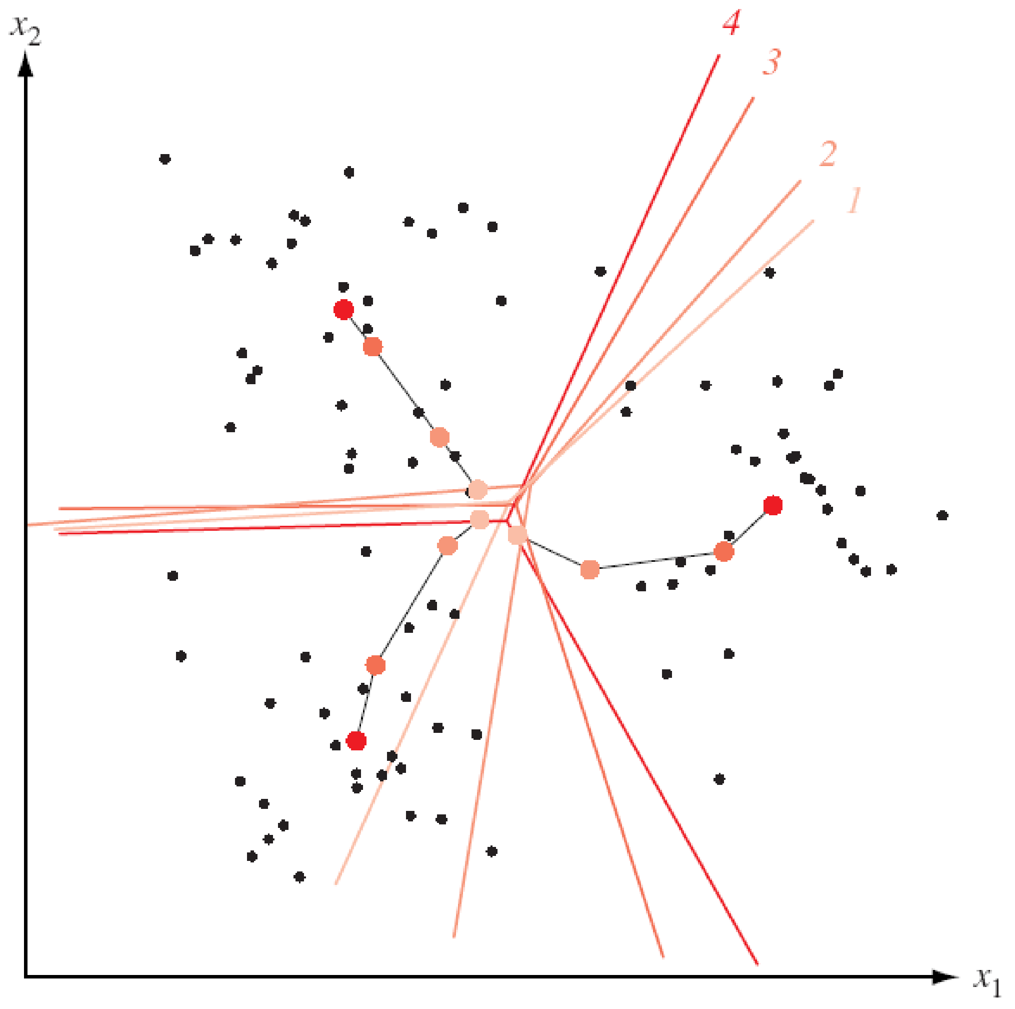
$$\boldsymbol{\mu}_j = \frac{\sum_{j=1}^{n} [P(\omega_i|\mathbf{x}_j)]^b \mathbf{x}_j}{\sum_{j=1}^{n} [P(\omega_i|\mathbf{x}_j)]^b} \tag{27}$$

and

$$P(\omega_i|\mathbf{x}_j) = \frac{(1/d_{ij})^{1/(b-1)}}{\sum_{r=1}^{c} (1/d_{rj})^{1/(b-1)}}, \qquad d_{ij} = ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2. \tag{28}$$

Algorithm 2 (Fuzzy K-means clustering)

```
1 begin initialize $n, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, P(\omega_i \mid \mathbf{x}_j), i = 1 \ldots, c;\ j = 1, \ldots, n$
2           normalize proabilities of cluster memberships by Eq. 25
3           do classify $n$ samples according to nearest $\boldsymbol{\mu}_i$
4               recompute $\boldsymbol{\mu}_i$ by Eq. 27
5               recompute $P(\omega_i \mid \mathbf{x}_j)$ by Eq. 28
6           until no change in $\boldsymbol{\mu}_i$ and $P(\omega_i \mid \mathbf{x}_j)$
7     return $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
8 end
```



At early iterations the means lie near the center of the full data set because each point has a non-negligible “membership” (i.e., probability) in each cluster. At later iterations the means separate and each membership tends toward the value 1.0 or 0.0.

The classical *K*-means algorithm is just of special case where the memberships for all points obey

$$P(\omega_i|\mathbf{x}_j) = \begin{cases} 1 & \text{if } \|\mathbf{x}_j - \boldsymbol{\mu}_i\| < \|\mathbf{x}_j - \boldsymbol{\mu}_{i'}\| \text{ for all } i' \neq i \\ 0 & \text{otherwise,} \end{cases}$$

Application

- Colour-Based Image Segmentation Using *K*-means

Step 1: Loading a colour image of tissue stained with hemotoxylin and eosin (H&E)

H&E image

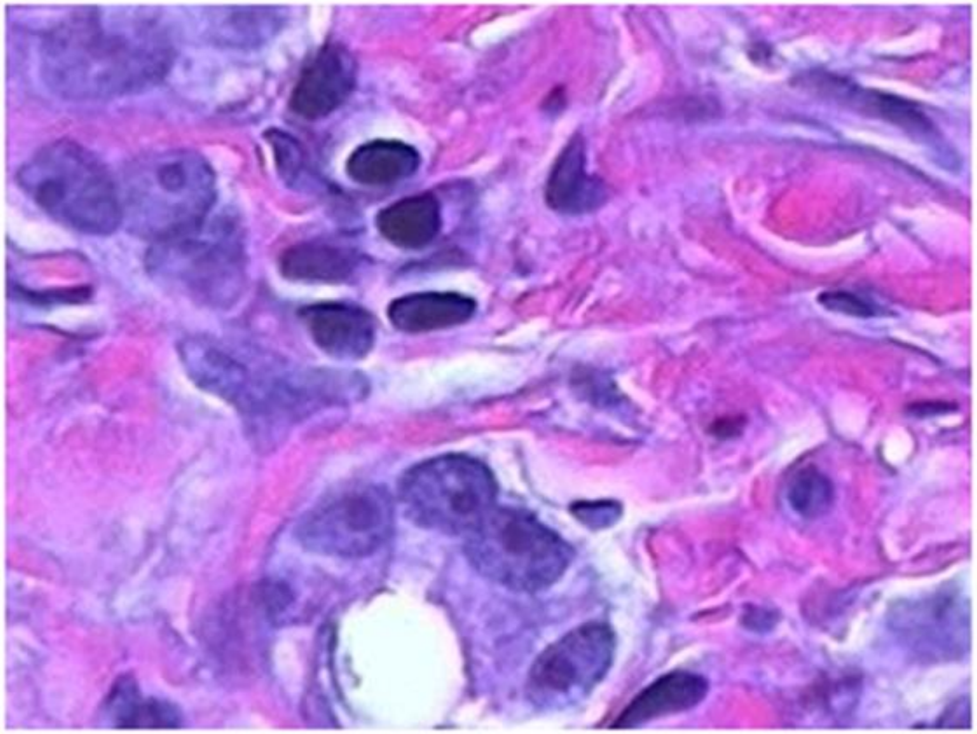


Image courtesy of Alan Partin, Johns Hopkins University

Application

- Colour-Based Image Segmentation Using *K*-means

Step 2: Convert the image from RGB colour space to L*a*b* colour space

- Unlike the RGB colour model, L*a*b* colour is designed to approximate human vision.
- There is a complicated transformation between RGB and L*a*b*.

$$(L^*, a^*, b^*) = T(R, G, B).$$

$$(R, G, B) = T'(L^*, a^*, b^*).$$

Application

- Colour-Based Image Segmentation Using *K*-means

  Step 3: Undertake clustering analysis in the (a*, b*) colour space with the *K*-means algorithm

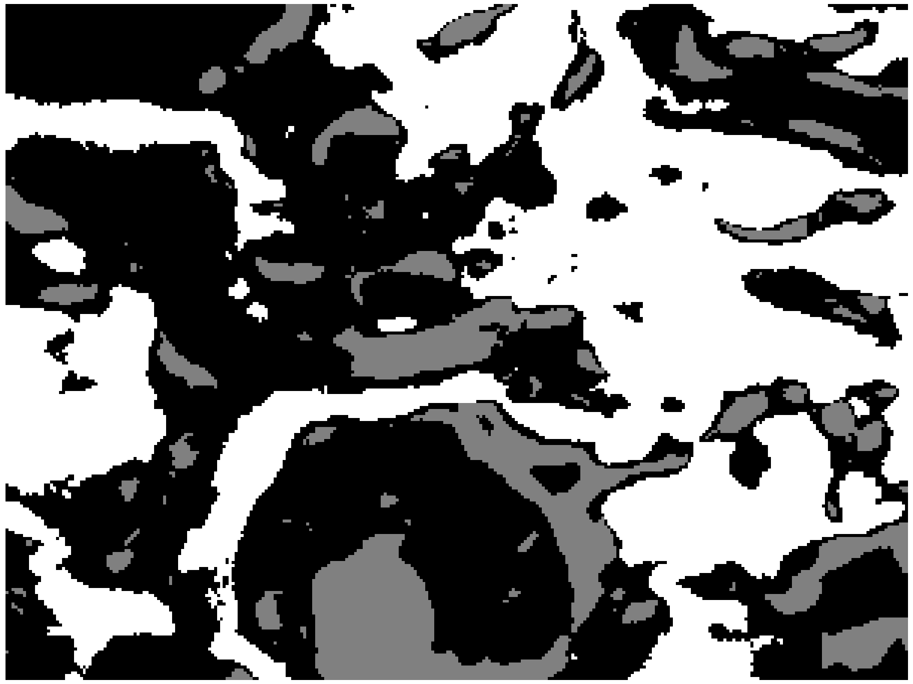
  - In the L*a*b* colour space, each pixel has a properties or feature vector: (L*, a*, b*).
  - Like feature selection, L* feature is discarded. As a result, each pixel has a feature vector (a*, b*).
  - Applying the *K*-means algorithm to the image in the a*b* feature space where $K = 3$ by applying the domain knowledge.

Application

- Colour-Based Image Segmentation Using *K*-means

  Step 4: Label every pixel in the image using the results from *K*-means clustering (indicated by three different grey levels)

image labeled by cluster index

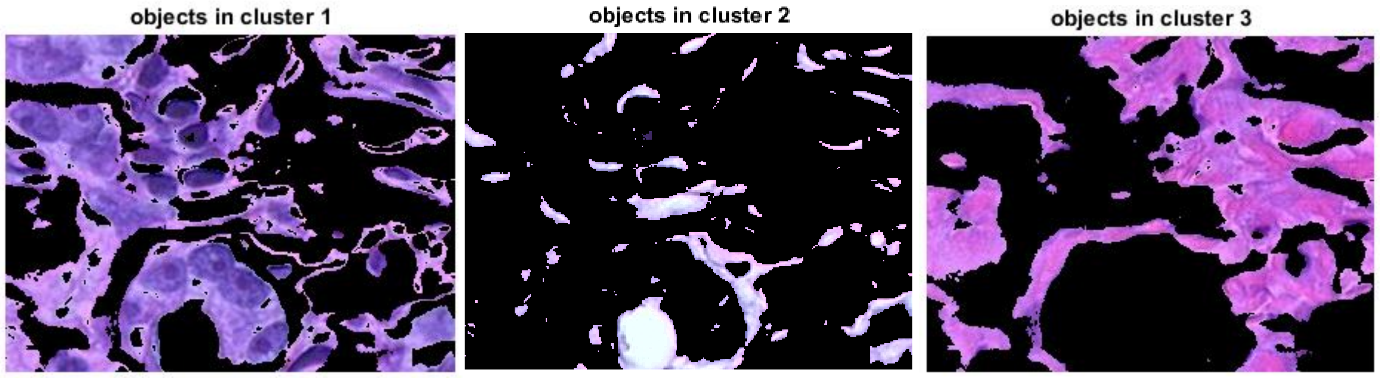


Application

- Colour-Based Image Segmentation Using *K*-means

Step 5: Create Images that Segment the H&E Image by Colour

- Apply the label and the colour information of each pixel to achieve separate colour images corresponding to three clusters.



"blue" pixels "white" pixels "pink" pixels

Application

- Colour-Based Image Segmentation Using *K*-means

Step 6: Segment the nuclei into a separate image with the L* feature

- In cluster 1, there are dark and light blue objects (pixels). The dark blue objects (pixels) correspond to nuclei (with the domain knowledge).
- L* feature specifies the brightness values of each colour.
- With a threshold for L*, we achieve an image containing the nuclei only.