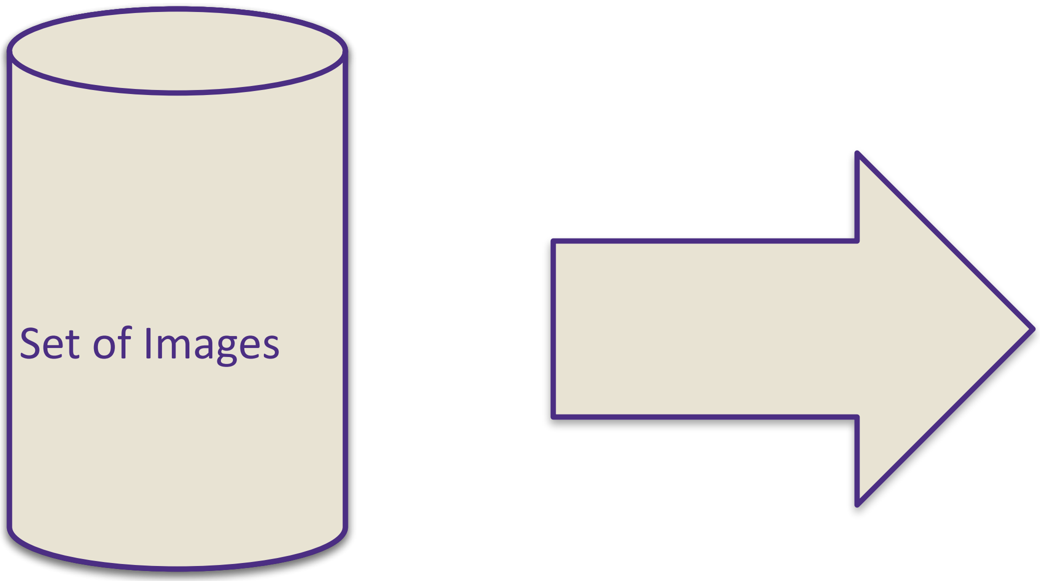
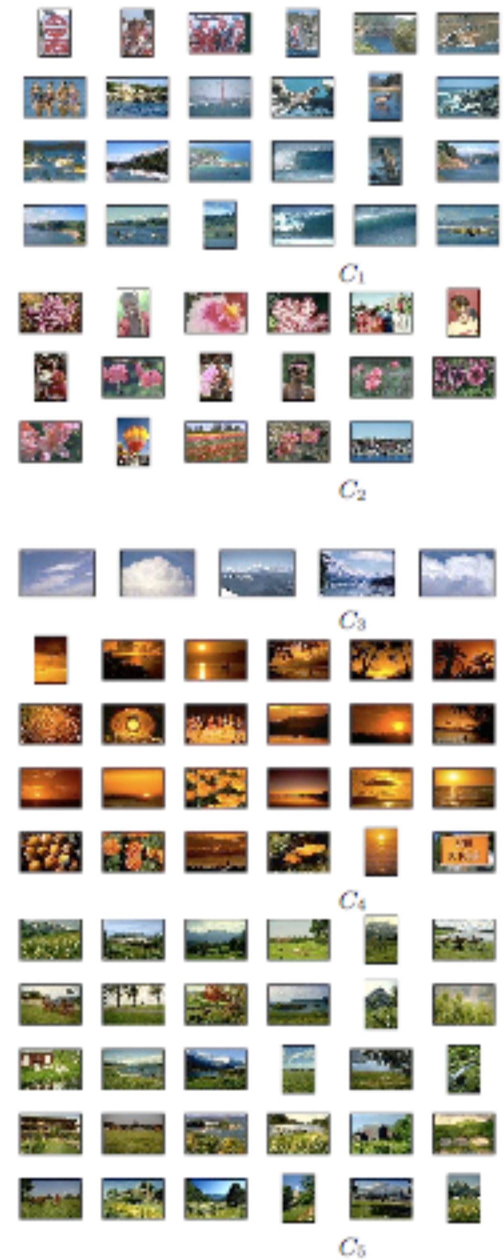
Clustering with k-means

UNIVERSITY of WASHINGTON

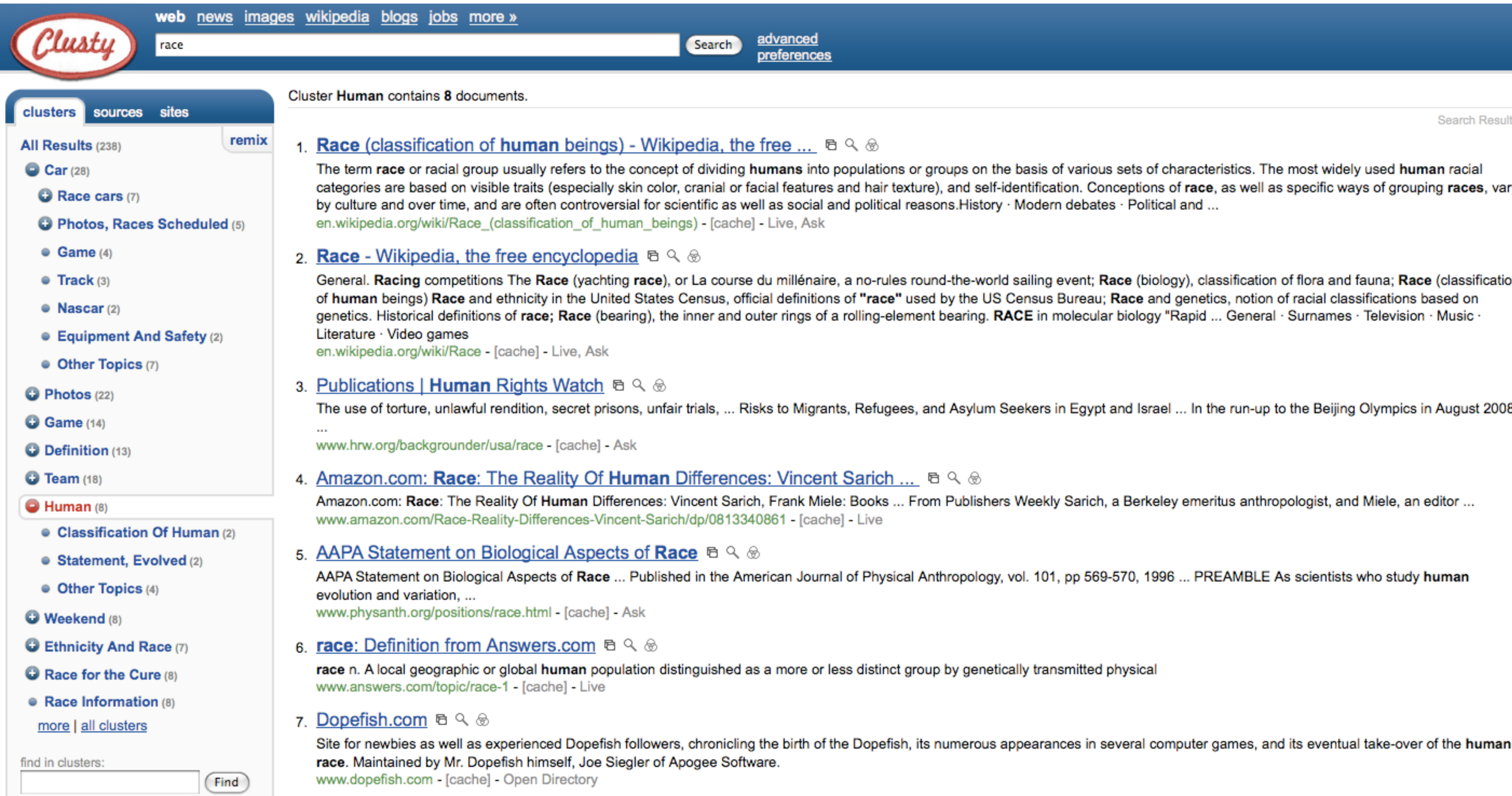
Clustering images



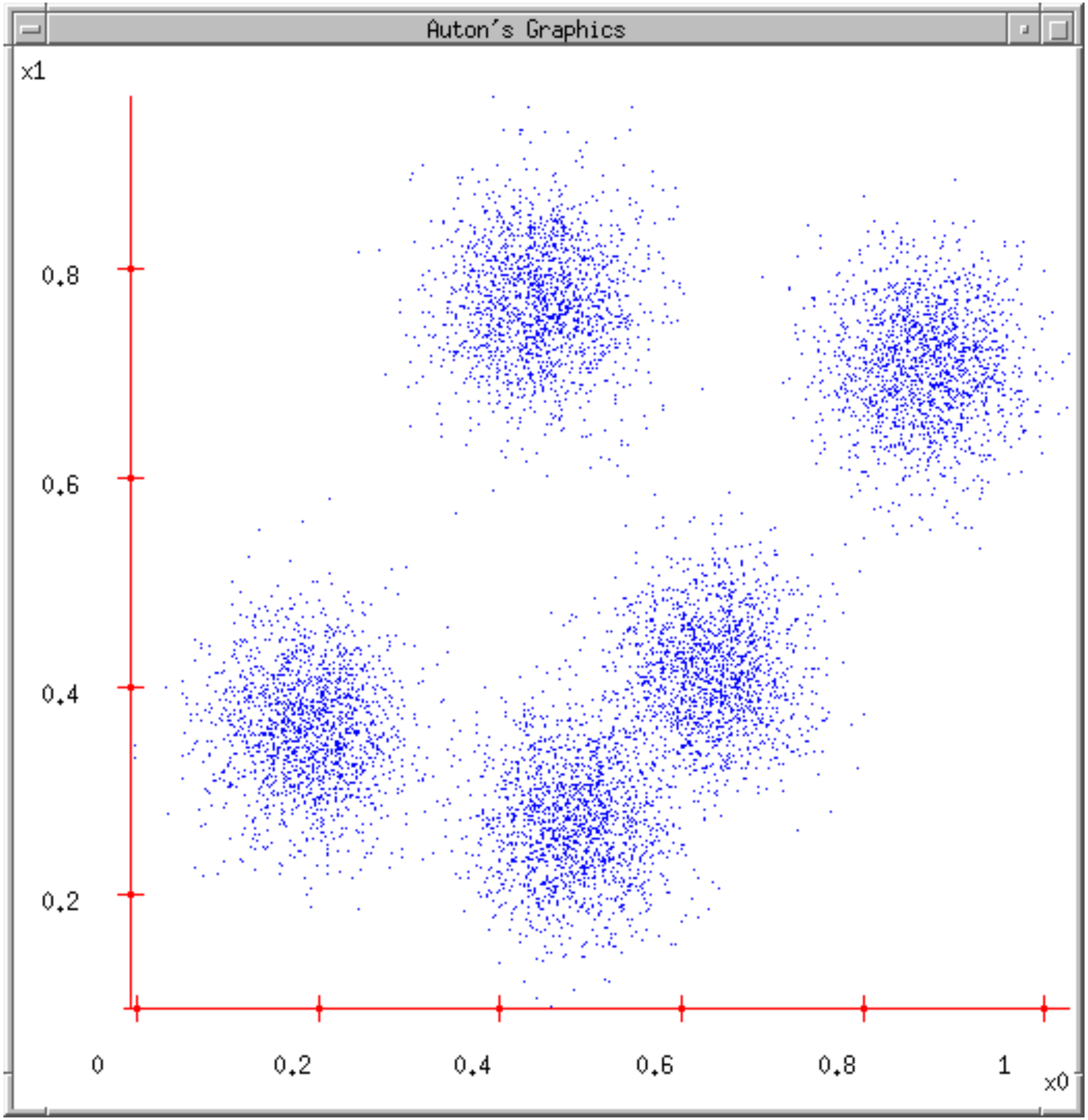


[Goldberger et al.]

Clustering web search results

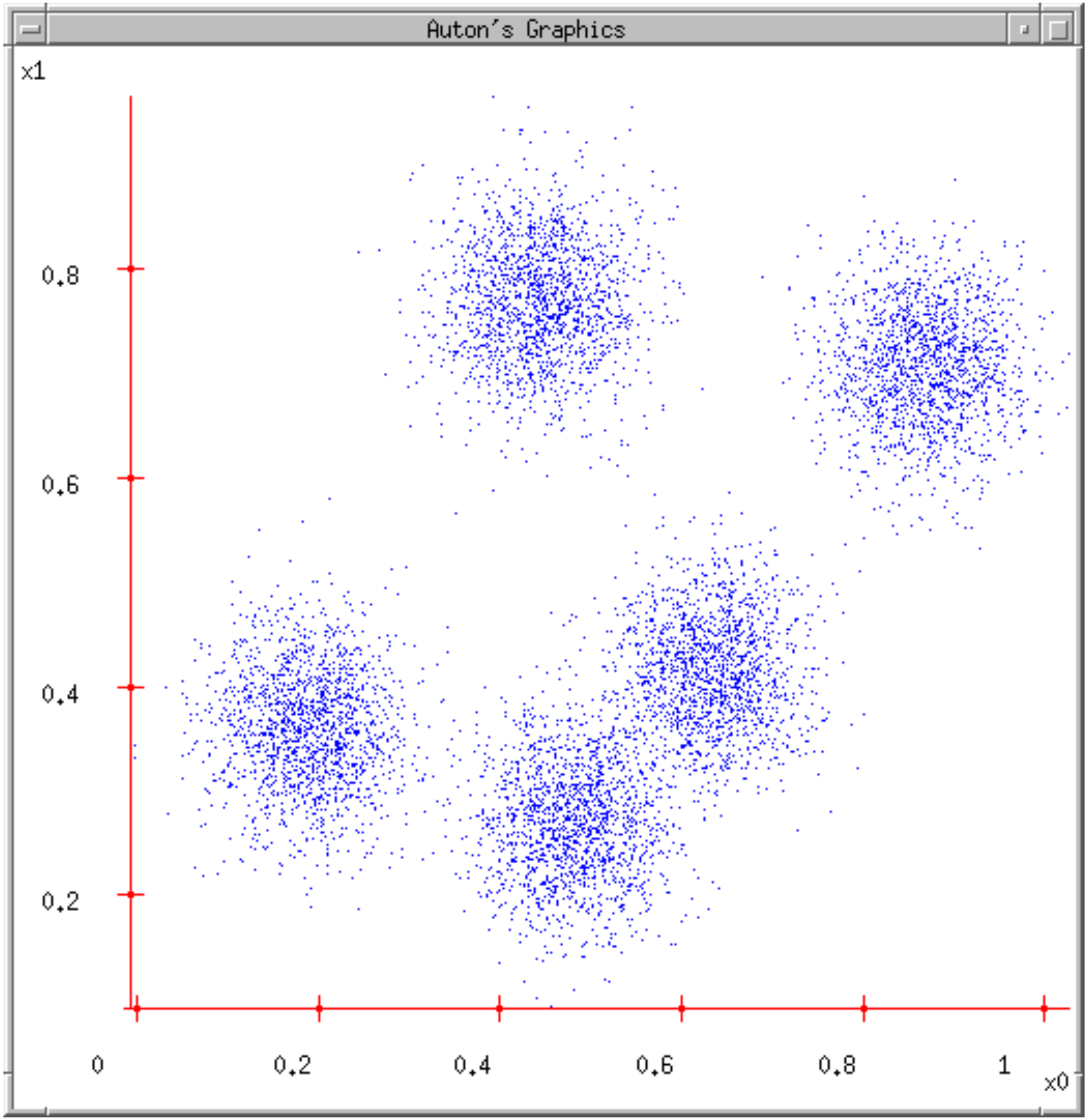


Some Data



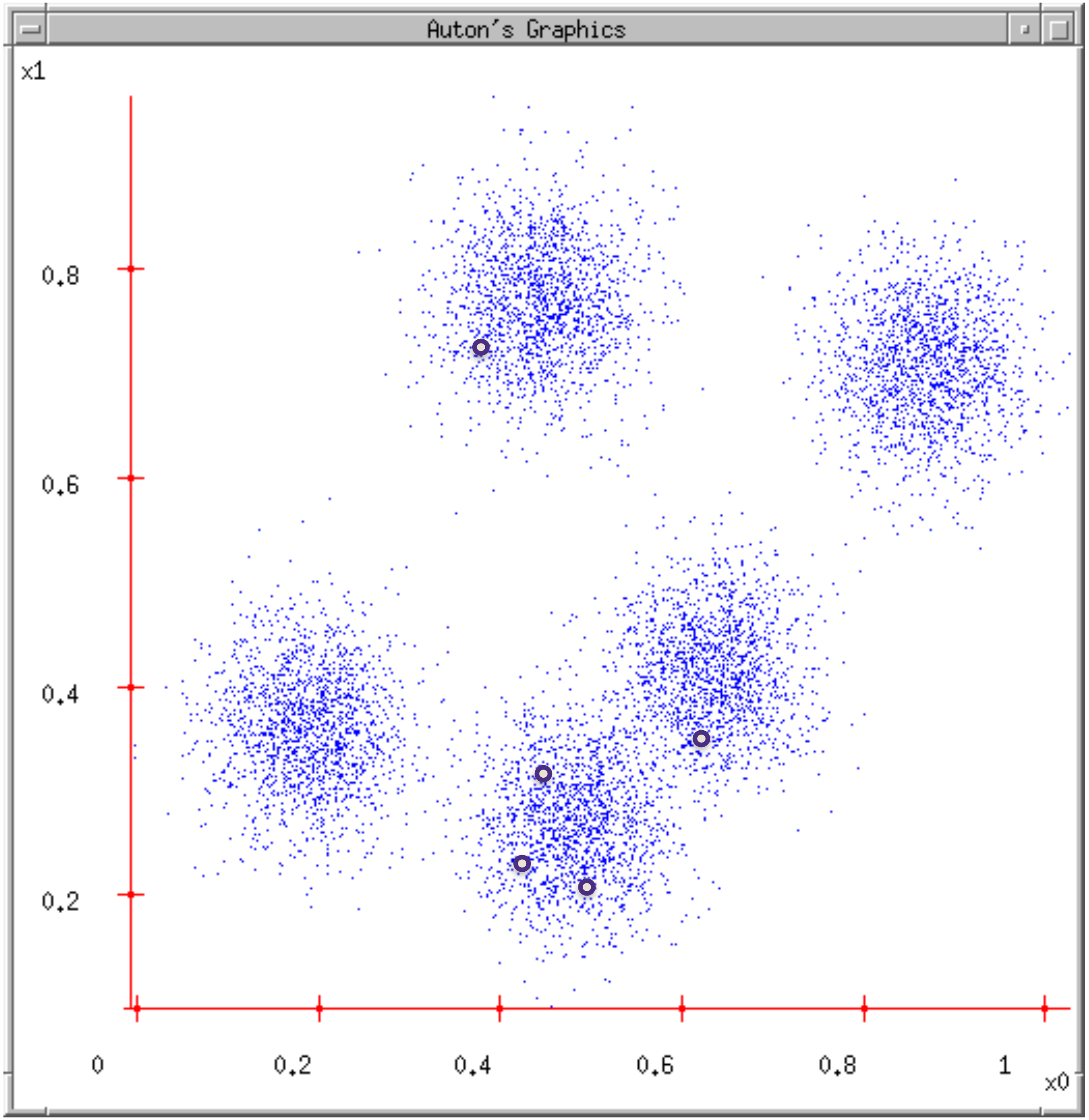
k-means

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



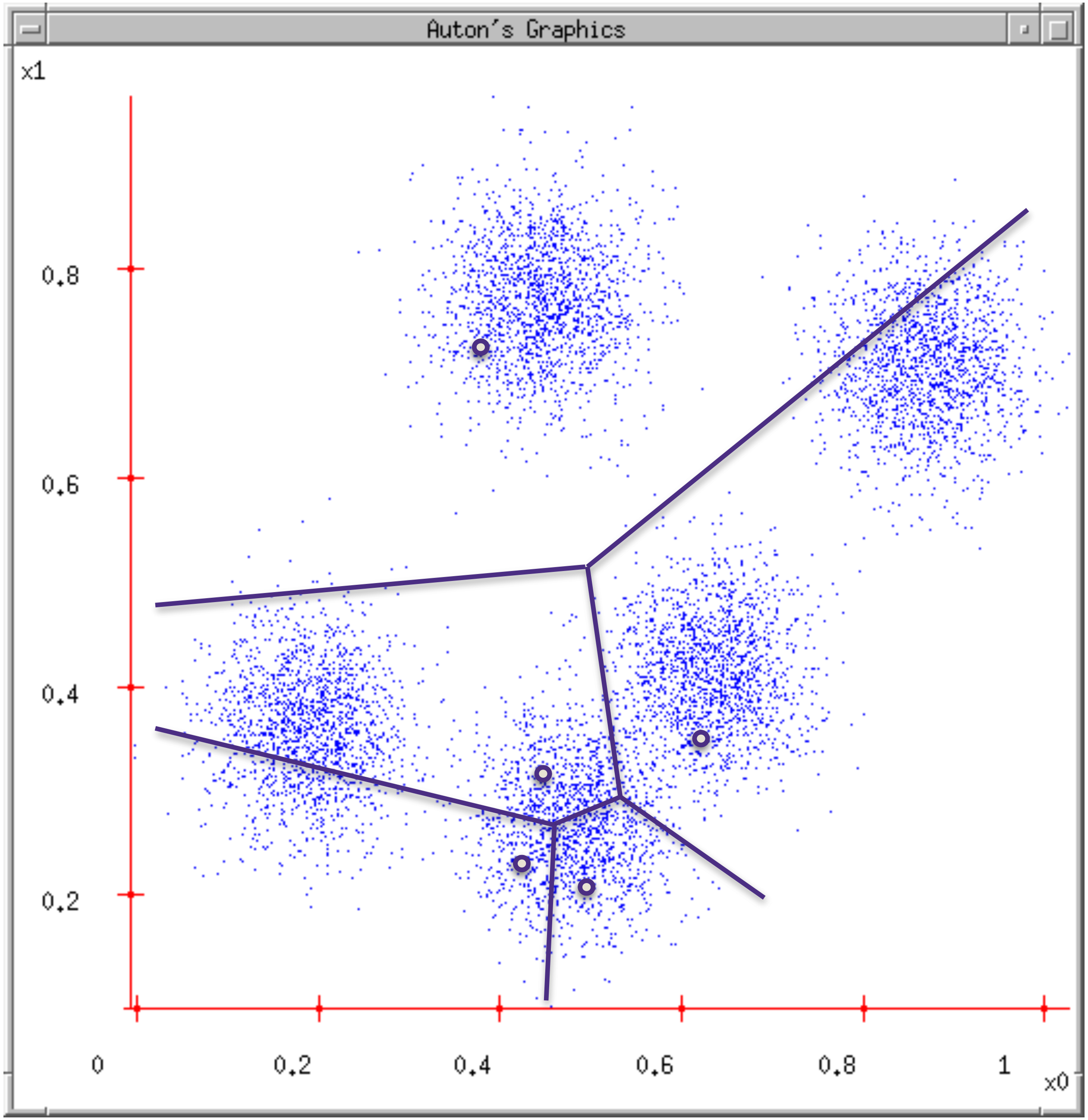
k-means

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



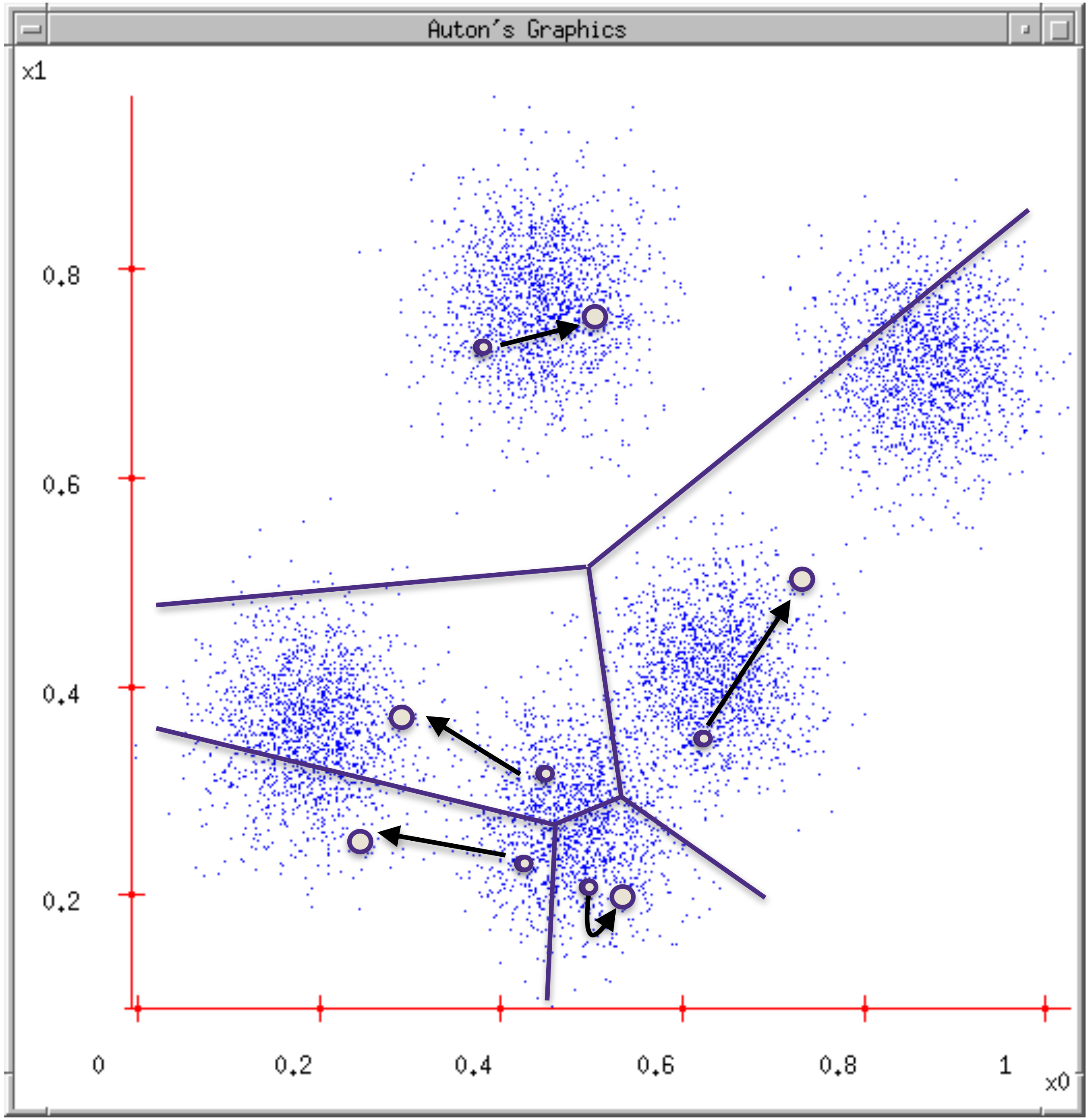
k-means

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



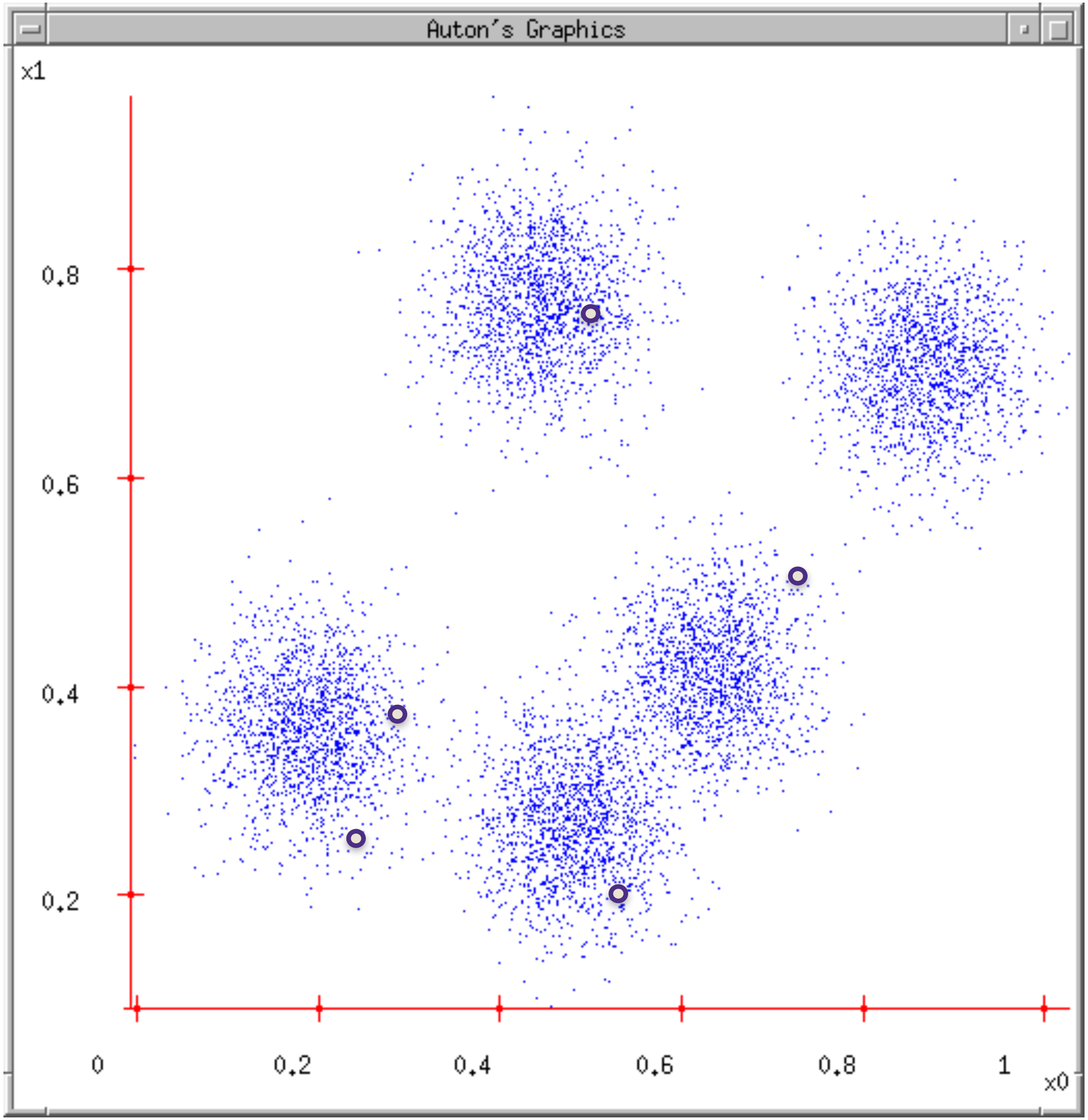
k-means

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

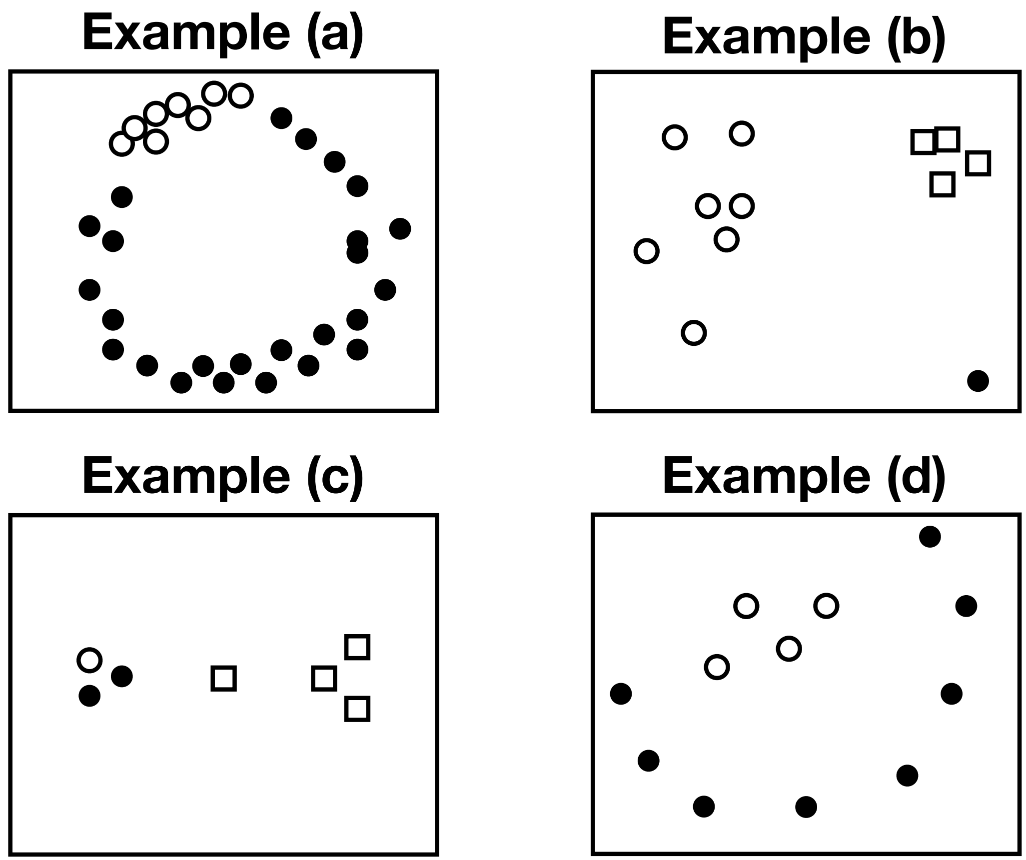


k-means

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



Which one is a snapshot of a converged k-means



k-means

- **Initialize** k centers (as random data points)
  - $\mu^{(0)} = (\mu_1^{(0)}, \dots , \mu_k^{(0)})$

- **Classify:** assign each point j∈{1,…n} to nearest center:
  - For each $j \in \{1,\ldots,n\}$, $C^{(t)}(j) \leftarrow \arg \min_{i\in\{1,\ldots,k\}} \|\mu_i^{(t)} - x_j\|_2^2$

- **Recenter: μ_i becomes centroid of its point:**
  - For each $i \in \{1,\ldots,k\}$, $\mu_i^{(t+1)} \leftarrow \arg\min_{\mu} \sum_{j:C^{(t)}(j)=i} \|\mu - x_j\|_2^2$
  - Equivalent to $\mu_i \leftarrow$ average of its points!

k-means

- **Initialize** k centers (as random data points)
  - $\mu^{(0)} = (\mu_1^{(0)}, \dots, \mu_k^{(0)})$
- **Classify:** assign each point j∈{1,…n} to nearest center:
  - For each $j \in \{1,\dots,n\}$, $C^{(t)}(j) \leftarrow \arg\min_{i\in\{1,\dots,k\}} \|\mu_i^{(t)} - x_j\|_2^2$
- **Recenter:** μ_i **becomes centroid of its point:**
  - For each $i \in \{1,\dots,k\}$, $\mu_i^{(t+1)} \leftarrow \arg\min_{\mu} \sum_{j:C^{(t)}(j)=i} \|\mu - x_j\|_2^2$
  - Equivalent to $\mu_i \leftarrow$ average of its points!

What does k-means do? Coordinate descent on

$$F(\mu, C) = \sum_{j=1}^{n} \|\mu_{C(j)} - x_j\|_2^2$$

Does k-means converge??

- k-means is trying to minimize the following objective

$$F(\mu, C) = \sum_{j=1}^{n} \|\mu_{C(j)} - x_j\|_2^2$$

- Via coordinate descent:
  - Fix μ, optimize C

Does k-means converge??

- k-means is trying to minimize the following objective

$$F(\mu, C) = \sum_{j=1}^{n} \|\mu_{C(j)} - x_j\|_2^2$$

- Via coordinate descent:
  - Fix μ, optimize C

$$\min_C \sum_{j=1}^{n} \|\mu_{C(j)} - x_j\|_2^2$$

    by solving n separate problems:

$$\min_{C(j)} \|\mu_{C(j)} - x_j\|_2^2$$

    whose solution is

$$C(j) \leftarrow \arg \min_{i \in \{1,\ldots,k\}} \|\mu_{C(j)} - x_j\|_2^2$$

Does k-means converge??

> k-means is trying to minimize the following objective

$$F(\mu, C) = \sum_{j=1}^{n} \|\mu_{C(j)} - x_j\|_2^2$$

> Via coordinate descent:

> > Fix C, optimize μ

$$\min_{\mu} \sum_{j=1}^{n} \|\mu_{C(j)} - x_j\|_2^2$$

by solving k separate problems

$$\min_{\mu_i} \sum_{j:C(j)=i} \|\mu_i - x_j\|_2^2$$

whose solution is

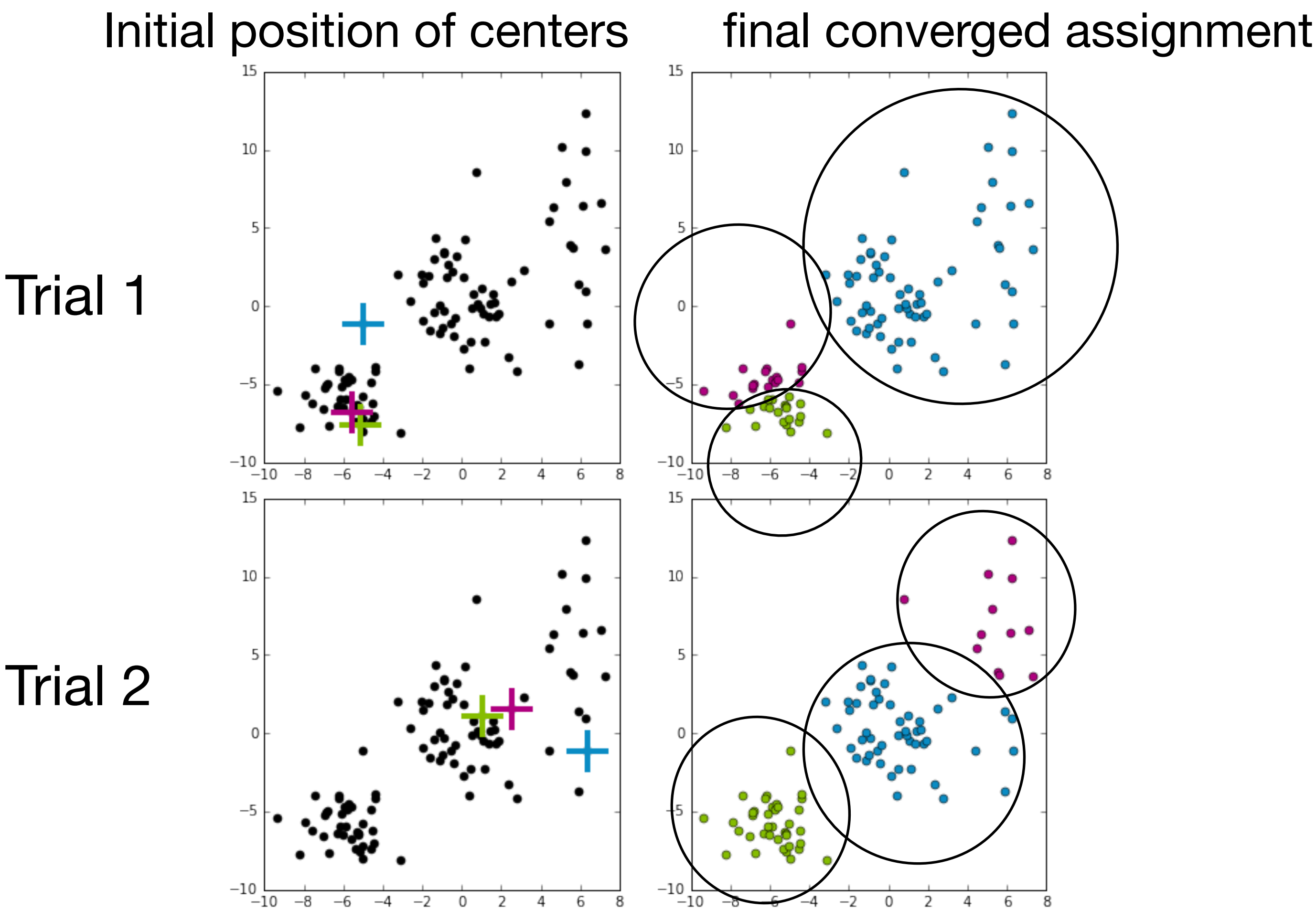
$$\mu_i \leftarrow \frac{1}{|\{j : C(j) = i\}|} \sum_{j:C(j)=i} x_j$$

Does k-means converge??

- there is only a finite set of values that $\{C(j)\}_{j=1}^{n}$ can take (k^n is large but finite)
- so there is only finite, k^n at most, values for μ also
- each time we update them, we will never increase the objective function $\sum_{i=1}^{k} \sum_{j:C(j)=i} \|x_j - \mu_i\|_2^2$
- the objective is lower bounded by zero
- after at most k^n steps, the algorithm must converge (as the assignments $\{C(j)\}_{j=1}^{n}$ cannot return to previous assignments in the course of k-means iterations)

Downside of k-means

- the final solution depends on the initialization
  (as it is a coordinate descent on a non-convex problem)



k-means ++: a smart initialization

Smart initialization:

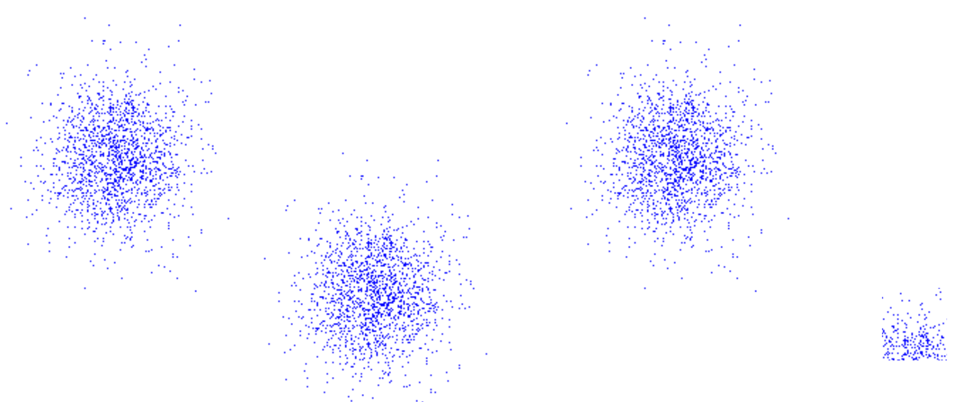
1. Choose first cluster center uniformly at random from data points

2. Repeat $k - 1$ times

  3. For each data point x_j, compute distance d_j to the nearest cluster center

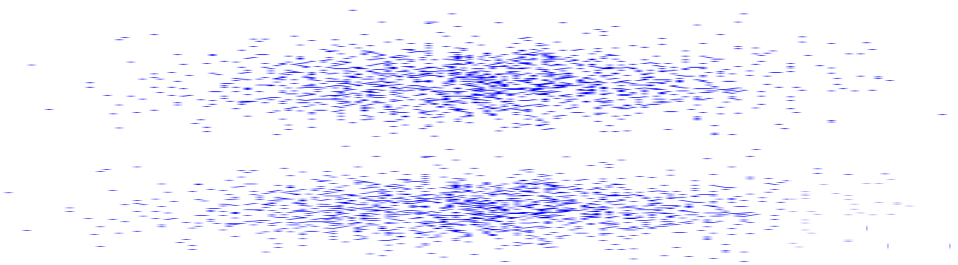
  4. Choose new cluster center from amongst data points, with probability of x_j being chosen proportional to $(d_j)^2$

- apply standard K-means after the initialization
- k-means++ achieves k-means error at most a factor of $\log k$ worse than the optimal [Arther,Vassilvitskii,2007]

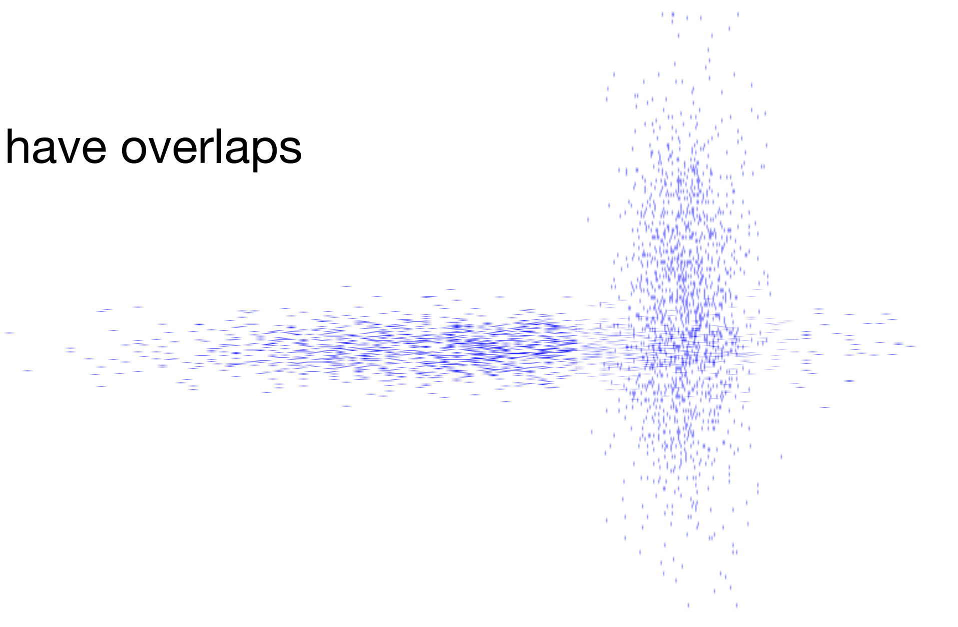


Downside of k-means

- Cluster shapes can be different

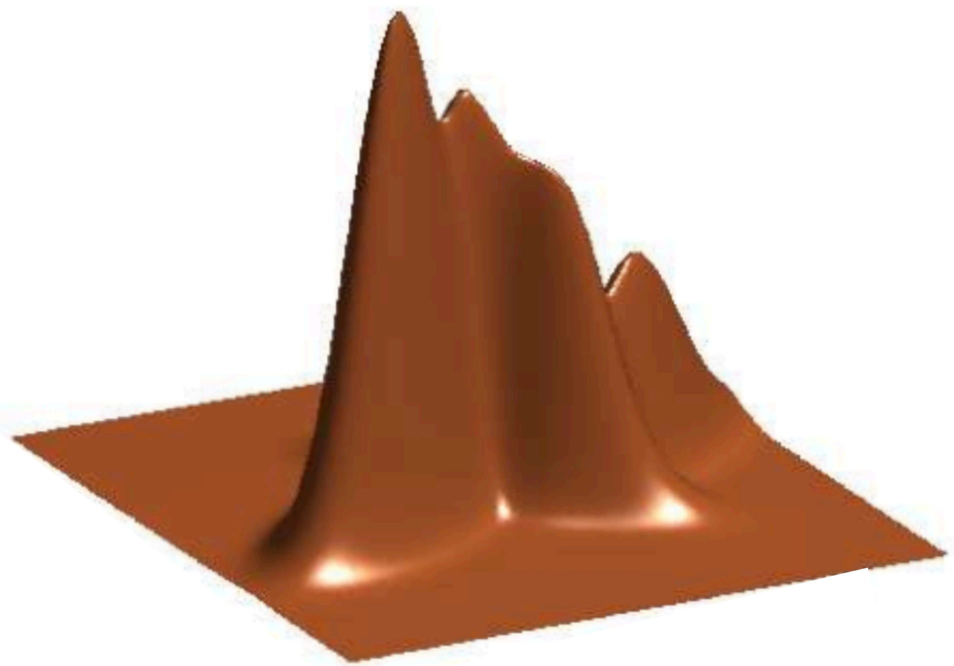


- Or clusters can have overlaps



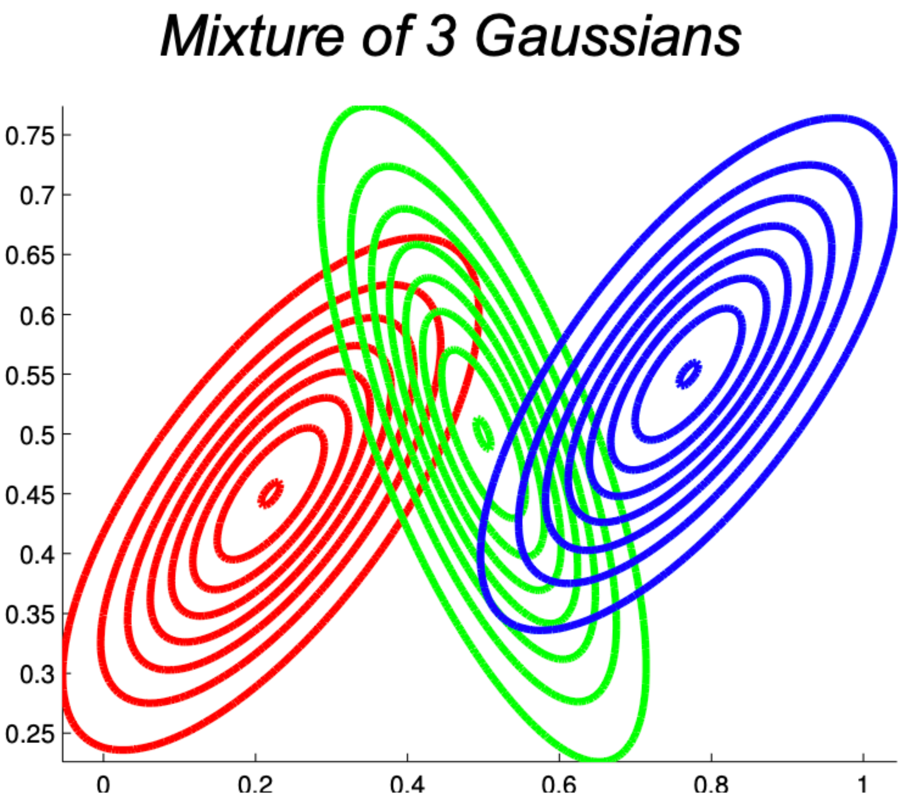
Solution: density estimation

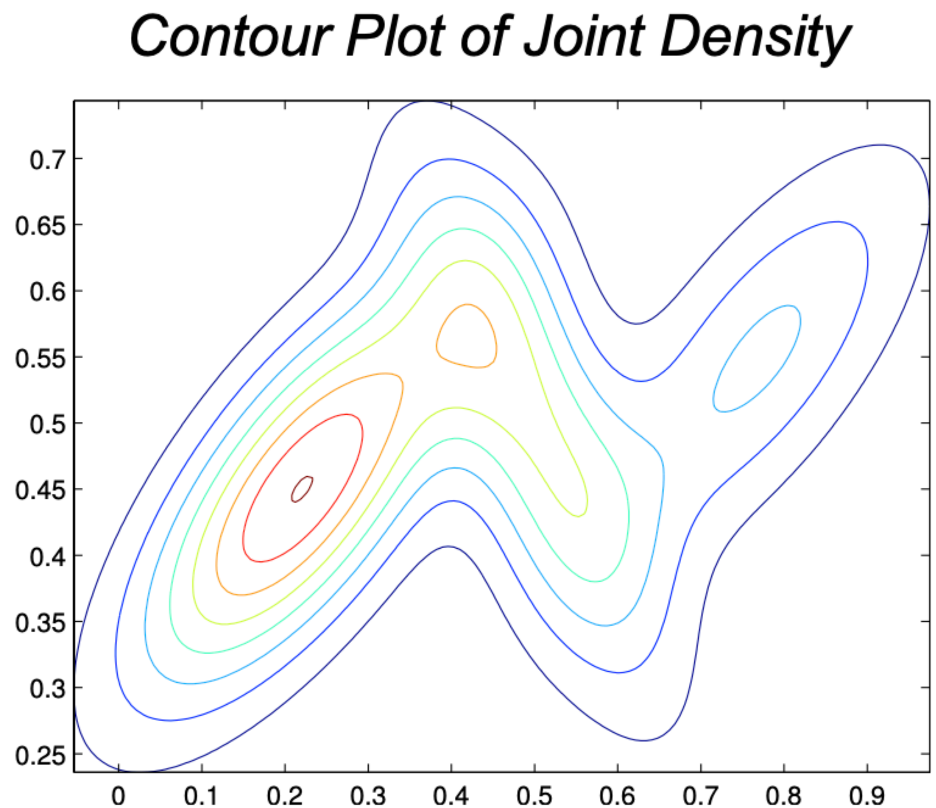
> Estimate probability density function from n i.i.d. samples $x_1, x_2, \ldots, x_n$



Density as mixture of Gaussians

> Approximate unknown density with a mixture of Gaussians



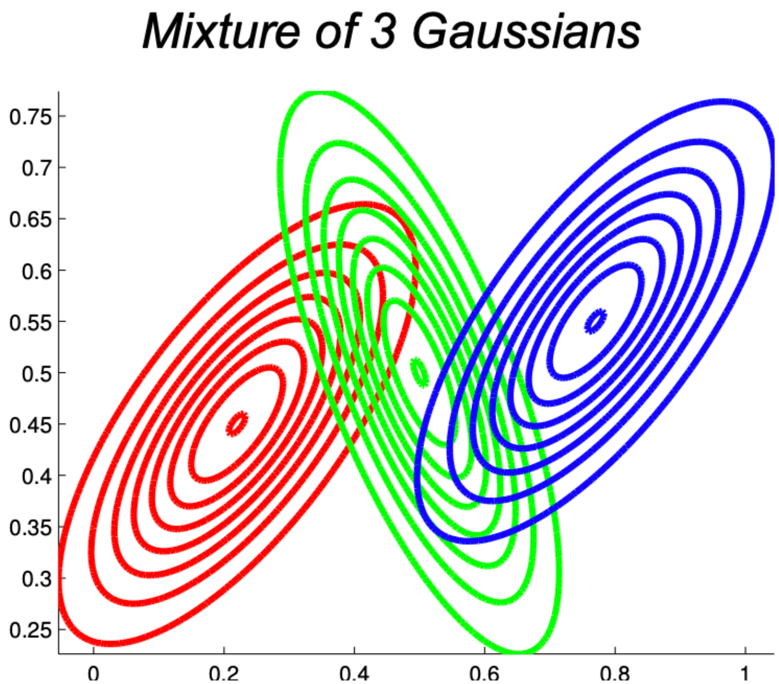


Mixture of Gaussians

$$P(x;\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

> Approximate unknown density with a mixture of Gaussians

$$P(x_j;\pi,\mu,\Sigma) =$$

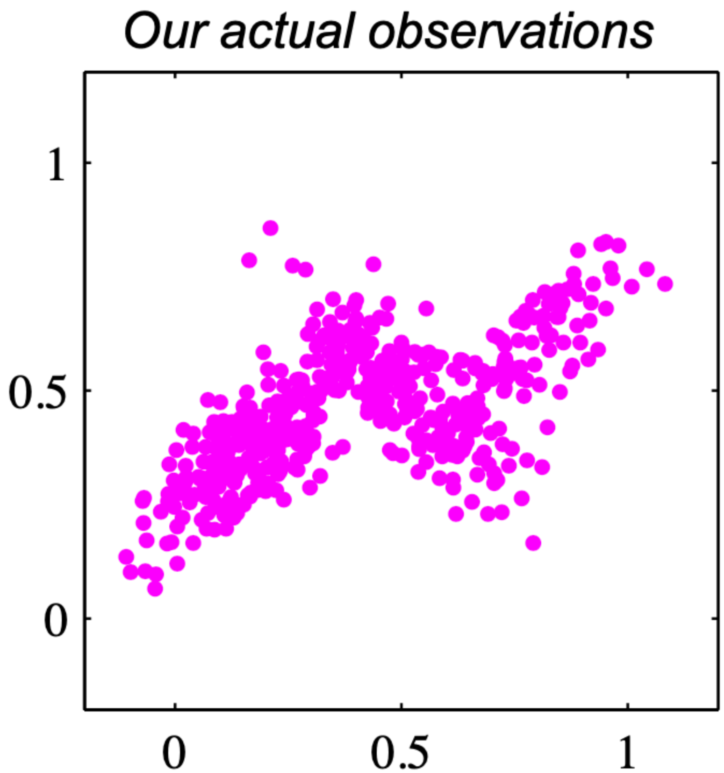


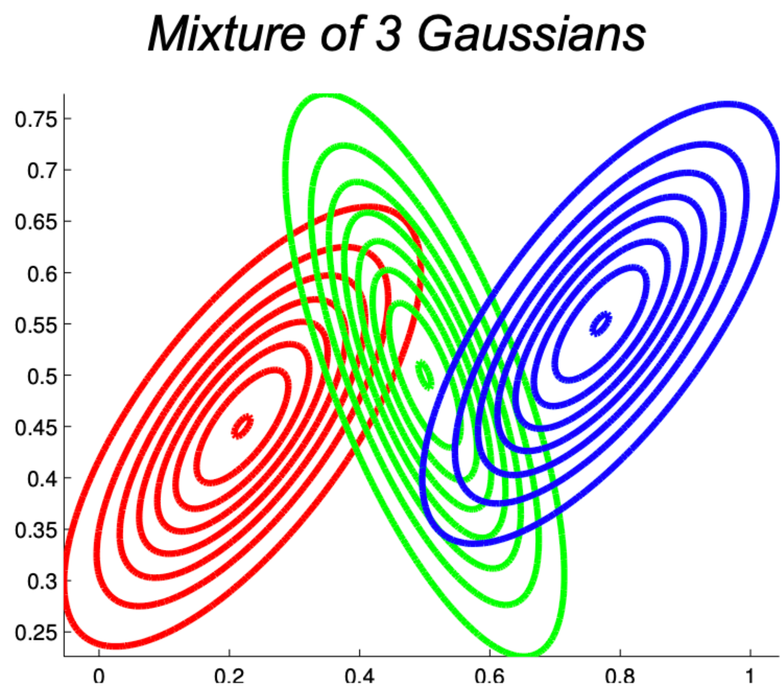
$$\sum_{i=1}^{k}\pi_i = 1$$

$$\pi_i \geq 0$$

Maximum likelihood solves clustering

$$\max_{\pi,\mu,\Sigma} \sum_{j=1}^{n} \log P(x_j; \pi, \mu, \Sigma)$$



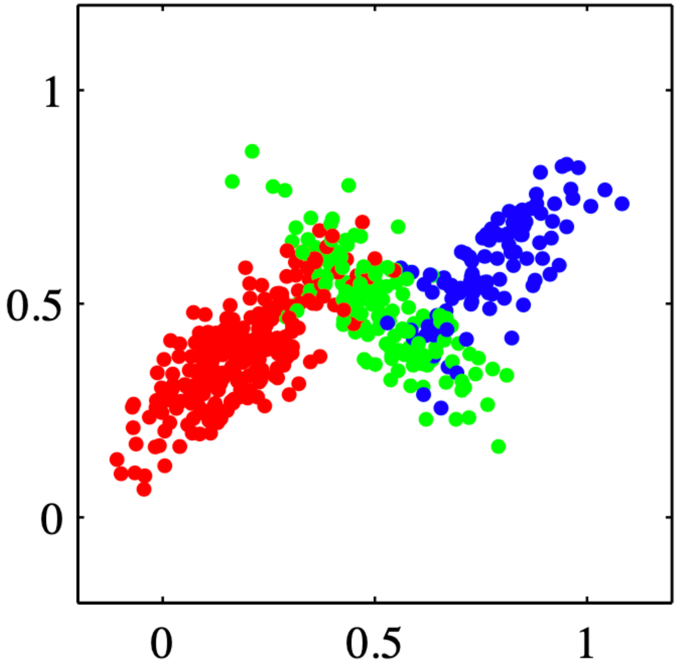


Maximum likelihood solves clustering

To assign clusters, we define latent cluster indicator $z_j \in \{1, \ldots, k\}$

Suppose for just now that we have z_j (true cluster indicator),

then we have $P(x_j; z_j, \pi, \mu, \Sigma) =$



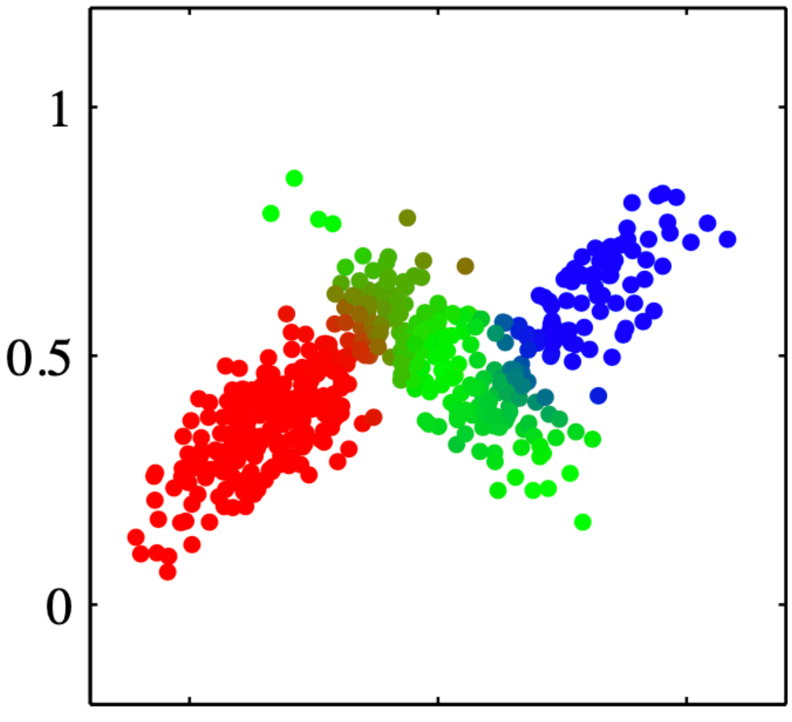
Complete data labeled by true cluster assignments

We can now infer the clusters for each sample using this formula

Maximum likelihood solves clustering

But in practice we do not know z_j's

but we can now infer the clusters, by computing the posterior probability on z_j's



Soft assignments to clusters

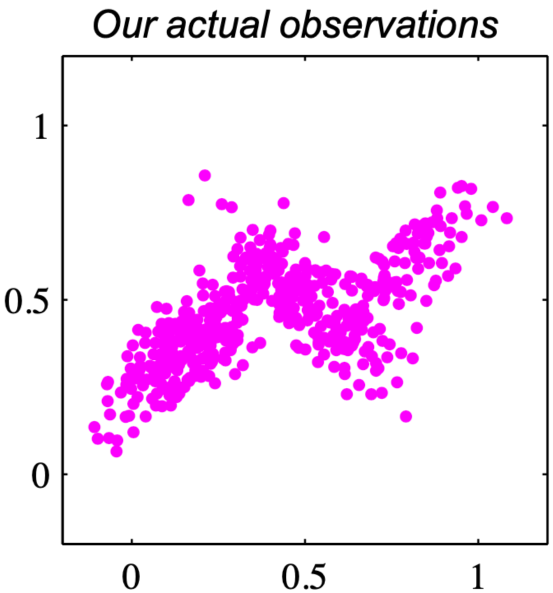
$$
\begin{aligned}
r_{ji} &= P(\text{sample j belongs to cluster i}) \\
&= P(z_j = i \mid x_j; \pi, \mu, \Sigma) \\
&= \frac{\pi_i P(x_j; \mu_i, \Sigma_i)}{\sum_{i'} \pi_{i'} P(x_j; \mu_{i'}, \Sigma_{i'})}
\end{aligned}
$$

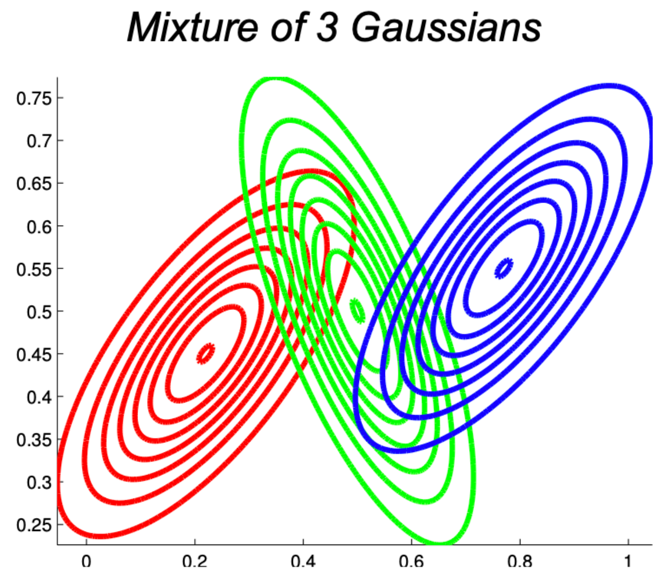
Recap: Mixture of Gaussians for clustering

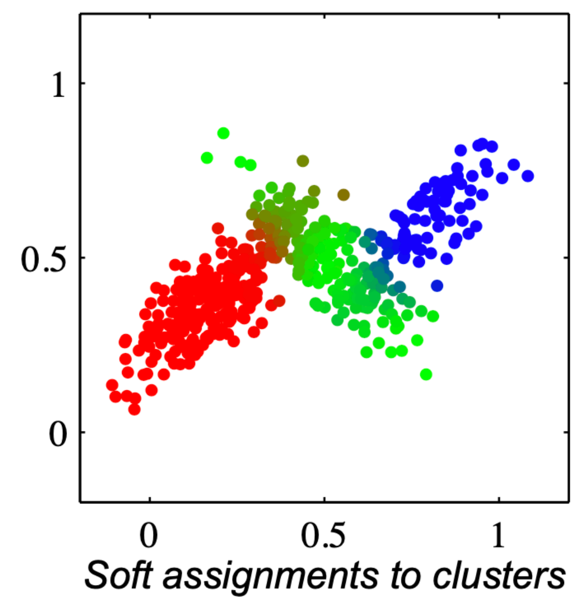
Given a set of samples

1. Fit a mixture of Gaussian model with maximum likelihood
2. Use posterior assignment probability for soft clustering

this can handle overlapping clusters, and clusters of various (oval) shapes and not just circles







Questions?