Carnegie Mellon

autonlab.org

Scalable Data Mining

Brigham Anderson, Andrew Moore, Dan Pelleg,
Alex Gray, Bob Nichols, Andy Connolly

The Auton Lab, Carnegie Mellon University

www.autonlab.org

Outline

- Kd-trees
  - Fast nearest-neighbor finding
  - Fast K-means clustering
  - Fast kernel density estimation

- Large-scale galactic morphology

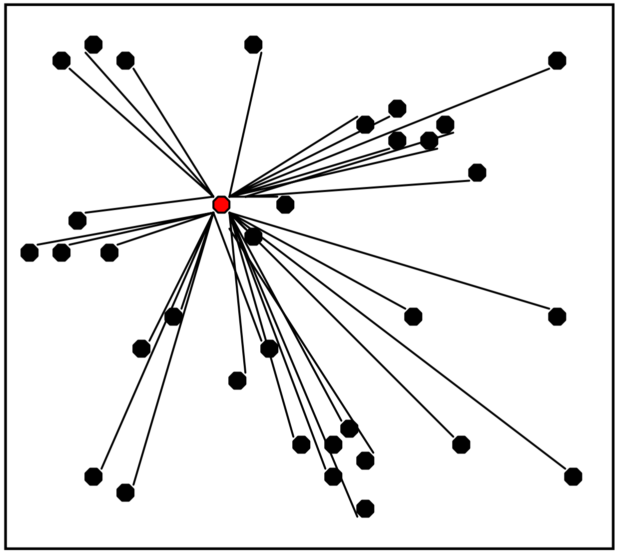
Outline

- Kd-trees
  - Fast nearest-neighbor finding
  - Fast K-means clustering
  - Fast kernel density estimation

- Large-scale galactic morphology

Nearest Neighbor - Naïve Approach

- Given a query point X.
- Scan through each point Y
- Takes O(N) time for each query!

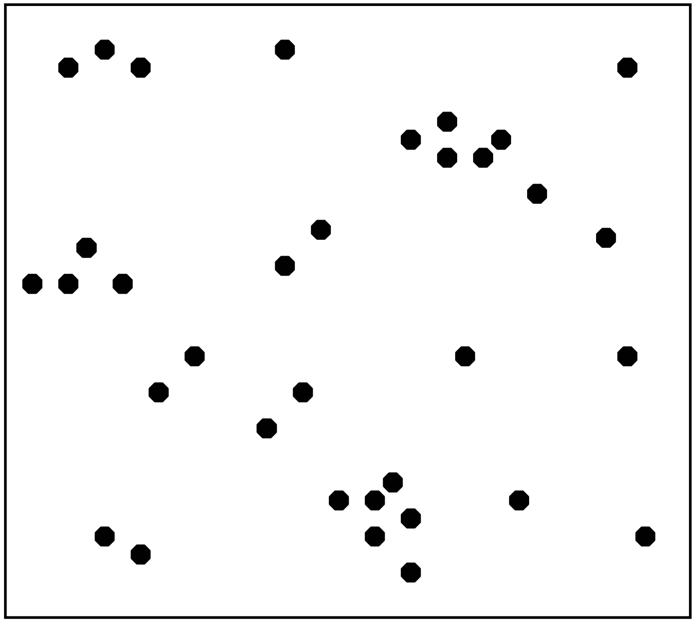


33 Distance Computations

Speeding Up Nearest Neighbor

- We can speed up the search for the nearest neighbor:
  - Examine nearby points first.
  - Ignore any points that are further then the nearest point found so far.
- Do this using a KD-tree:
  - Tree based data structure
  - Recursively partitions points into axis aligned boxes.

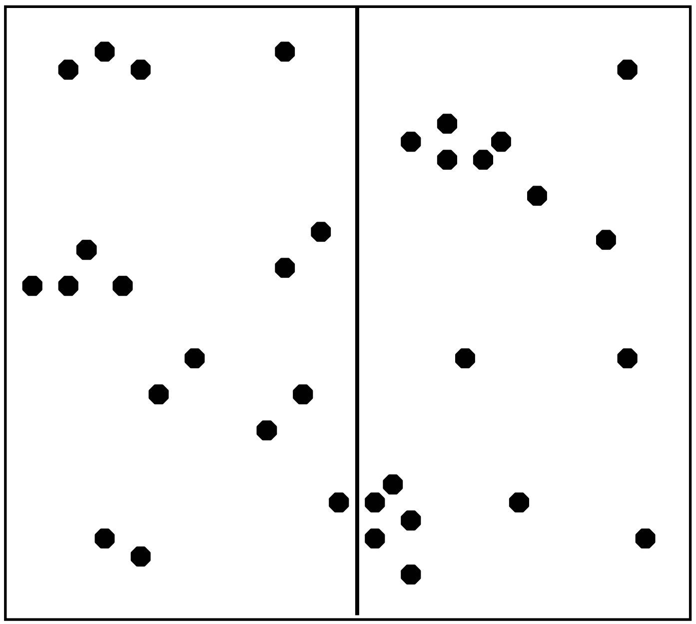
KD-Tree Construction

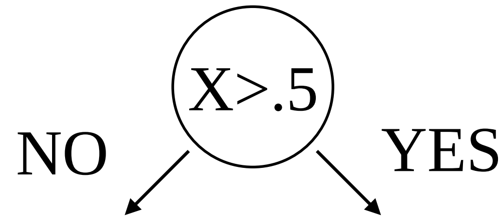


Pt	X	Y
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
...	...	...

We start with a list of n-dimensional points.

KD-Tree Construction



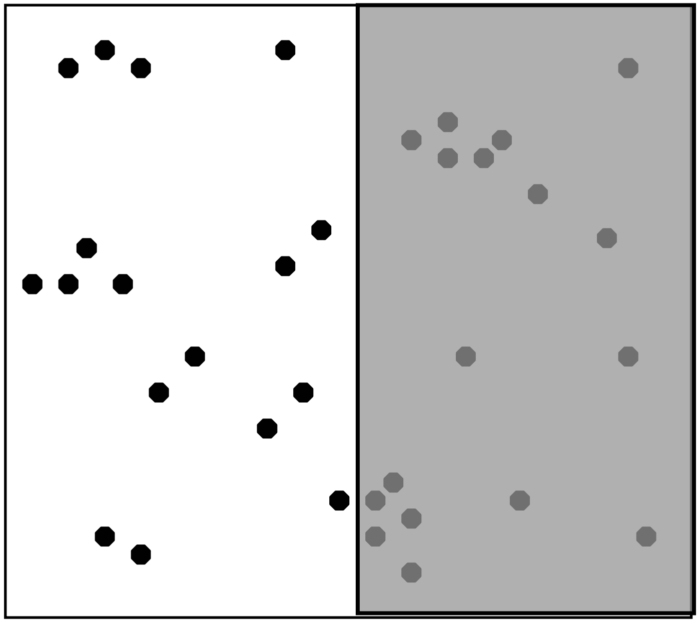


Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

Pt	X	Y
2	1.00	4.31
…	…	…

We can split the points into 2 groups by choosing a dimension X and value V and separating the points into X > V and X <= V.

KD-Tree Construction



X>.5

NO

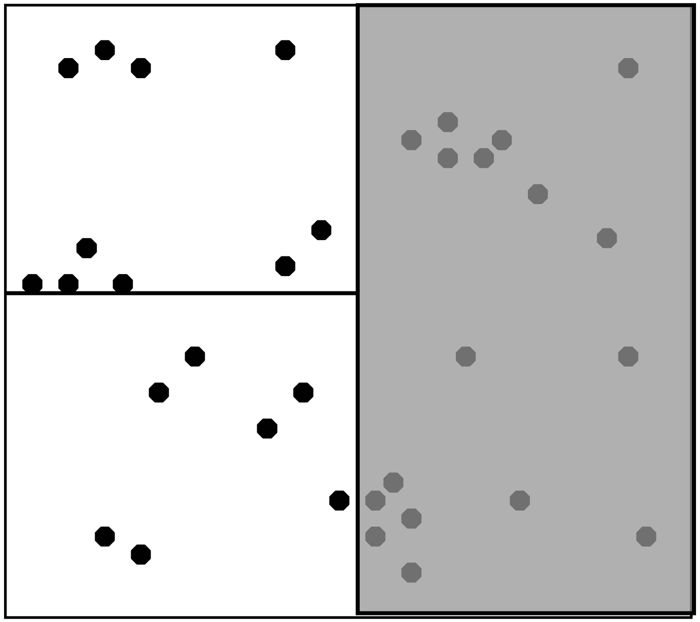
Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

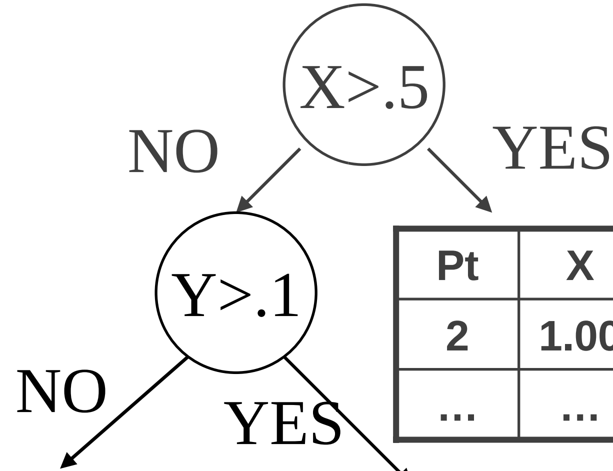
YES

Pt	X	Y
2	1.00	4.31
…	…	…

We can then consider each group separately and possibly split again (along same/different dimension).

KD-Tree Construction





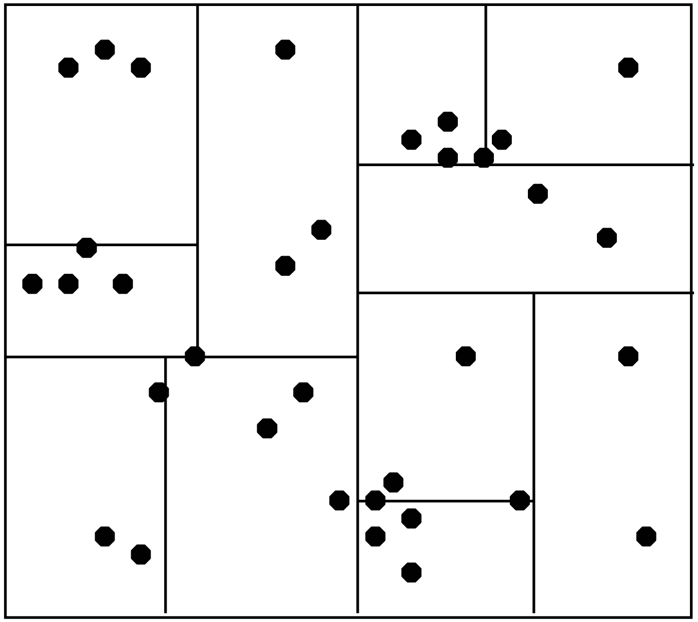
Pt	X	Y
2	1.00	4.31
...	...	...

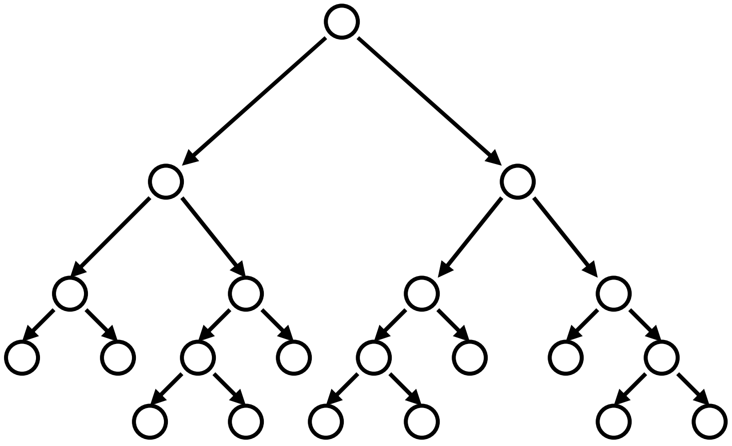
Pt	X	Y
3	0.13	2.85
...	...	...

Pt	X	Y
1	0.00	0.00
...	...	...

We can then consider each group separately and possibly split again (along same/different dimension).

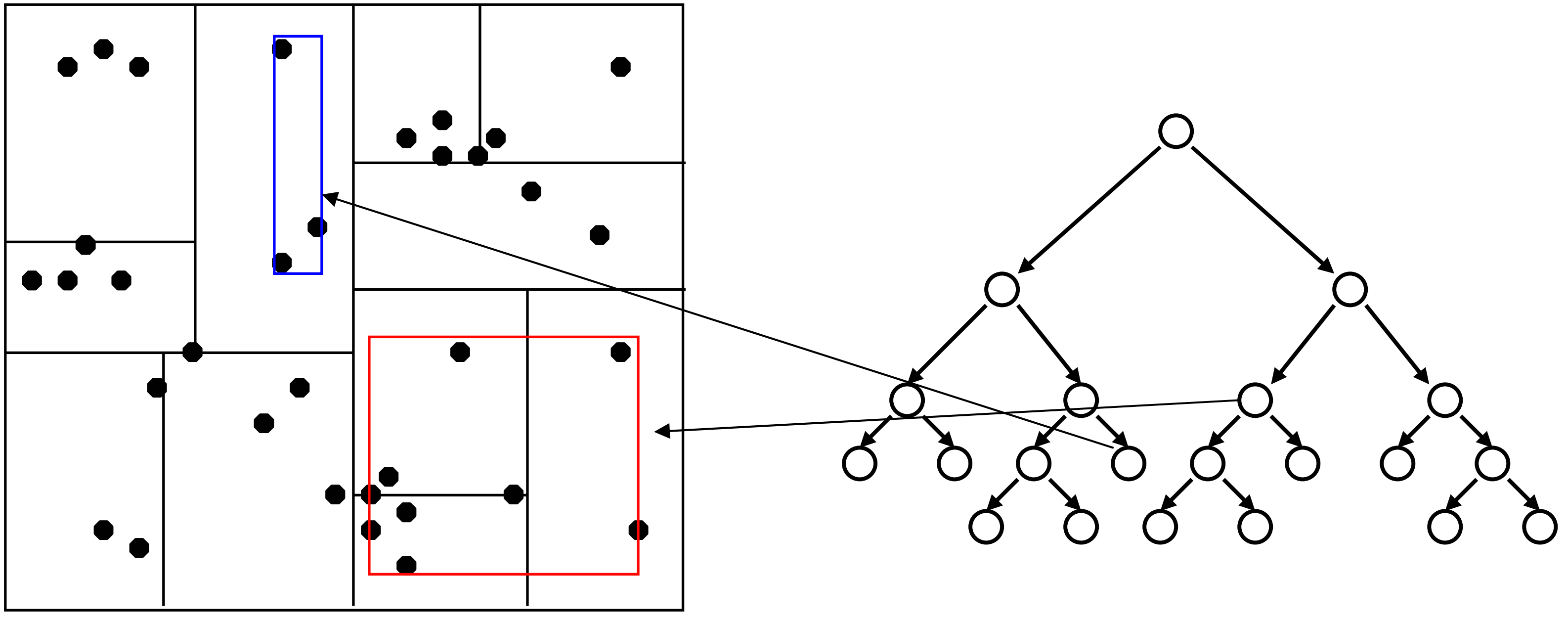
KD-Tree Construction





We can keep splitting the points in each set to create a tree structure. Each node with no children (leaf node) contains a list of points.

KD-Tree Construction



We will keep around one additional piece of information at each node. The (tight) bounds of the points at or below this node.

KD-Tree Construction

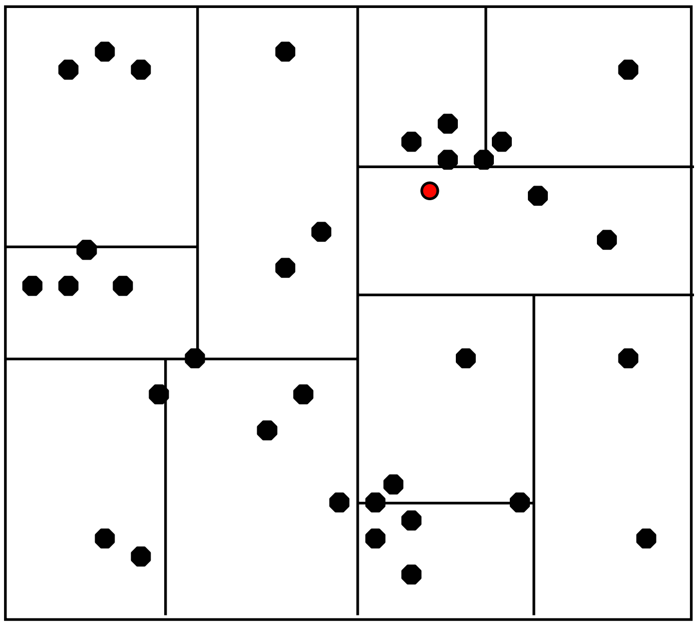
Use heuristics to make splitting decisions:

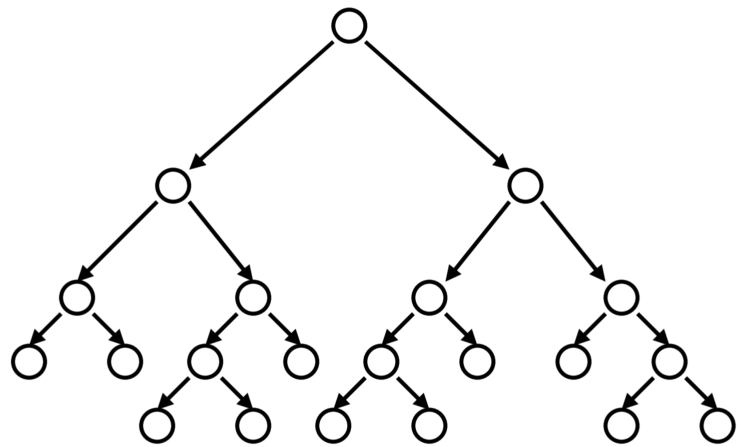
- Which dimension do we split along? Widest
- Which value do we split at? Median of value of that split dimension for the points.
- When do we stop? When there are fewer then m points left OR the box has hit some minimum width.

Outline

- ✓ Kd-trees
  - ➢ Fast nearest-neighbor finding
  - ❑ Fast K-means clustering
  - ❑ Fast kernel density estimation
- ❑ Large-scale galactic morphology

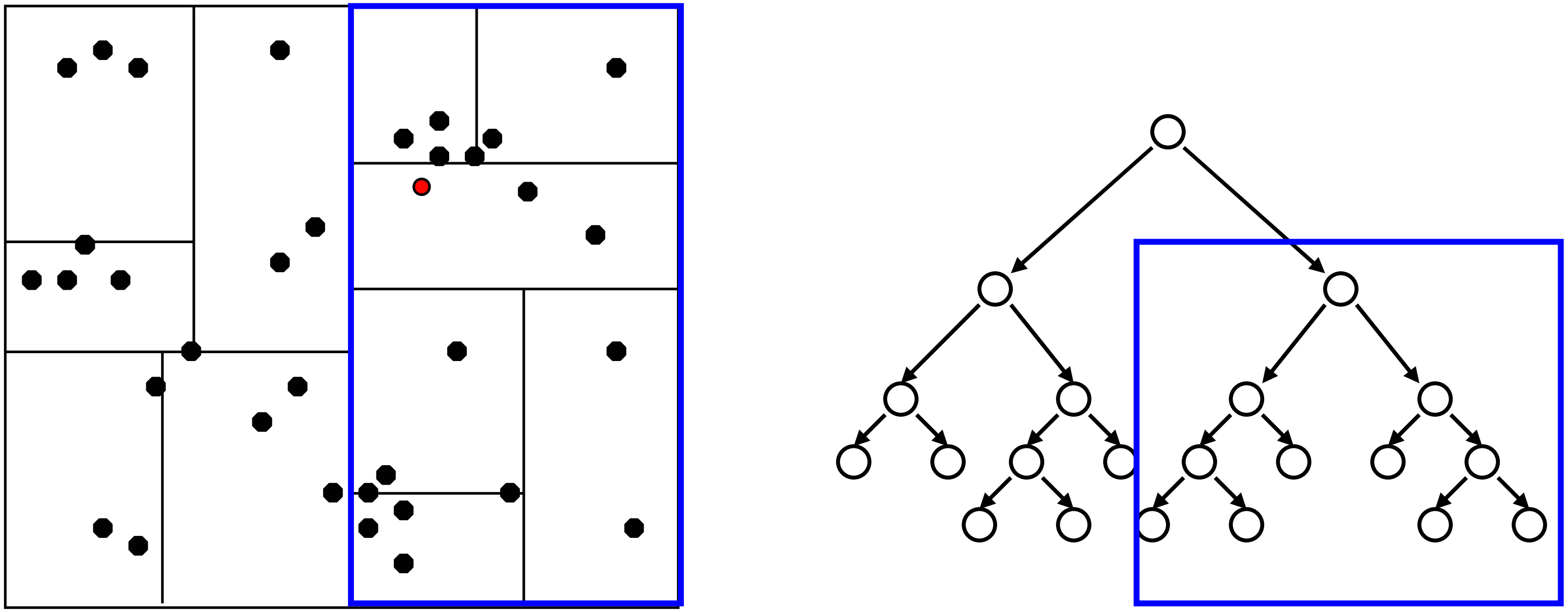
Nearest Neighbor with KD Trees





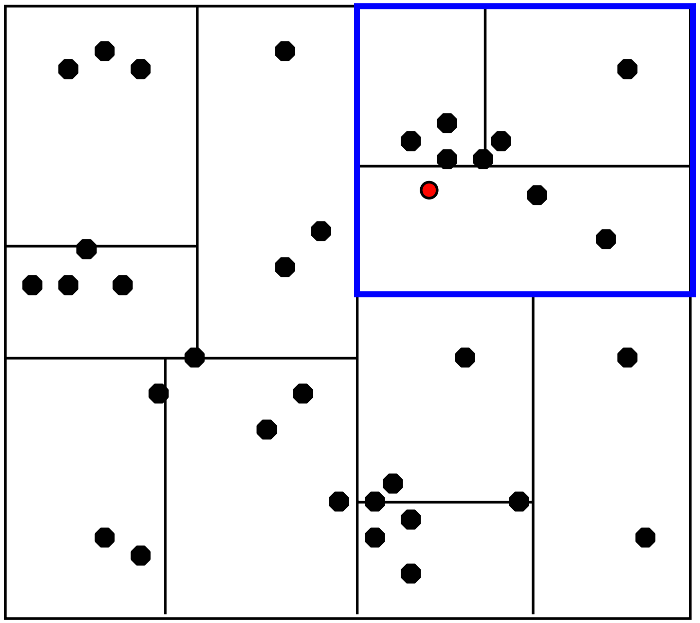
We traverse the tree looking for the nearest neighbor of the query point.

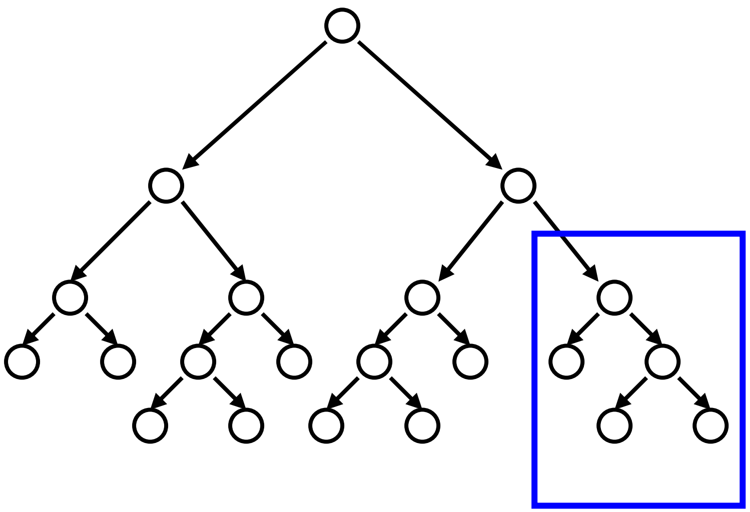
Nearest Neighbor with KD Trees



<u>Examine nearby points first</u>: Explore the branch of the tree that is closest to the query point first.

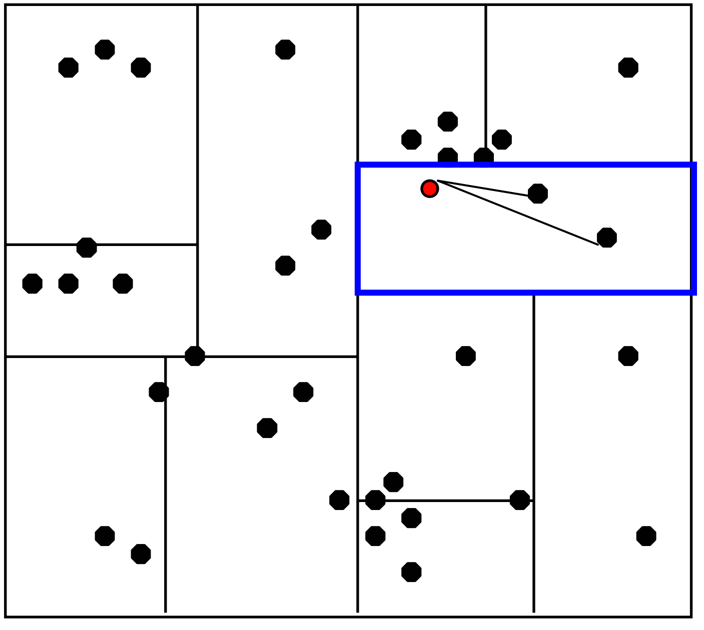
Nearest Neighbor with KD Trees

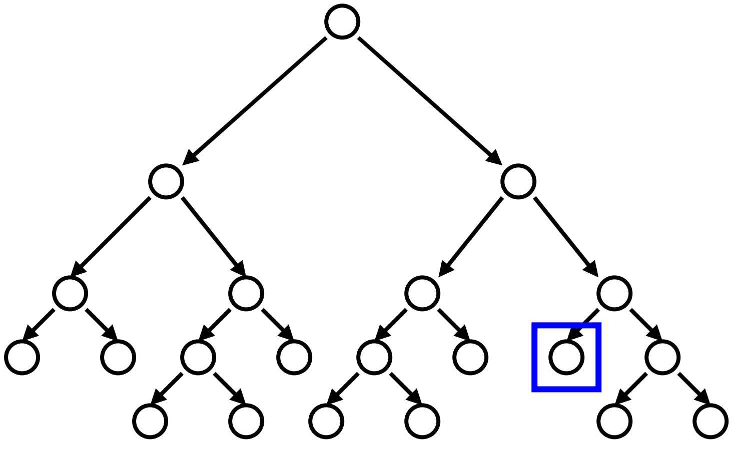




Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

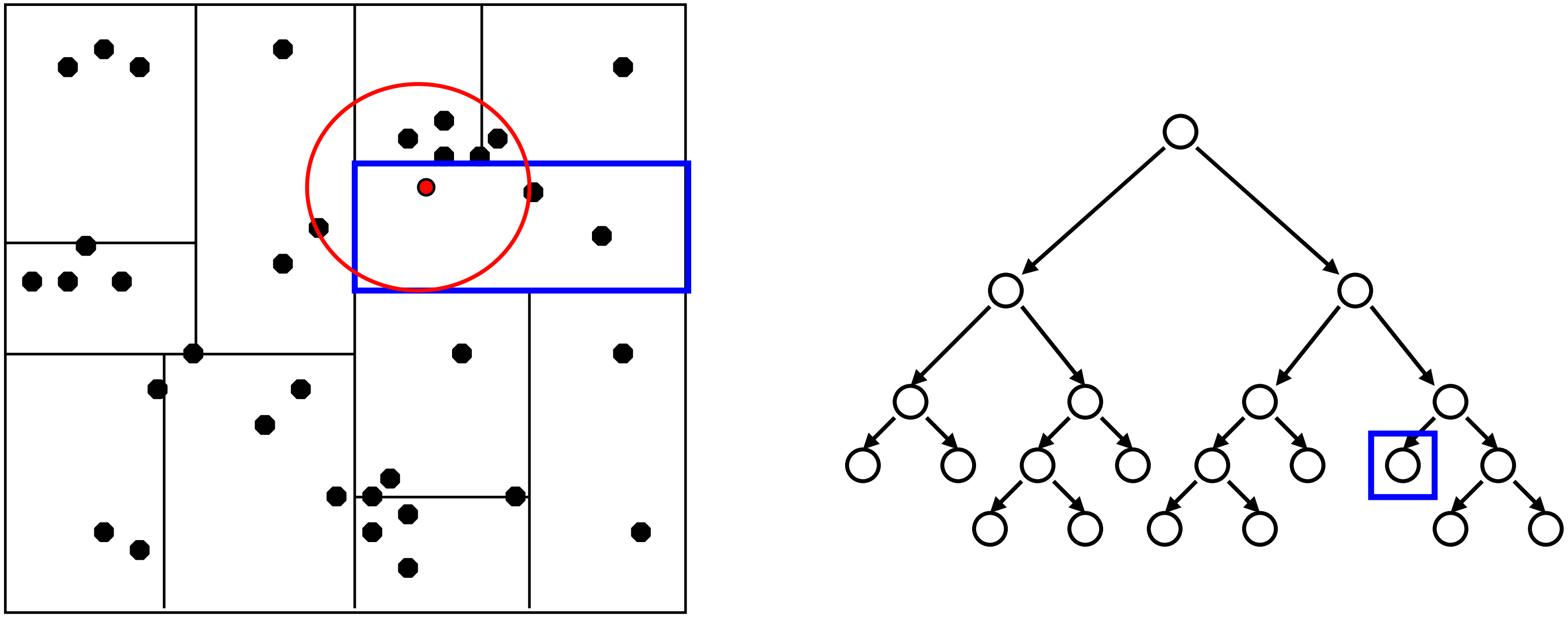
Nearest Neighbor with KD Trees





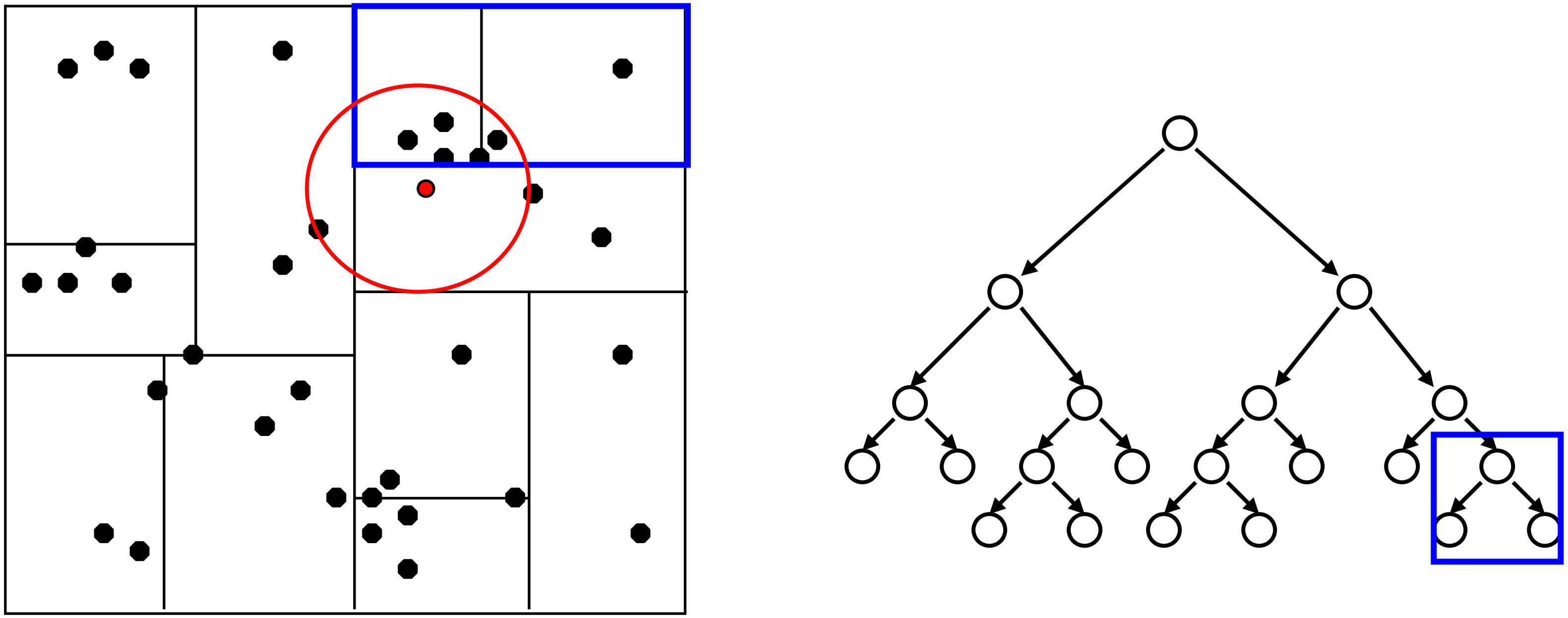
When we reach a leaf node: compute the distance to each point in the node.

Nearest Neighbor with KD Trees



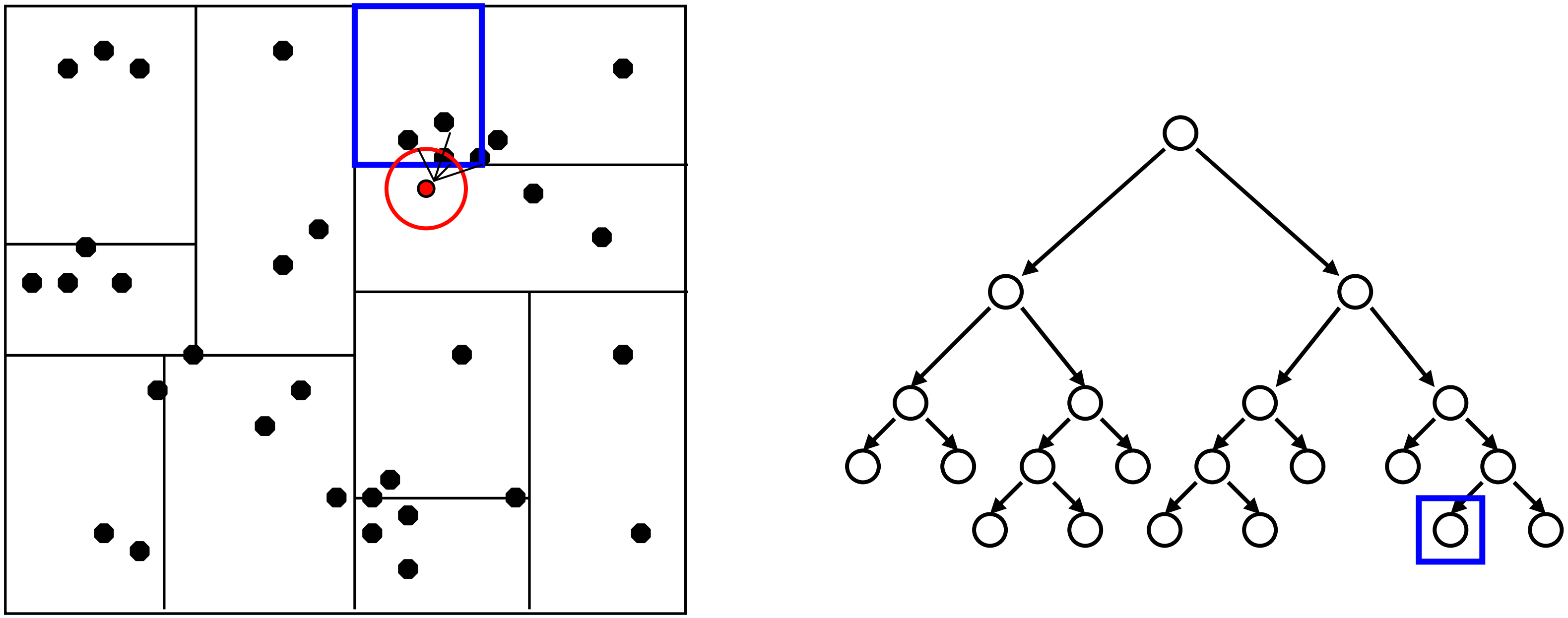
When we reach a leaf node: compute the distance to each point in the node.

Nearest Neighbor with KD Trees



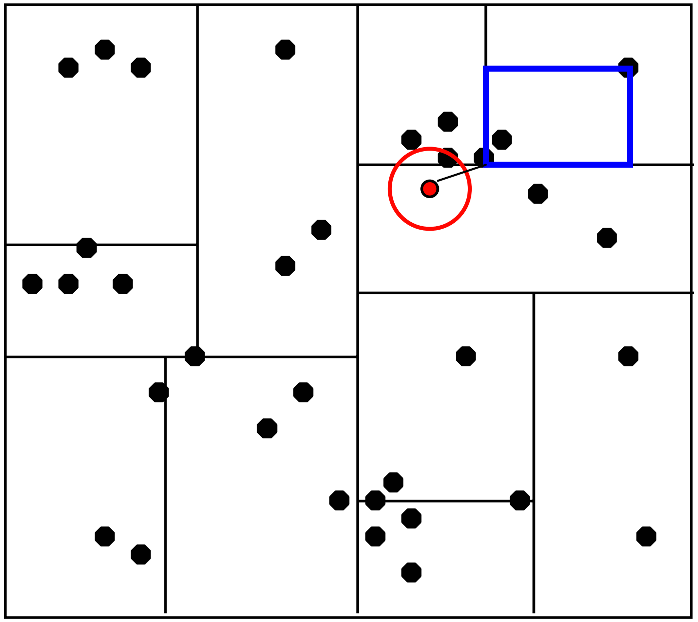
Then we can backtrack and try the other branch at each node visited.

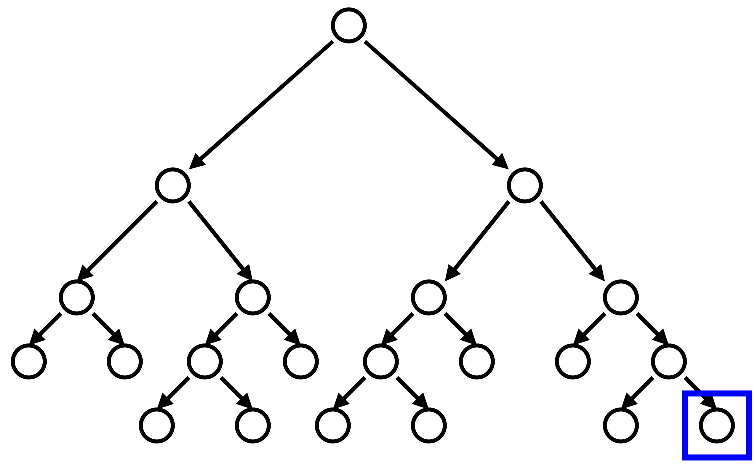
Nearest Neighbor with KD Trees



Each time a new closest node is found, we can update the distance bounds.

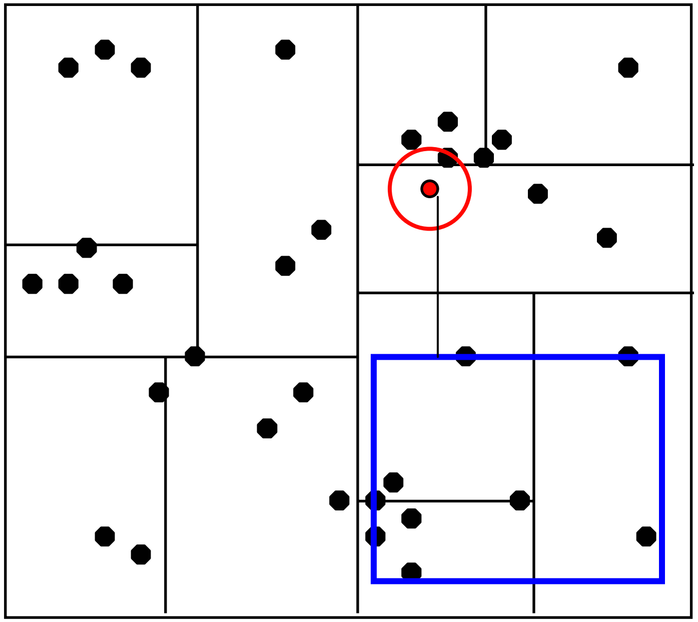
Nearest Neighbor with KD Trees

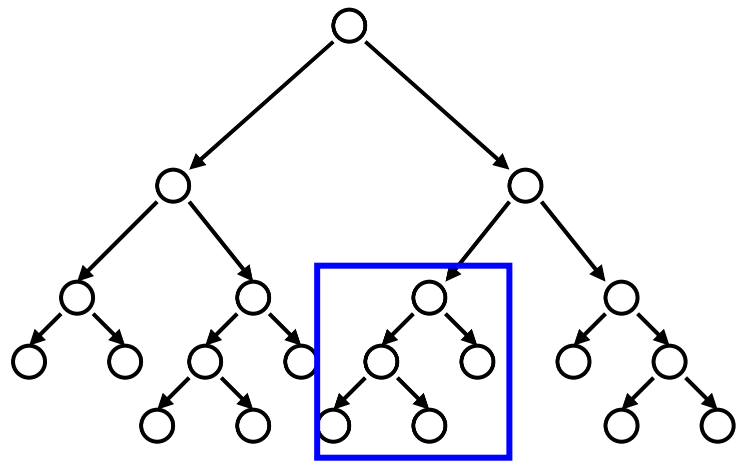




Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

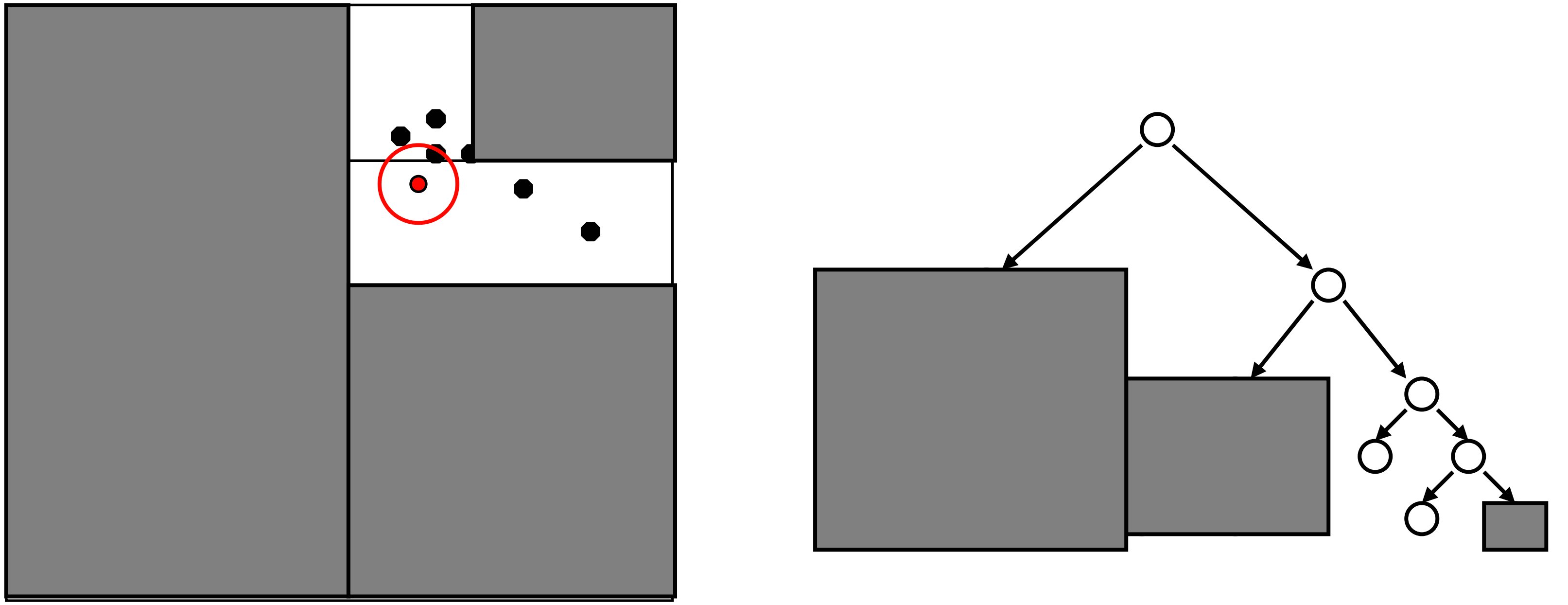
Nearest Neighbor with KD Trees





Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Metric Trees

- Kd-trees rendered worse-than-useless in higher dimensions
- Only requires metric space (a well-behaved distance function)

Outline

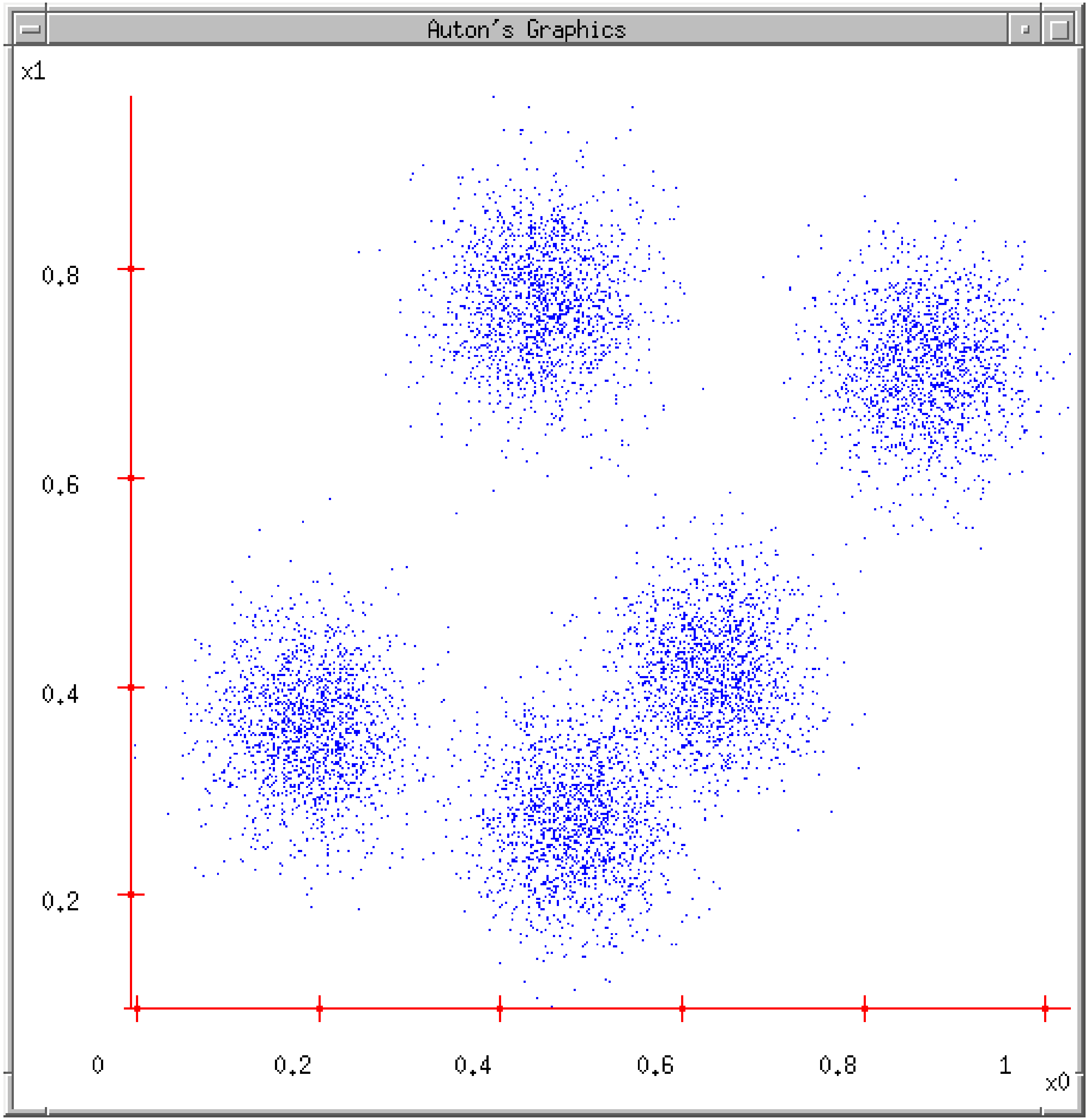
- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ➢ Fast K-means clustering
  - ❑ Fast kernel density estimation

- ❑ Large-scale galactic morphology

What does k-means do?

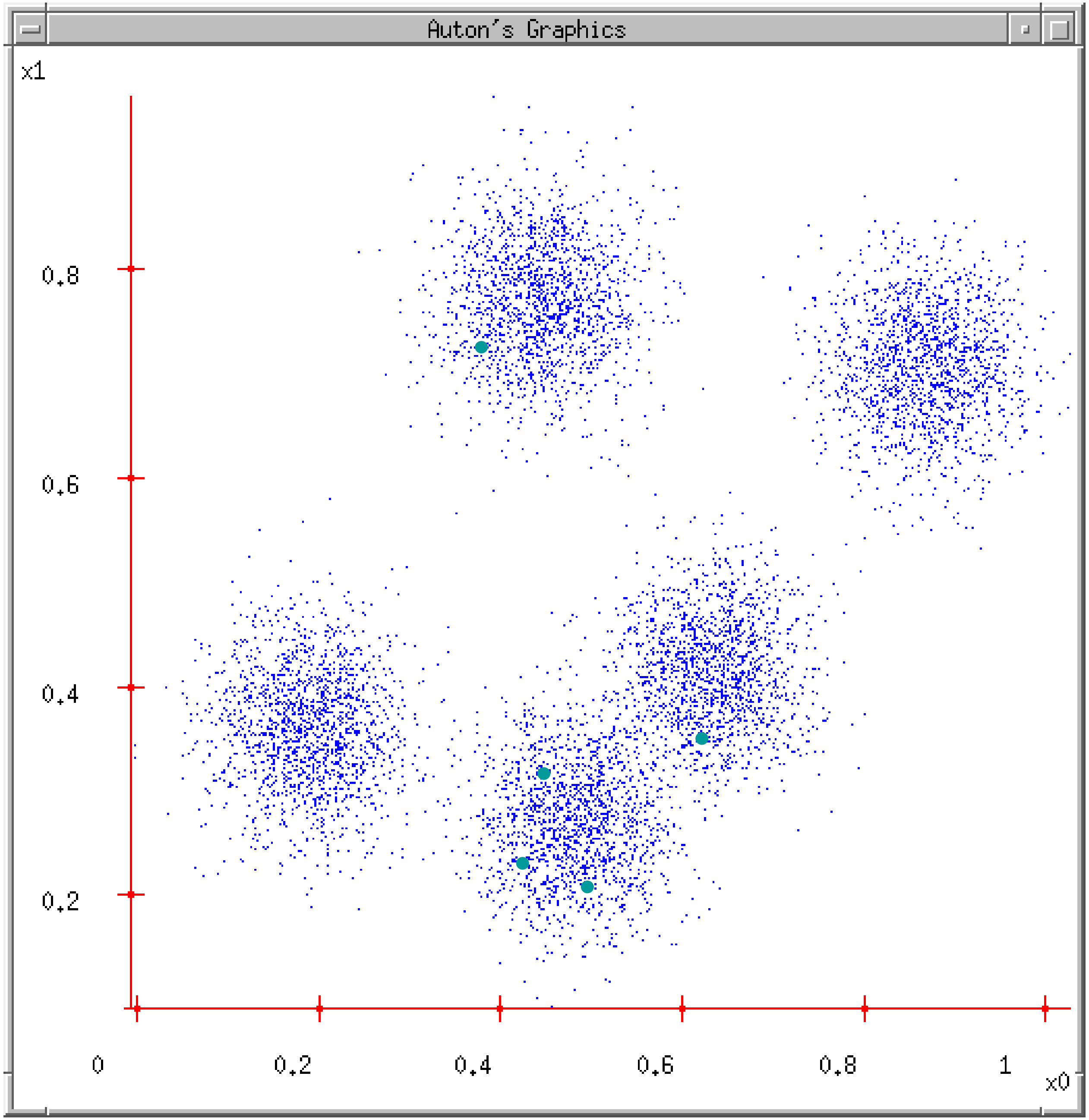
K-means

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



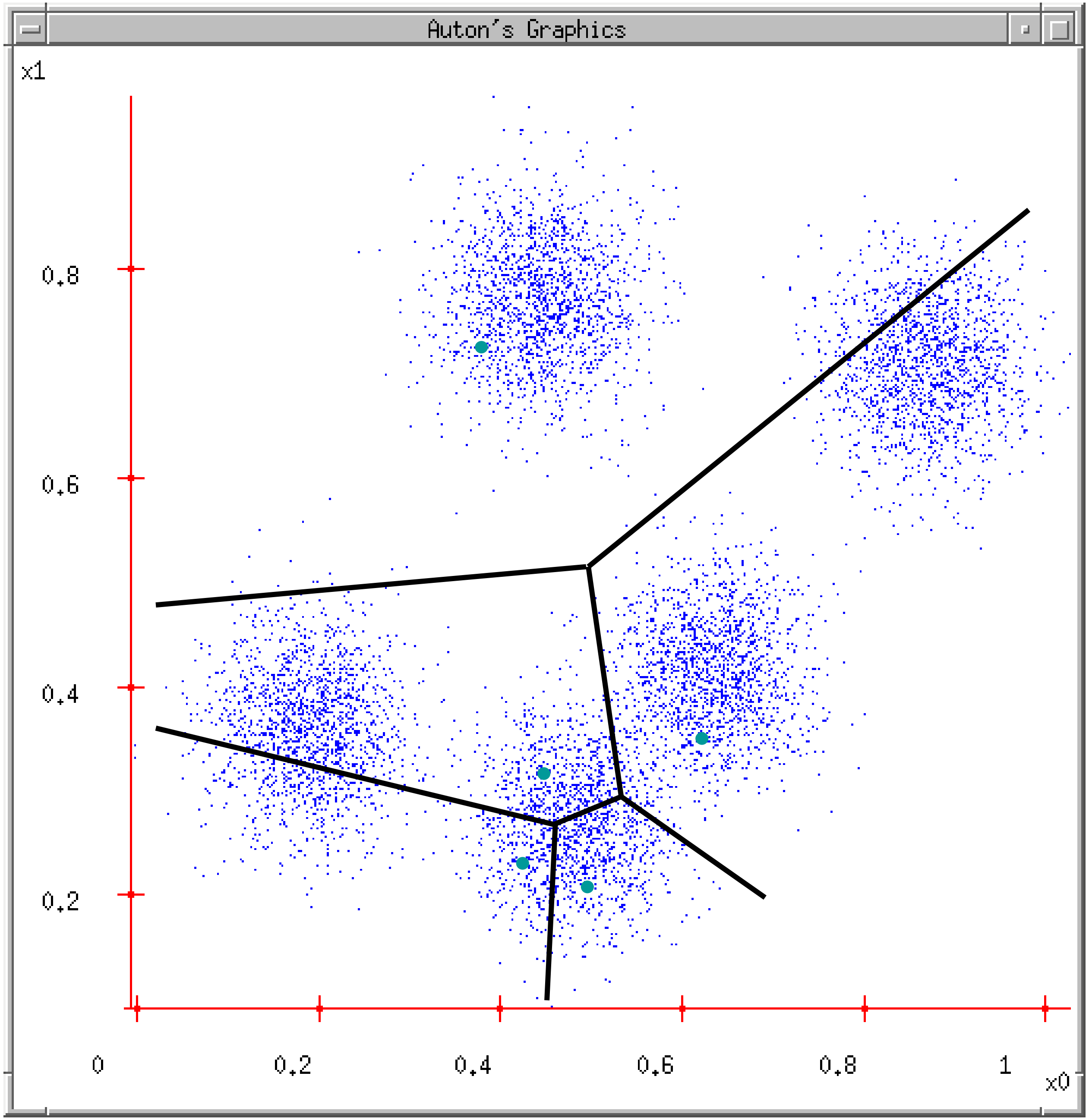
K-means

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



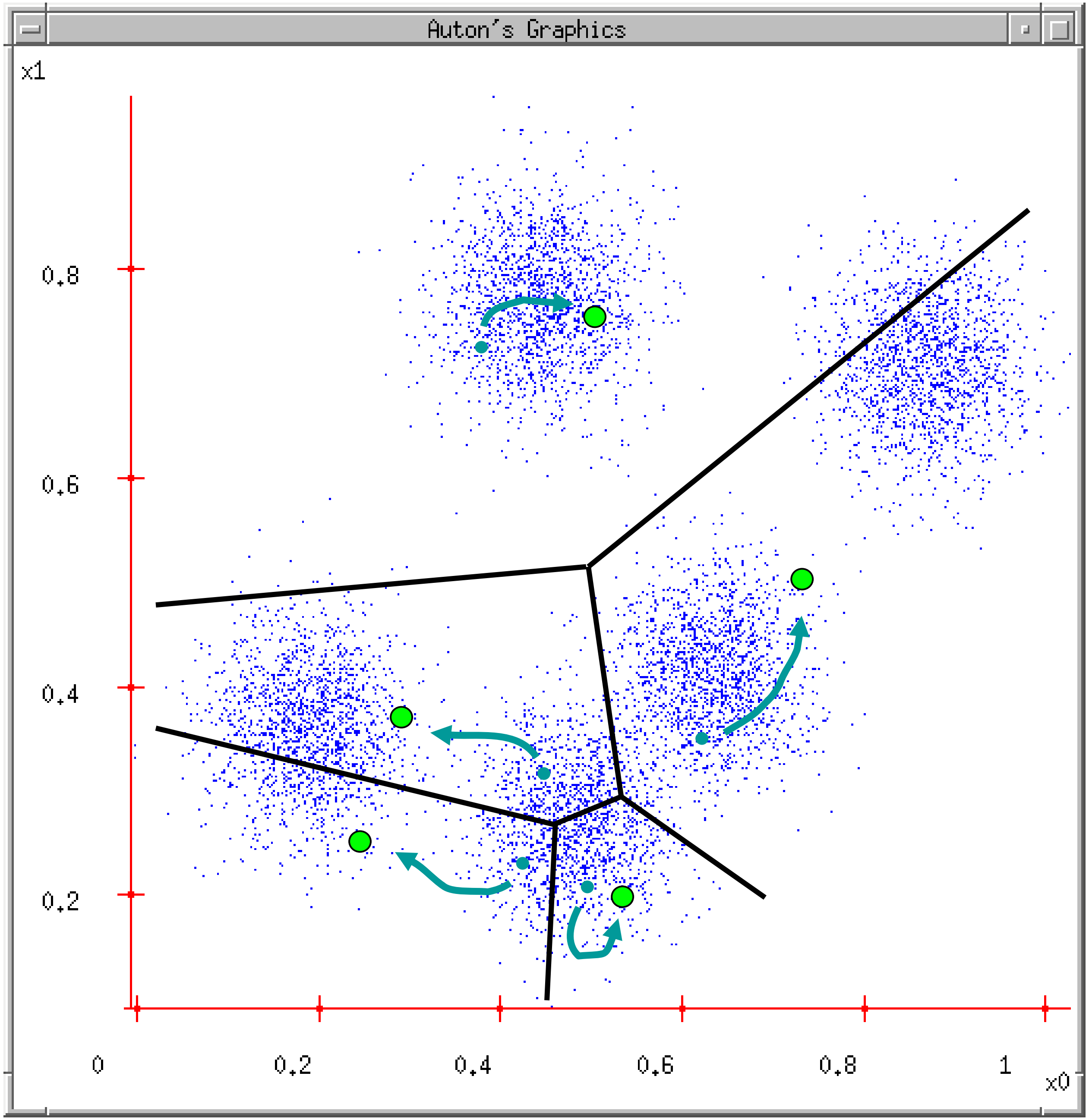
K-means

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)



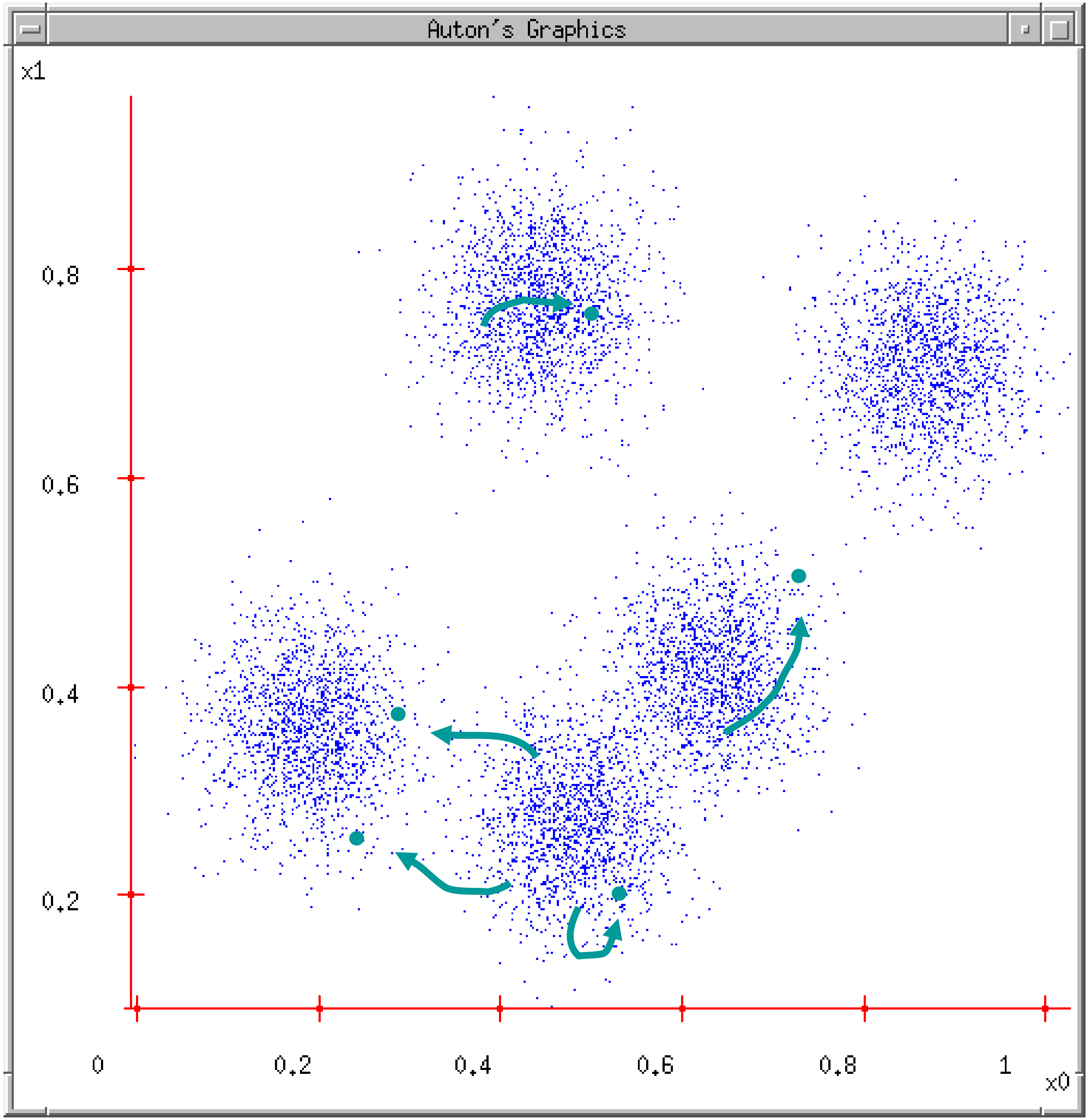
K-means

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!



- For basic tutorial on kmeans, see any machine learning text book , or www.cs.cmu.edu/~awm/tutorials/kmeans.html

K-means

1. As[...] clusters they'd like. *(e.g. k=5)*

2. Randomly gues[...] cluster Center locations

3. Each datapoint [...] out which Center it's closest to.

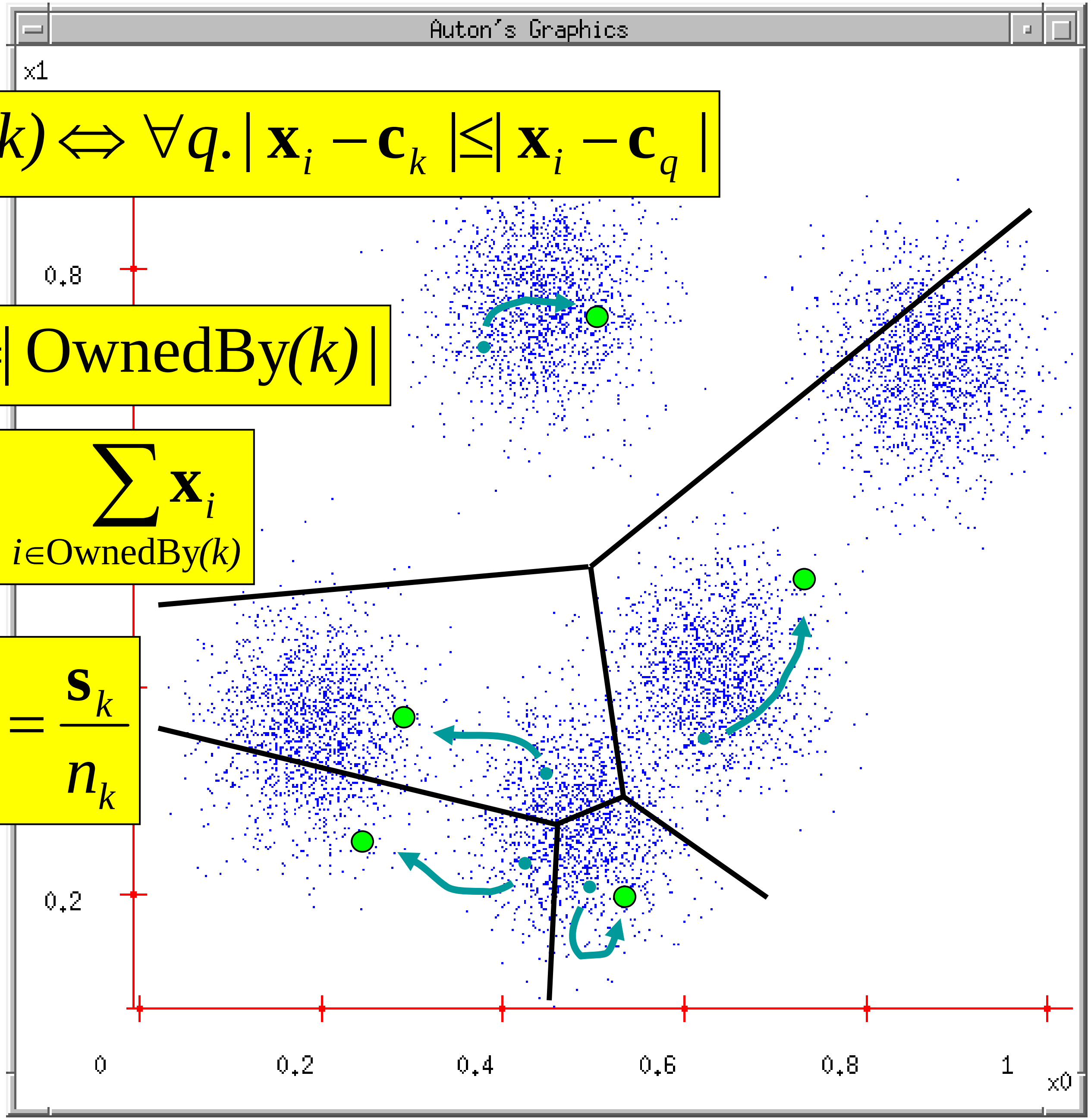
4. Each Center fin[...] the centroid of t[...] points it owns

$$i \in \mathrm{OwnedBy}(k) \Leftrightarrow \forall q. |\mathbf{x}_i - \mathbf{c}_k| \le |\mathbf{x}_i - \mathbf{c}_q|$$

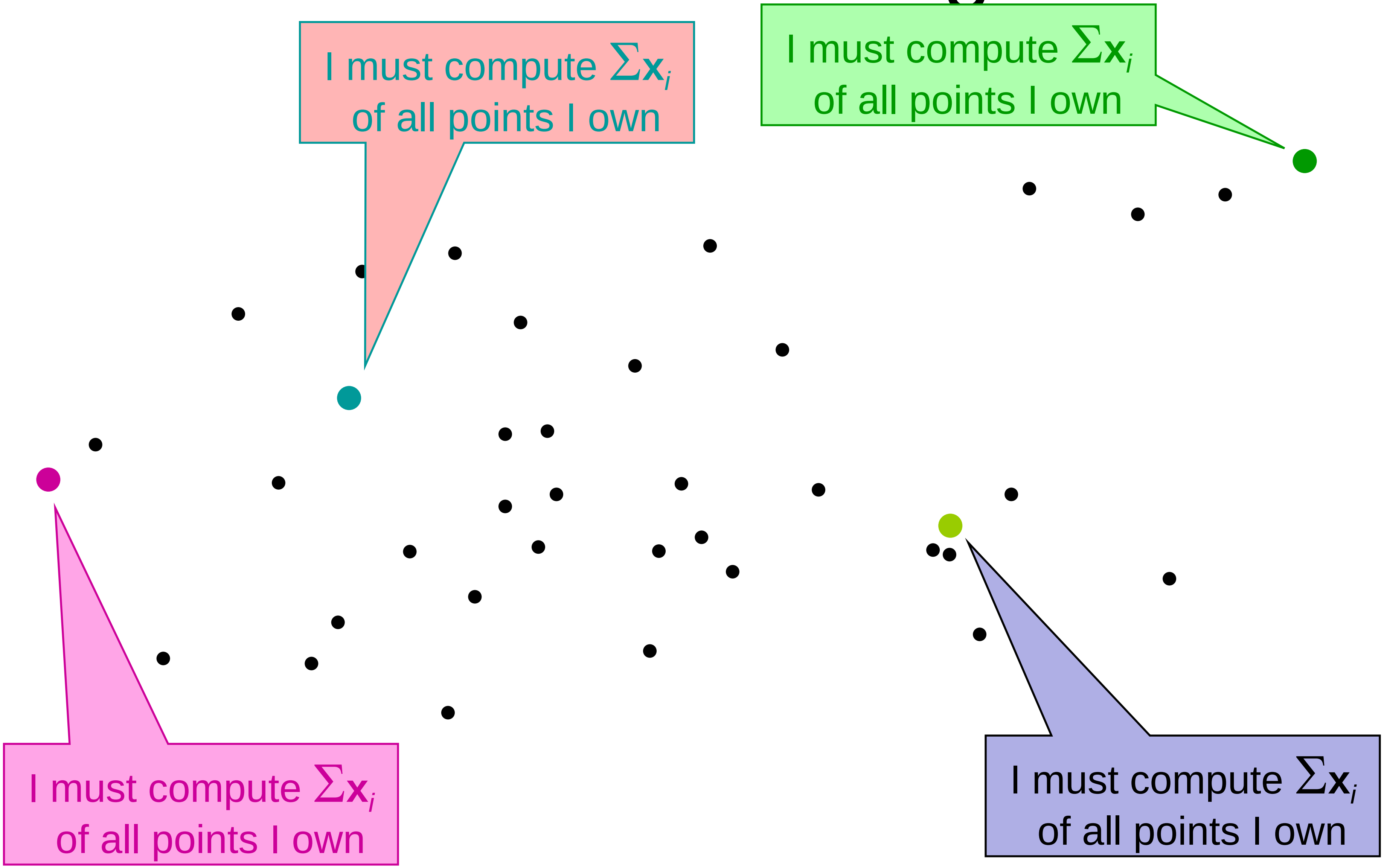
$$n_k = |\mathrm{OwnedBy}(k)|$$

$$\mathbf{s}_k = \sum_{i \in \mathrm{OwnedBy}(k)} \mathbf{x}_i$$

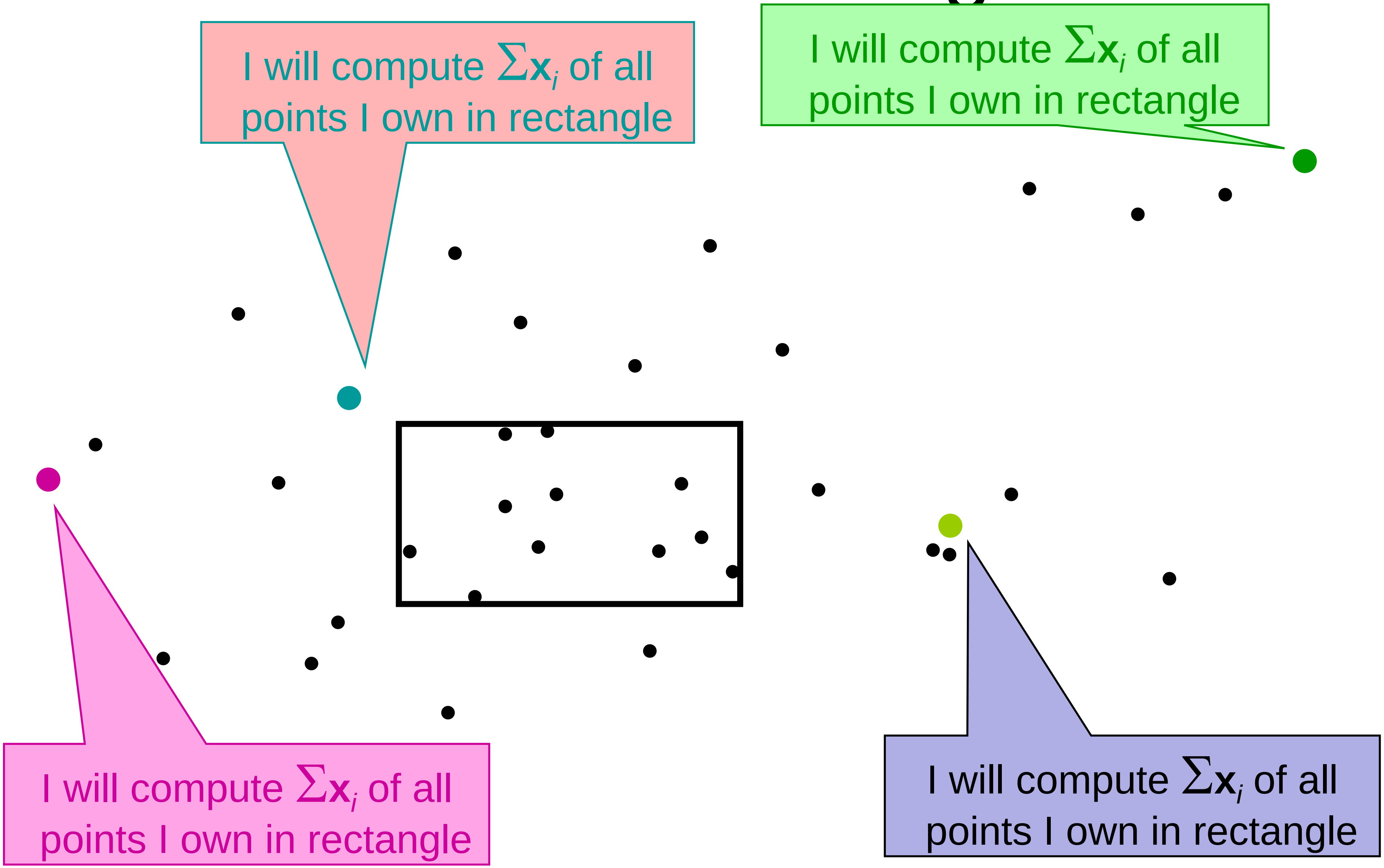
$$\mathbf{c}_k := \frac{\mathbf{s}_k}{n_k}$$



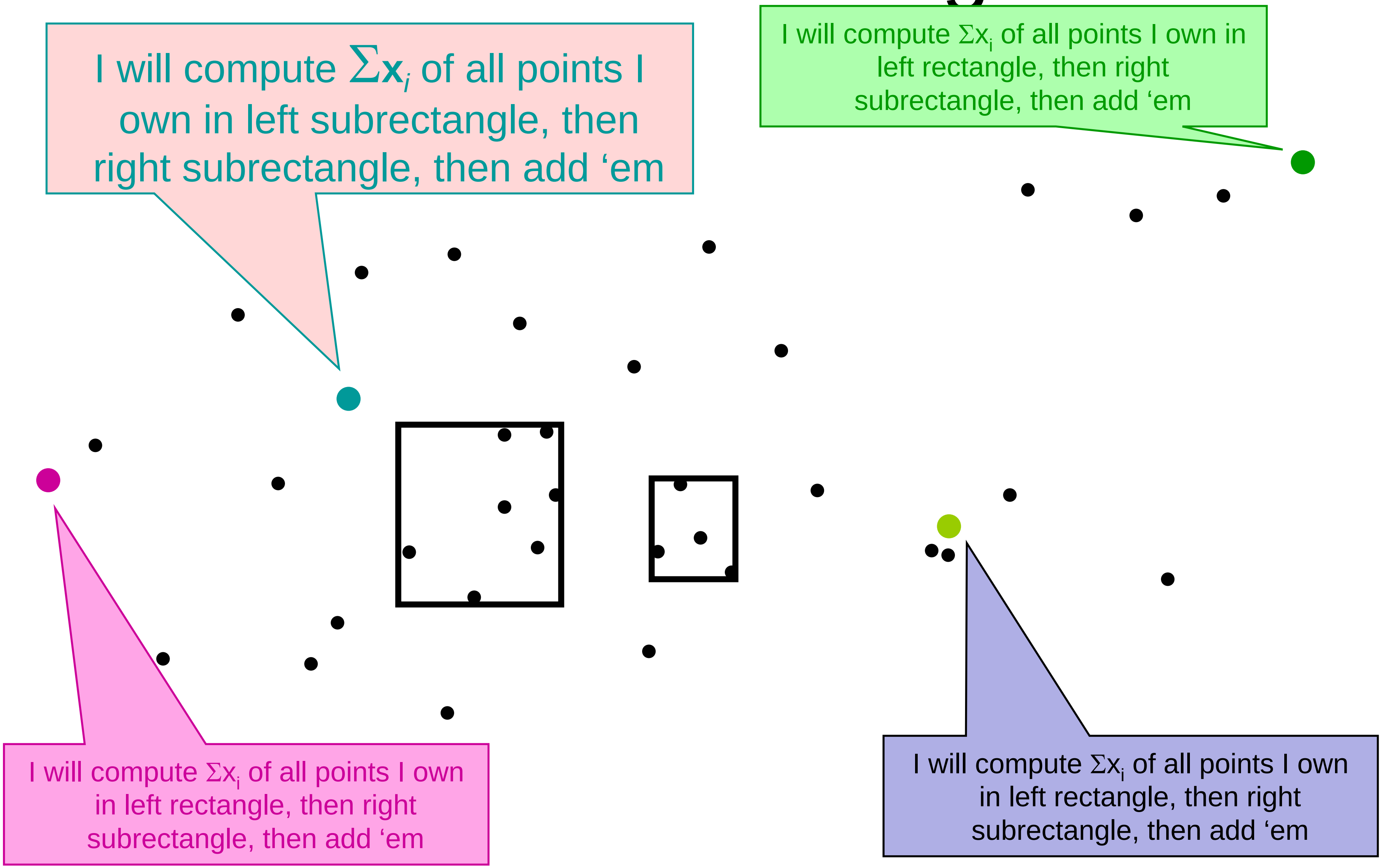
K-means search guts



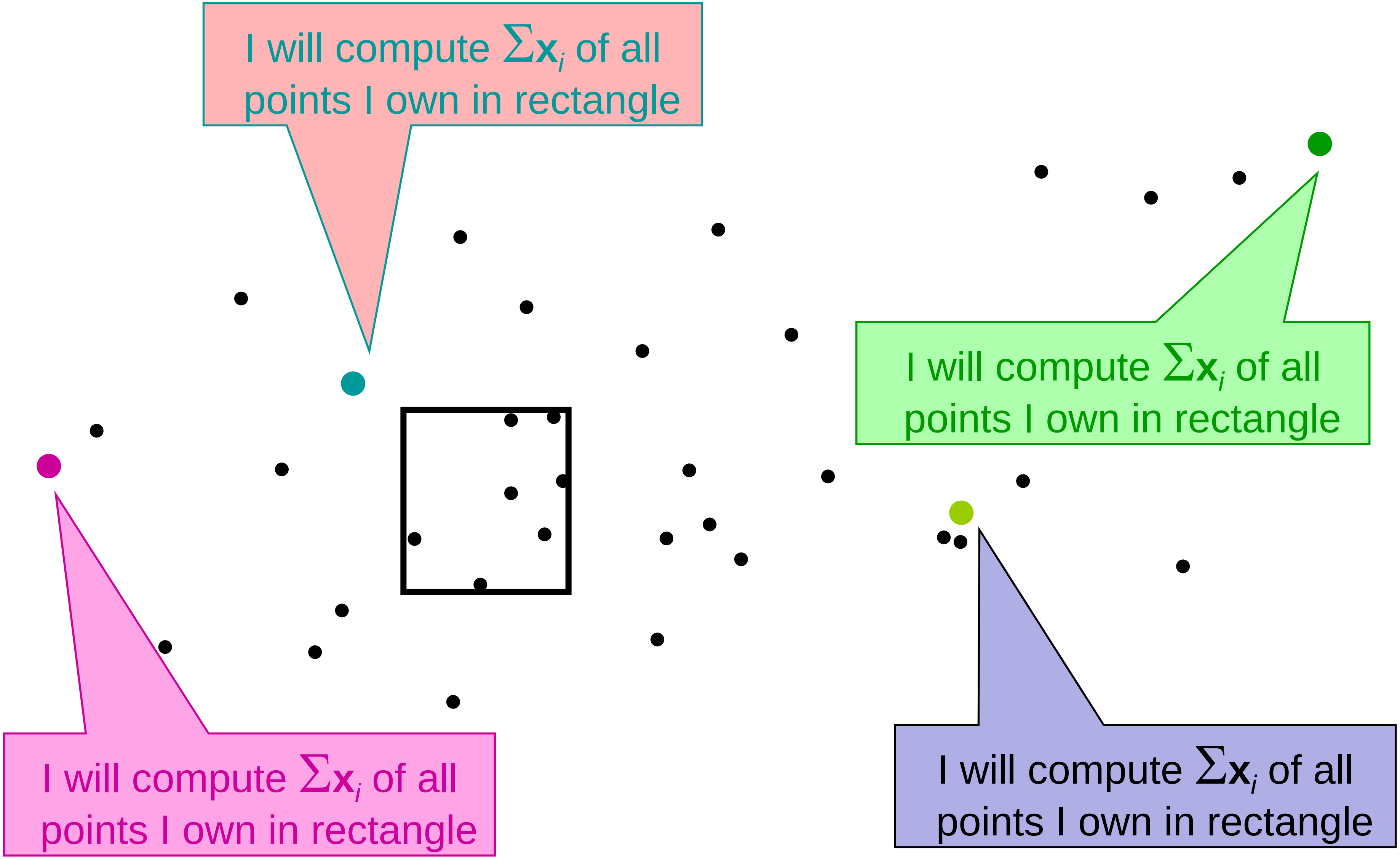
K-means search guts



K-means search guts

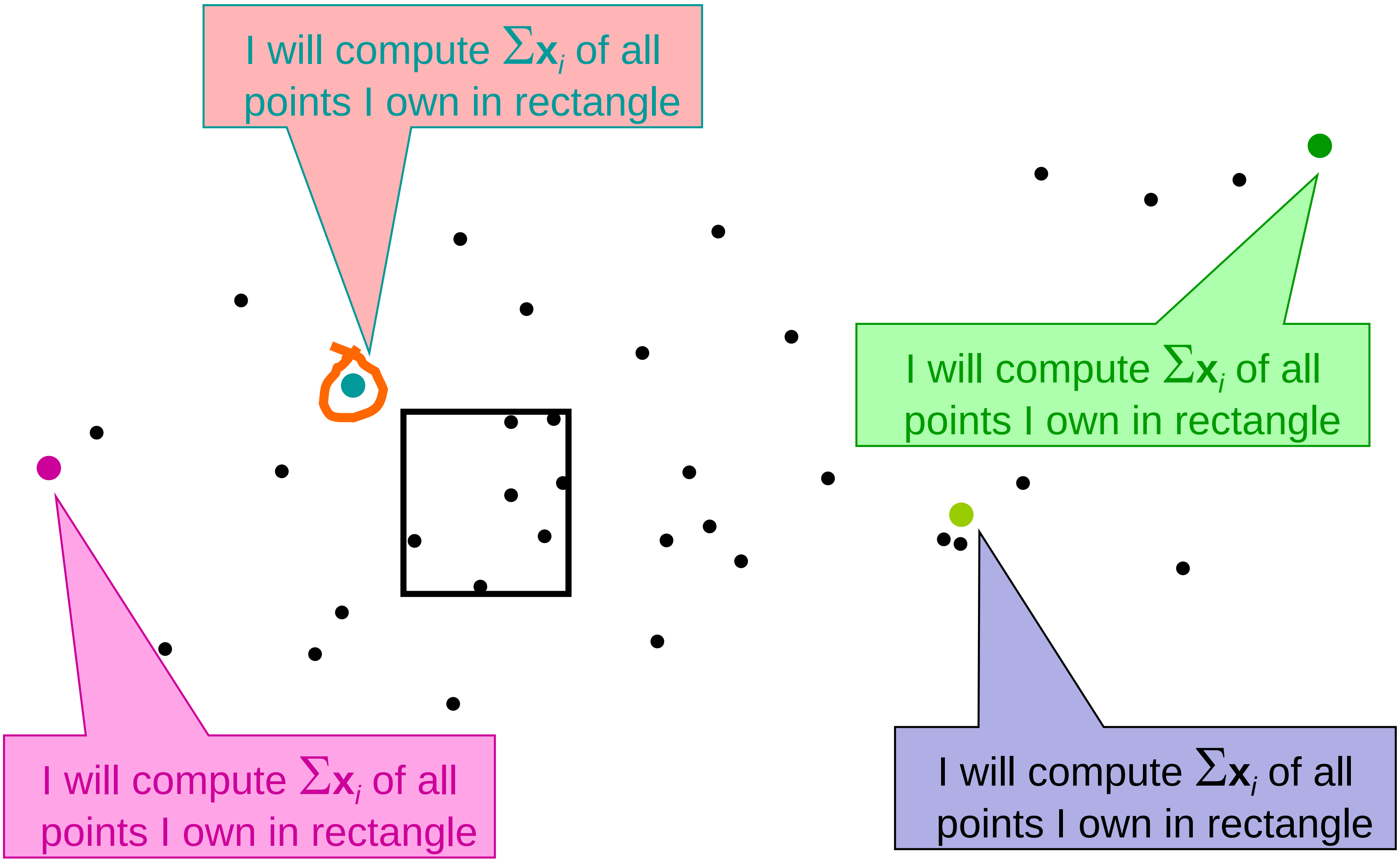


In recursive call…



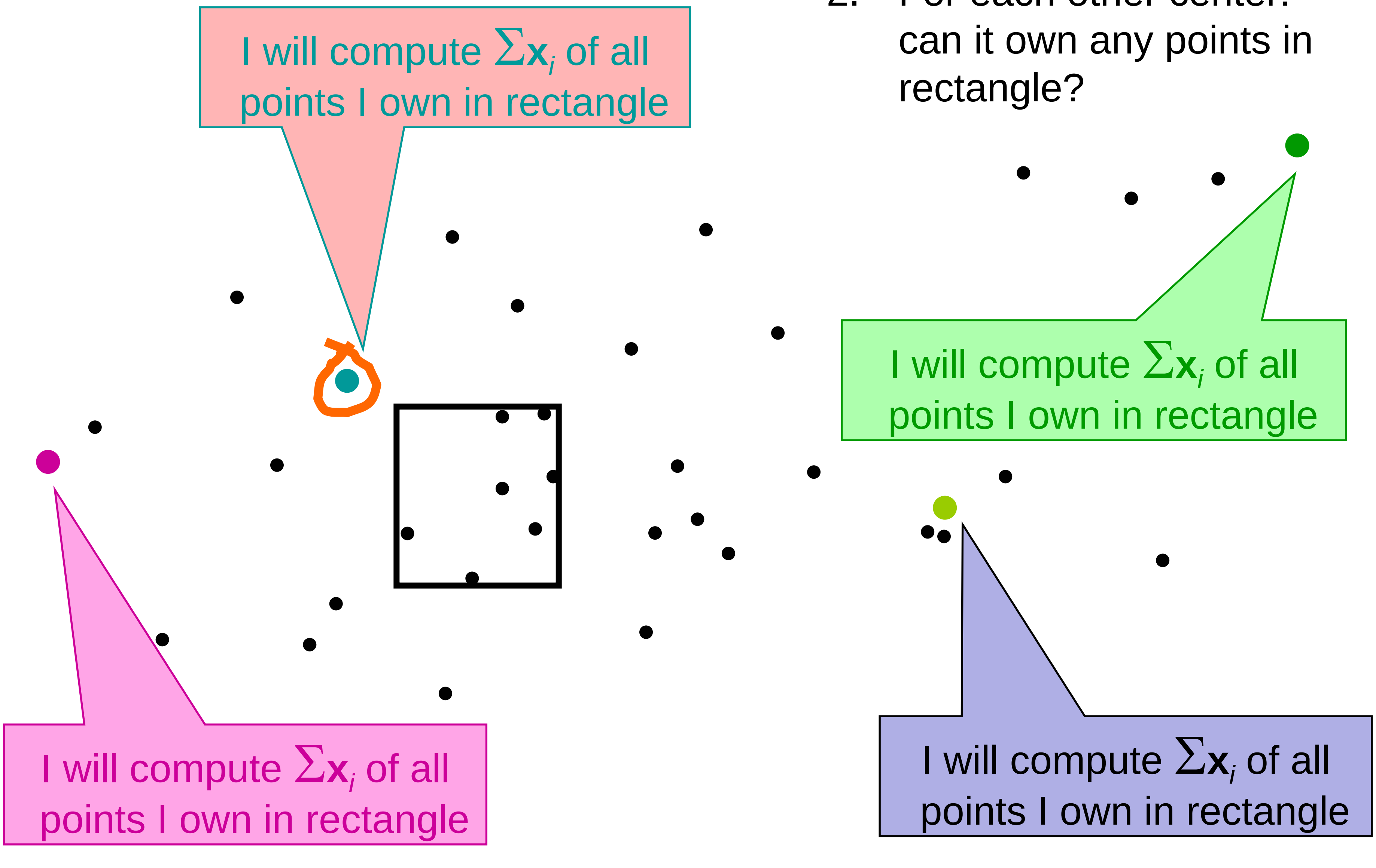
In recursive call…

1. Find center nearest to rectangle



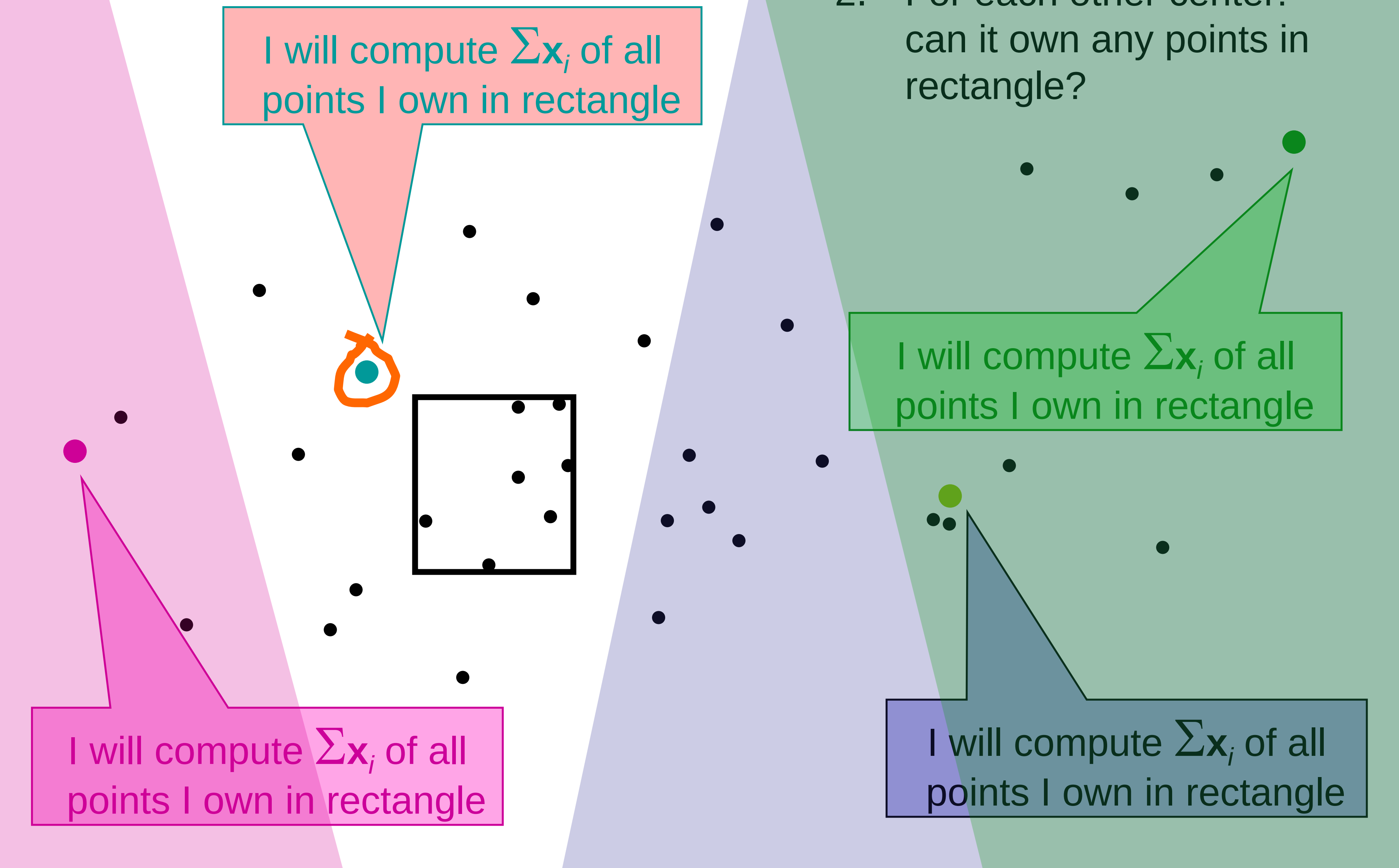
In recursive call…

1. Find center nearest to rectangle
2. For each other center: can it own any points in rectangle?



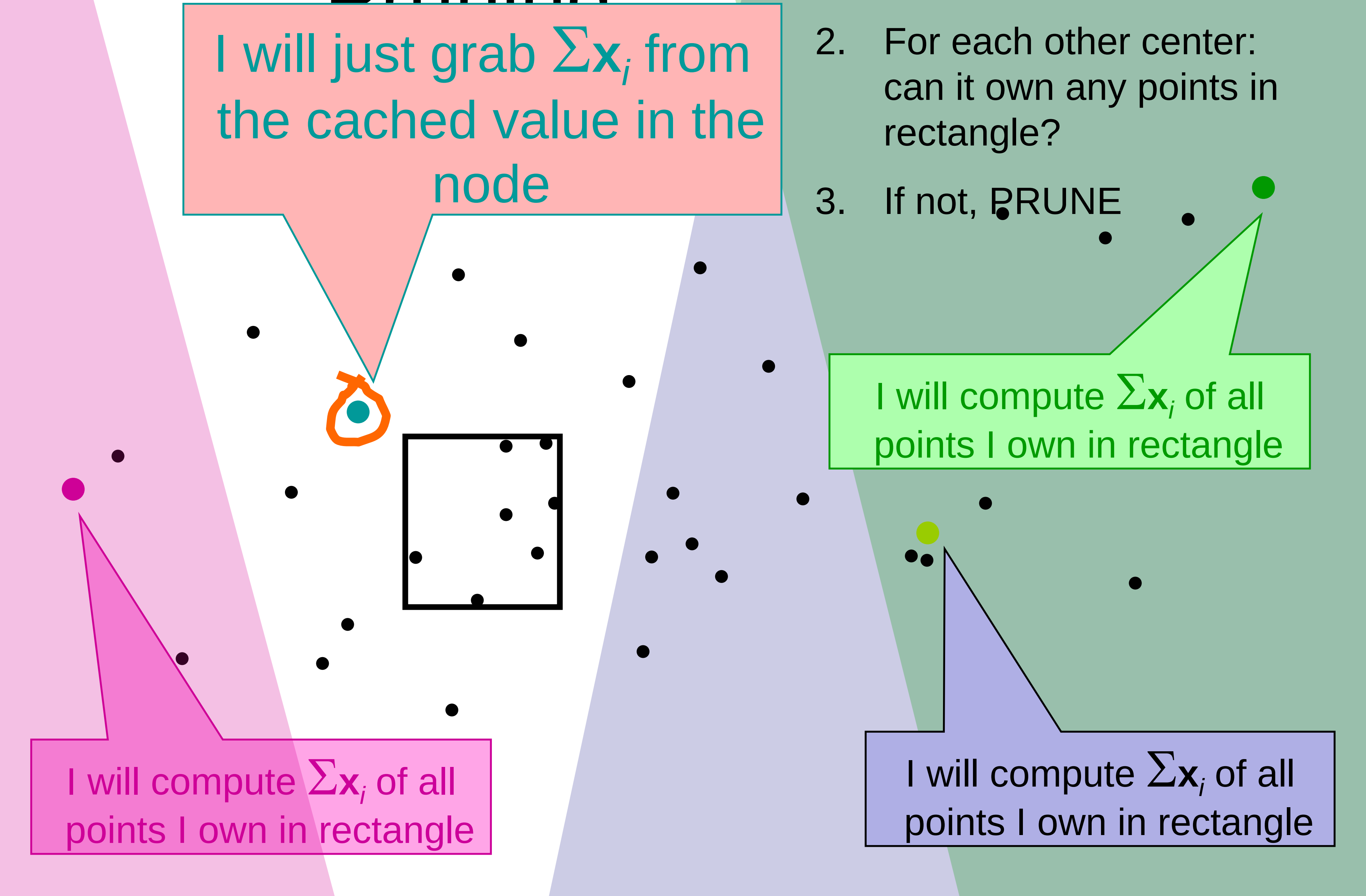
In recursive call…

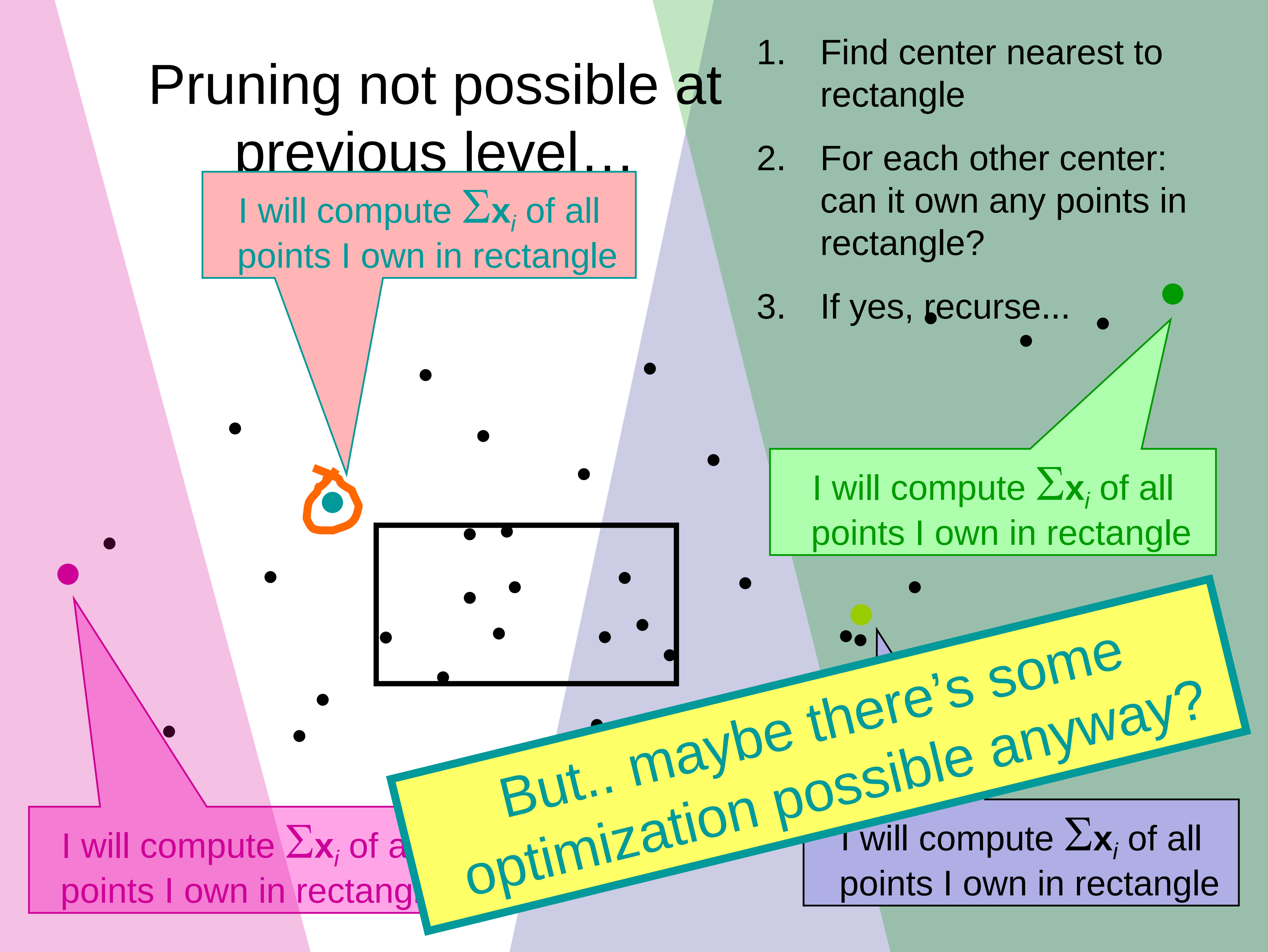
1. Find center nearest to rectangle
2. For each other center: can it own any points in rectangle?



Pruning

1. Find center nearest to rectangle
2. For each other center: can it own any points in rectangle?
3. If not, PRUNE

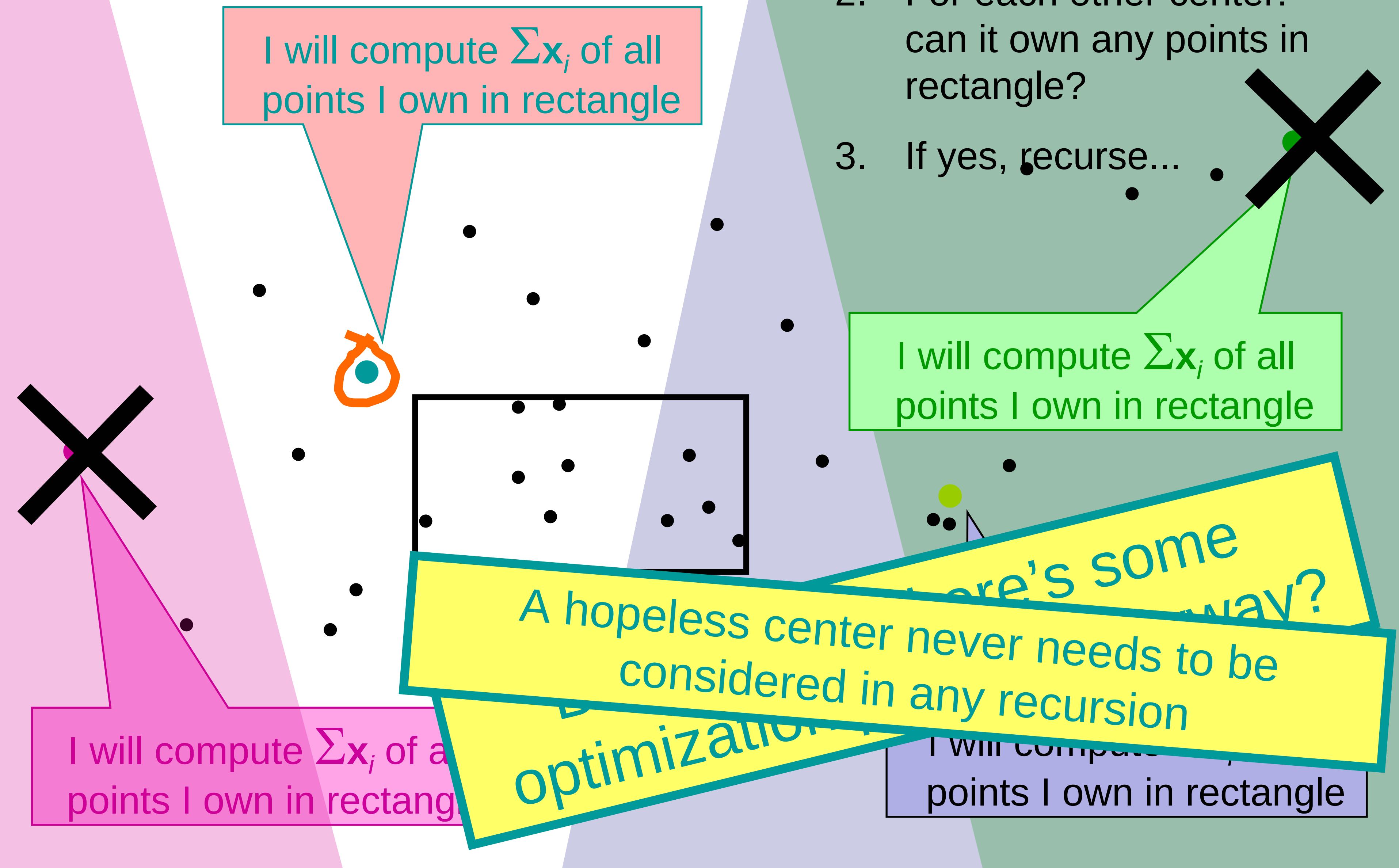




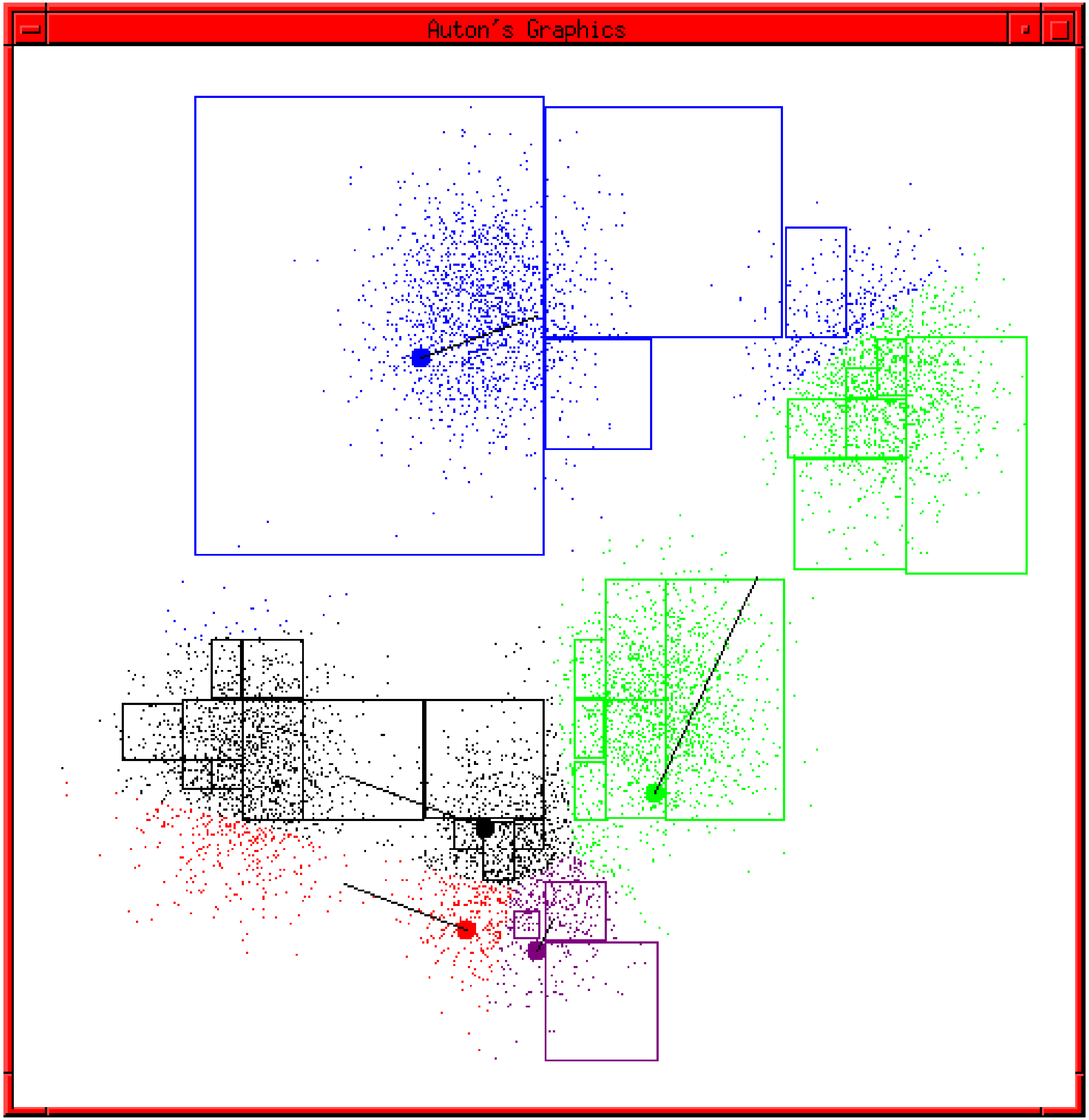
Pruning not possible at previous level...
1. Find center nearest to rectangle
2. For each other center: can it own any points in rectangle?
3. If yes, recurse...
I will compute Σx_i of all points I own in rectangle
I will compute Σx_i of all points I own in rectangle
I will compute Σx_i of a
points I own in rectang
I will compute Σx_i of all points I own in rectangle
But.. maybe there's some optimization possible anyway?

Blacklisting

1. Find center nearest to rectangle
2. For each other center: can it own any points in rectangle?
3. If yes, recurse...



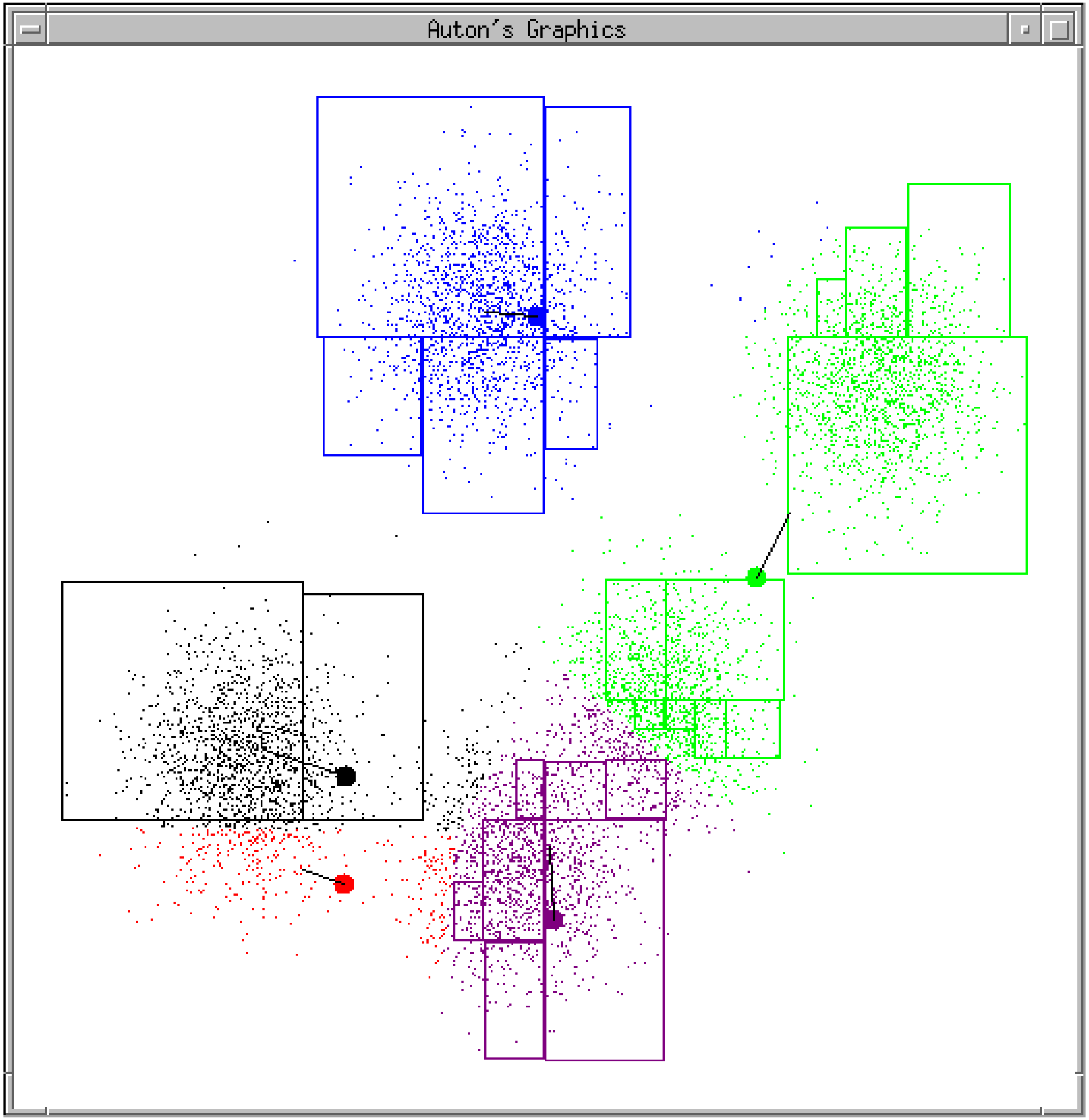
Example



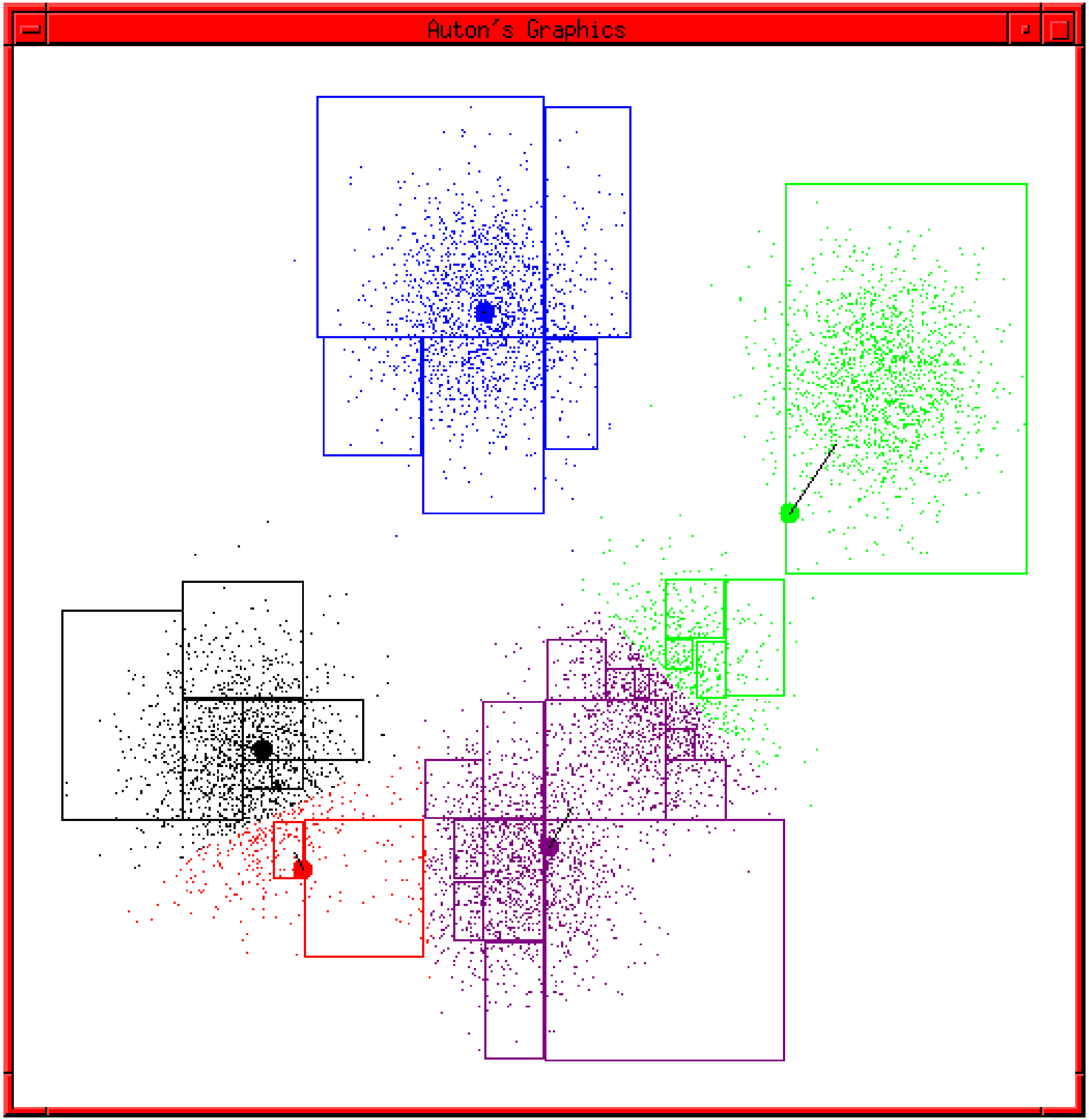
Example generated by:

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 *)*

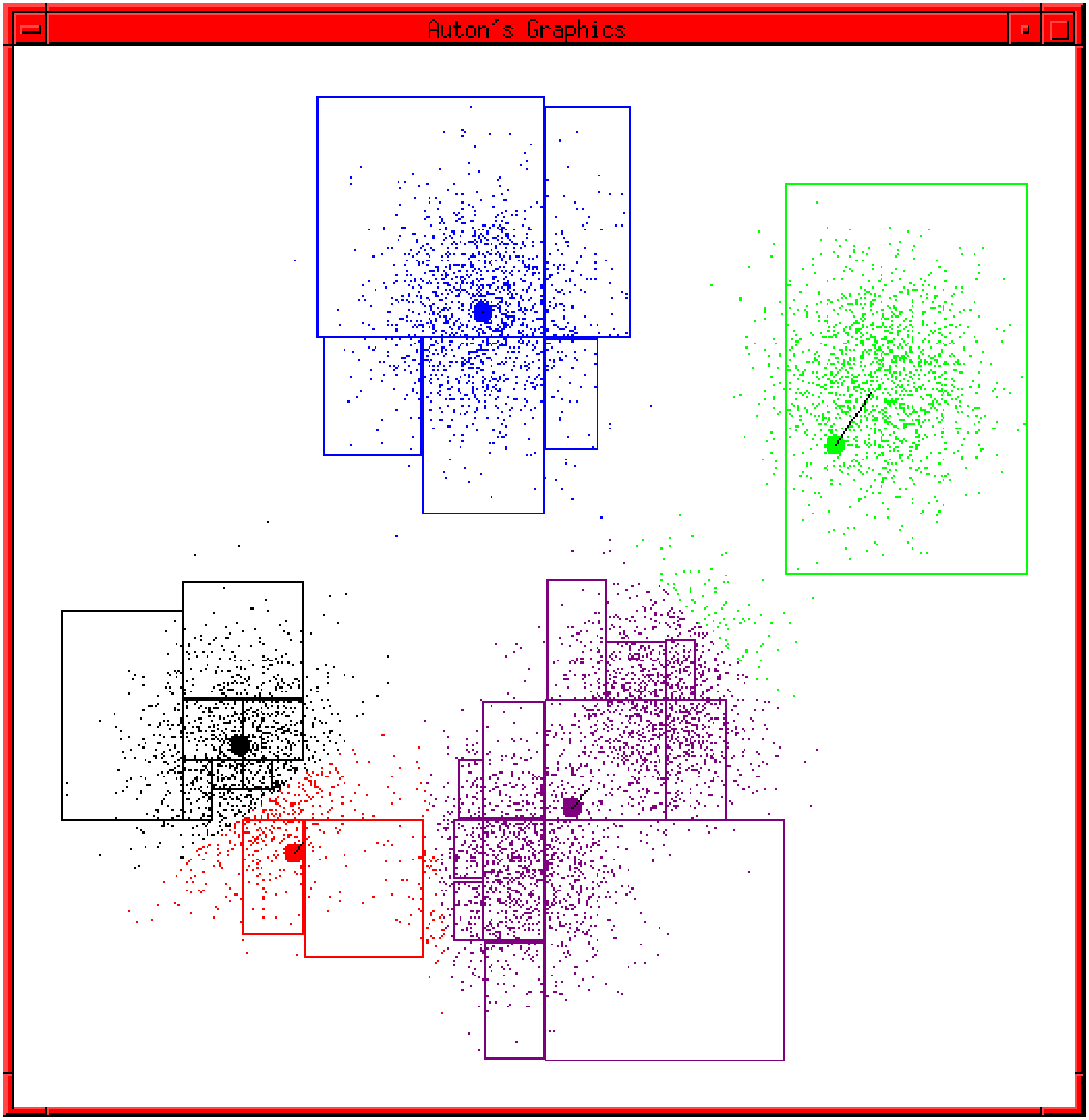
K-means continues …



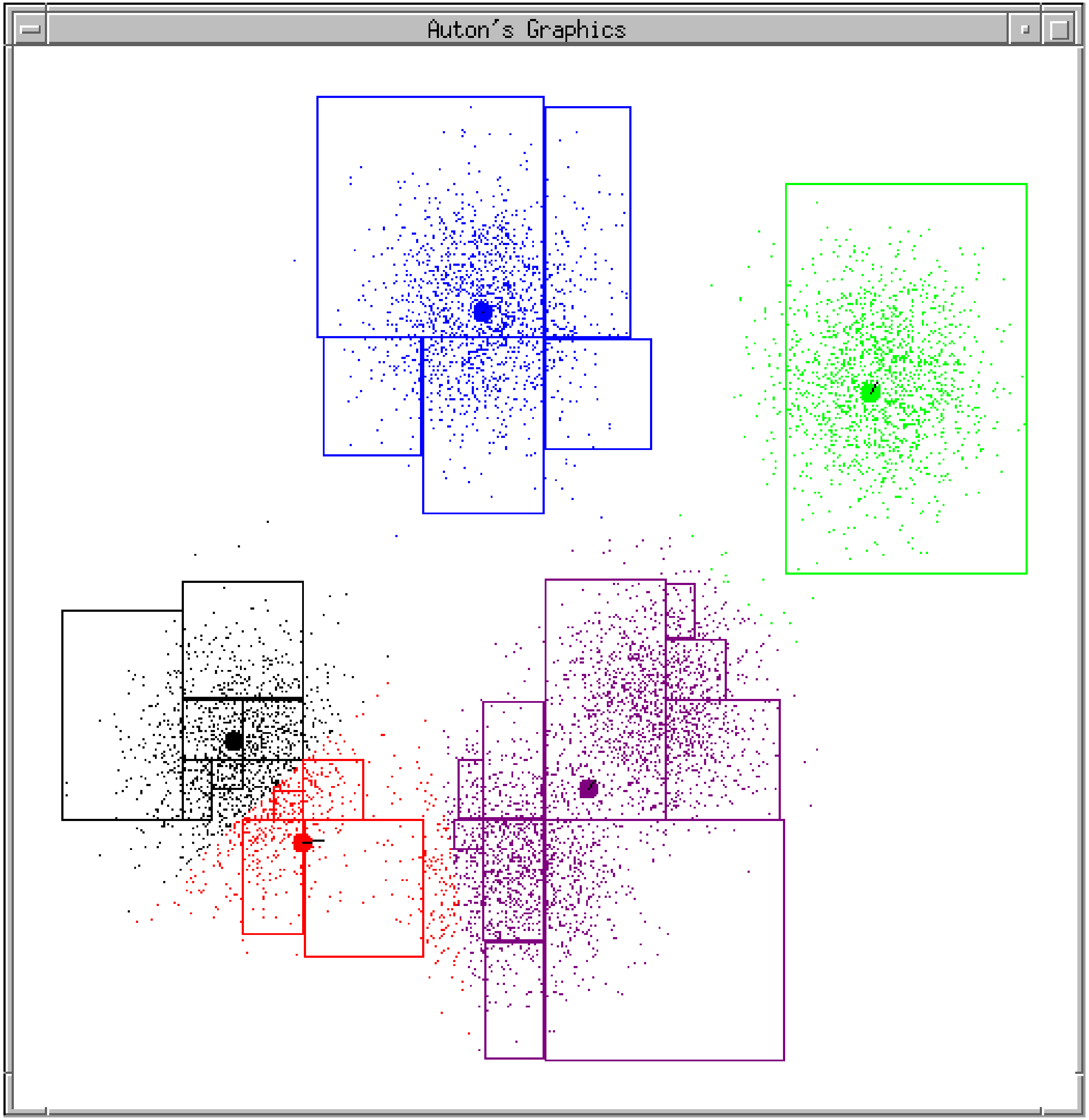
K-means continues …



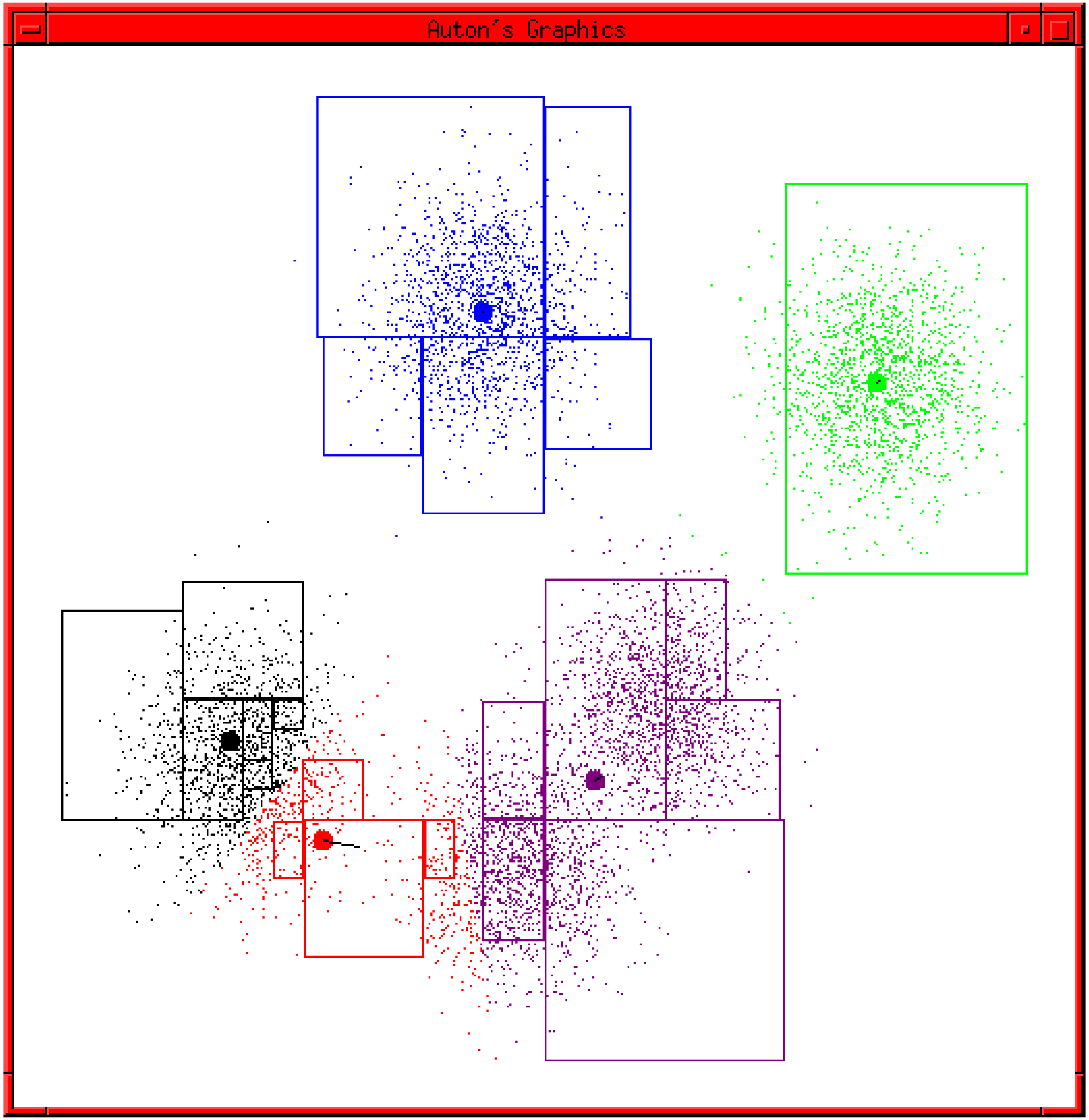
K-means continues …



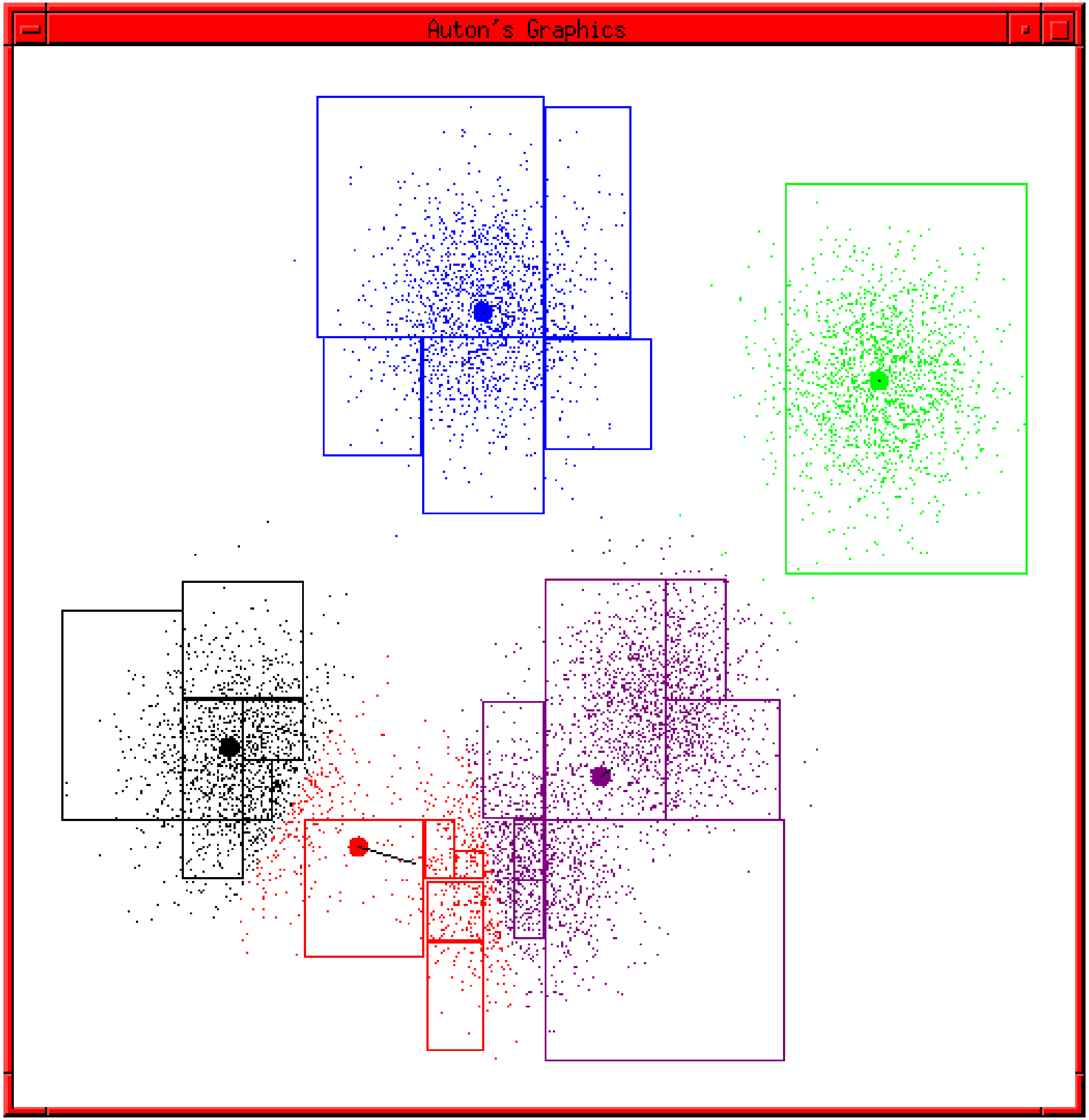
K-means continues …



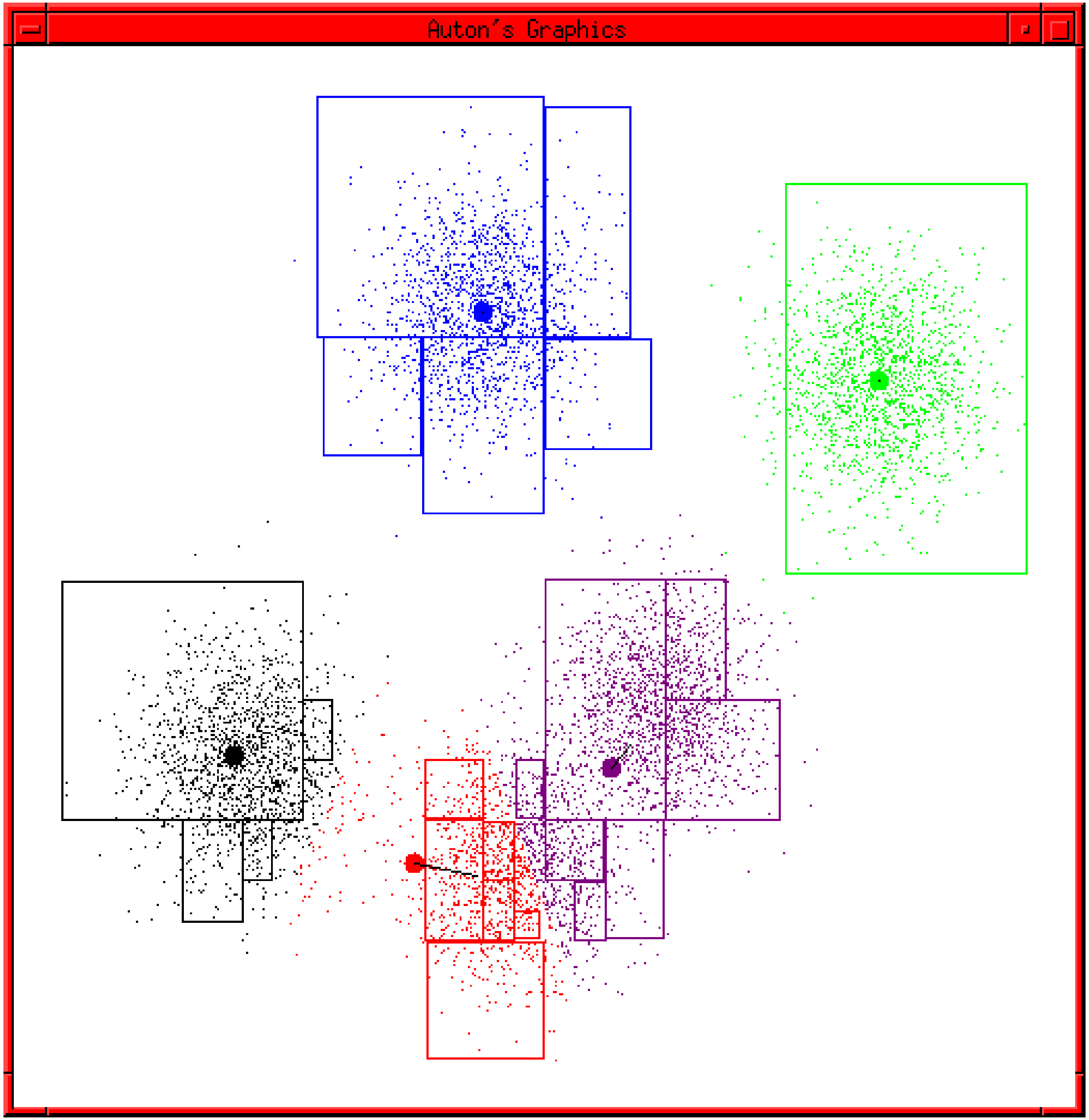
K-means continues …



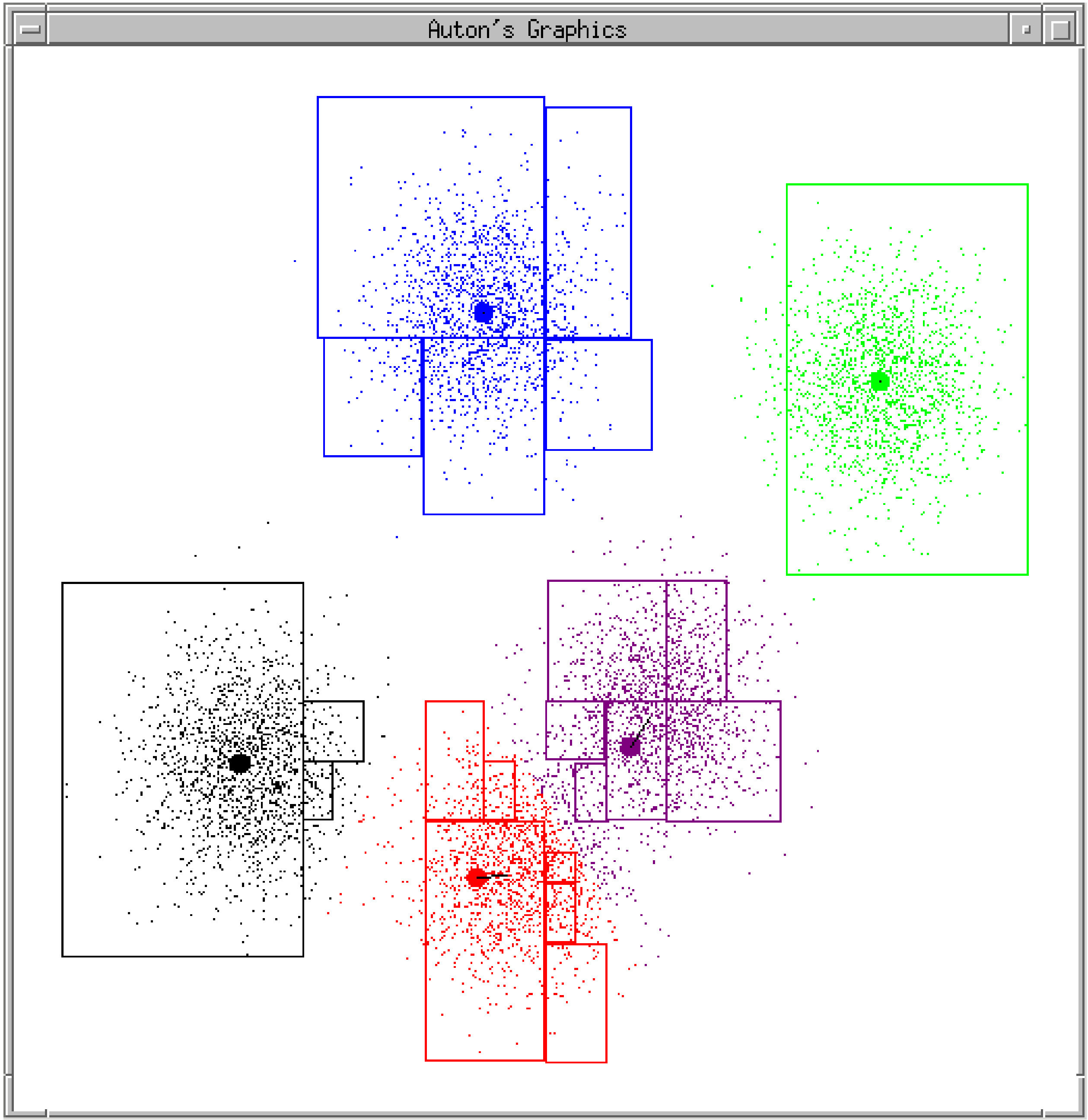
K-means continues …



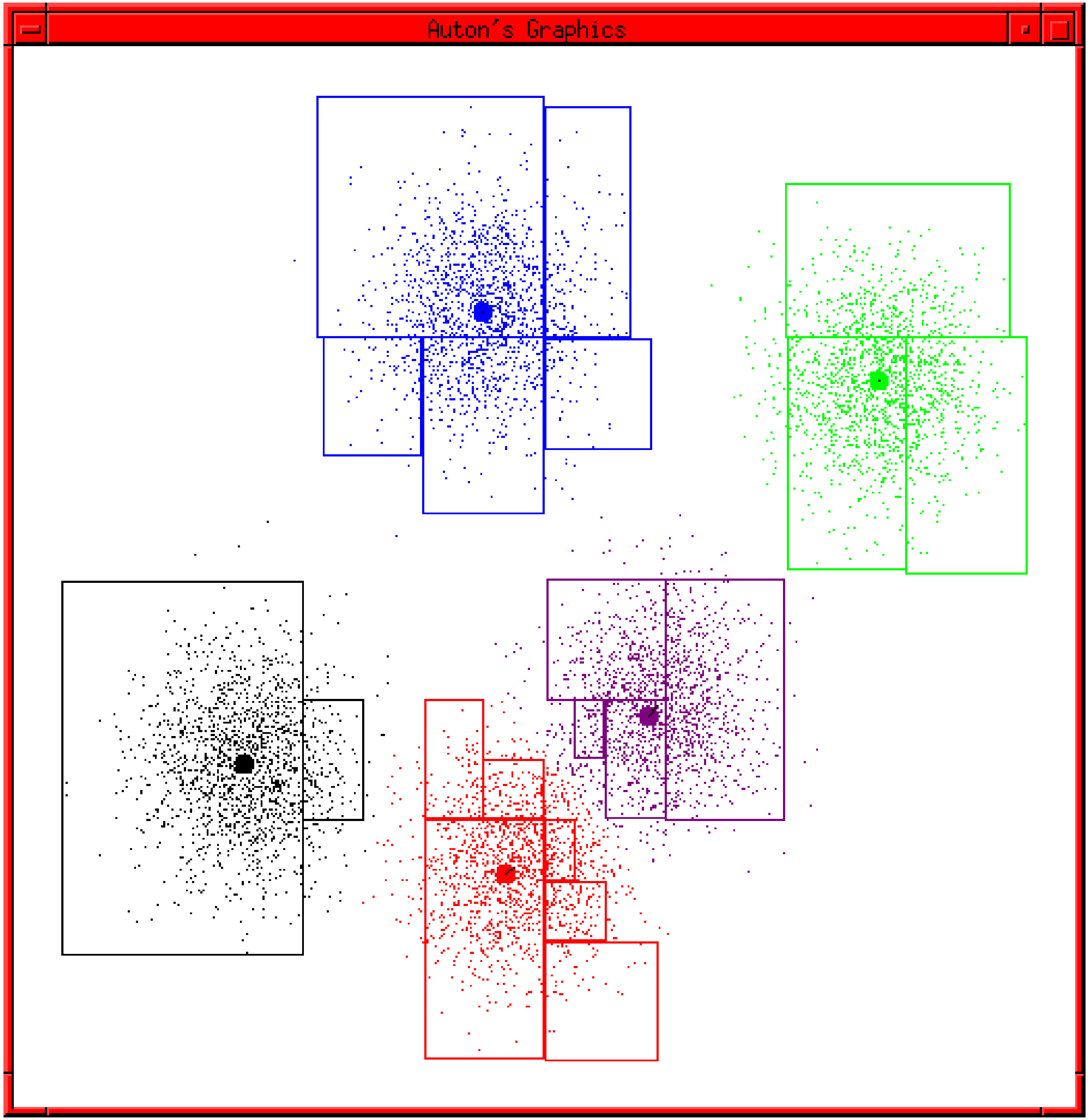
K-means continues …



K-means continues …



K-means terminates



Comparison to a linear algorithm

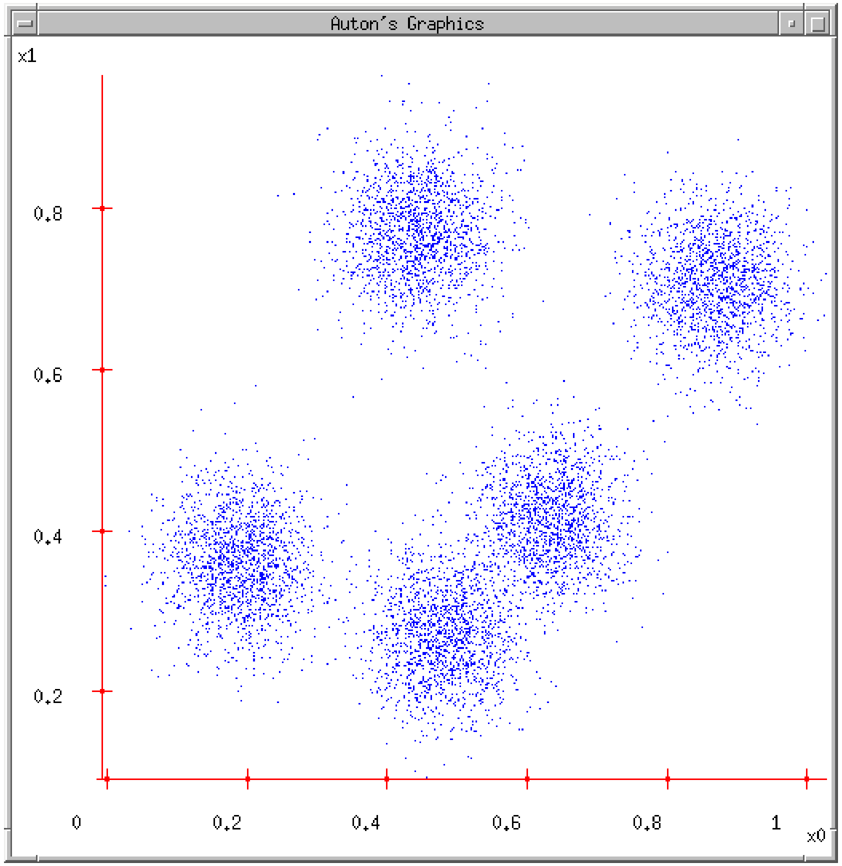
points	blacklisting	naive	speedup
50000	2.02	52.22	25.9
100000	2.16	134.82	62.3
200000	2.97	223.84	75.3
300000	1.87	328.80	176.3
433208	3.41	465.24	136.6

Astrophysics data (2-dimensions)

Outline

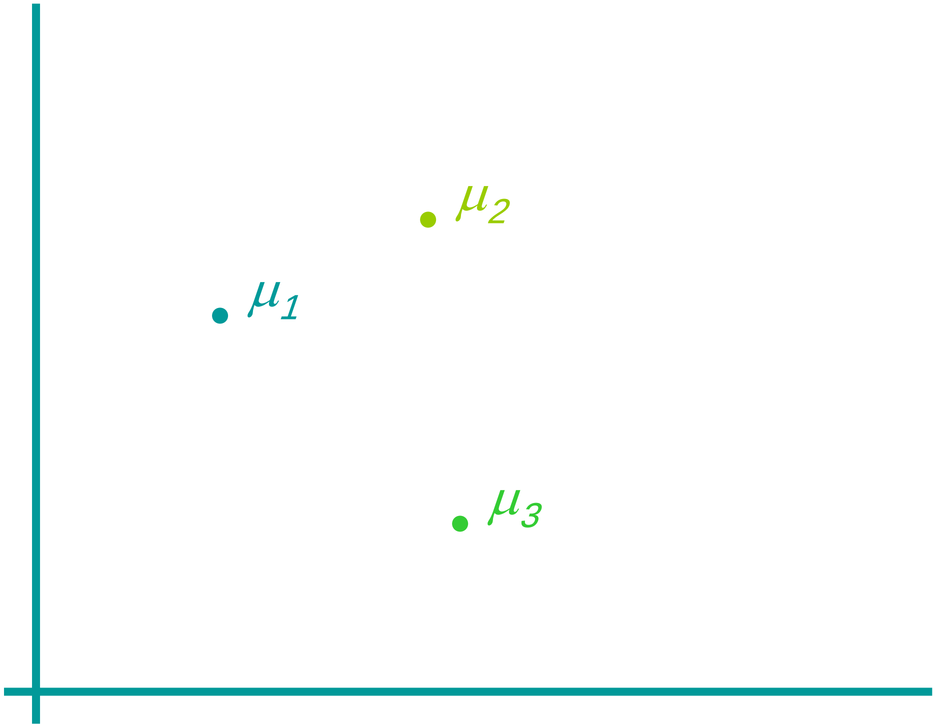
- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ➢ Fast kernel density estimation
- ❑ Large-scale galactic morphology

What if we want to do density estimation with multimodal or clumpy data?



The GMM assumption

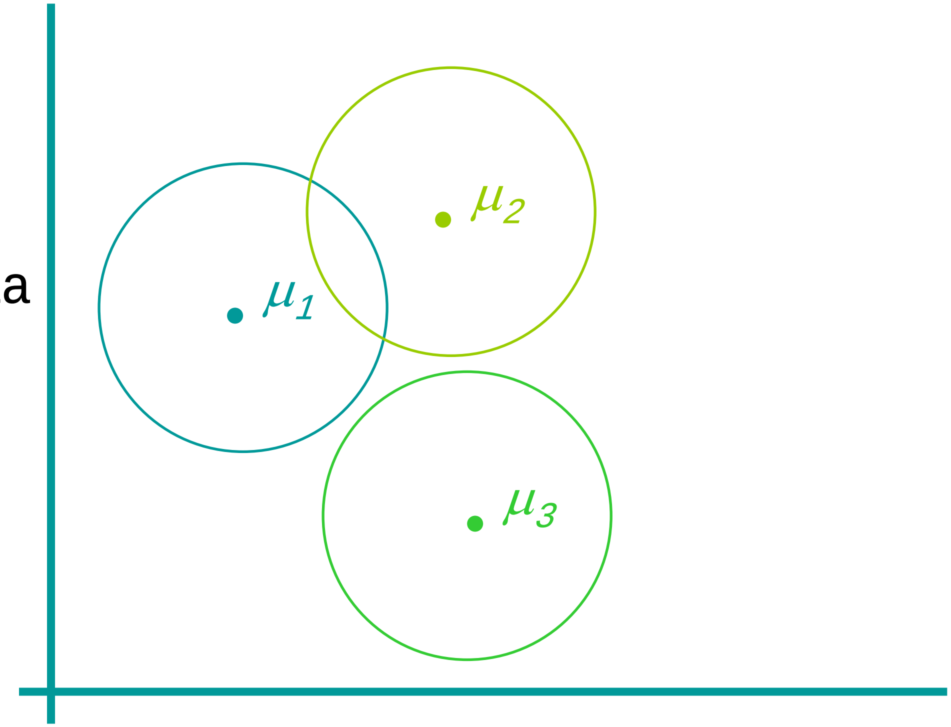
- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i



The GMM assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \mathbf{I}$

Assume that each datapoint is generated according to the following recipe:

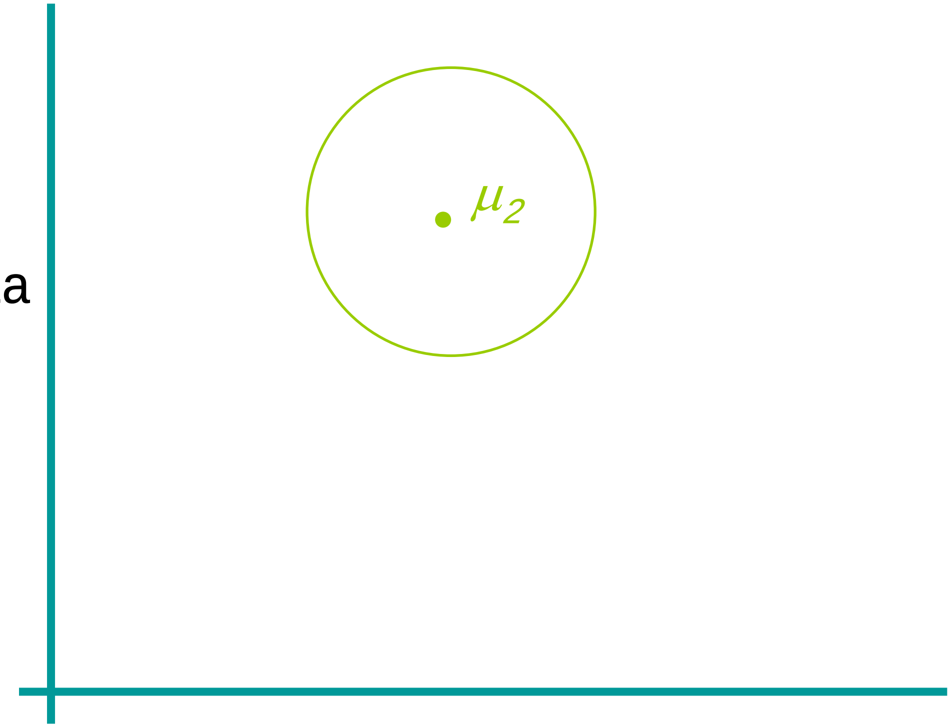


The GMM assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \boldsymbol{I}$

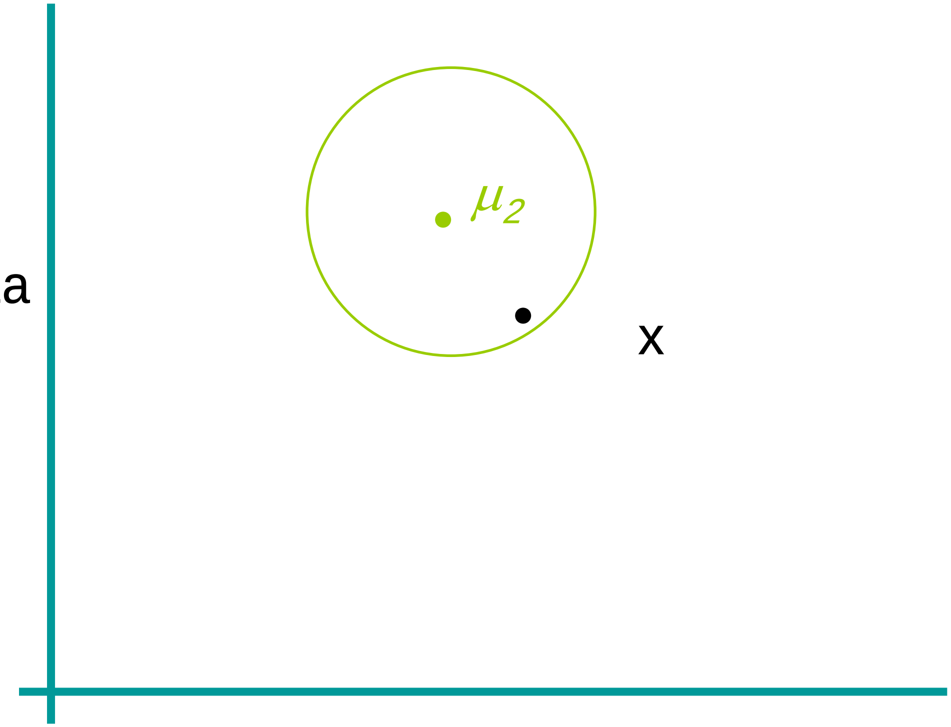
Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component i with probability $P(\omega_i)$.



The GMM assumption

- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2 \boldsymbol{I}$

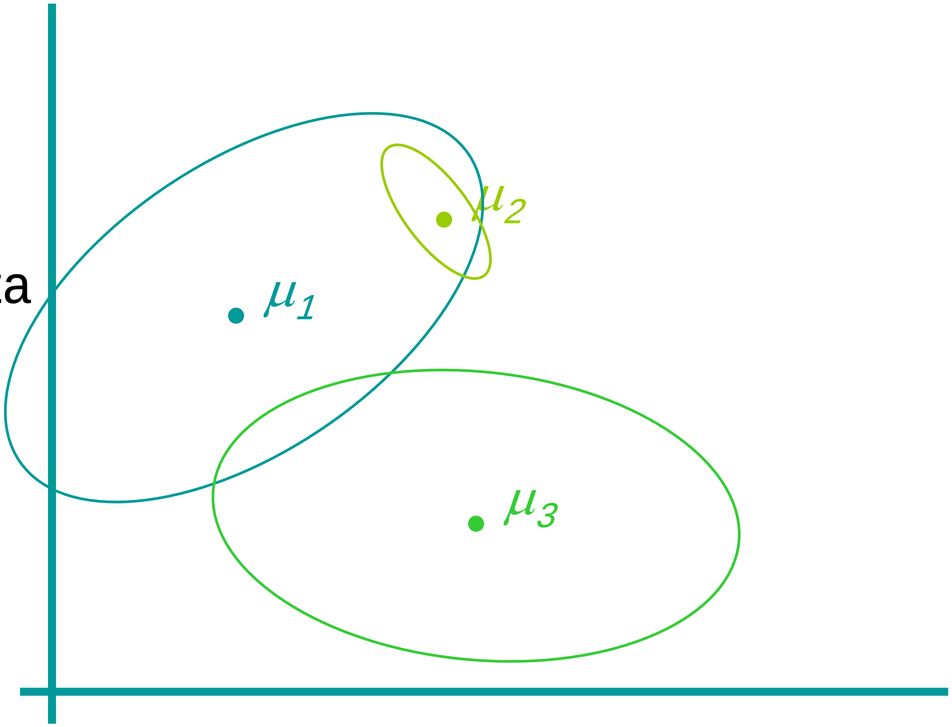


Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component i with probability $P(\omega_i)$.
2. Datapoint ~ N(μ_i, $\sigma^2 \boldsymbol{I}$)

The General GMM assumption

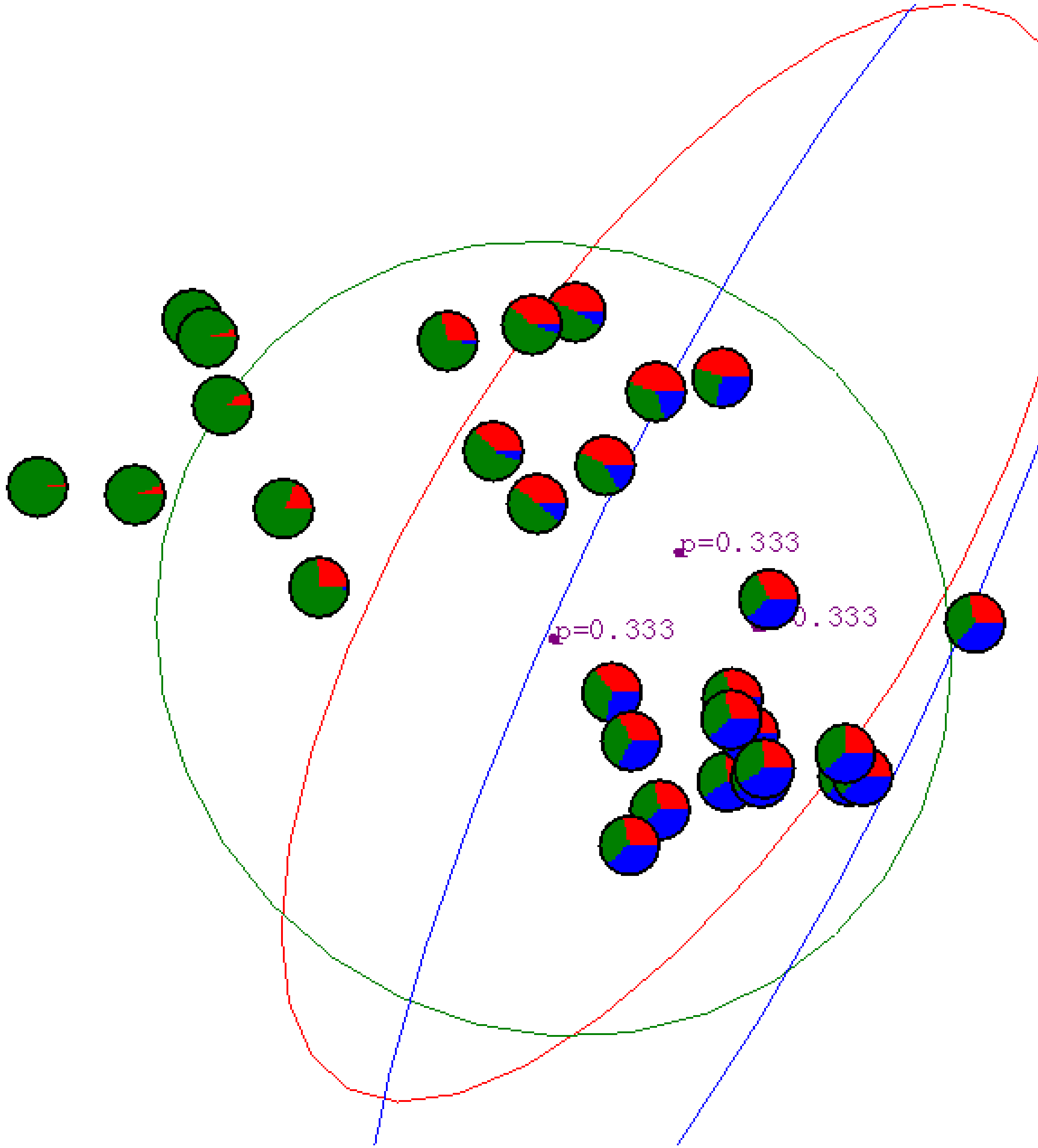
- There are k components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i



Assume that each datapoint is generated according to the following recipe:

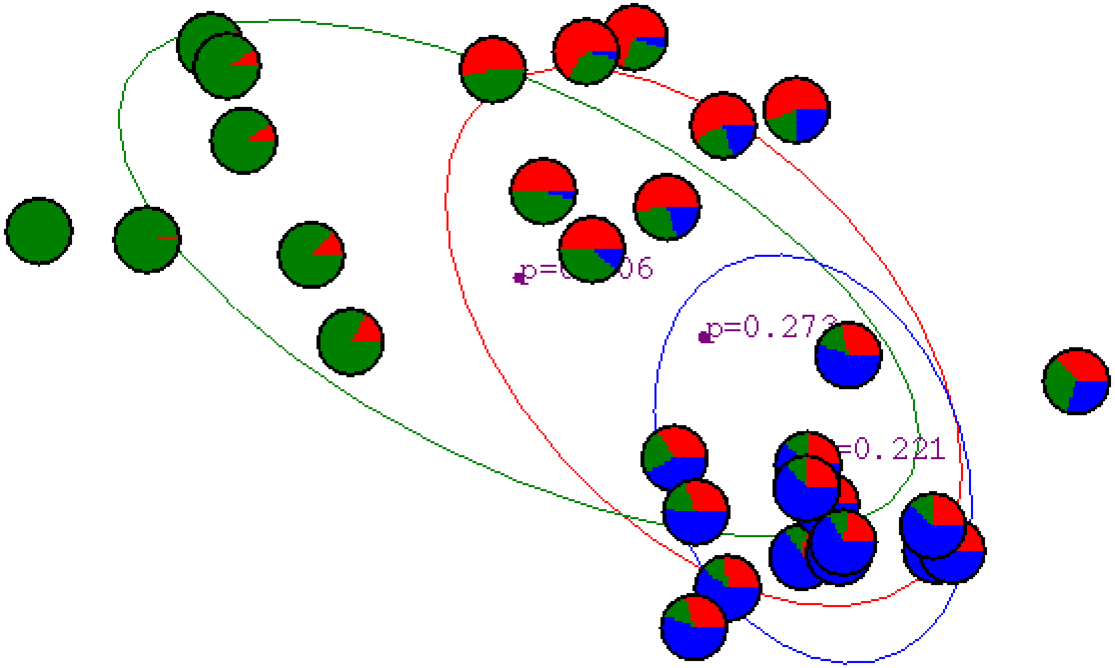
1. Pick a component at random. Choose component i with probability $P(\omega_i)$.
2. Datapoint ~ $N(\mu_i, \Sigma_i)$

Gaussian Mixture Example: Start

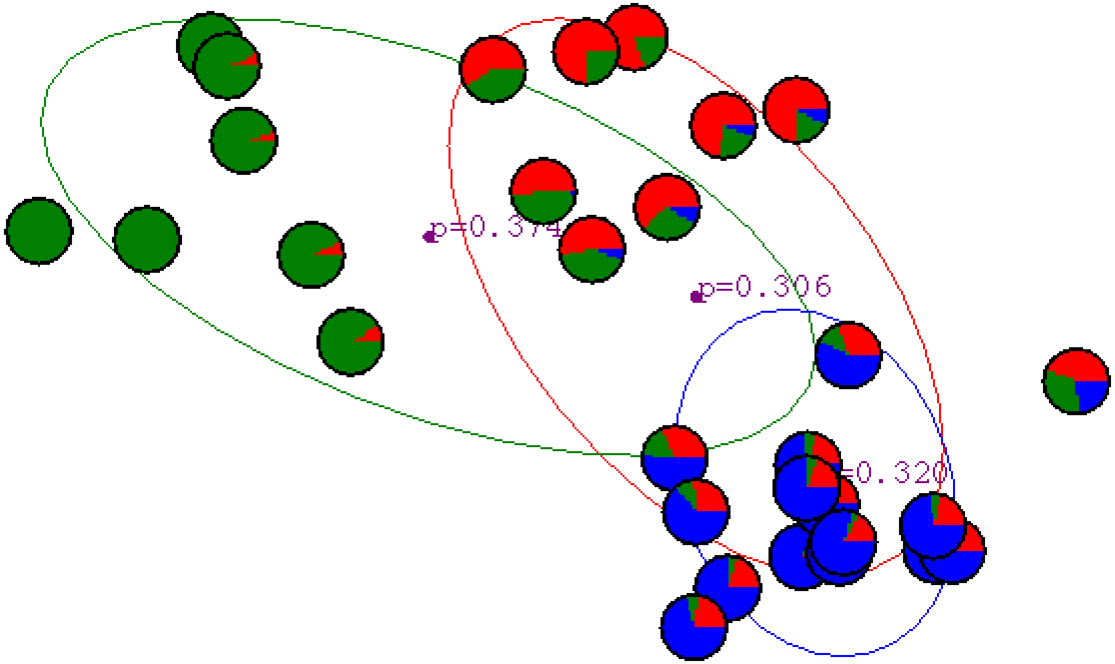


Advance apologies: in Black and White this example will be incomprehensible

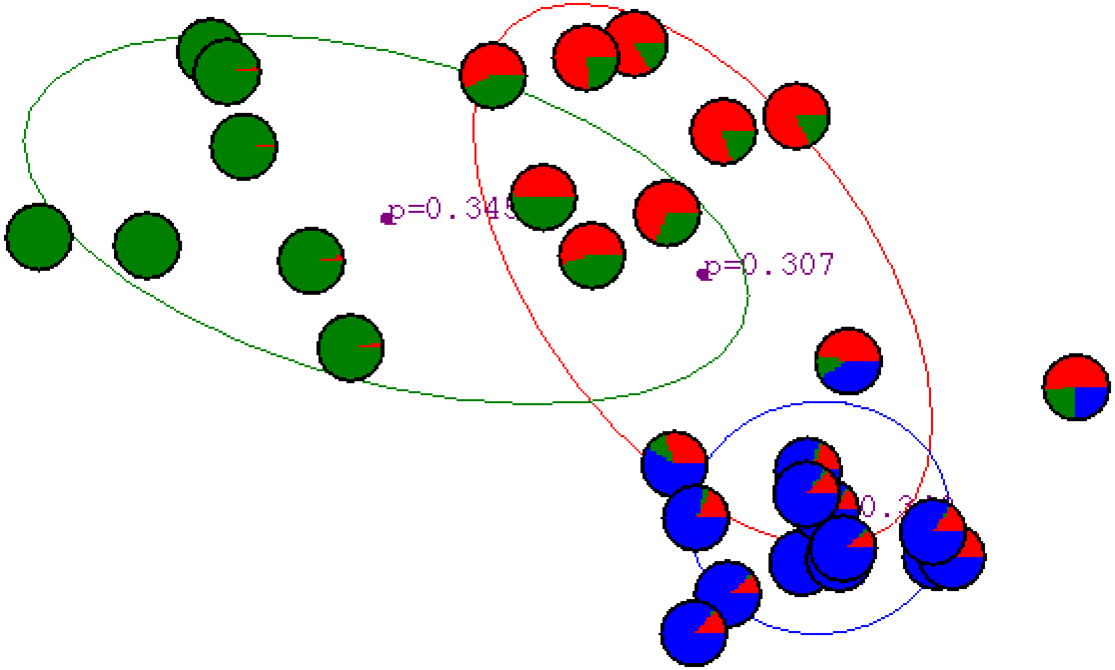
After first iteration



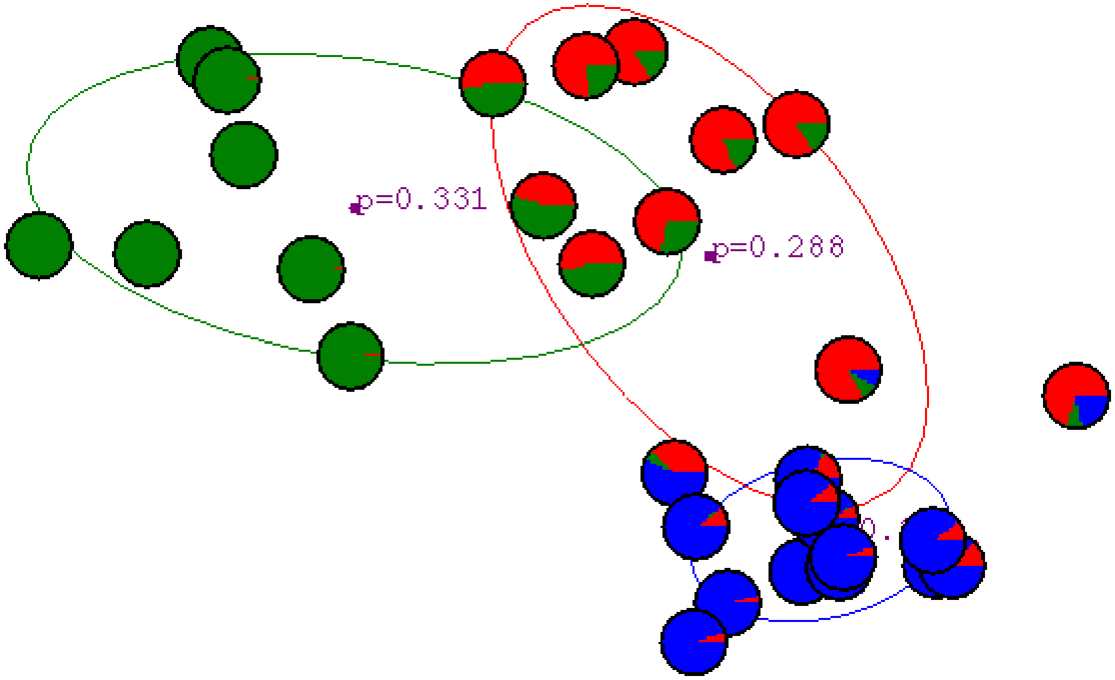
After 2nd iteration



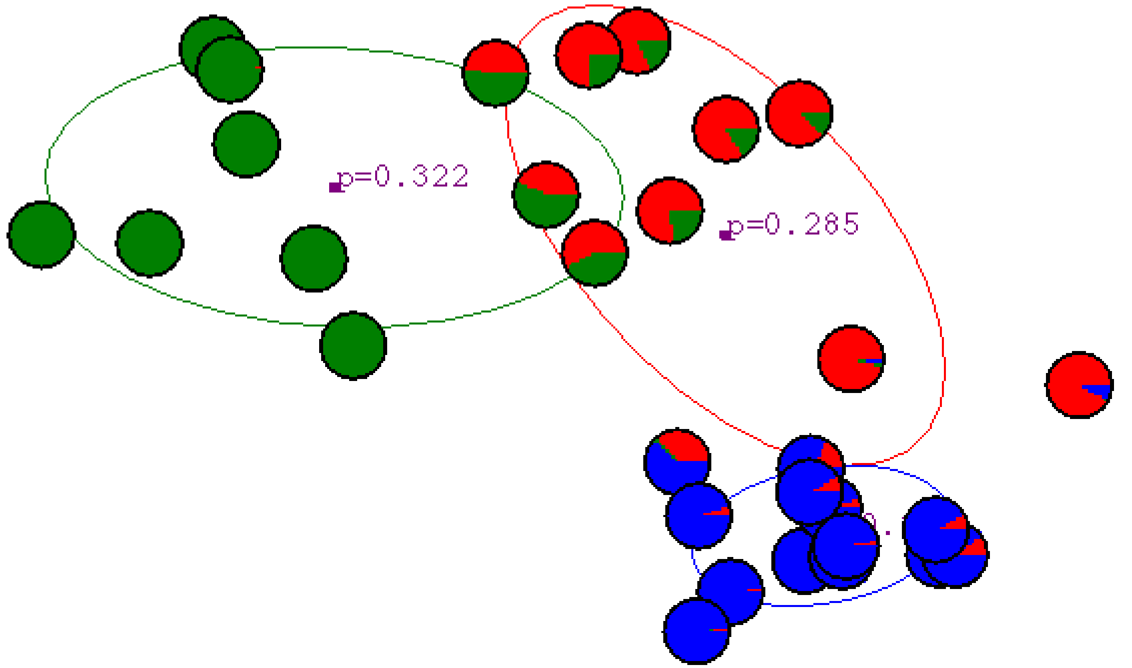
After 3rd iteration



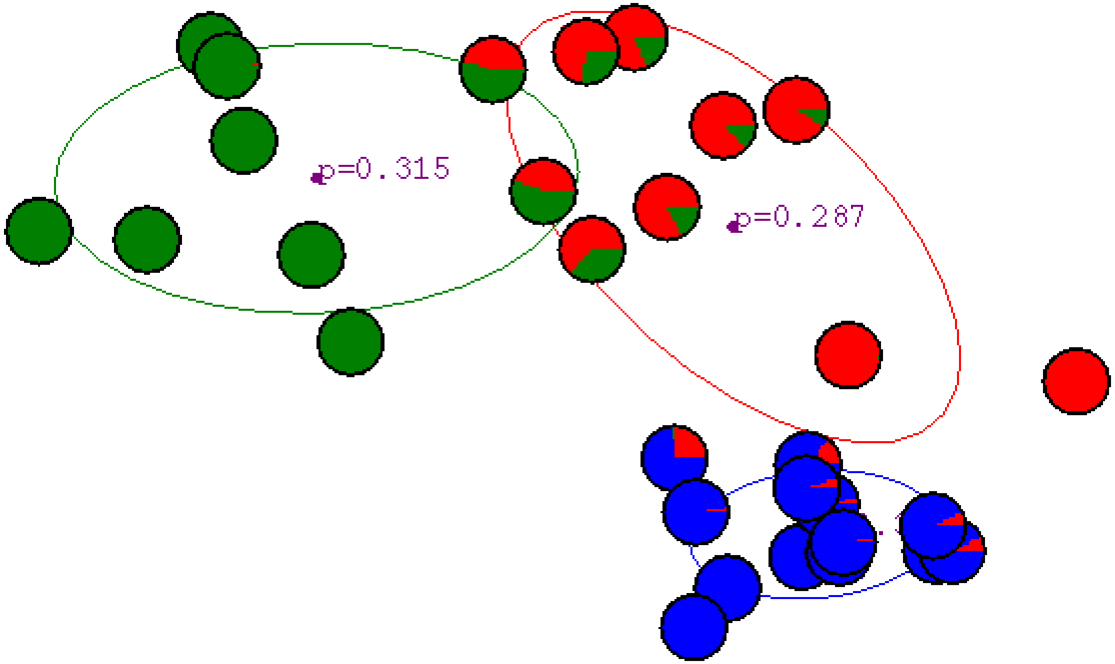
After 4th iteration



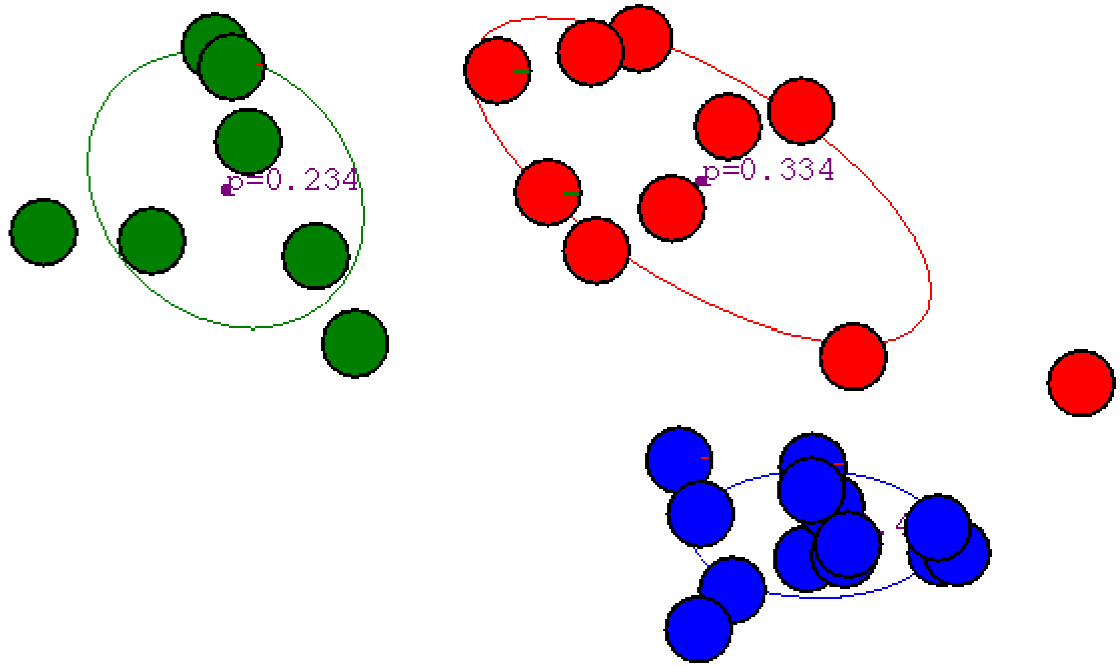
After 5th iteration



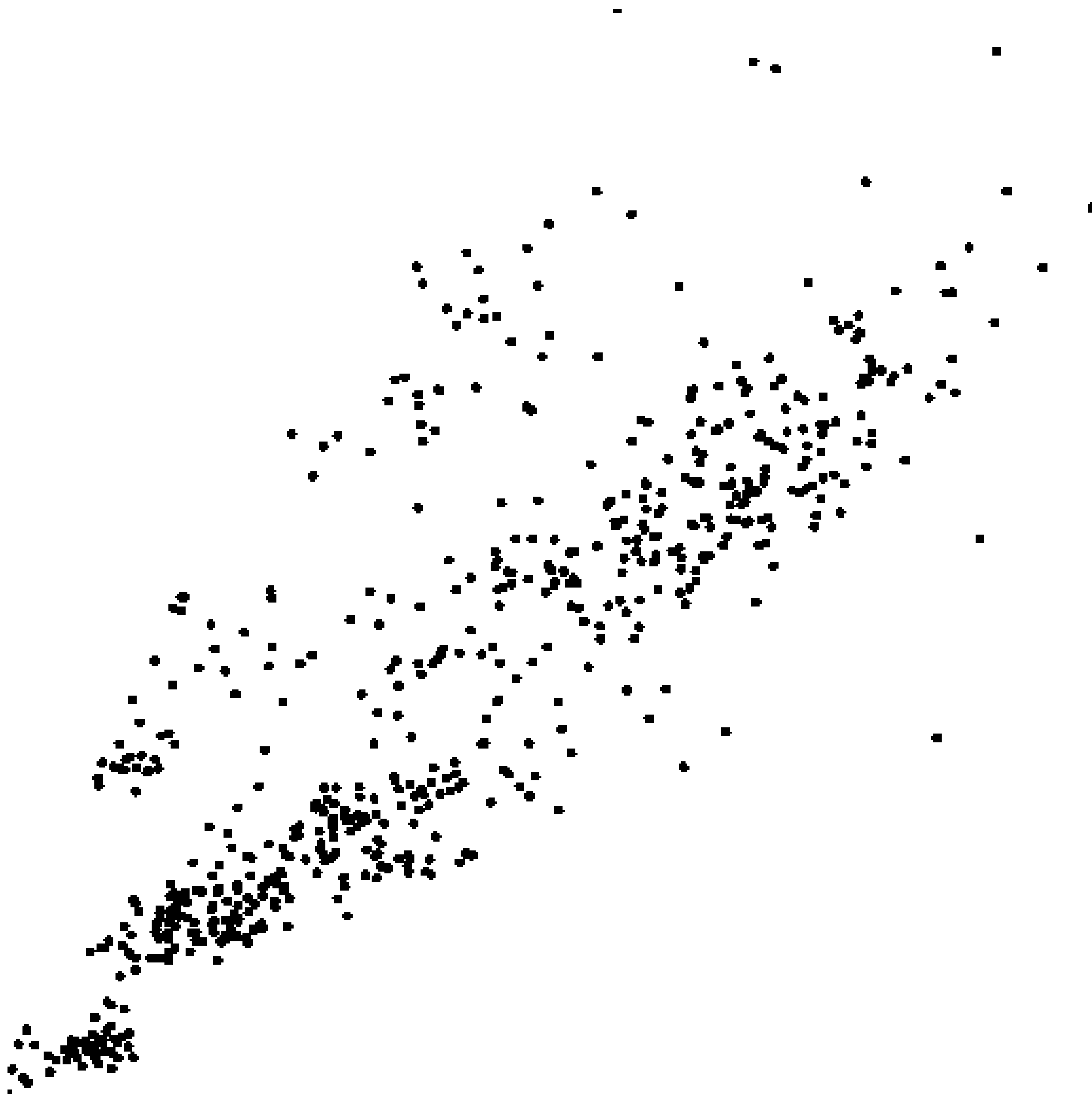
After 6th iteration



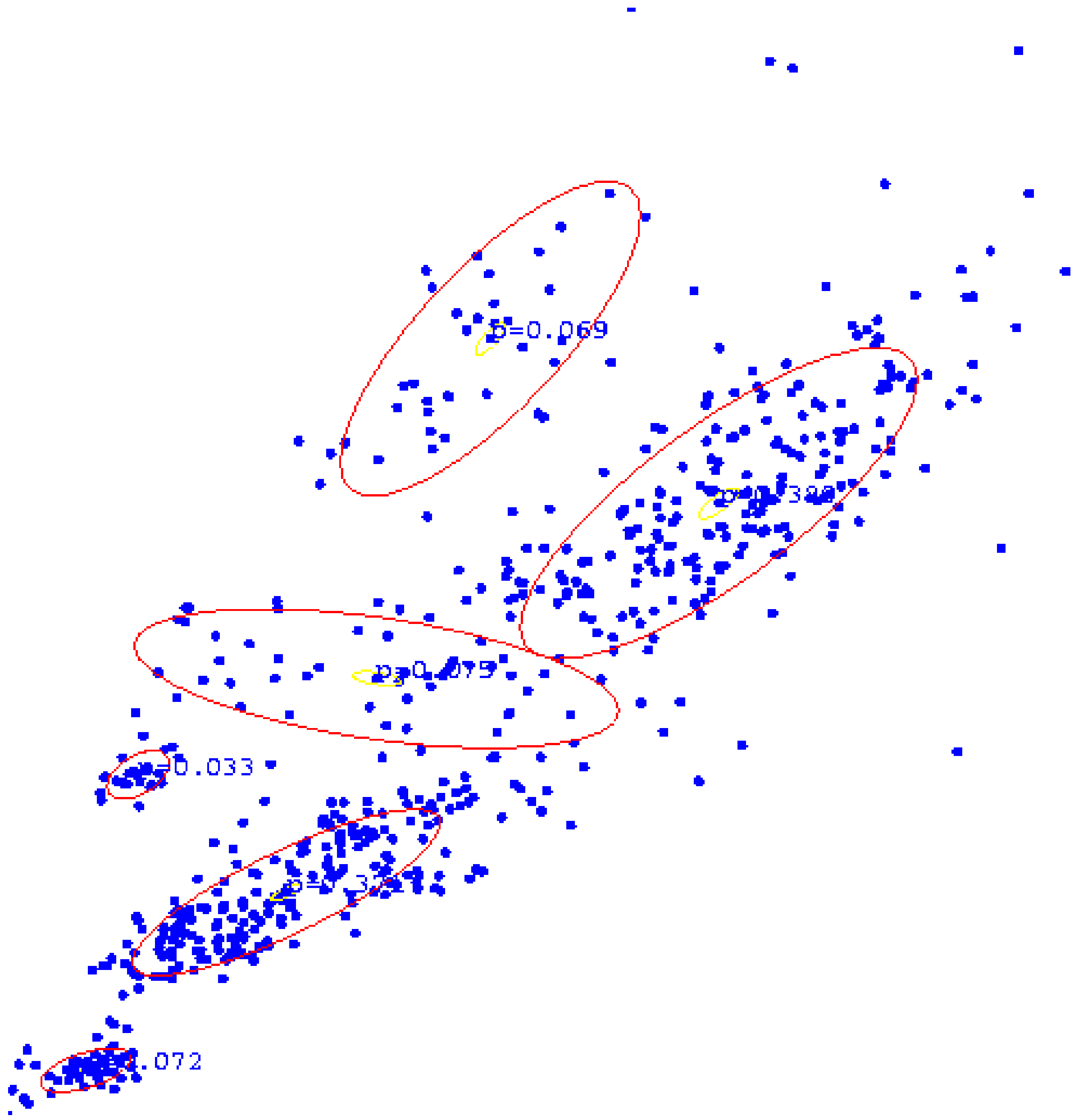
After 20th iteration



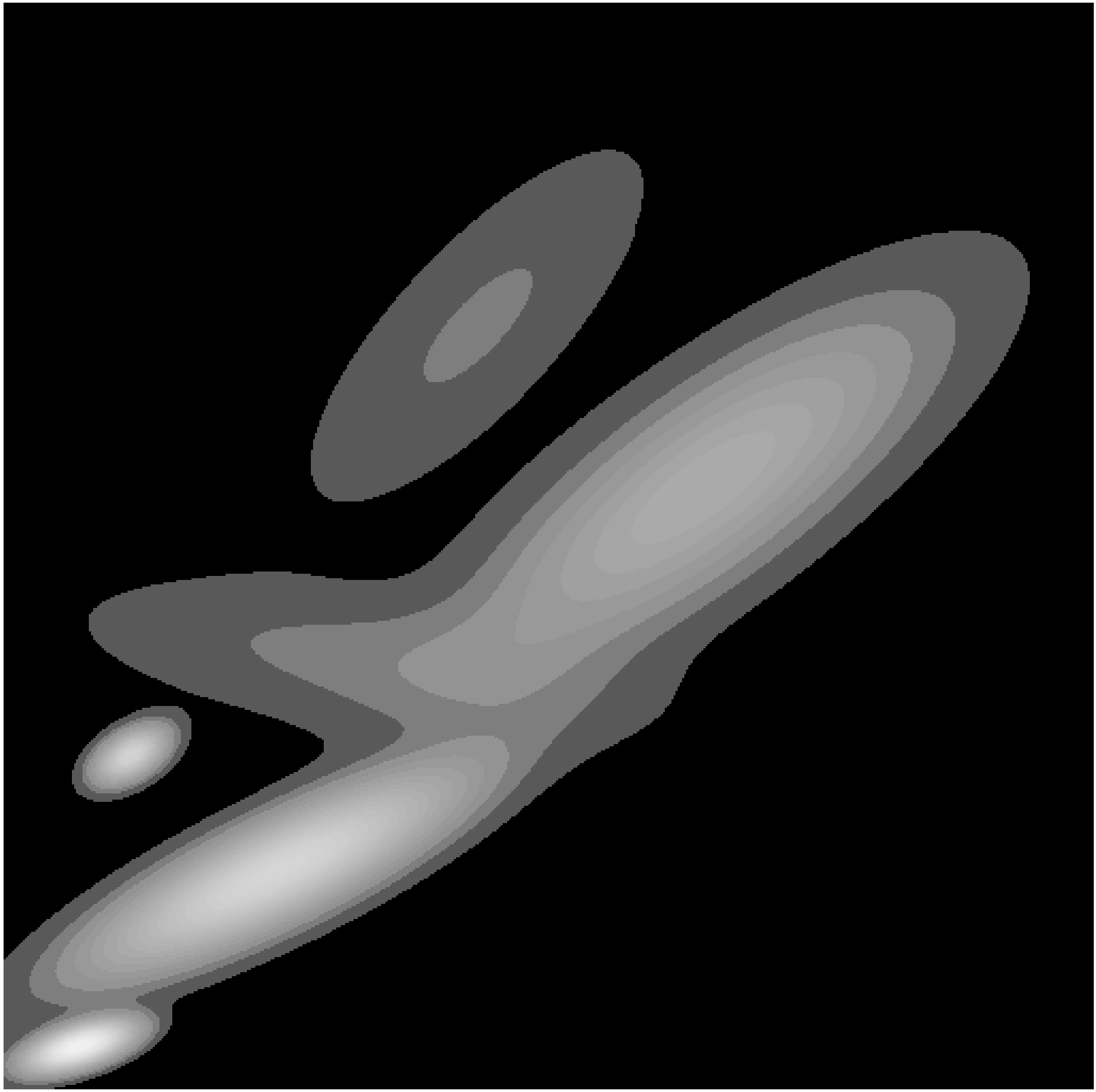
Some Bio Assay data



GMM clustering of the assay data

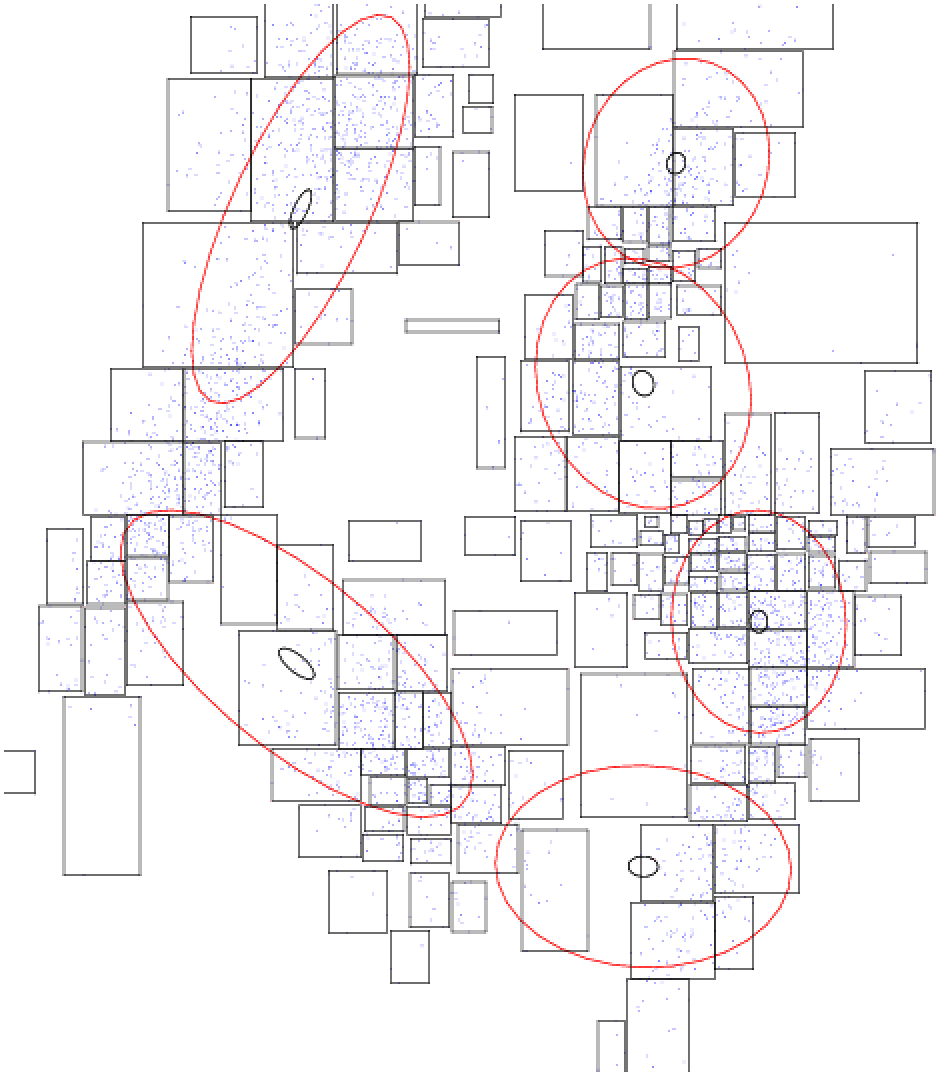


Resulting Density Estimator

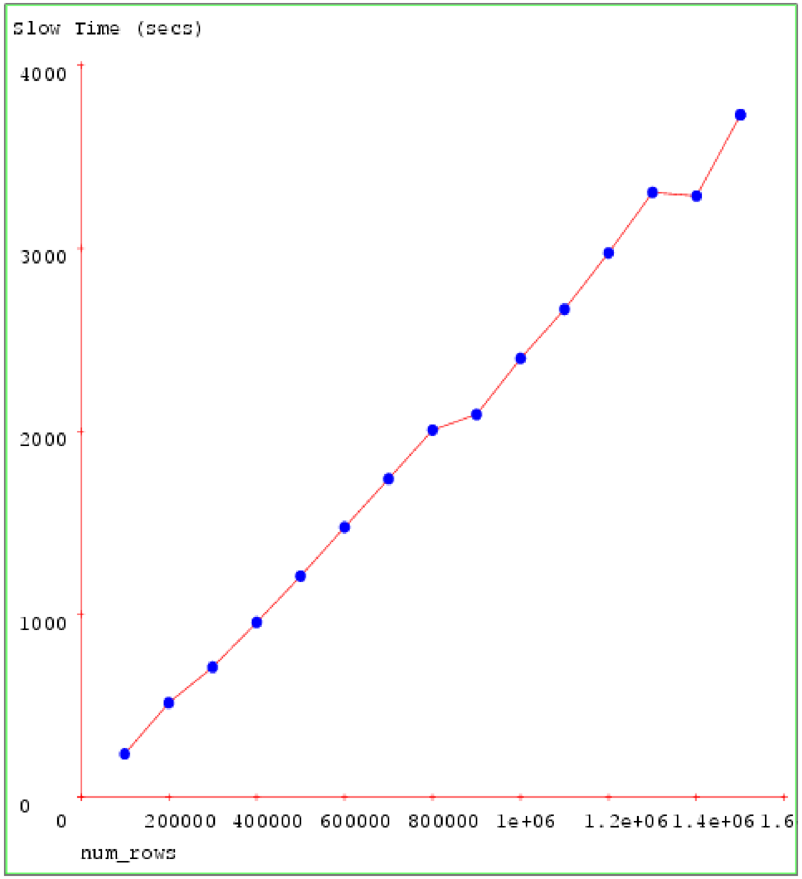


- For basic tutorial on Gaussian Mixture Models, see Hastie et al or Duda, Hart & Stork books , or www.cs.cmu.edu/~awm/tutorials/gmm.html

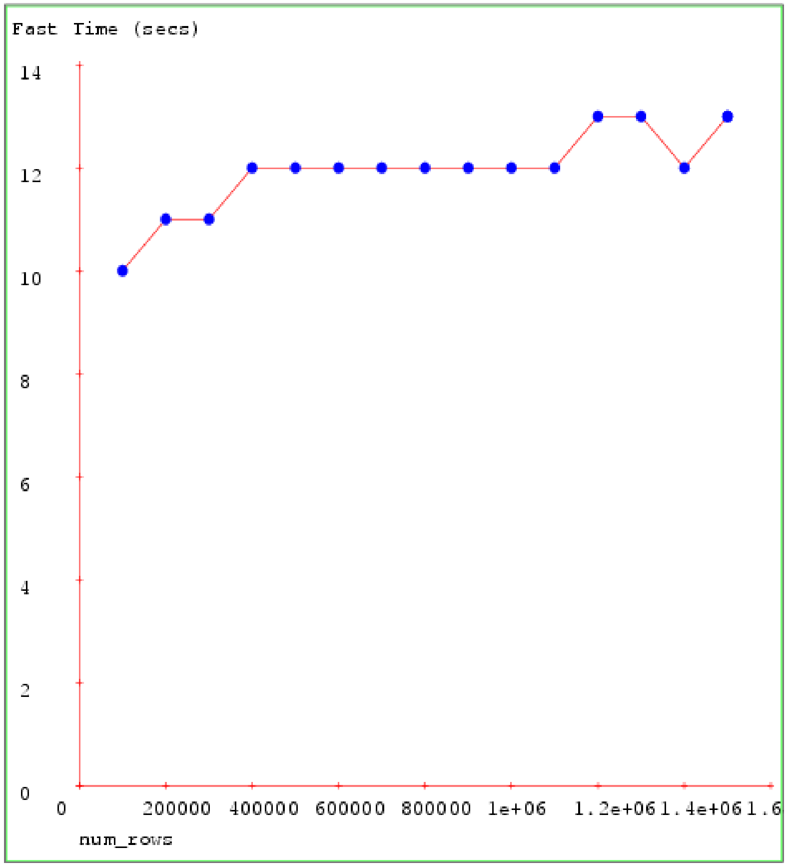
Nodes visited during an EM pass



Sky Survey Data: Time Taken by the Slow method.



Sky Survey Data: Time Taken by the Fast method



Uses for kd-trees and cousins

- K-Means clustering [Pelleg and Moore, 99], [Moore 2000]
- Kernel Density Estimation [Deng & Moore, 1995], [Gray & Moore 2001]
- Kernel-density-based clustering [Wong and Moore, 2002]
- Gaussian Mixture Model [Moore, 1999]

- Kernel Regression [Deng and Moore, 1995]
- Locally weighted regression [Moore, Schneider, Deng, 97]

- Kernel-based Bayes Classifiers [Moore and Schneider, 97]
- N-point correlation functions [Gray and Moore 2001]

Also work by Priebe, Ramikrishnan, Schaal, D'Souza, Elkan,…

Papers (and software): www.autonlab.org

Uses for kd-trees and cousins

- K-Means clustering [Pelleg and Moore, 99], [Moore 2000]
- Kernel Density Estimation [Deng & Moore, 1995], [Gray & Moore 2001]
- Kernel-density-based clustering [Wong and Moore, 2002]
- Gaussian Mixture Model [Moore, 1999]

- Kernel Regression [Deng and Moore, 1995]
- Locally weighted regression [Moore, Schneider, Deng, 97]

- Kernel-based Bayes Classifiers [Moore and Schneider, 97]
- N-point correlation functions [Gray and Moore 2001]

Also work by Priebe, Ramikrishnan, Schaal, D'Souza, Elkan,…

Papers (and software): www.autonlab.org

Outline

- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ✓ Fast kernel density estimation
- ➢ Large-scale galactic morphology

Outline

- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ✓ Fast kernel density estimation
- ➢ Large-scale galactic morphology
  - ❑ GMorph
    - ❑ Memory-based
    - ❑ PCA

Galactic Morphology via Eigengalaxies

Brigham S. Anderson, CMU
Andrew Moore, CMU
Andy Connolly, U. of Pittsburgh Astrophysics
Bob Nichols, CMU Astrophysics
Mariangela Bernardi, CMU Astrophysics


Image space (~4096 dim)

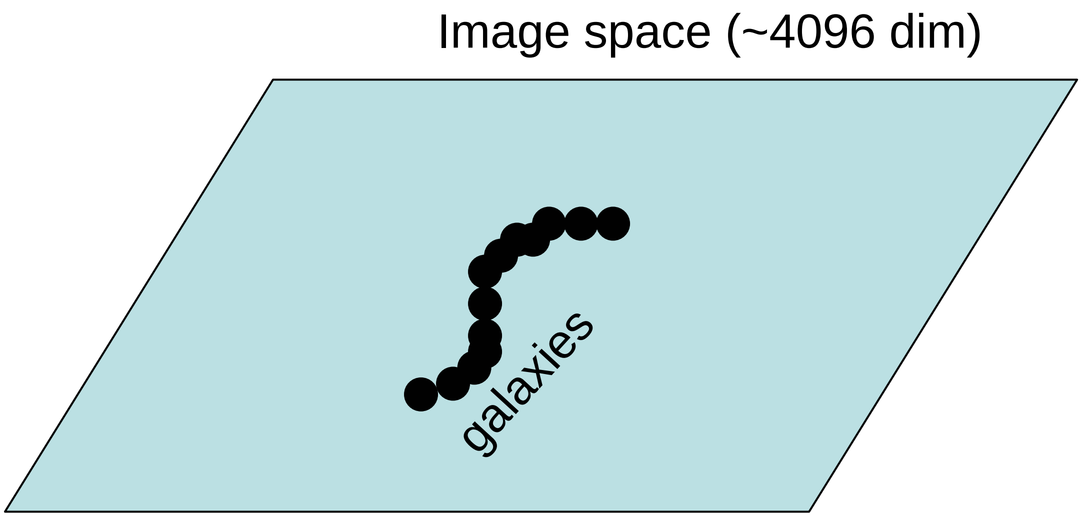


Image space (~4096 dim)
galaxies

Disk	Bulge
Exponential profile (Freeman, 1970)	de Vaucouleur's profile (1953)
2-d disk	3-d oblate spheroid
$I(r) \propto exp(-\frac{r}{r_d})$	$I(r) \propto exp\left(-b\left(\frac{r}{r_b}\right)^{\frac{1}{4}}\right)$
	

Model Parameters

μ_x μ_y	the x and y offset of the galactic center from the image center $(pixels)$
F_b F_d	total integrated flux of bulge or disk $(erg \cdot cm^2/sec)$
r_b r_d	bulge or disk scale length. $(pixels)$
ϵ_b	apparent bulge ellipticity $(unitless)$.
γ_{inc}	disk inclination $(degrees)$. Rotation toward viewer.
γ_b γ_d	bulge or disk angle of rotation $(degrees)$.
sky	background offset $(flux/cm^2)$
$Sersic$	a bulge shape parameter that is fixed to the value 4 for all expts

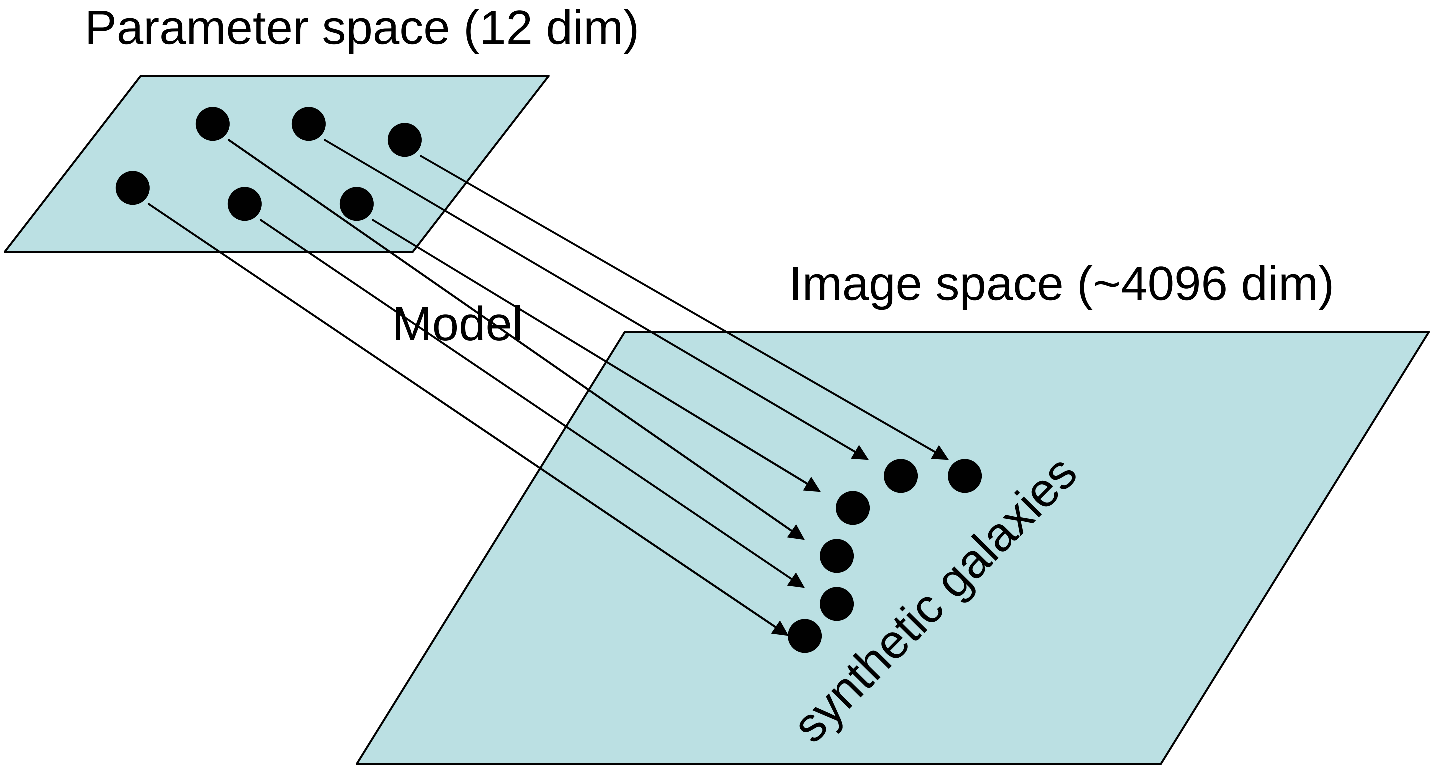
Algorithm	Method	Speed	Weakness
Naive deconvolve & fit	e.g., descent	?	sensitive to noise
GIM2d (Simard, 2002)	Simulated annealing	~3 min	
Galfit (Peng, 2002)	Levenberg-Marquadt	~30 sec	local minima
GMORPH	Memory-based	~1 sec	
1-d approaches	descent	<1 sec	bias

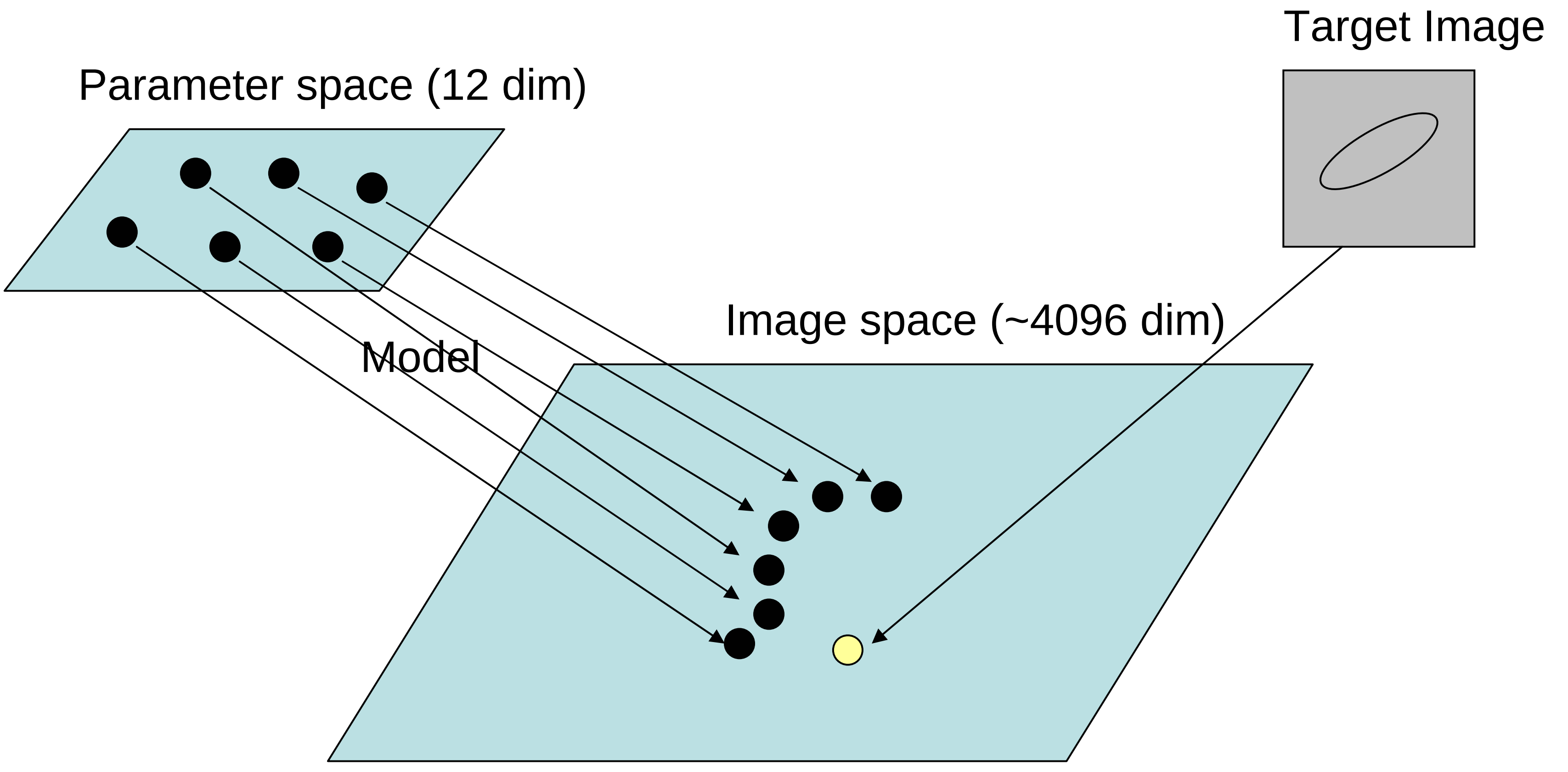
Outline

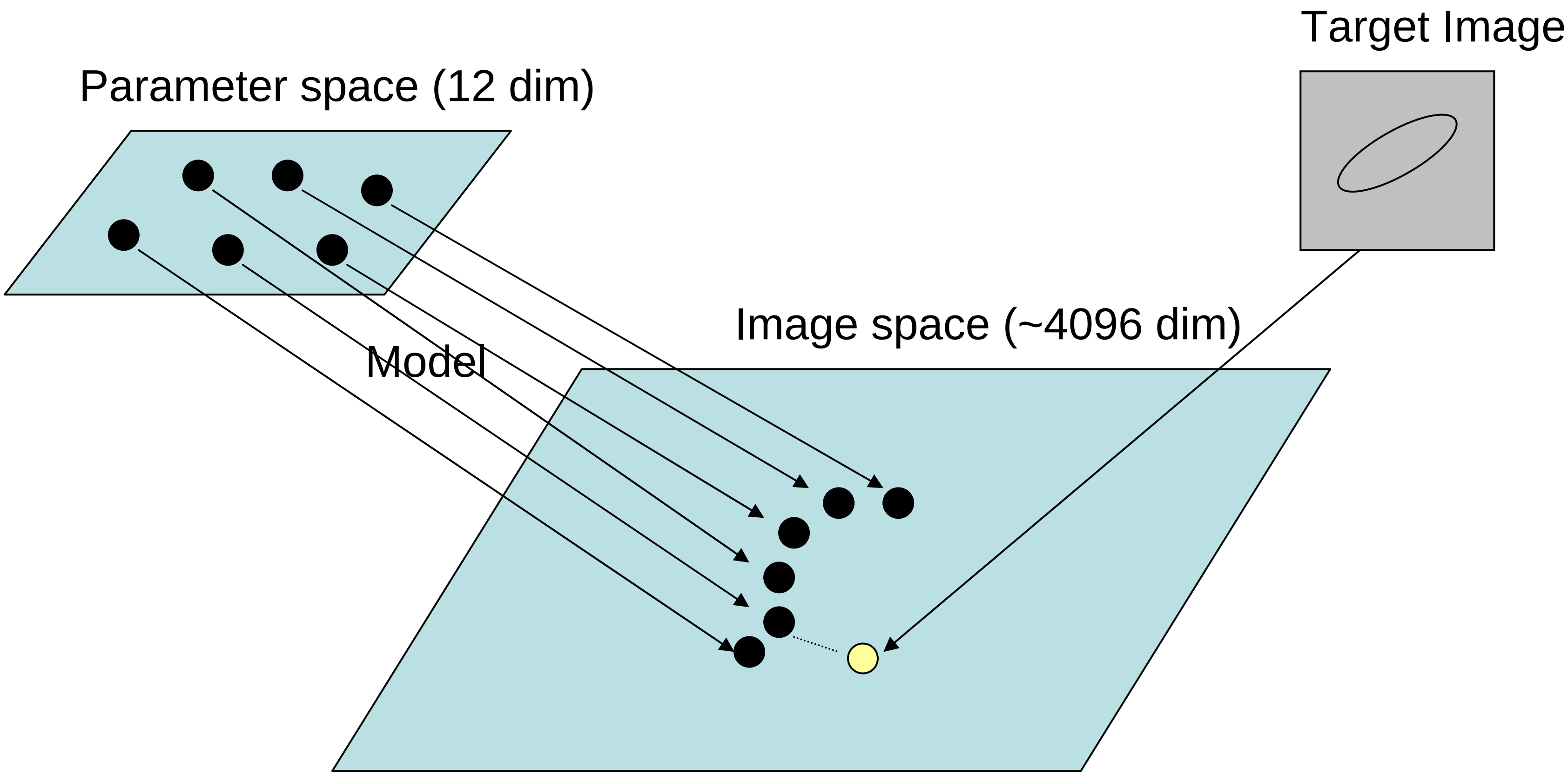
- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ✓ Fast kernel density estimation
- ✓ Large-scale galactic morphology
  - ➢ GMorph
    - ❑ Memory-based
    - ❑ PCA

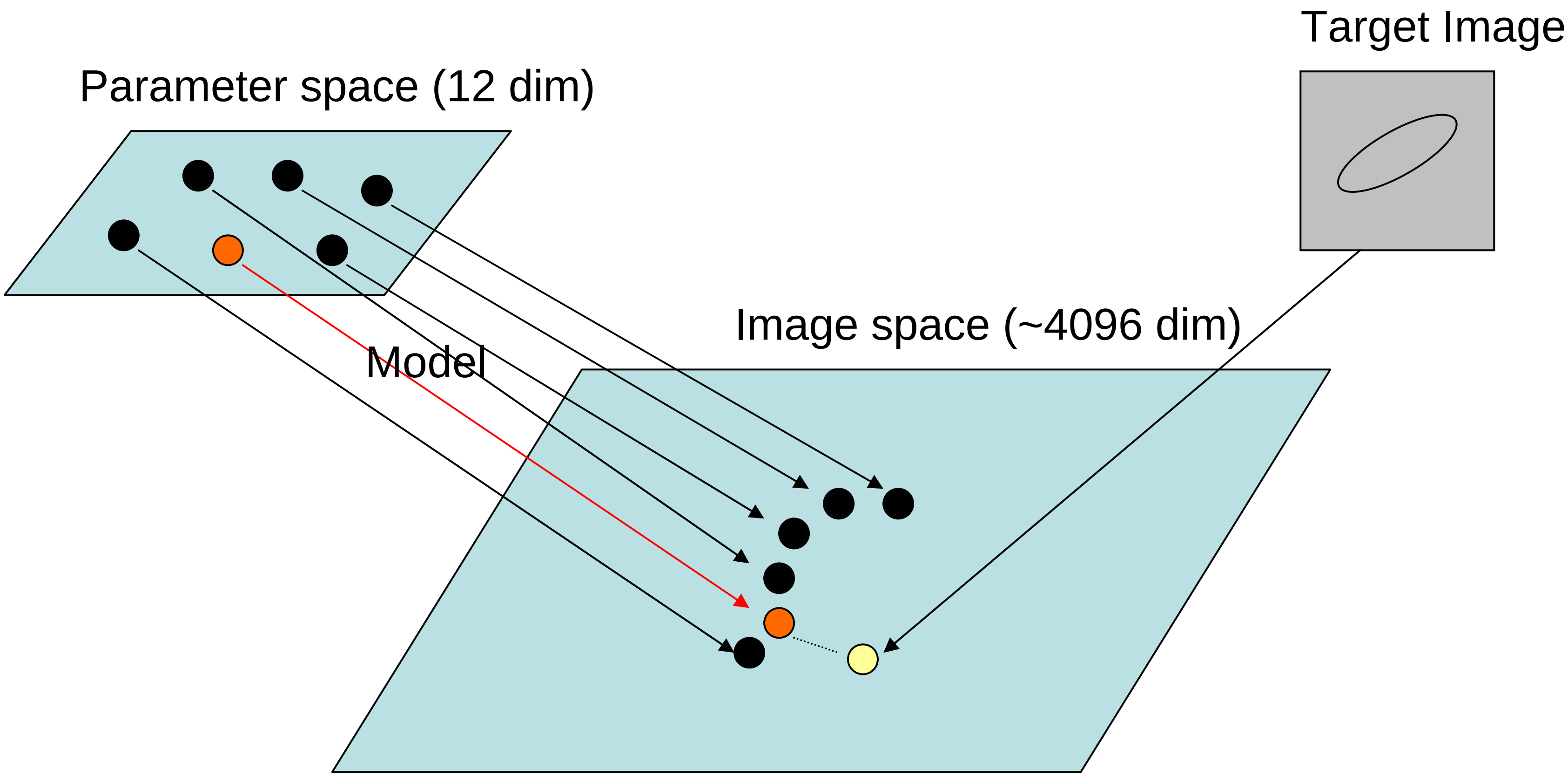
Outline

- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ✓ Fast kernel density estimation
- ✓ Large-scale galactic morphology
  - ➢ GMorph
    - ➢ Memory-based
    - ❑ PCA


Parameter space (12 dim)
Image space (~4096 dim)
Model
synthetic galaxies

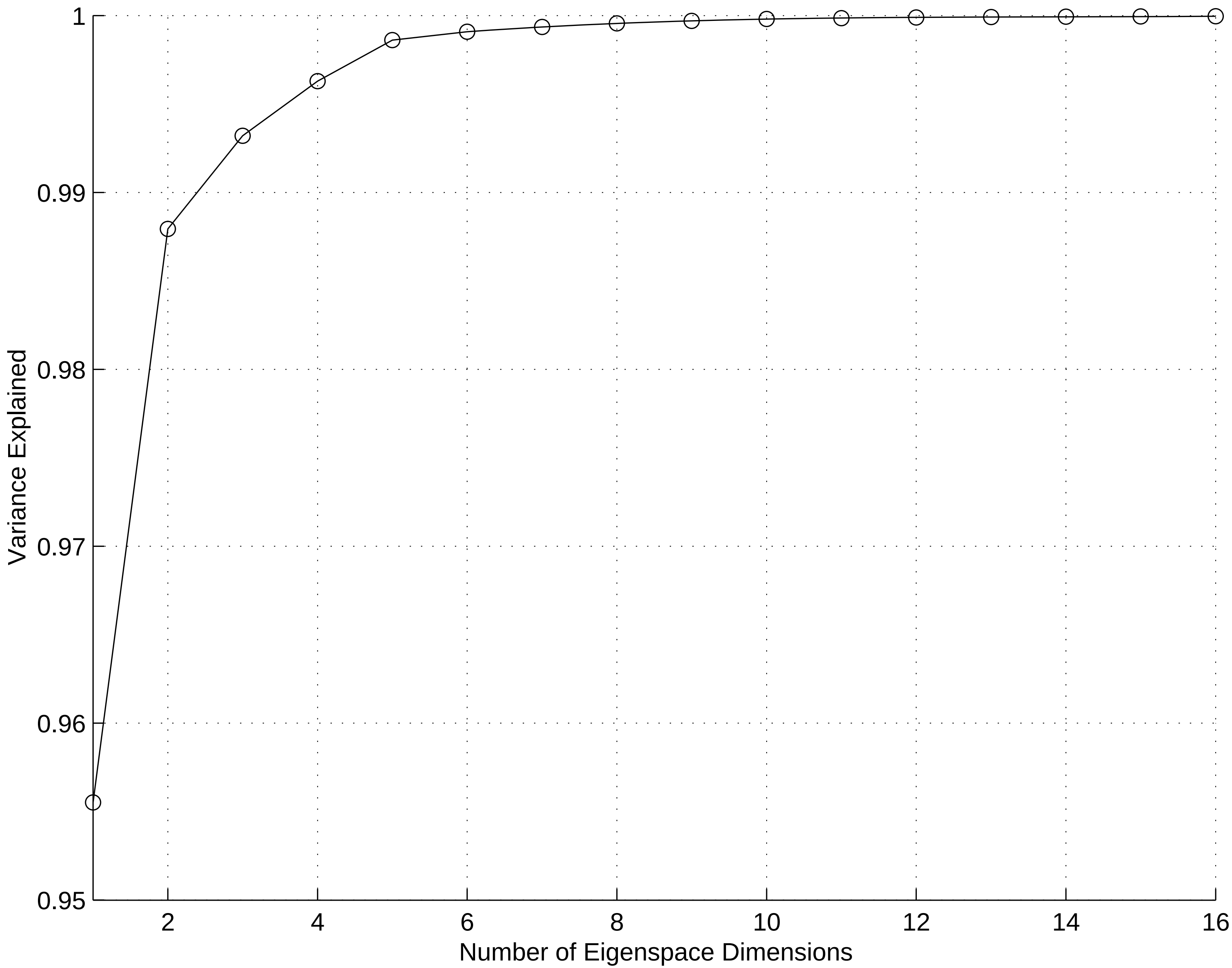

Target Image
Parameter space (12 dim)
Image space (~4096 dim)
Model

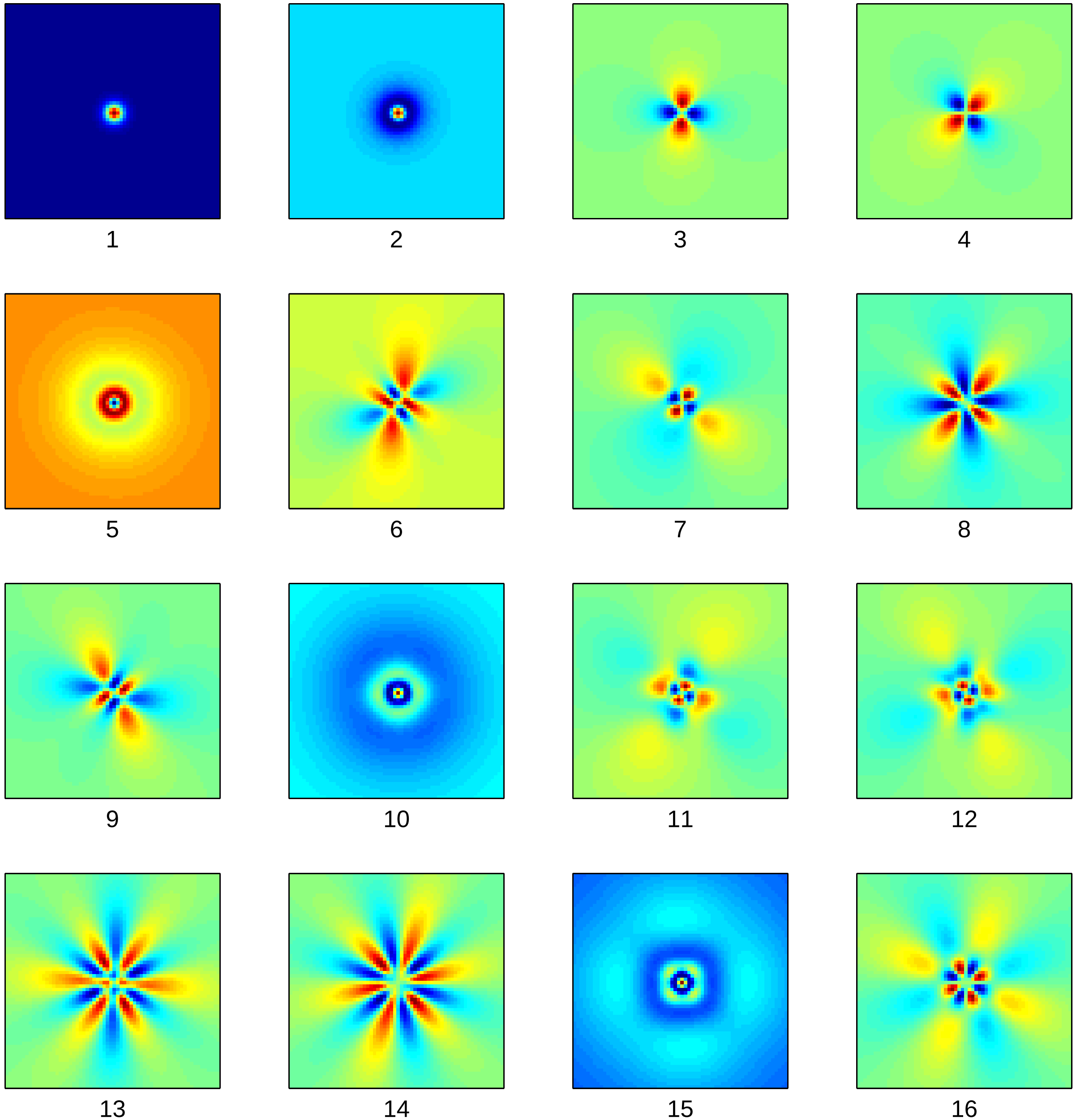

Target Image
Parameter space (12 dim)
Image space (~4096 dim)
Model

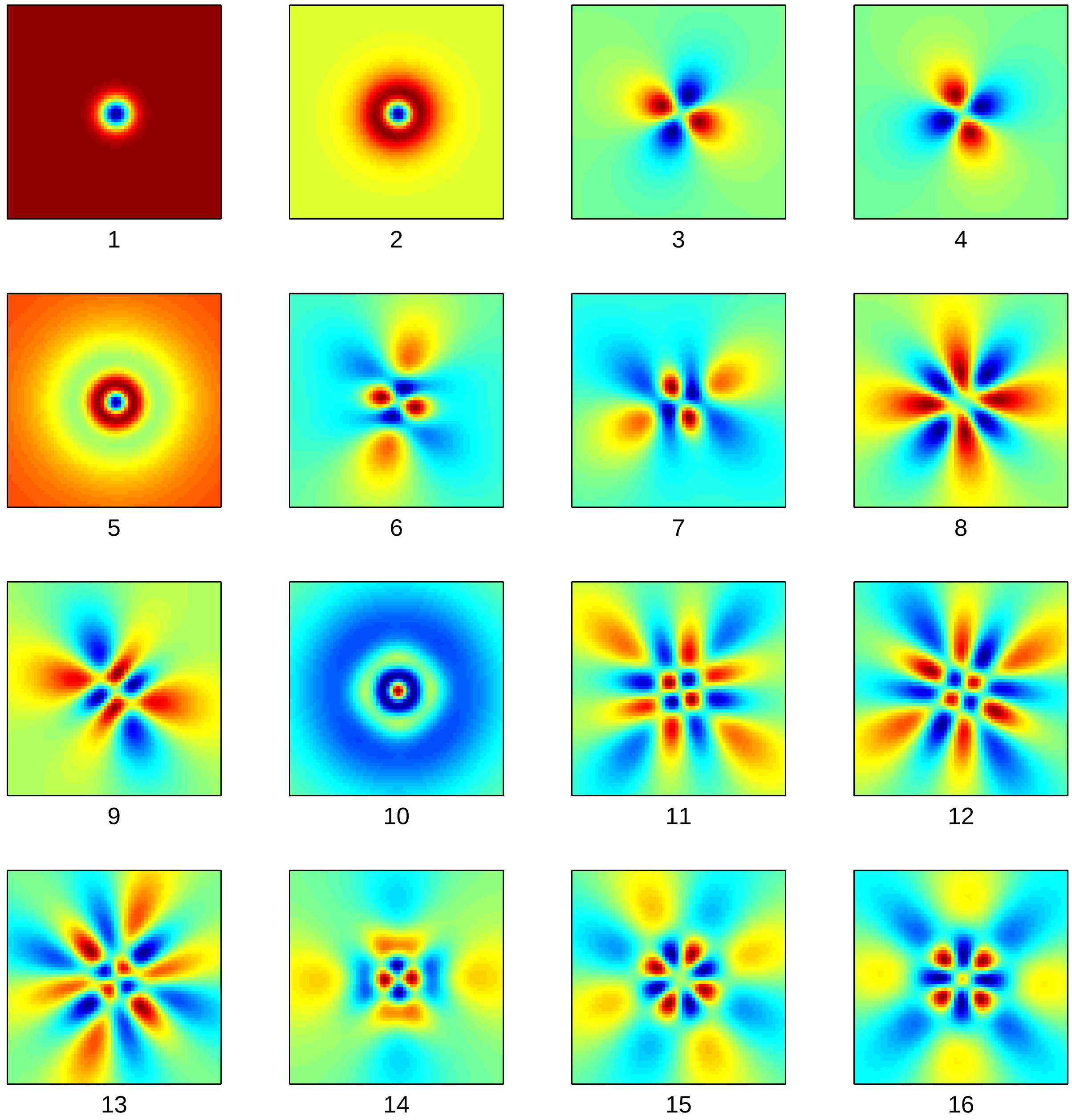

Target Image
Parameter space (12 dim)
Image space (~4096 dim)
Model

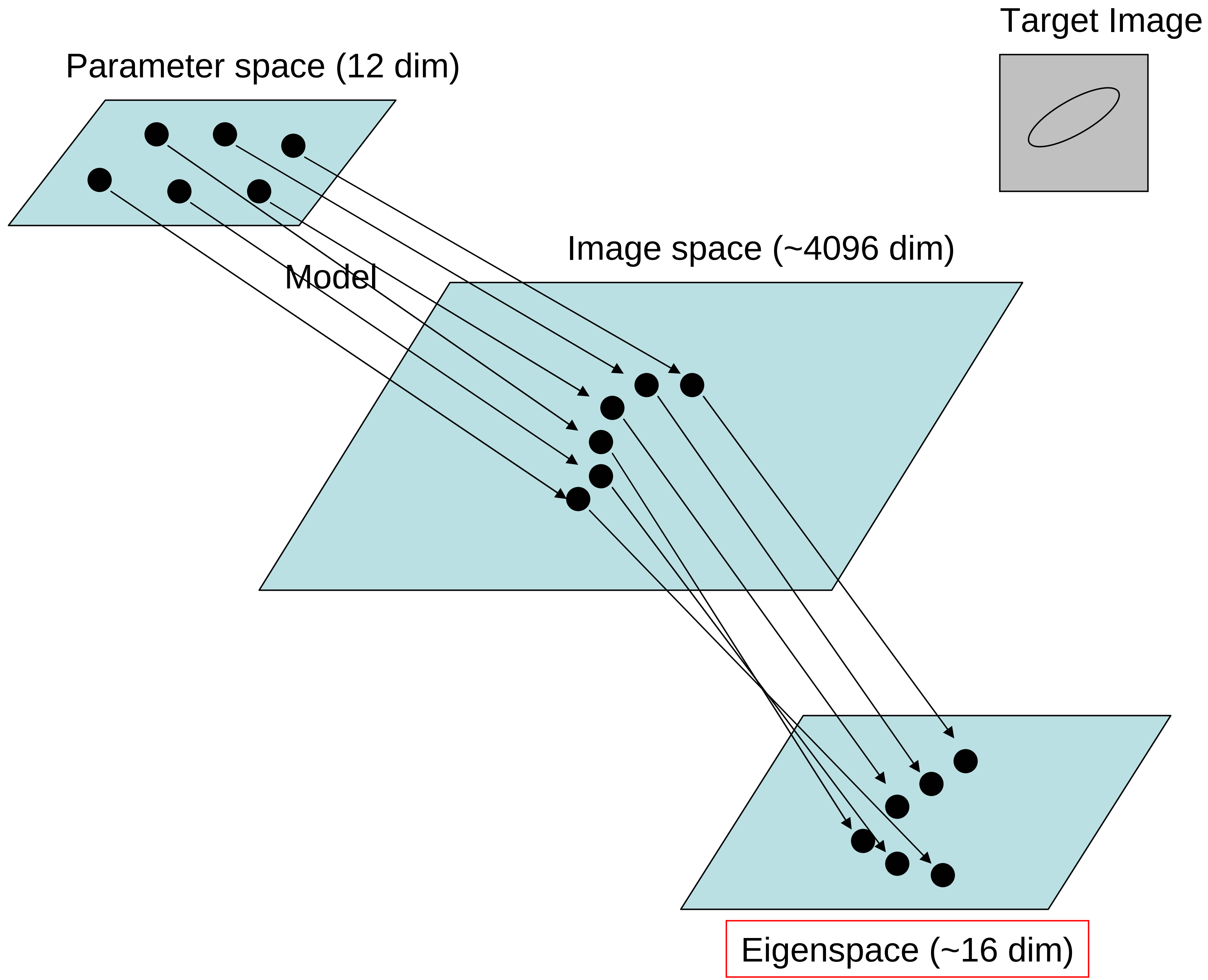
Outline

- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ✓ Fast kernel density estimation
- ✓ Large-scale galactic morphology
  - ➢ GMorph
    - ✓ Memory-based
    - ➢ PCA

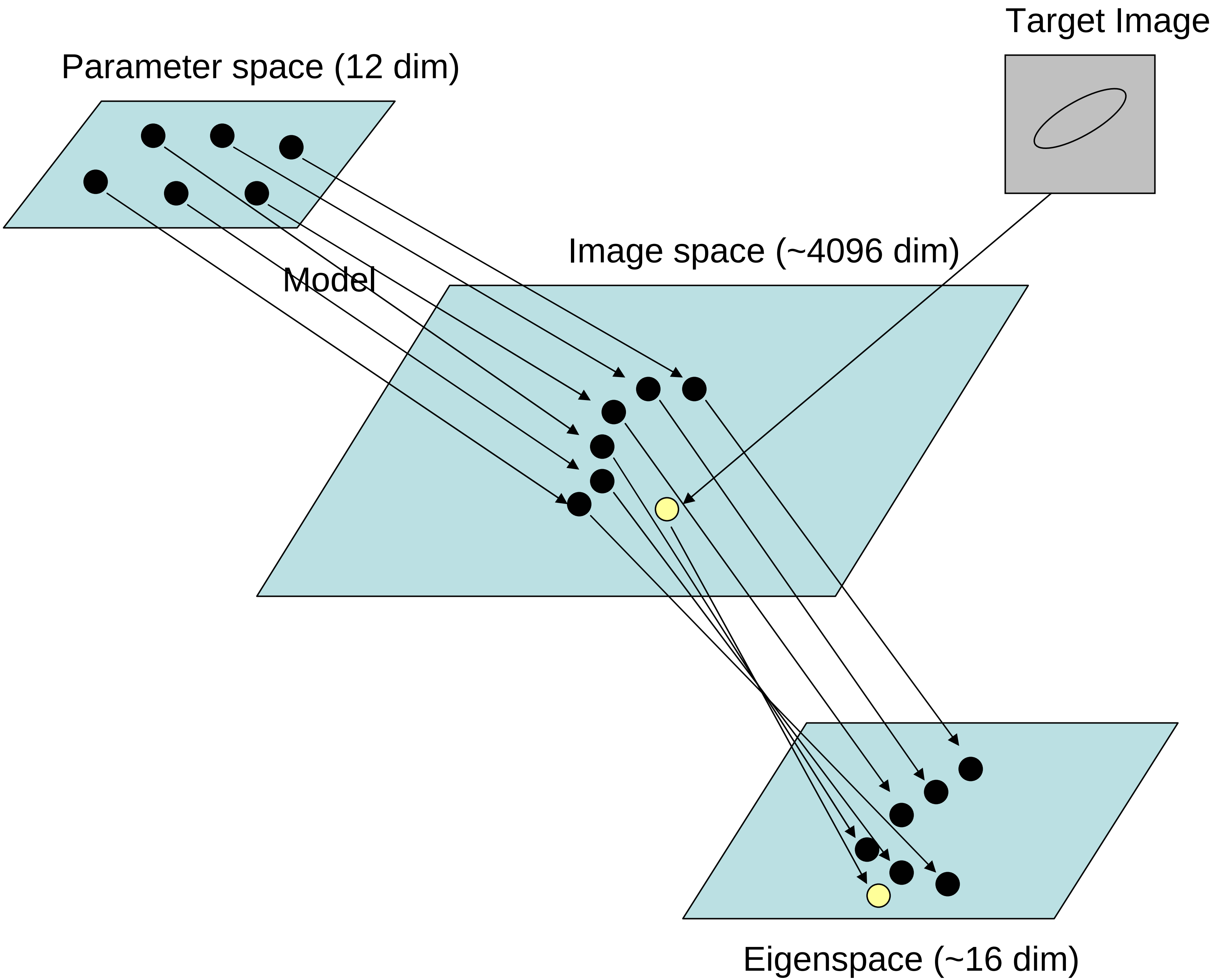

1
0.99
0.98
0.97
0.96
0.95
2
4
6
8
10
12
14
16
Number of Eigenspace Dimensions
Variance Explained

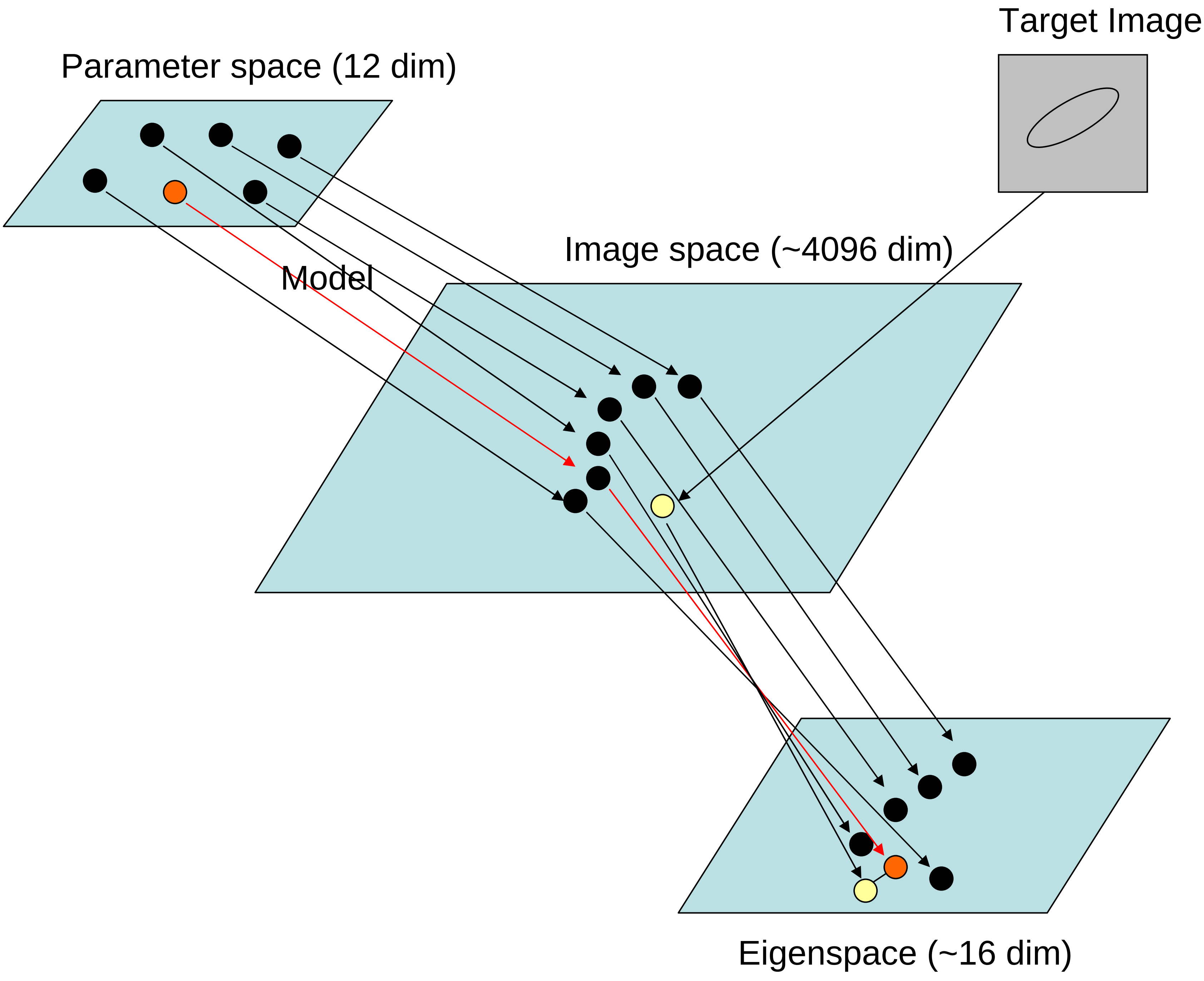

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16



Target Image
Parameter space (12 dim)
Image space (~4096 dim)
Model
Eigenspace (~16 dim)

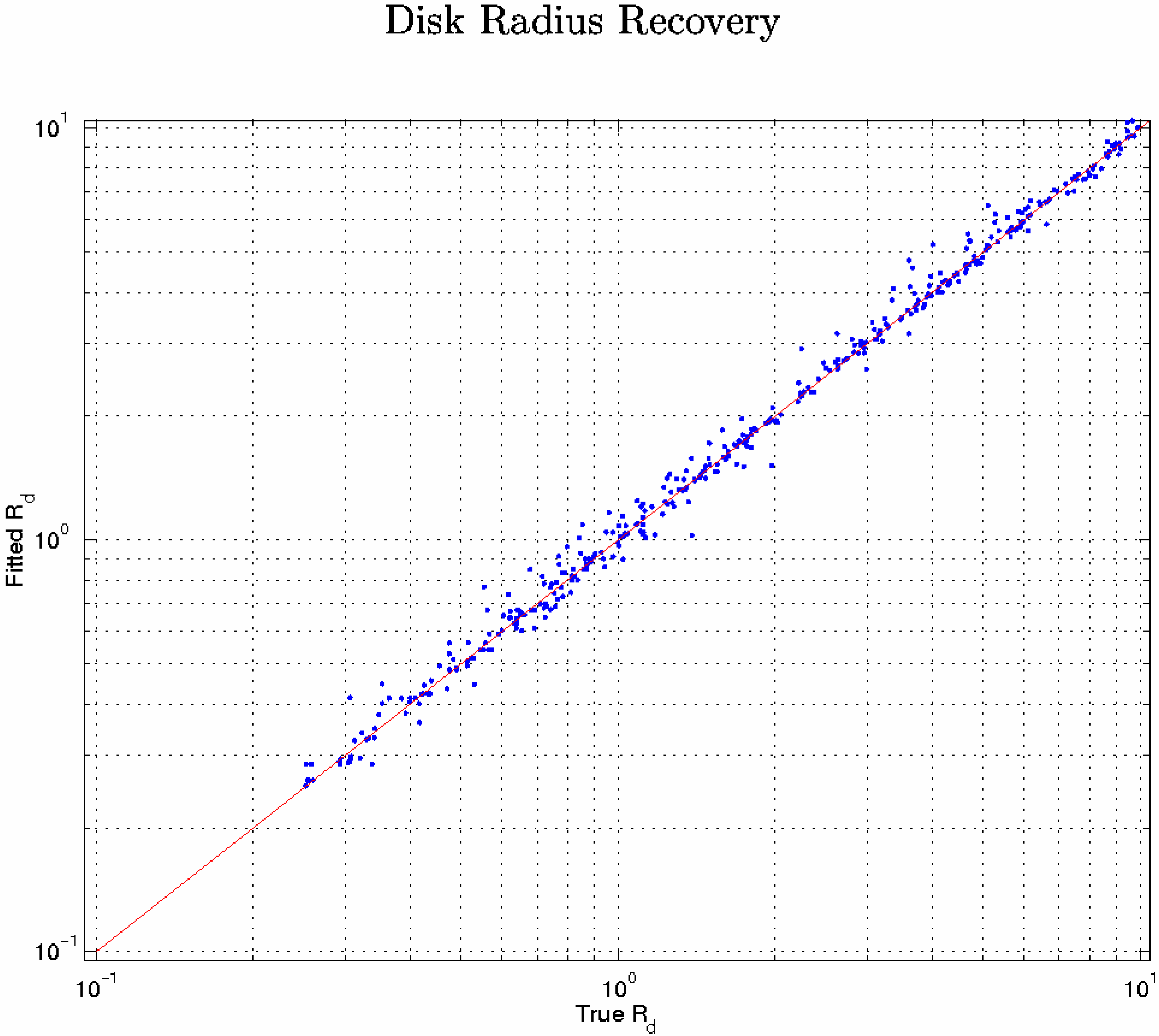

Target Image
Parameter space (12 dim)
Image space (~4096 dim)
Model
Eigenspace (~16 dim)

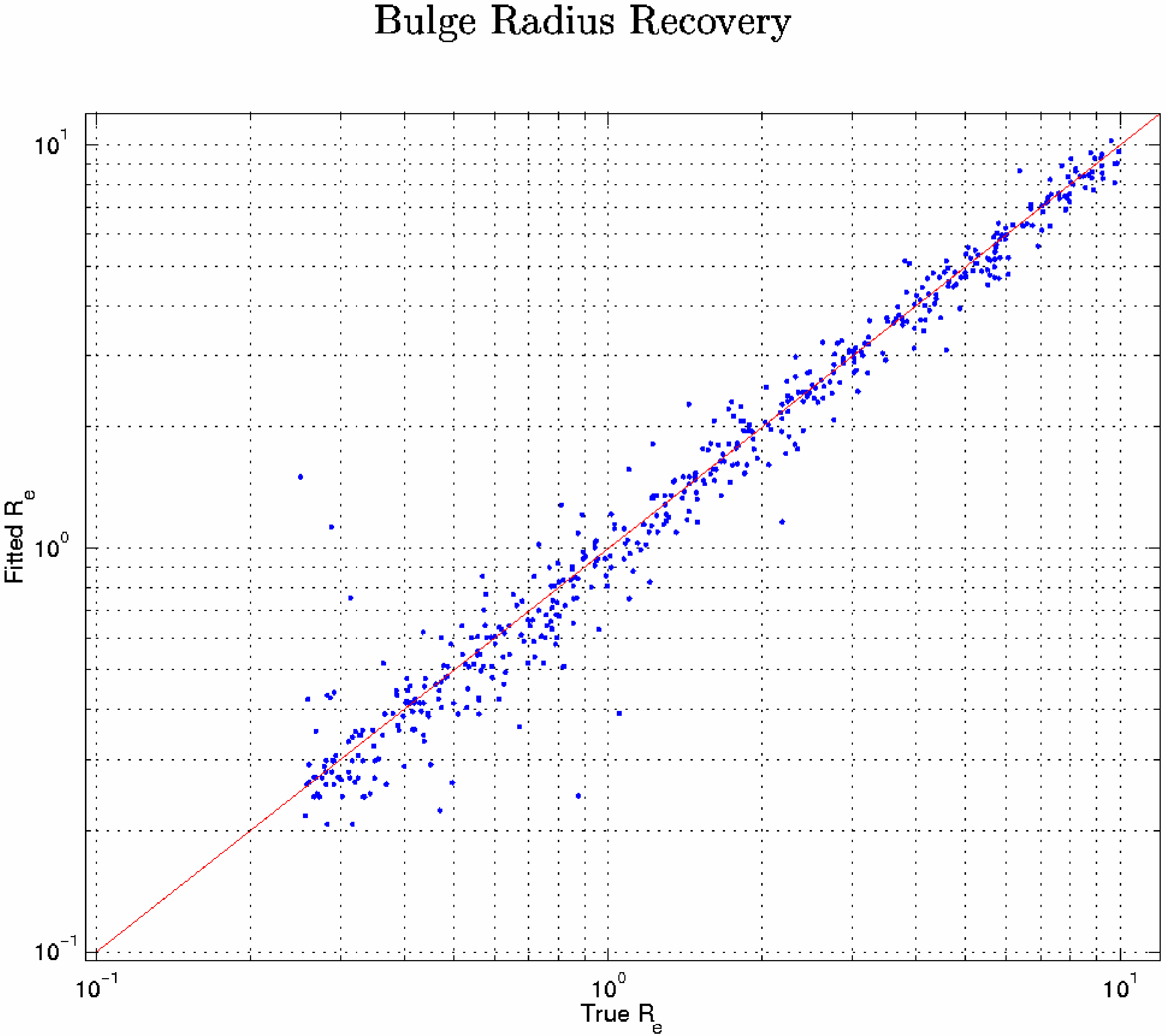

Target Image
Parameter space (12 dim)
Image space (~4096 dim)
Model
Eigenspace (~16 dim)

snapshot

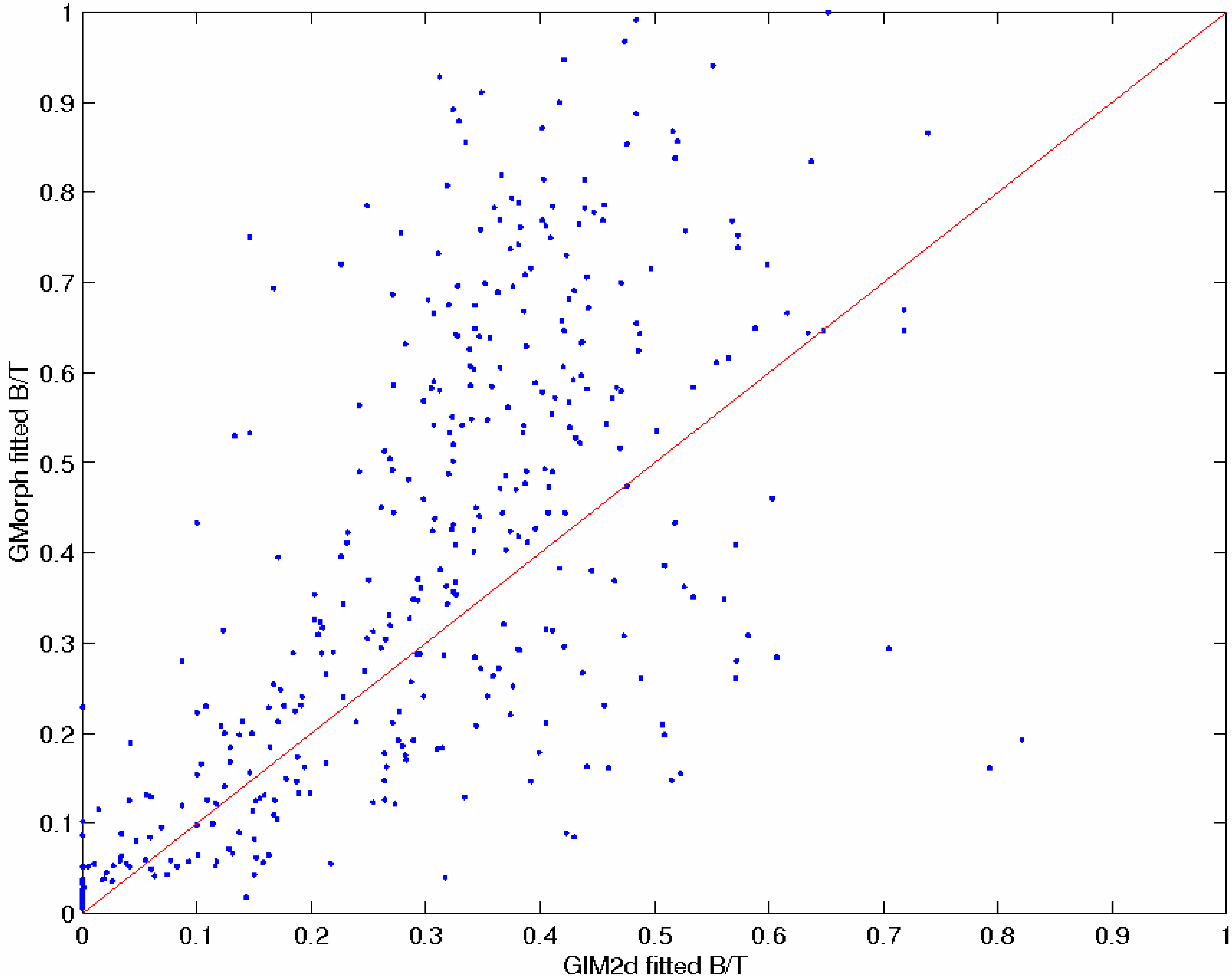
Outline

- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ✓ Fast kernel density estimation
- ✓ Large-scale galactic morphology
  - ➢ GMorph
    - ✓ Memory-based
    - ✓ PCA
    - ➢ Results


Disk Radius Recovery
Fitted R_d
True R_d
10^1
10^0
10^-1
10^-1
10^0
10^1


Bulge Radius Recovery
10^1
10^0
10^{-1}
Fitted R_e
10^{-1}
10^0
10^1
True R_e

GIM2d Comparison (Bulge-to-Total Flux ratio)



Outline

- ✓ Kd-trees
  - ✓ Fast nearest-neighbor finding
  - ✓ Fast K-means clustering
  - ✓ Fast kernel density estimation

See AUTON website for papers

- ✓ Large-scale galactic morphology
  - ✓ GMorph
    - ✓ Memory-based
    - ✓ PCA
    - ✓ Results

paper