This lecture is partially based on an excellent set of lecture slides by Andrew Moore, available at: http://www.cs.cmu.edu/~awm/tutorials

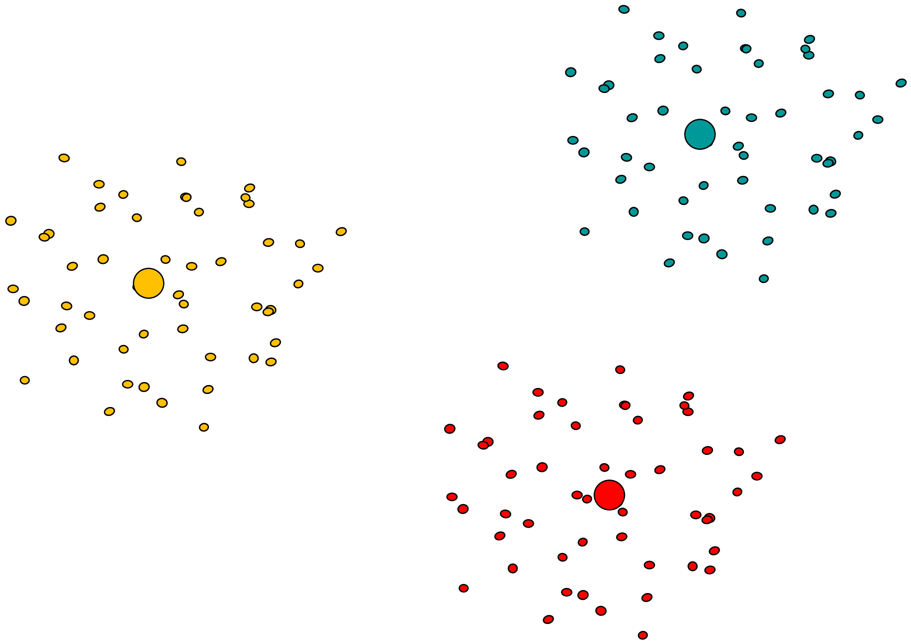
Machine Learning for Cities
CUSP-GX 5006.001, Spring 2019

Lecture 6: More Clustering Algorithms

Recap from last week

In the last lecture, we saw how Bayesian methods (Naïve Bayes/EM) could be used for a range of applications, including **supervised learning** (classification), **semi-supervised learning** (classification with labeled and unlabeled data), and **unsupervised learning** (clustering).

Today we will take a step back to better understand clustering- what it is and why it is useful- and examine a variety of other clustering methods.



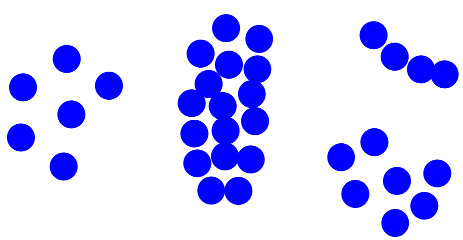
Clustering data

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

Given a population of individuals, we want to **identify** and **characterize** the underlying subgroups (or “clusters”) of individuals, and possibly use these subgroups for **prediction**.

We want to explain the data in terms of its natural groupings.

How many groups are there?
Who belongs to which group?
Characteristics of each group?



2-D example

Example: identify congressional voting blocs.

How well do blocs correspond to party affiliation?
Are there relevant blocs within a party?
Are there “mavericks” that vote across party lines?
How much do blocs vary with proposed legislation?

Clustering data

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

Given a population of individuals, we want to **identify** and **characterize** the underlying subgroups (or “clusters”) of individuals, and possibly use these subgroups for **prediction**.

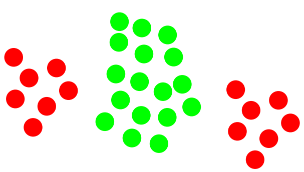
How does clustering differ from prediction?

There may not be any single output that we are trying to predict.

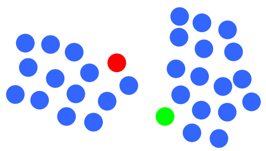
We are interested in an underlying **structure** that explains or predicts many characteristics of the population.

Clustering can sometimes improve prediction performance.

Improve model-based classification by learning multiple models per class.



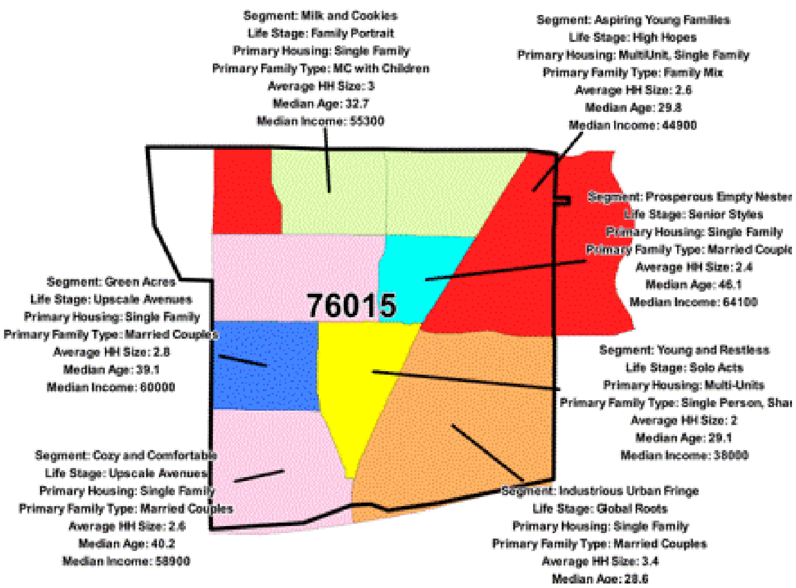
We may have little or no labeled training data for learning class models, but lots of unlabeled data.



Applications of clustering

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

An important application of clustering is to improve **prediction** of an individual's characteristics or behavior, using information obtained from other members of their inferred group.



Customer database segmentation: To which subgroups should I market my product, and how should I target them? (Similarly, voter database segmentation)

Patient database segmentation: Different subgroups of patients may benefit from different treatment regimens.

Applications of clustering

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

Group-based trajectory modeling identifies subgroups of the population, and uses these subgroups to predict how an individual's behavior will change over the course of his/her life.

Daniel Nagin (CMU) developed these methods and applied them to **predict** juvenile delinquency and criminal behavior.

This figure shows the predicted and actual trajectories of an individual's number of criminal convinctions, varying with age. Three trajectory groups were identified: never (71%), limited to adolescence (22%), and chronic offenders (7%).

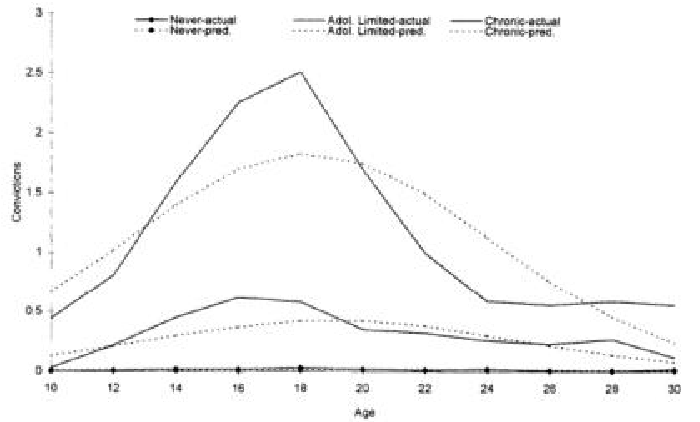


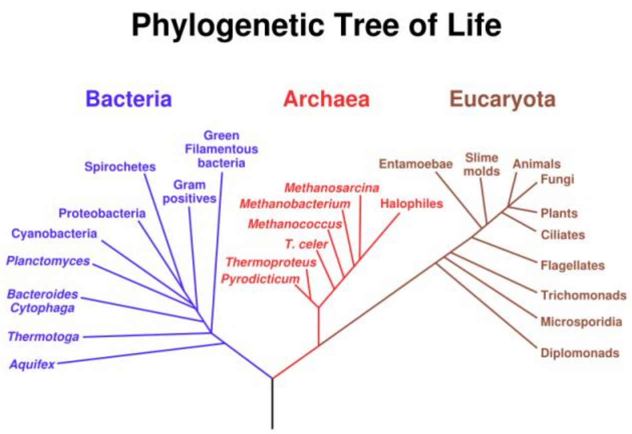
Figure 1. Trajectories of number of convictions (Cambridge sample). Adol. = adolescent; pred. = predicted.

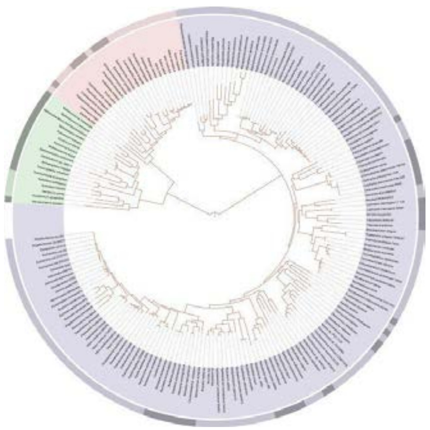
Nagin, D. S. 1999. "Analyzing Developmental Trajectories: A Semi-parametric, Group-based Approach." *Psychological Methods*, 4: 139-177.

Applications of clustering

<u>Main goal</u>: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

Clustering is also very commonly used in **evolutionary biology** to create "phylogenetic trees" relating different species and showing when the species diverged from a common ancestor.



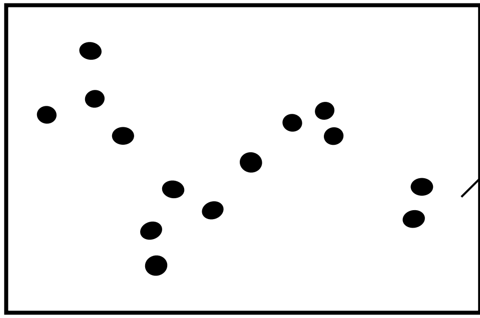


Here we are interested in learning the **hierarchy** of clusters!

Here is a simple phylogenetic tree developed manually by evolutionary biologists.

Now much more detailed trees can be generated automatically by clustering DNA sequences, enhancing our understanding of evolution.

Hierarchical clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define “nearest” clusters?

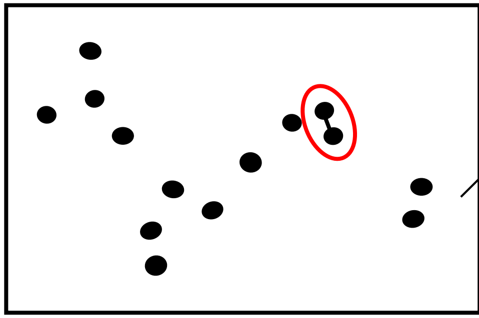
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

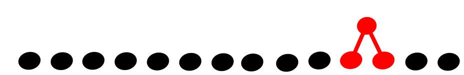
$$D(C,C') = \min_{x \in C,\, x' \in C'} d(x,x')$$

Single-link clustering

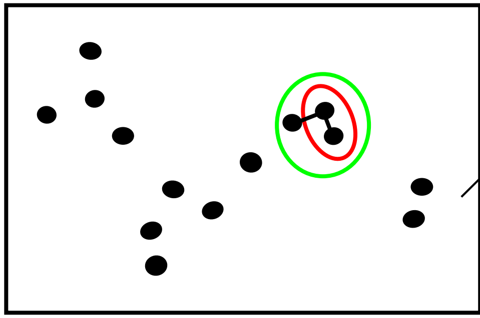
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 1 merge)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

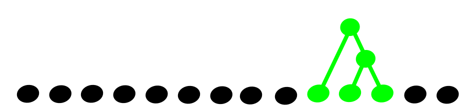
$$D(C,C') = \min_{x \in C,\, x' \in C'} d(x,x')$$

Single-link clustering

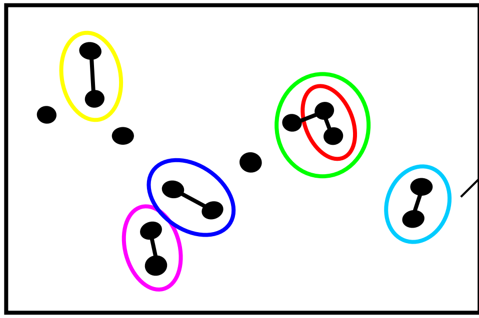
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 2 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define "nearest" clusters?

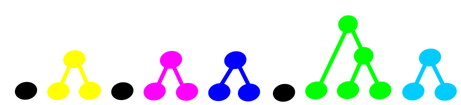
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

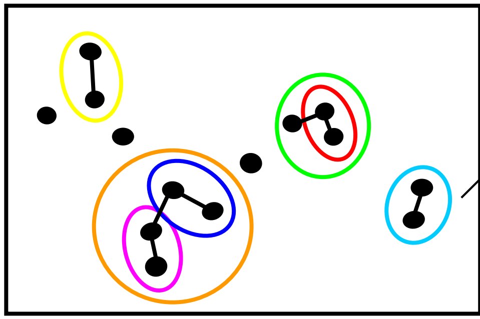
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 6 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

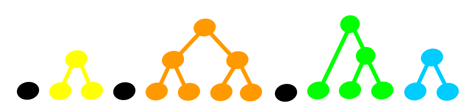
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

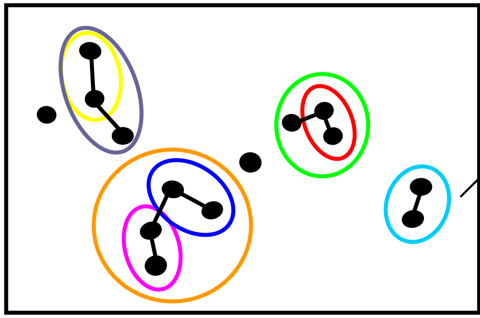
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 7 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define “nearest” clusters?

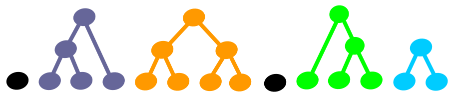
$$D(C,C') = \min_{x \in C,\, x' \in C'} d(x,x')$$

Single-link clustering

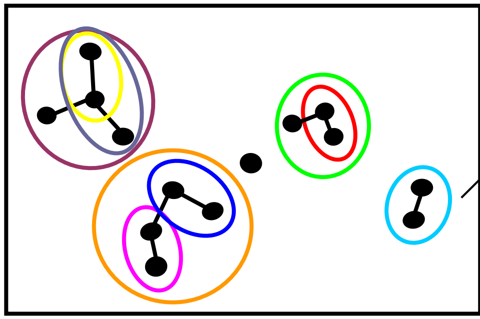
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 8 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

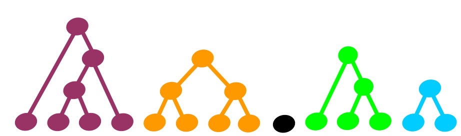
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

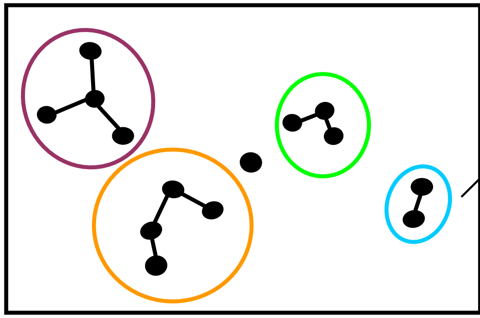
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 9 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

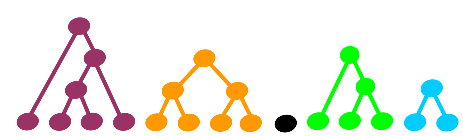
$$D(C, C') = \min_{x \in C,\, x' \in C'} d(x, x')$$

Single-link clustering

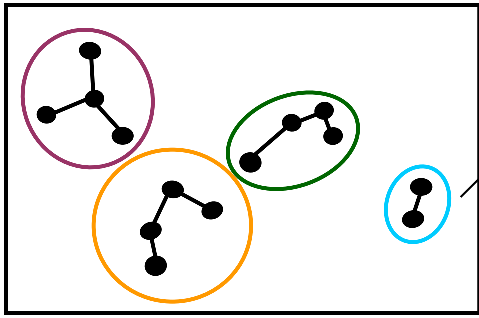
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 9 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define "nearest" clusters?

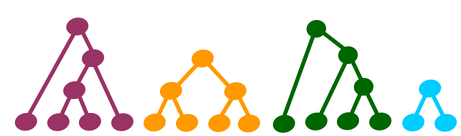
$$D(C, C') = \min_{x \in C,\, x' \in C'} d(x, x')$$

Single-link clustering

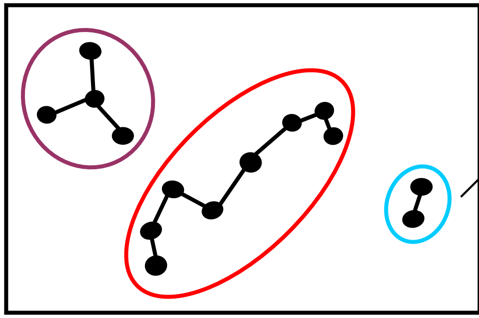
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 10 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define "nearest" clusters?

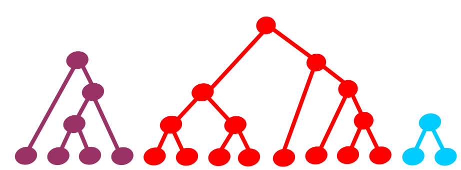
$$D(C,C') = \min_{x \in C,\, x' \in C'} d(x,x')$$

Single-link clustering

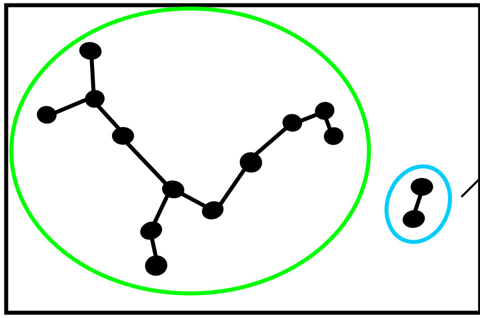
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 11 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define "nearest" clusters?

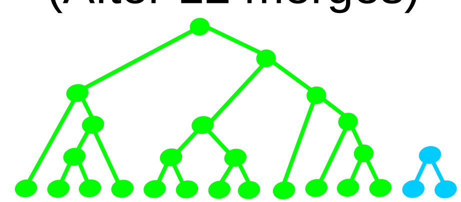
$$D(C, C') = \min_{x \in C,\, x' \in C'} d(x, x')$$

Single-link clustering

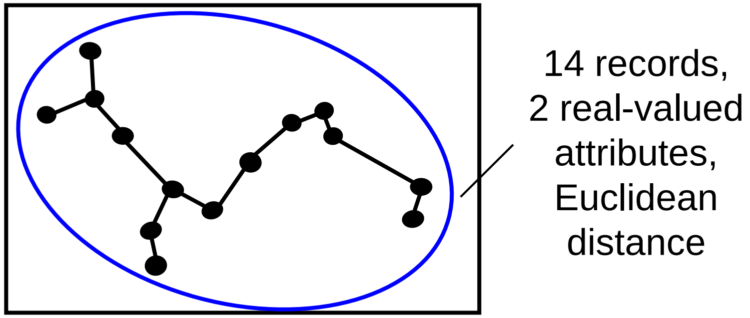
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 12 merges)



Single-link clustering



Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

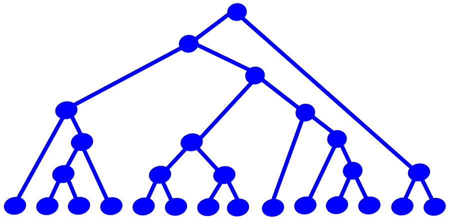
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

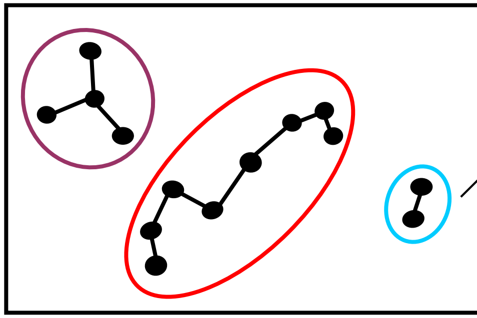
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Done!



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

$$D(C,C') = \min_{x \in C,\, x' \in C'} d(x,x')$$

Single-link clustering

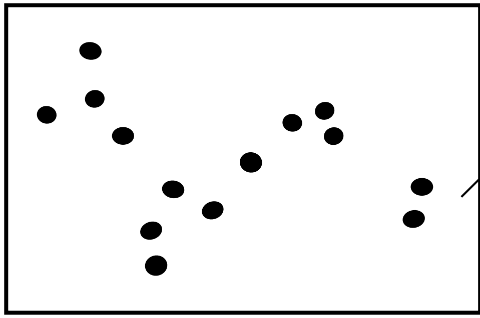
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or “chains”) as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C, C') = \max_{x \in C,\, x' \in C'} d(x, x')$$

Complete-link clustering

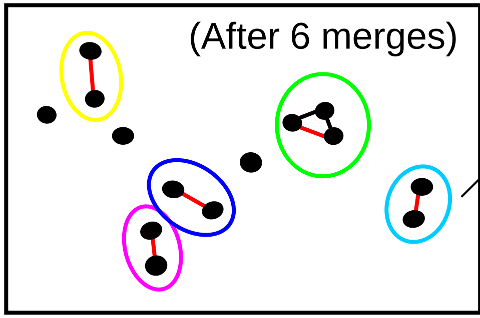
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define "nearest" clusters?

$$D(C, C') = \max_{x \in C,\, x' \in C'} d(x, x')$$

Complete-link clustering

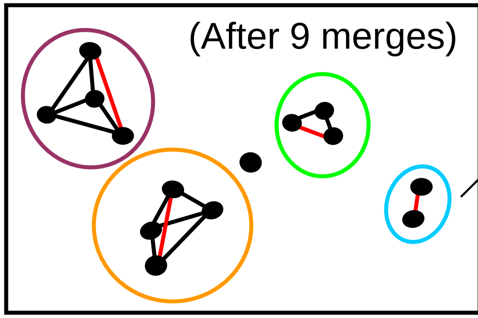
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C,C') = \max_{x \in C,\, x' \in C'} d(x,x')$$

Complete-link clustering

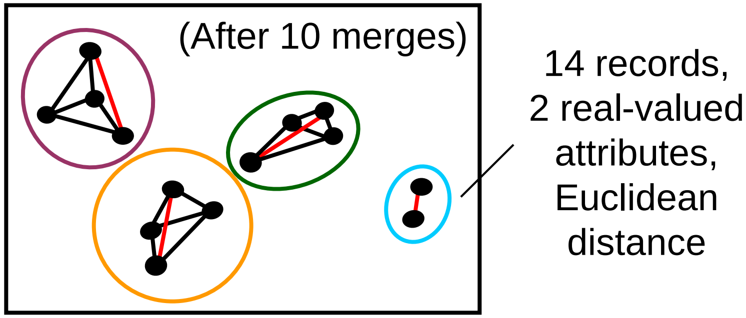
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

$$D(C,C') = \max_{x \in C,\, x' \in C'} d(x,x')$$

Complete-link clustering

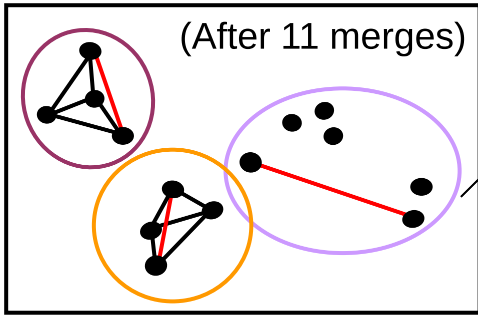
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or “chains”) as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define “nearest” clusters?

$$D(C,C') = \max_{x \in C,\, x' \in C'} d(x,x')$$

Complete-link clustering

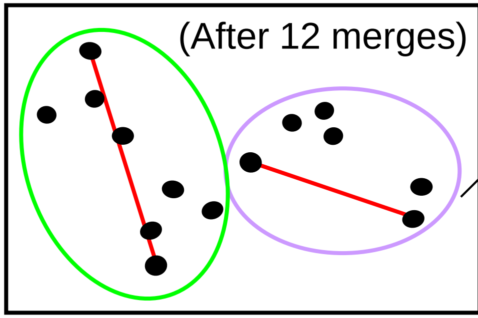
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or “chains”) as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C, C') = \max_{x \in C,\, x' \in C'} d(x, x')$$

Complete-link clustering

<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Hierarchical clustering

<u>Top-down hierarchical clustering</u>

- Start with one cluster containing all N records.
- Choose the "worst" cluster, and optimally partition it into two distinct clusters.
- Repeat until all points are in separate clusters.

<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Top-down clustering is hard, so we focus on the bottom-up approach.

Some fancy hierarchical clustering algorithms alternate bottom-up and top-down stages.

<u>Advantages of hierarchical clustering</u>

You get an entire cluster hierarchy, not just a single clustering.

Easy to compare different numbers of clusters k: just cut the longest k-1 links and optimize some measure of "goodness" (more on that later).

(Any disadvantages?)

K-means clustering

Hierarchical clustering is a simple, useful clustering method, but it gives you no guarantees on the quality of the resulting groups.

An alternative is to define some objective function that describes the quality of the groups, and attempt to optimize that function.

Assume that all attributes are real-valued, so we can compute the centroid of any cluster (i.e. the mean value of each attribute)

Possible objective:
minimize distortion

$$\sum_{C_k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

This is the sum of squared errors for each data point x_i, assuming that each x_i is mapped to the closest cluster center μ_k.

Possible moves for state-space search
Change position of cluster center μ_k
Change mapping of point x_i to center μ_k
Change number of clusters K

How to perform this search efficiently?

K-means clustering

Hierarchical clustering is a simple, useful clustering method, but it gives you no guarantees on the quality of the resulting groups.

An alternative is to define some objective function that describes the quality of the groups, and attempt to optimize that function.

Assume that all attributes are real-valued, so we can compute the centroid of any cluster (i.e. the mean value of each attribute)

Possible objective:
minimize distortion

$$\sum_{C_k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

This is the sum of squared errors for each data point x_i, assuming that each x_i is mapped to the closest cluster center μ_k.

Possible moves for state-space search

Change position of cluster center μ_k

Change mapping of point x_i to center μ_k

Change number of clusters K

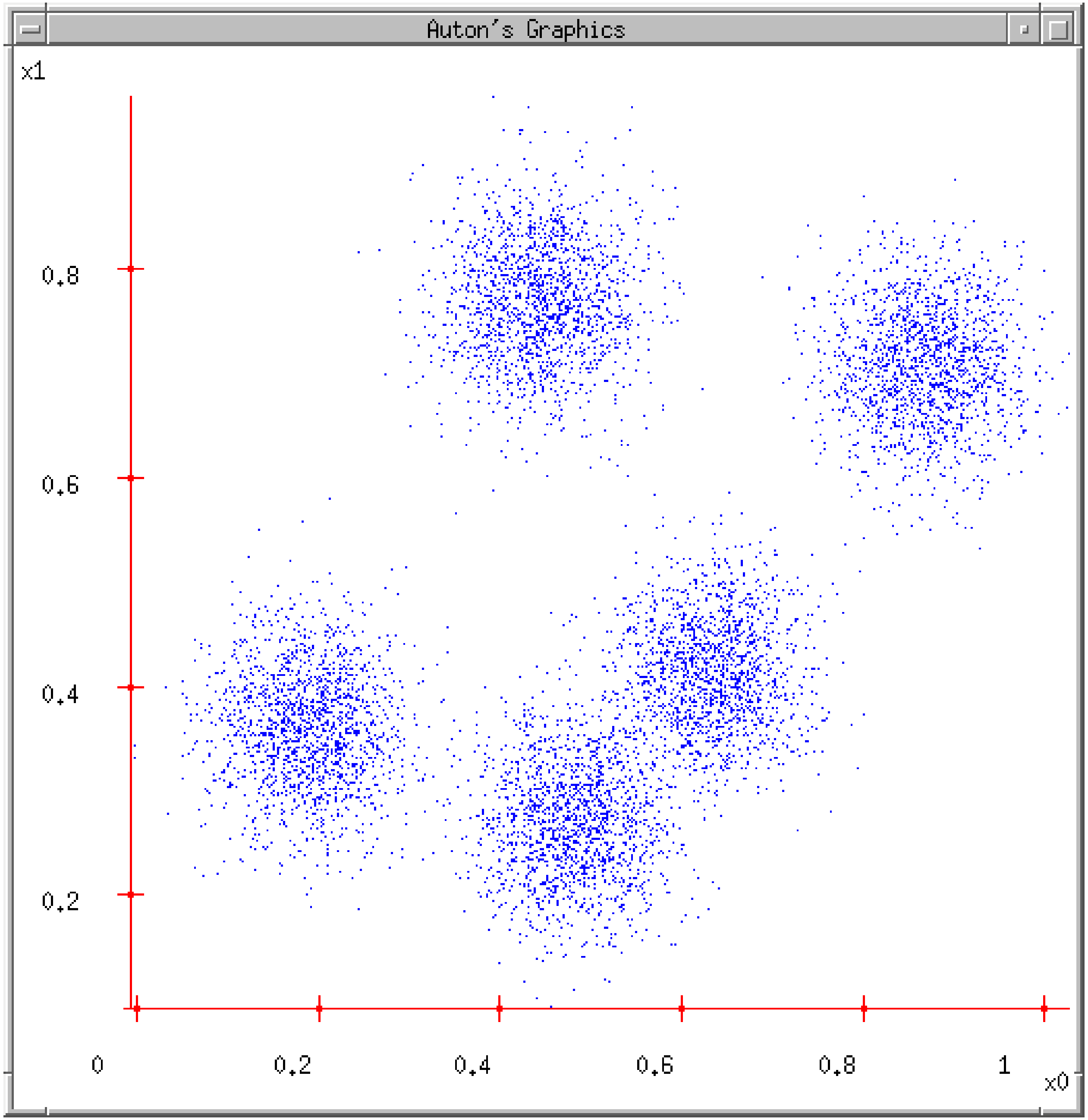
1. Moving μ_k to centroid of all points in C_k reduces distortion

2. Mapping point x_i to nearest center μ_k reduces distortion.

Keep number of centers fixed, and alternate between these two moves.

K-means

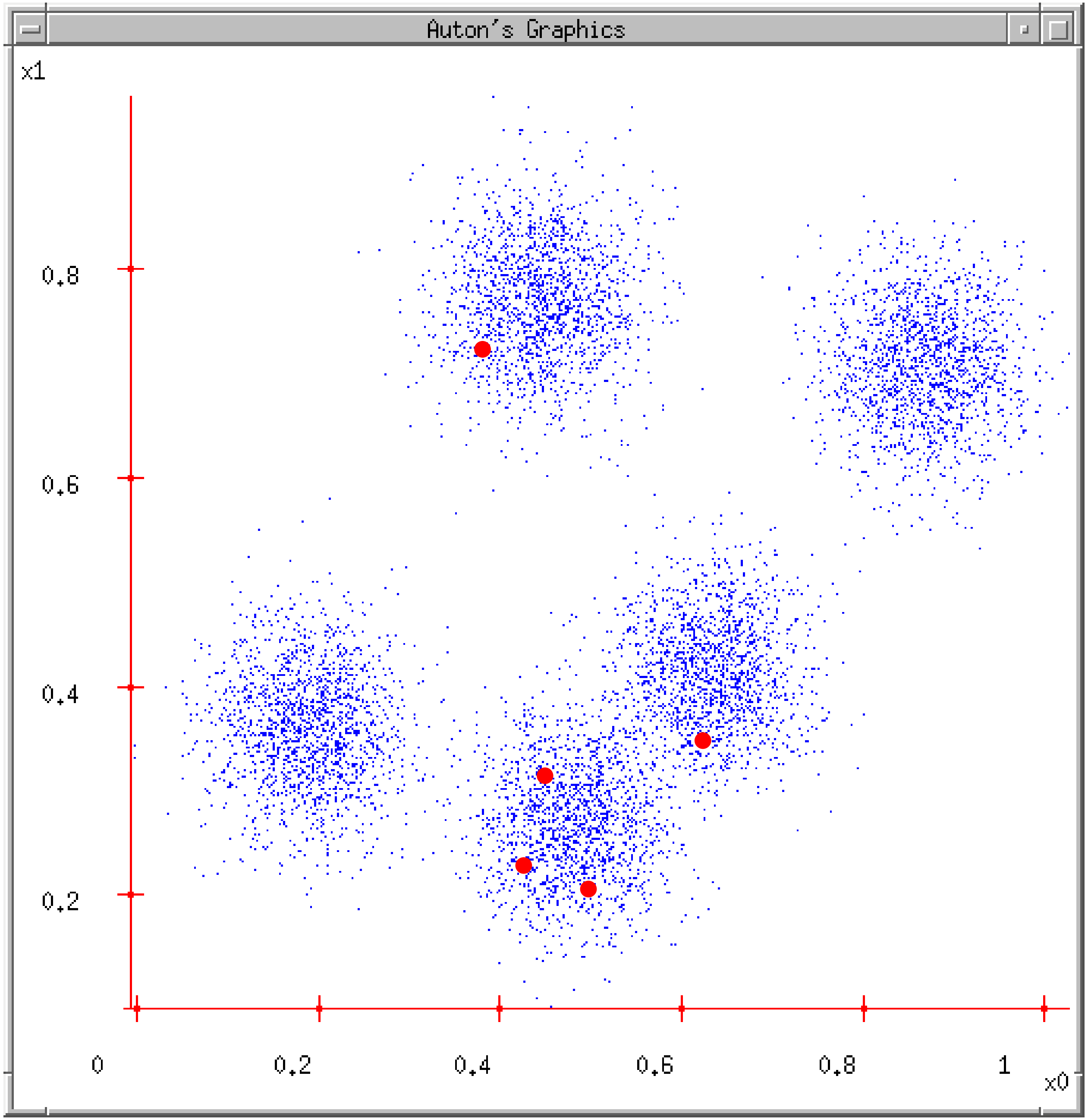
1. Ask user how many clusters they'd like. (e.g. k = 5)



Thanks to Andrew Moore for providing this example.

K-means

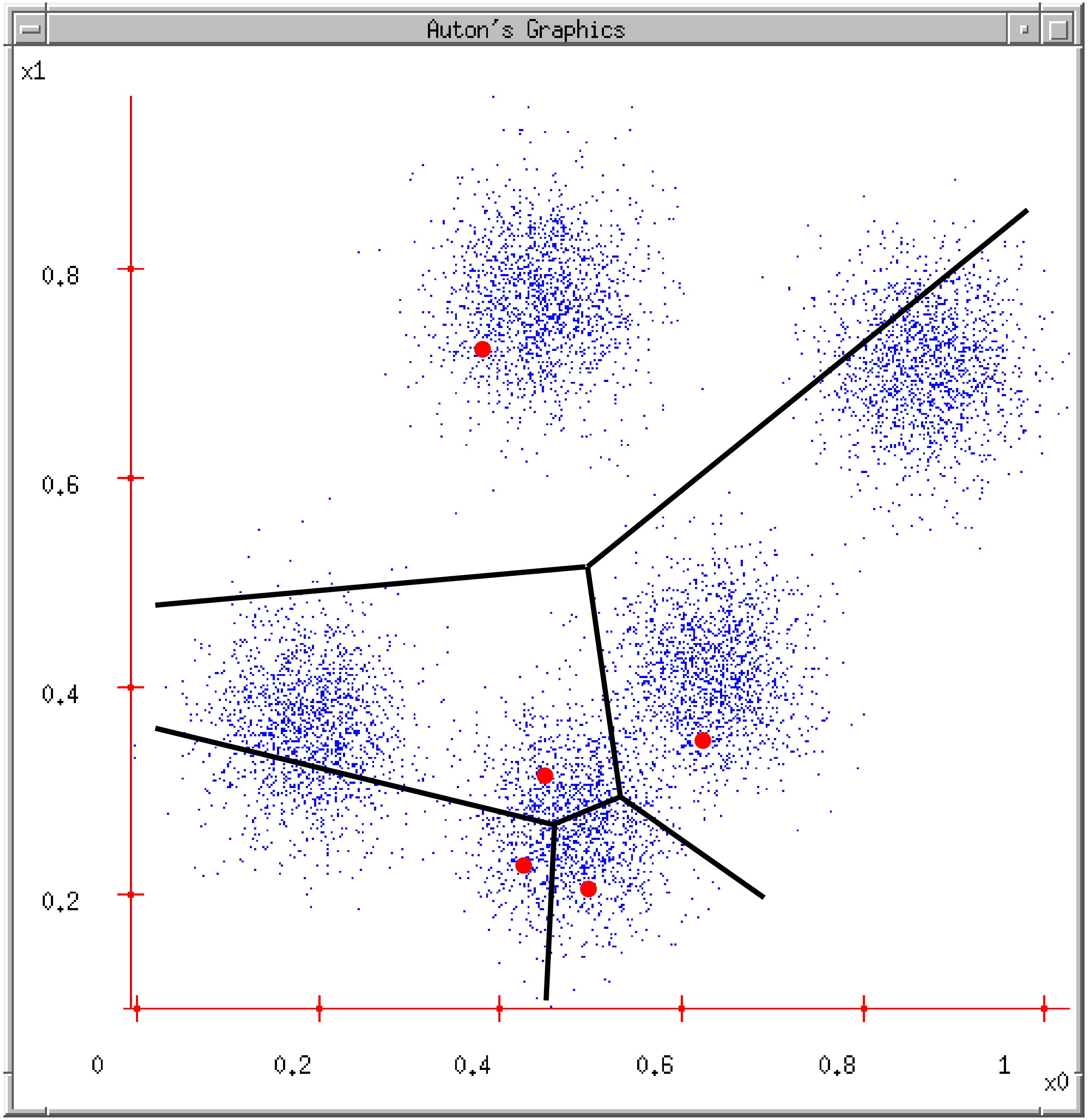
1. Ask user how many clusters they'd like. (e.g. k = 5)
2. Randomly guess k cluster center locations



Thanks to Andrew Moore for providing this example.

K-means

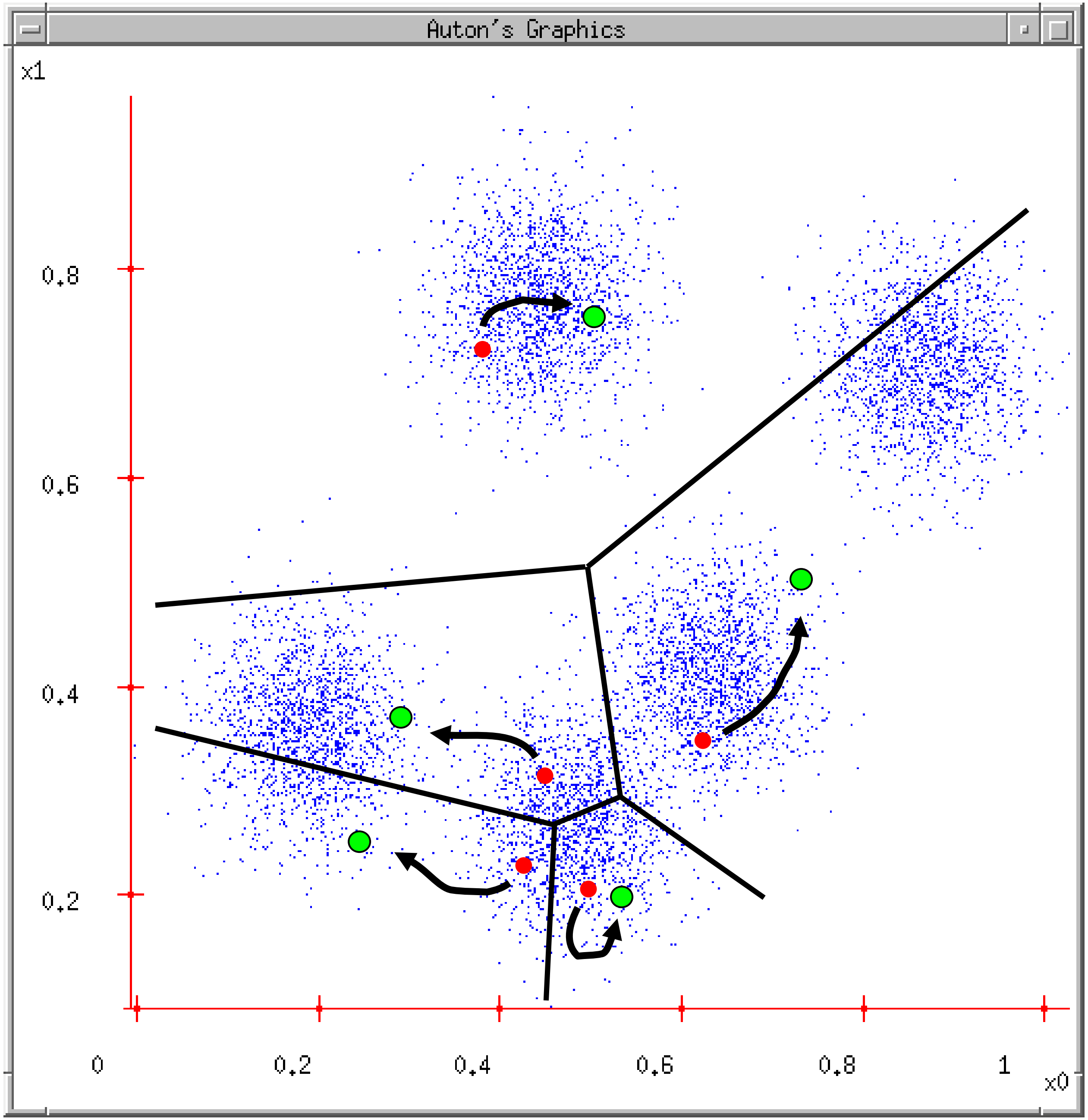
1. Ask user how many clusters they'd like. (e.g. k = 5)
2. Randomly guess k cluster center locations
3. Each datapoint finds out which center it's closest to.



Thanks to Andrew Moore for providing this example.

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, and moves there.

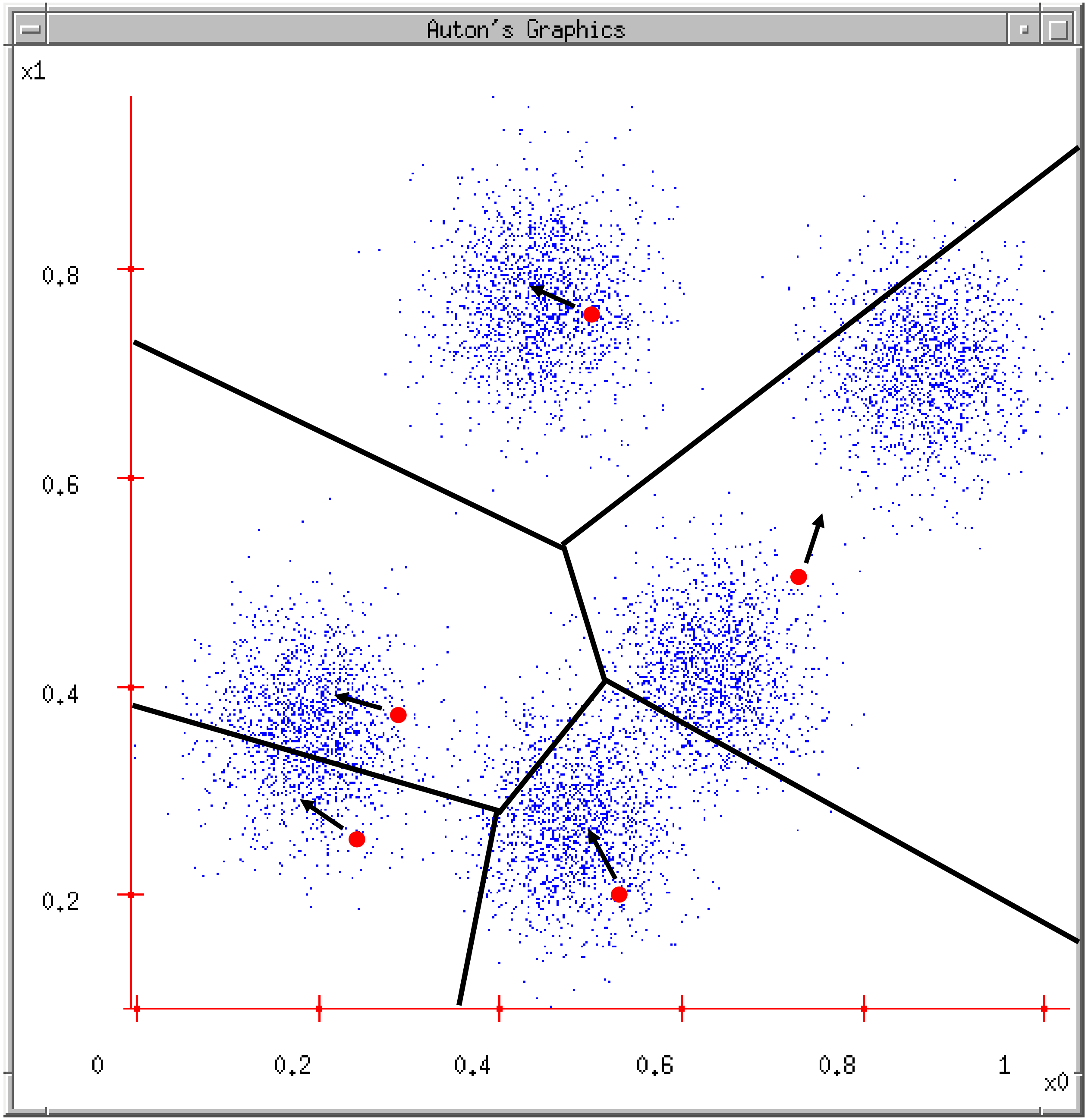


Thanks to Andrew Moore for providing this example.

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, and moves there.

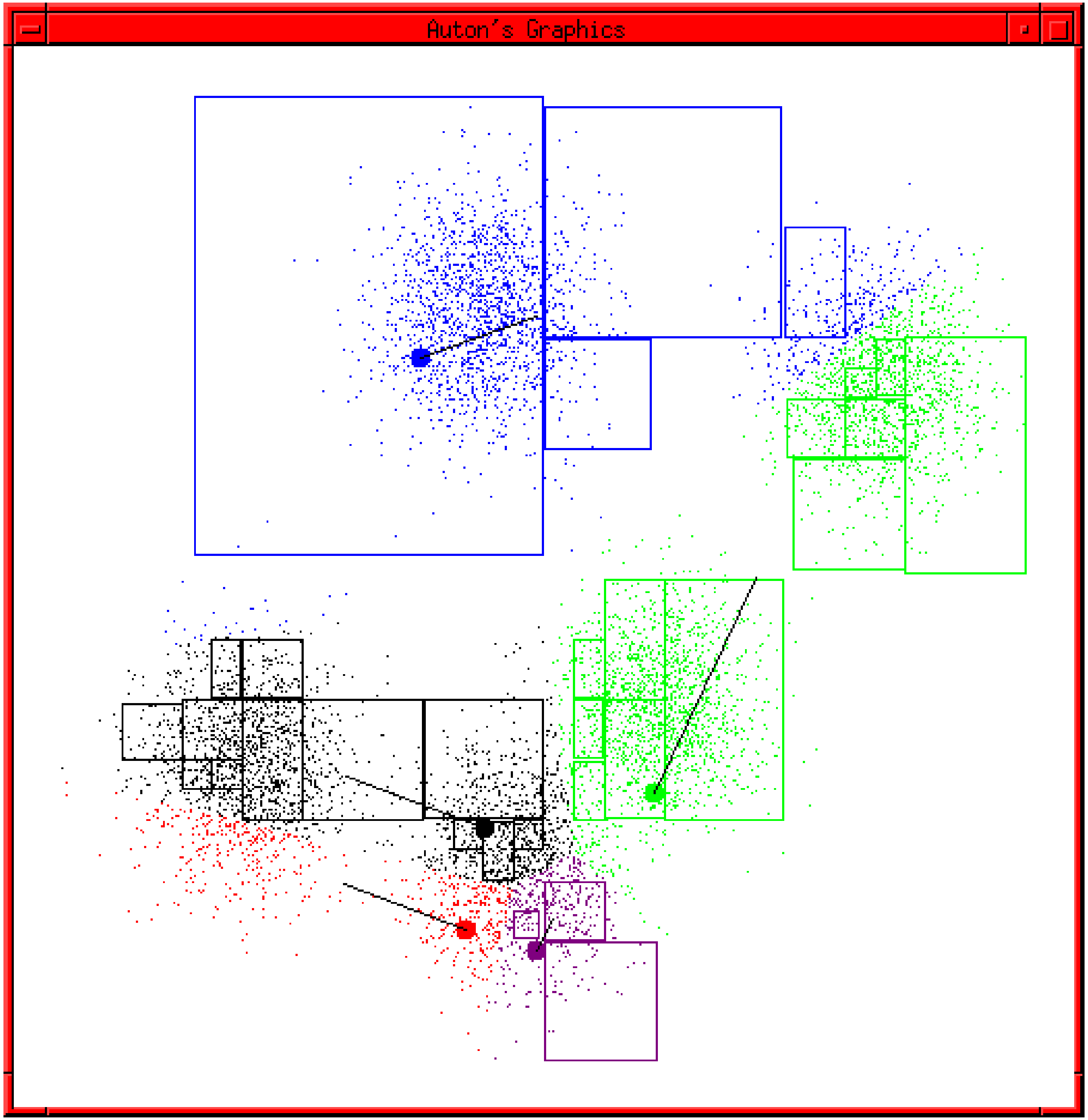
Repeat steps 3-4 until convergence!



Thanks to Andrew Moore for providing this example.

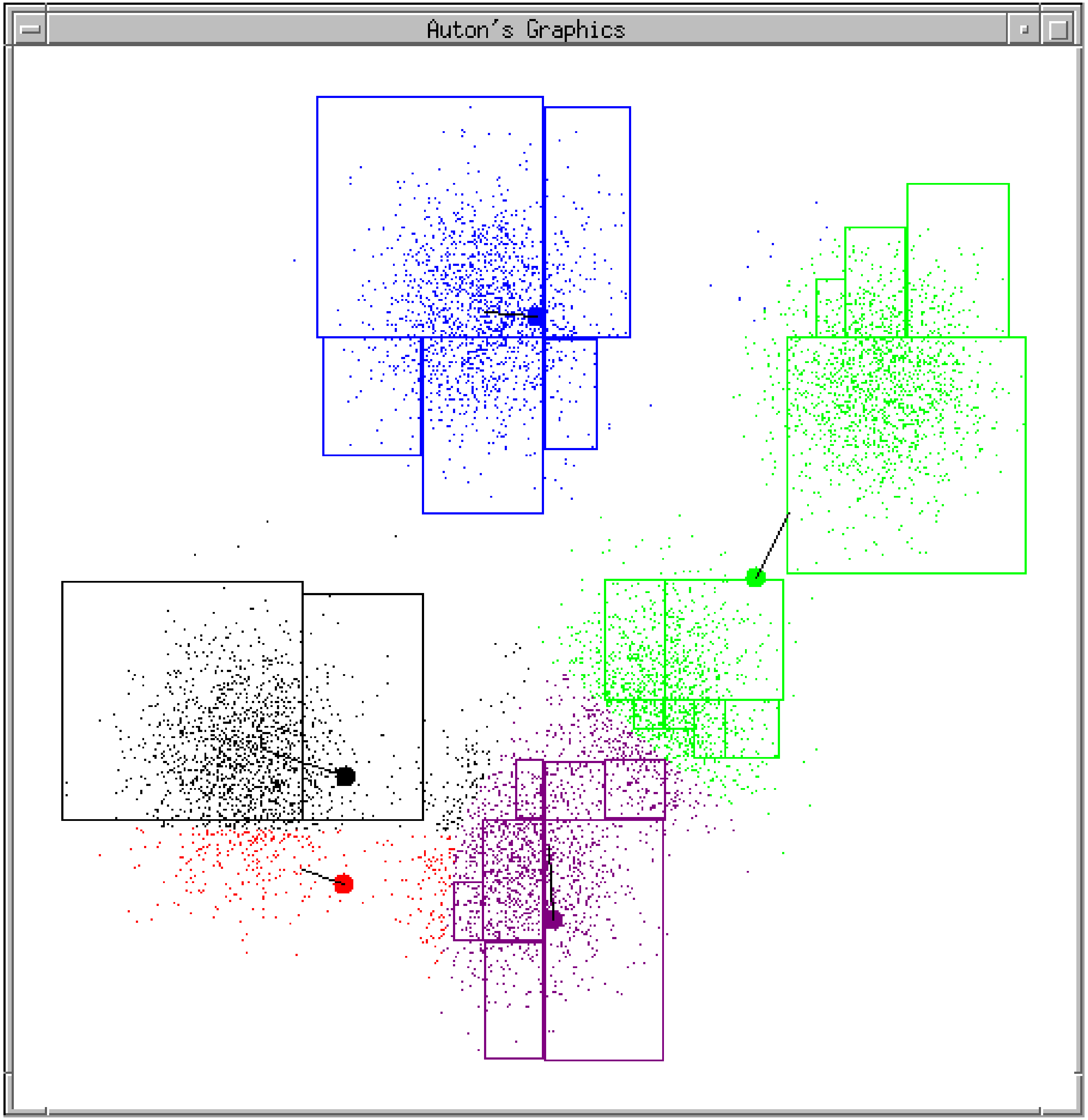
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



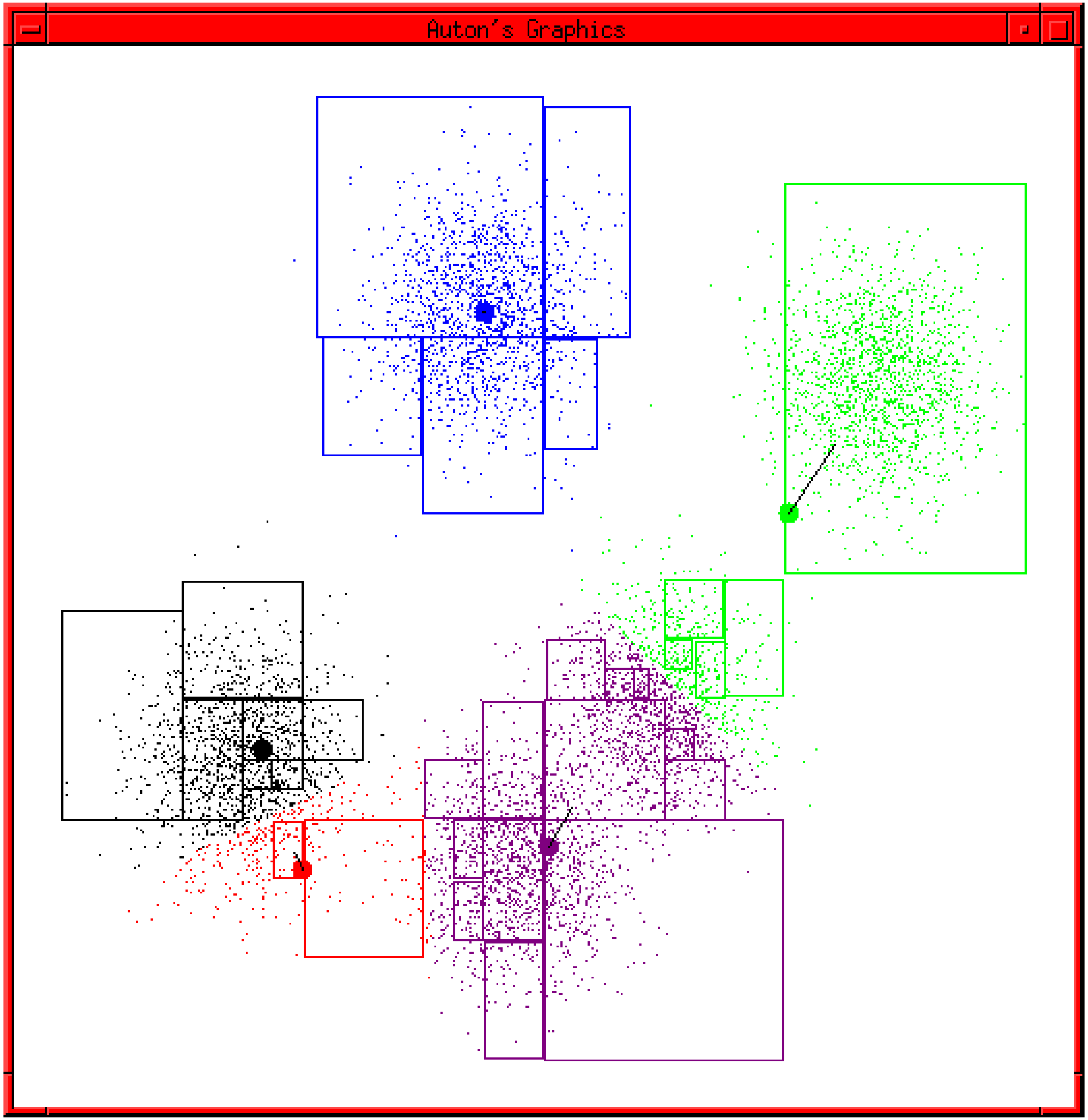
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



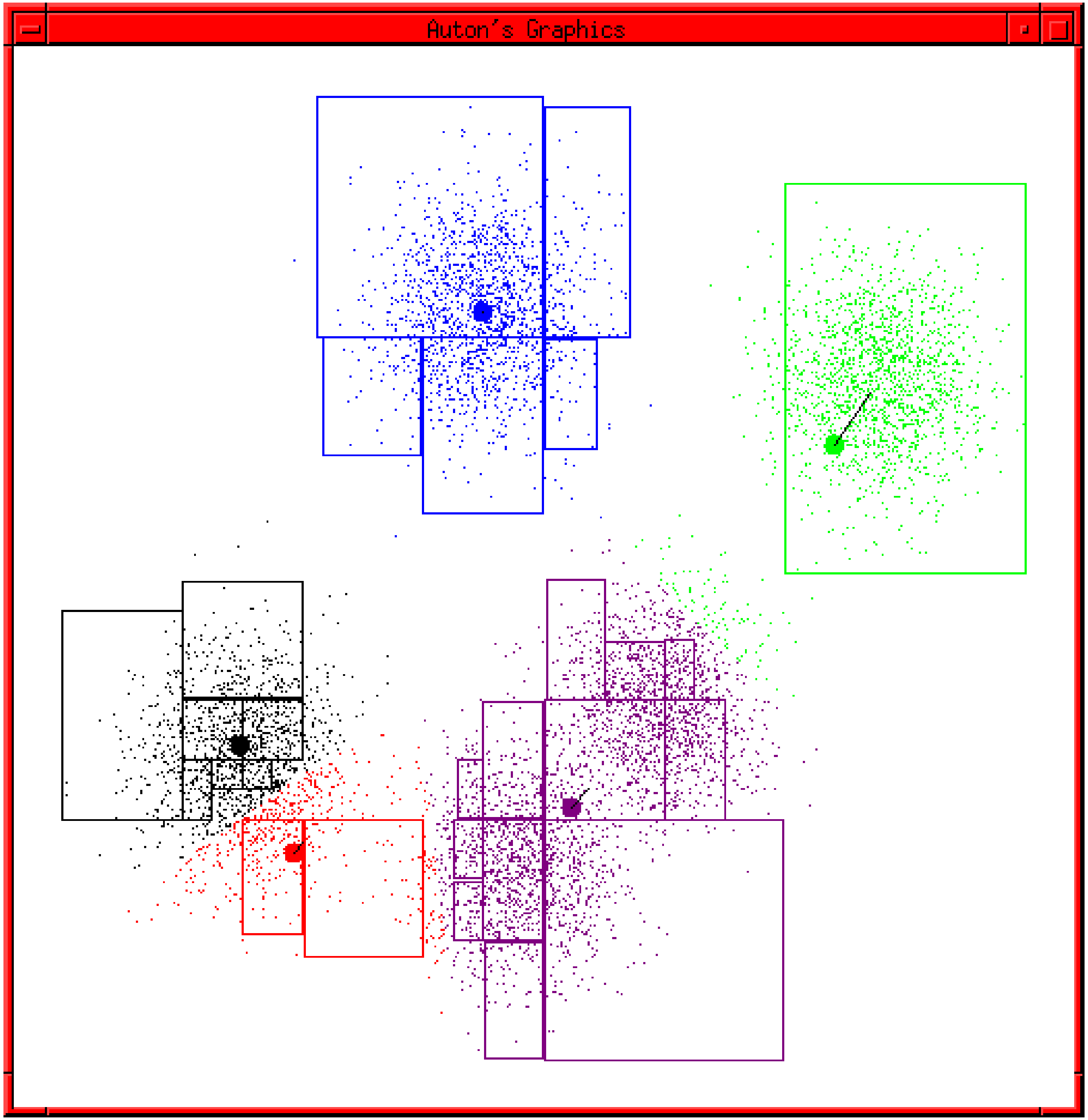
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



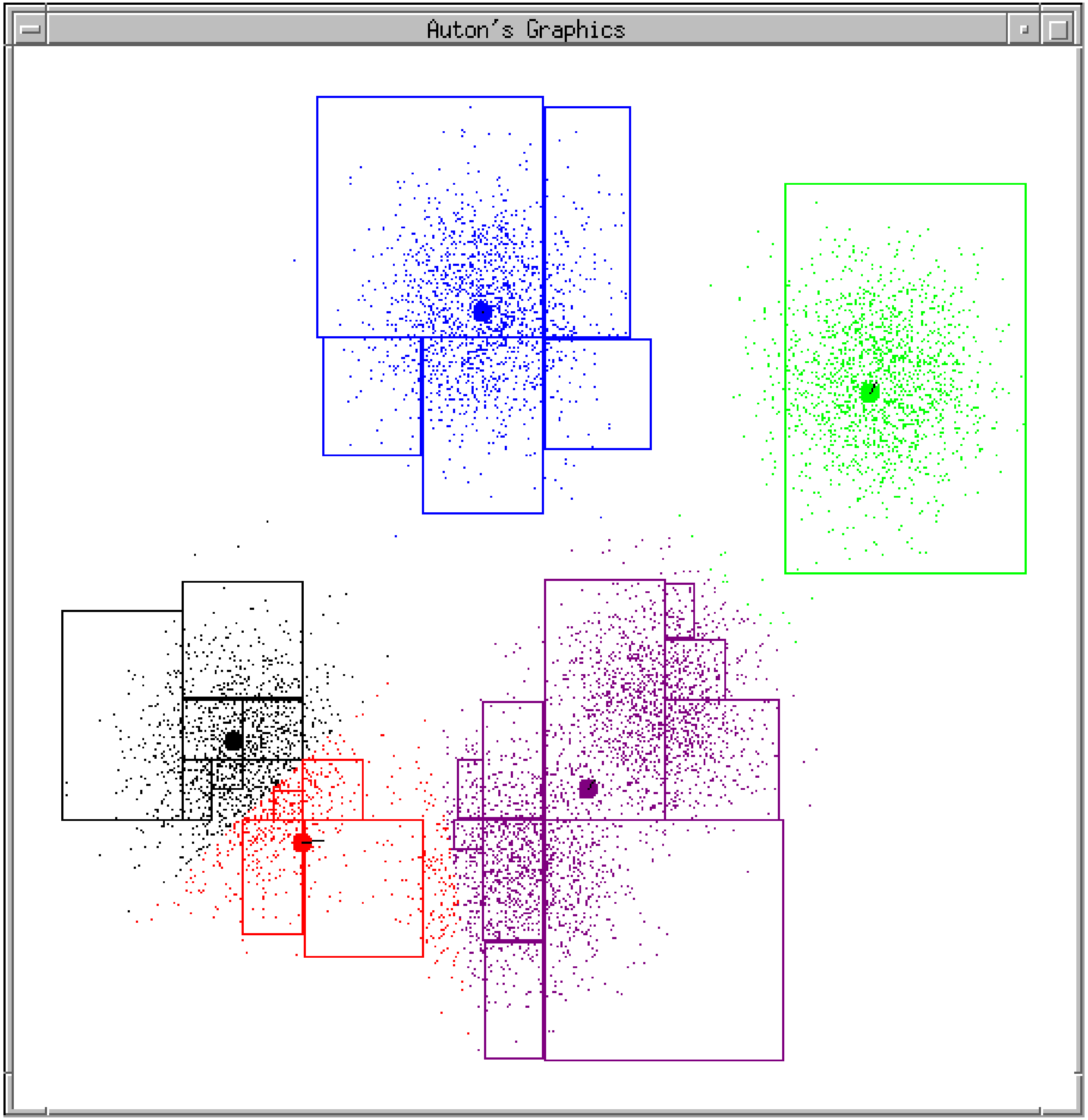
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



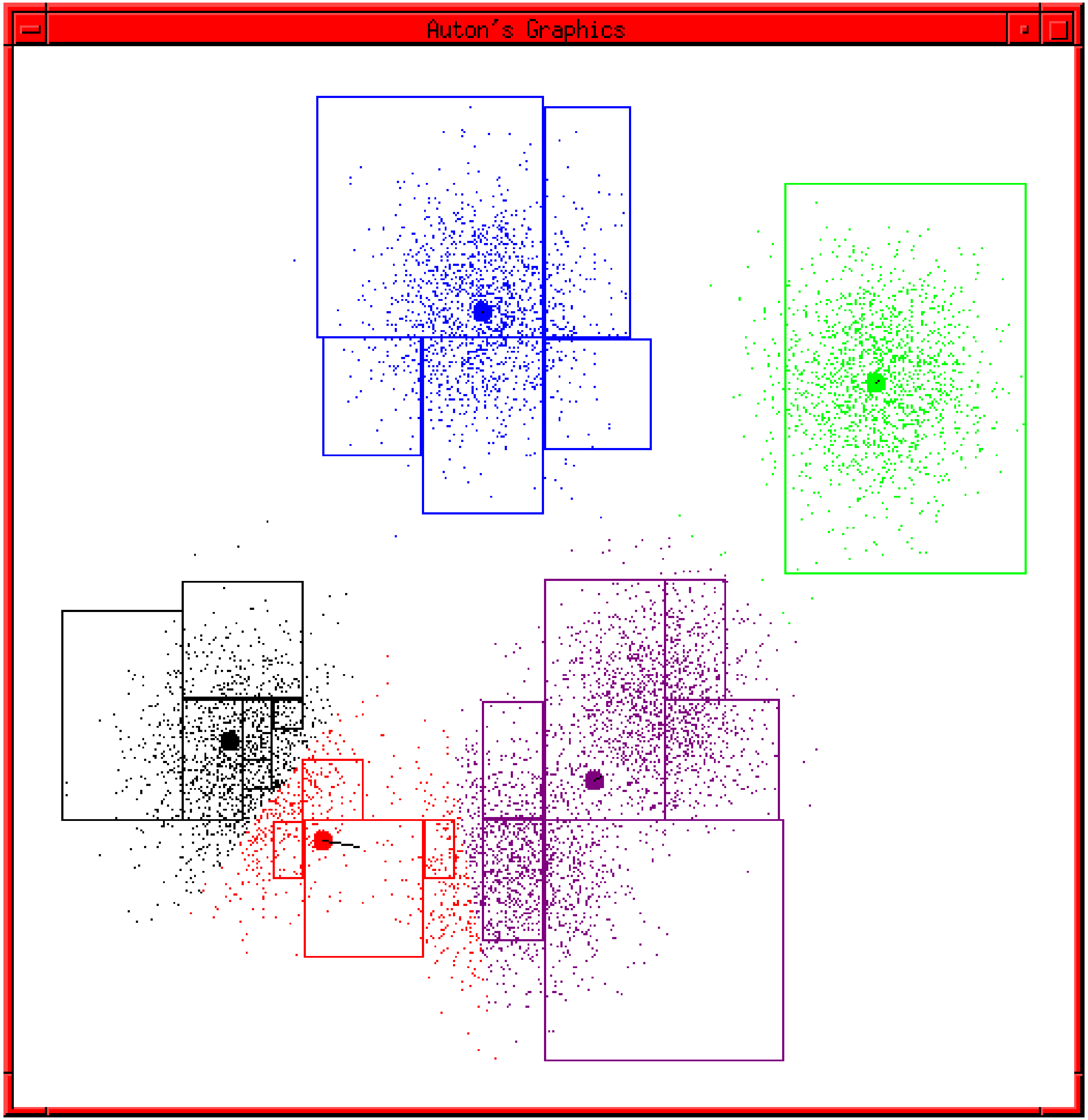
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



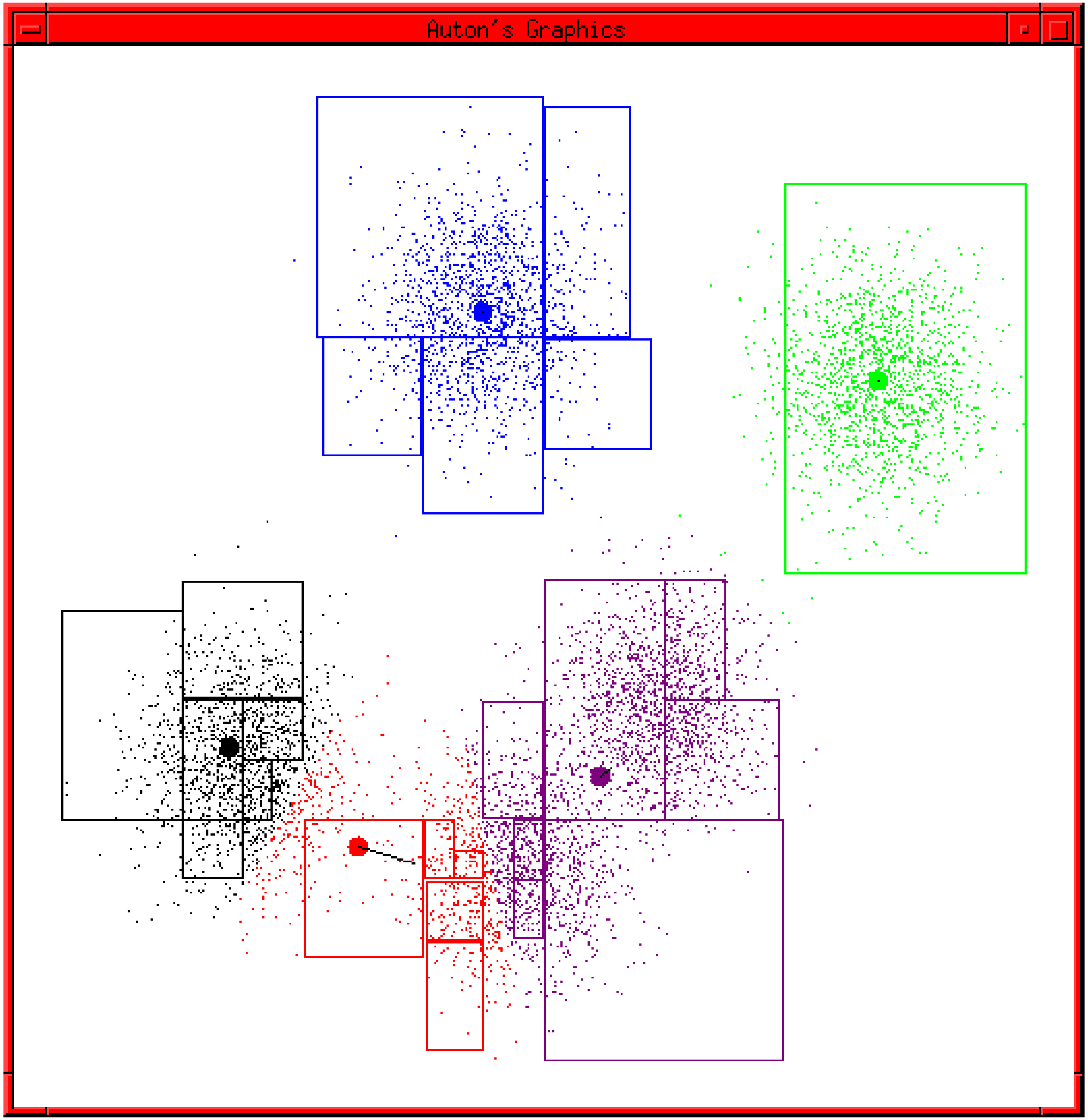
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



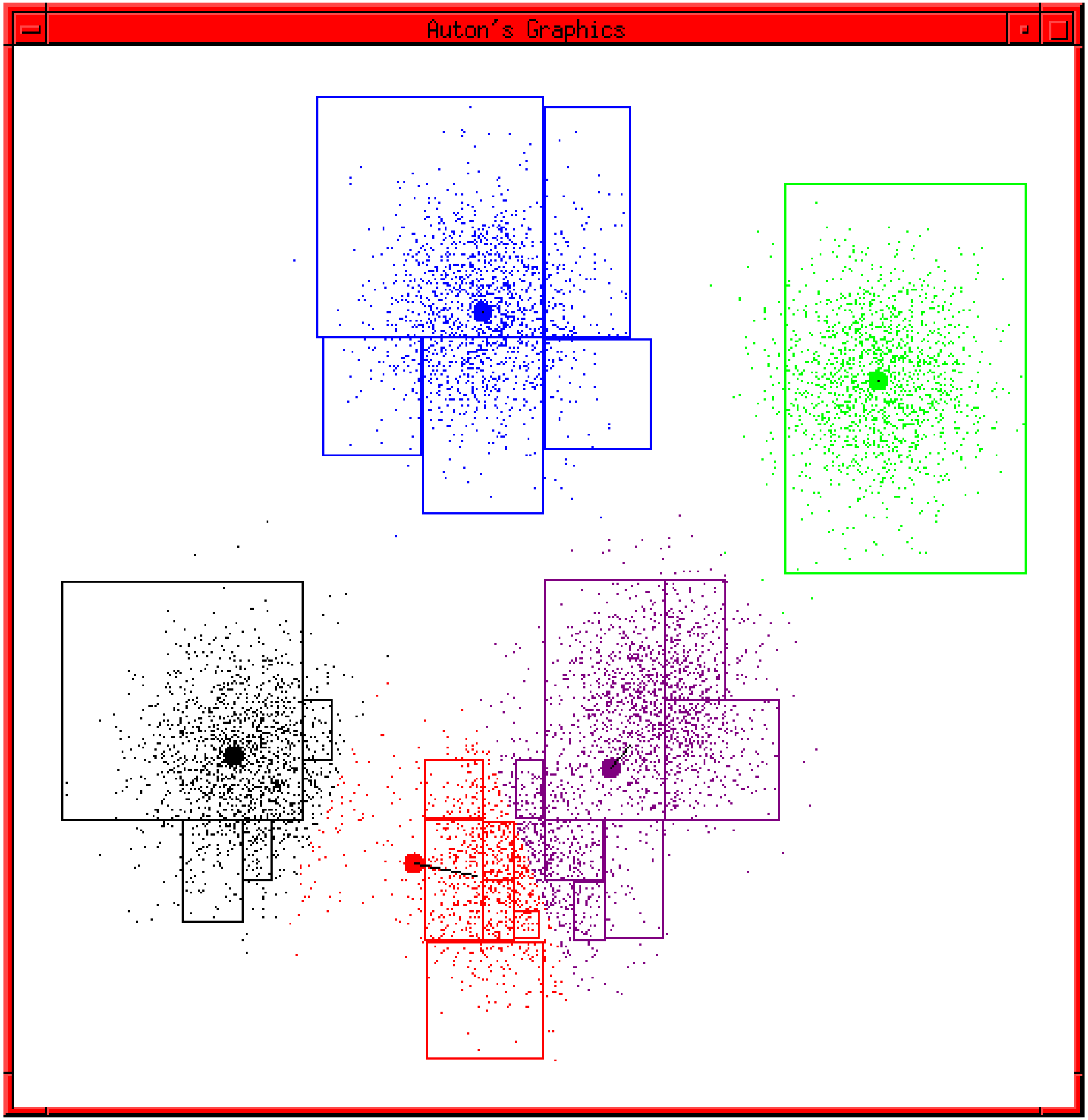
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



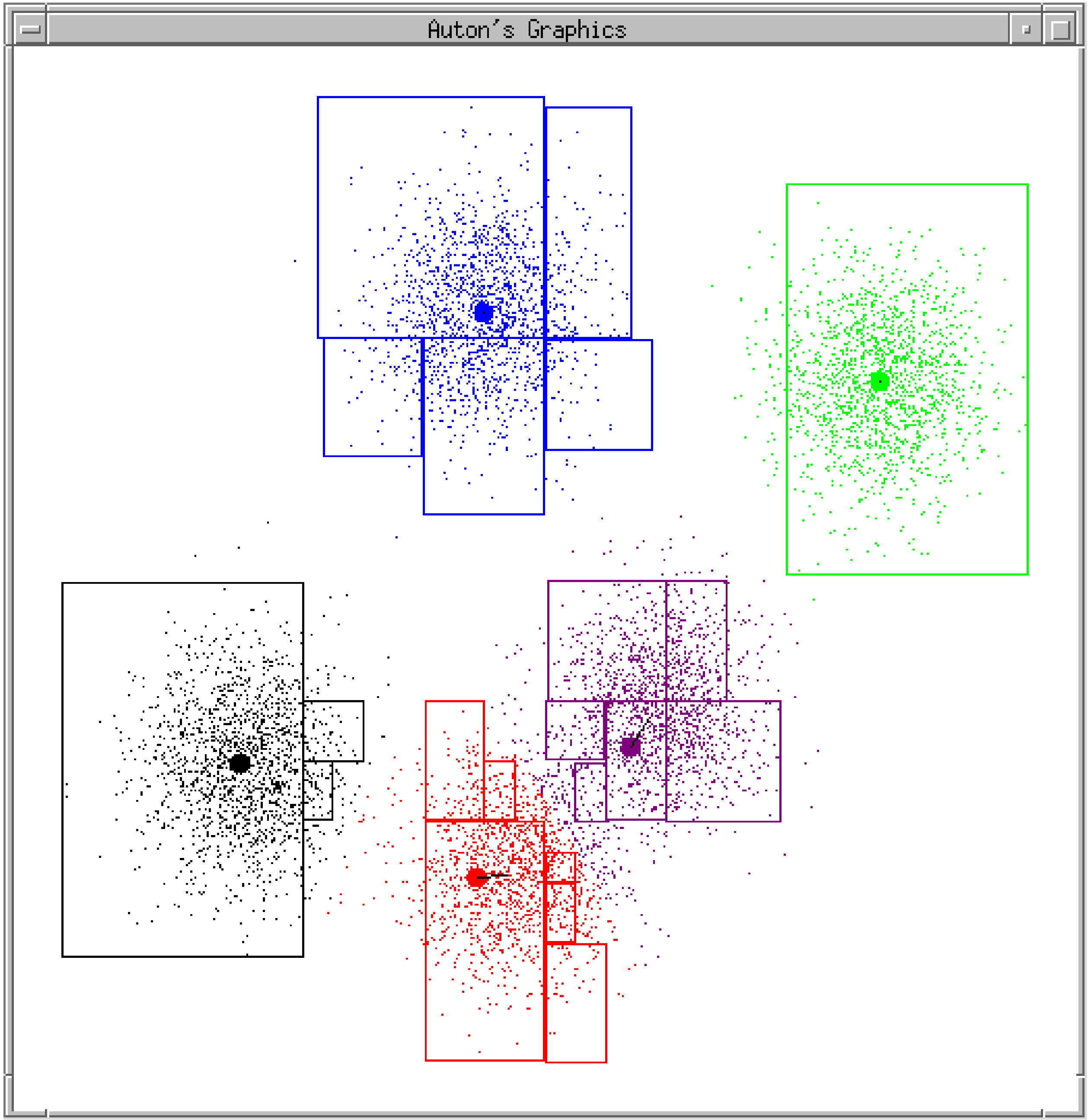
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



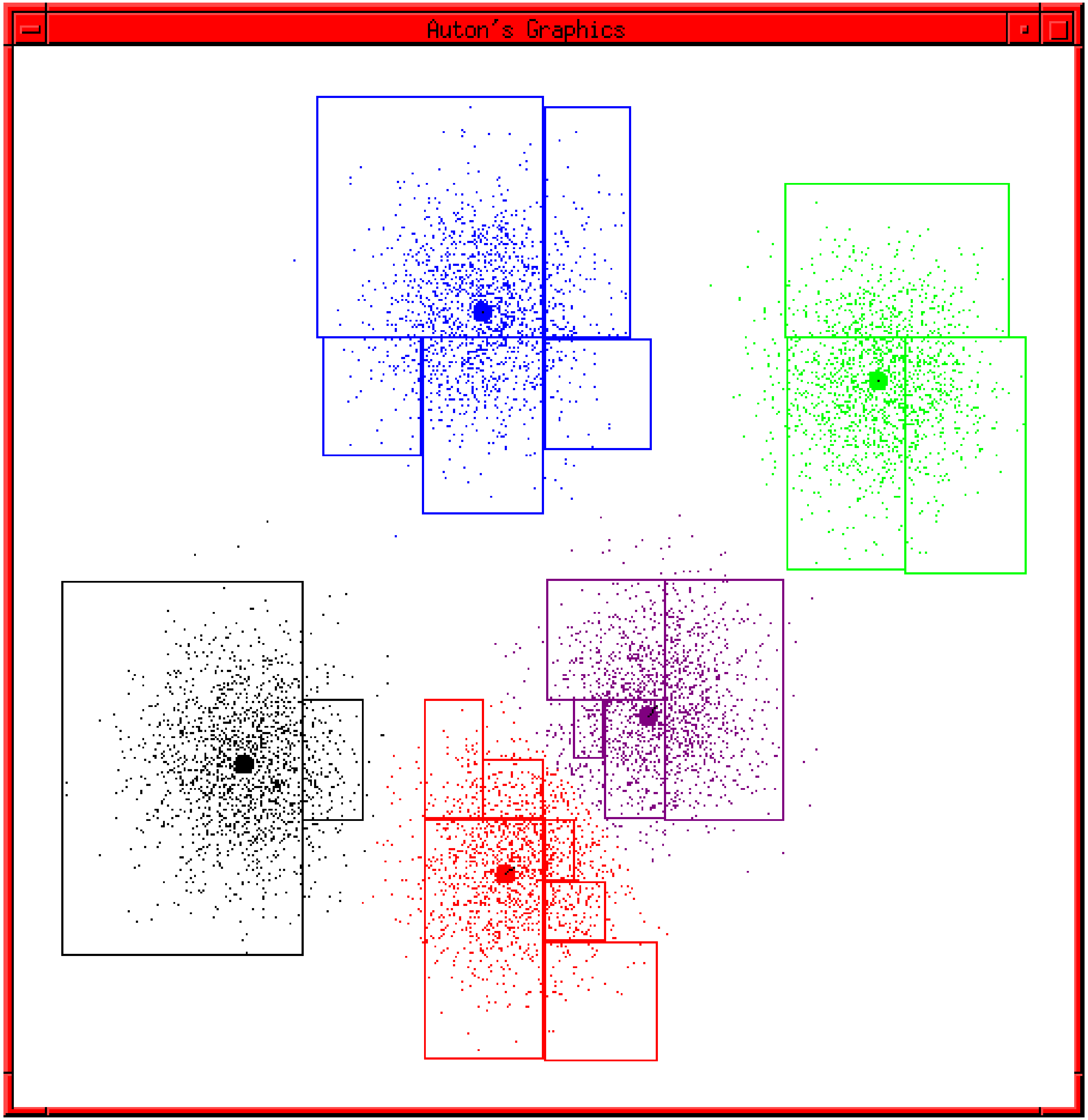
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



K-means

Here's the final result when the algorithm converges.



K-means clustering

Question 1: Will k-means always converge to a solution?

Yes! Distortion decreases monotonically, and it will end in a state where neither type of move will further reduce the distortion.

Question 2: Will k-means always find the optimal solution?

No! Here is one example where k-means converges to a locally optimal solution that does not have the global minimum distortion.

K-means clustering

Question 1: Will k-means always converge to a solution?

Yes! Distortion decreases monotonically, and it will end in a state where neither type of move will further reduce the distortion.

Question 2: Will k-means always find the optimal solution?

No! Here is one example where k-means converges to a locally optimal solution that does not have the global minimum distortion.

Question 3: How can we avoid these poor local optima?

K-means is a form of hill-climbing, so we can use many of the same tricks!

1. Run multiple times with different start states, and choose the best result.
2. Allow some moves that increase distortion, as in simulated annealing.
3. Choose a start state that is less likely to result in a poor local optimum.

Center 1 = randomly chosen data point
Center 2 = data point that's farthest from center 1
Center 3 = data point that's farthest from the closest of centers 1 and 2, etc.

K-means clustering

Question 4: How can we choose the number of centers k?

Run k-means multiple times with different values of k, and choose the k with minimum distortion???

Bad idea! Minimum distortion occurs when k = N, and each cluster contains only one point.

Better idea: choose the k that minimizes a measure of distortion with a penalty for more complex models.

Schwarz criterion: minimize (distortion + λk), where λ is a constant, proportional to (# of attributes) x log(# of records).

(Many other criteria have also been considered!)

In general, we can use this criterion to pick the “best” of any set of clusterings, e.g. which links to cut for a given hierarchical clustering.

K-means vs. EM

Both k-means and EM are iterative algorithms that **model** each cluster and **assign** points to clusters. How do these algorithms differ?

Assignment: k-means makes **hard** cluster assignments (each point is mapped to a single cluster), while EM makes **soft** assignments (each point is assigned a probability distribution over clusters).

Modeling: k-means models only the mean (centroid) of each cluster, while EM models the mean and **covariance matrix** (or with naïve Bayes assumption, the **variance** of each attribute).

Equivalence: EM reduces to k-means when covariance matrix is diagonal, and all variances are equal and very small.

K-means is biased toward spherical clusters; EM+NB is biased to axis-aligned ellipses; EM with full covariance matrix can model non-axis-aligned ellipses.

Speed: k-means is much faster and consumes much less memory in practice. (the tradeoff: EM can better model the data, as measured by log-likelihood.)

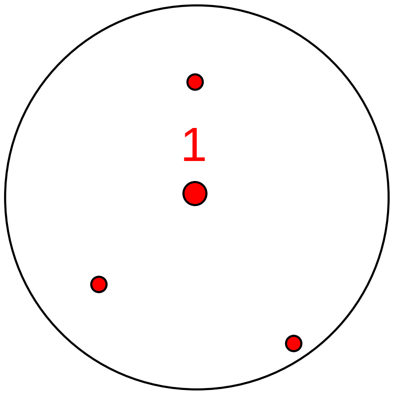
Leader clustering

"massive streaming data"
Billions of sensor readings
for a complex system

What if the dataset is so huge that we can only look at each data point once before discarding it?

We keep only a small subset of representative points ("leaders") and summary information about the points similar to each leader.

For each data point x_i: if x_i is within distance T of any leader, add to nearest leader's group, otherwise make x_i a leader.



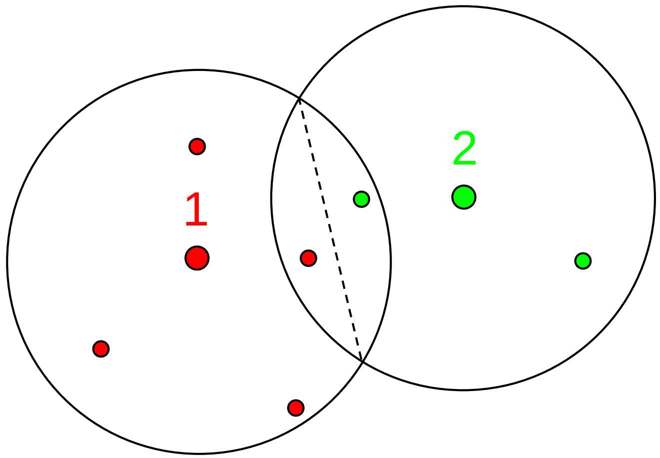
Leader clustering

“massive streaming data”
Billions of sensor readings
for a complex system

What if the dataset is so huge that we can only look at each data point once before discarding it?

We keep only a small subset of representative points (“leaders”) and summary information about the points similar to each leader.

For each data point x_i: if x_i is within distance T of any leader, add to nearest leader’s group, otherwise make x_i a leader.

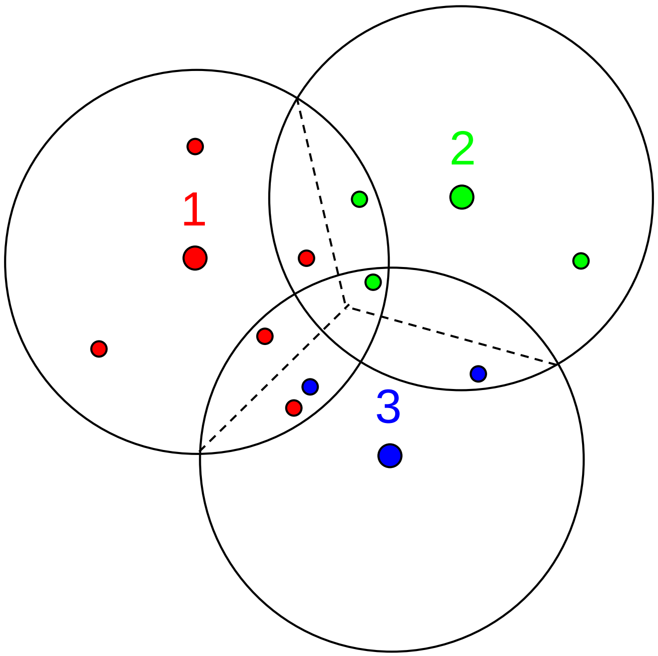


Leader clustering

"massive streaming data"
Billions of sensor readings
for a complex system

What if the dataset is so huge that we can only look at each data point once before discarding it?

We keep only a small subset of representative points ("leaders") and summary information about the points similar to each leader.



For each data point x_i: if x_i is within distance T of any leader, add to nearest leader's group, otherwise make x_i a leader.

Advantages of leader clustering

Very fast: only need to look at leaders for each point

Guarantees group diameter < 2T, group leaders at least T apart

Disadvantages of leader clustering

Order-dependent: first point always a leader, initial clusters tend to be larger

Points may not be assigned to nearest cluster center

The many uses of clustering

Clustering provides a useful **summary** of a large dataset, representing the N data records using only k << N clusters.

This can be used for **exploratory data analysis**, enabling us to understand the underlying sub-structure of the data.

Value of k?
Shape?
Size?
Hierarchy?

Important to choose correct k value

k usually small

We can often improve the performance of model-based prediction by learning separate models for each group.

We can also use clustering to detect **anomalies**, by finding points that are far from any cluster center.

Choose k large as possible

We can use clustering to speed up instance-based prediction (e.g. k-NN) by reducing the training set size.

Finally, we can use clustering to handle massive streaming data by maintaining only the cluster summaries in memory.

References

- Scikit-learn clustering documentation:
  http://scikit-learn.org/stable/modules/clustering.html
- C.C. Aggarwal and C.K. Reddy, eds. *Data Clustering: Algorithms and Applications*, 2014.
  http://www.crcnetbase.com/doi/book/10.1201/b15410
- A.K. Jain et al. Data clustering: a review. *ACM Computing Surveys* 31(3), 1999.
- The Auton Laboratory (www.autonlab.org) has very fast k-means (and X-means) software, created by D. Pelleg.