Clustering

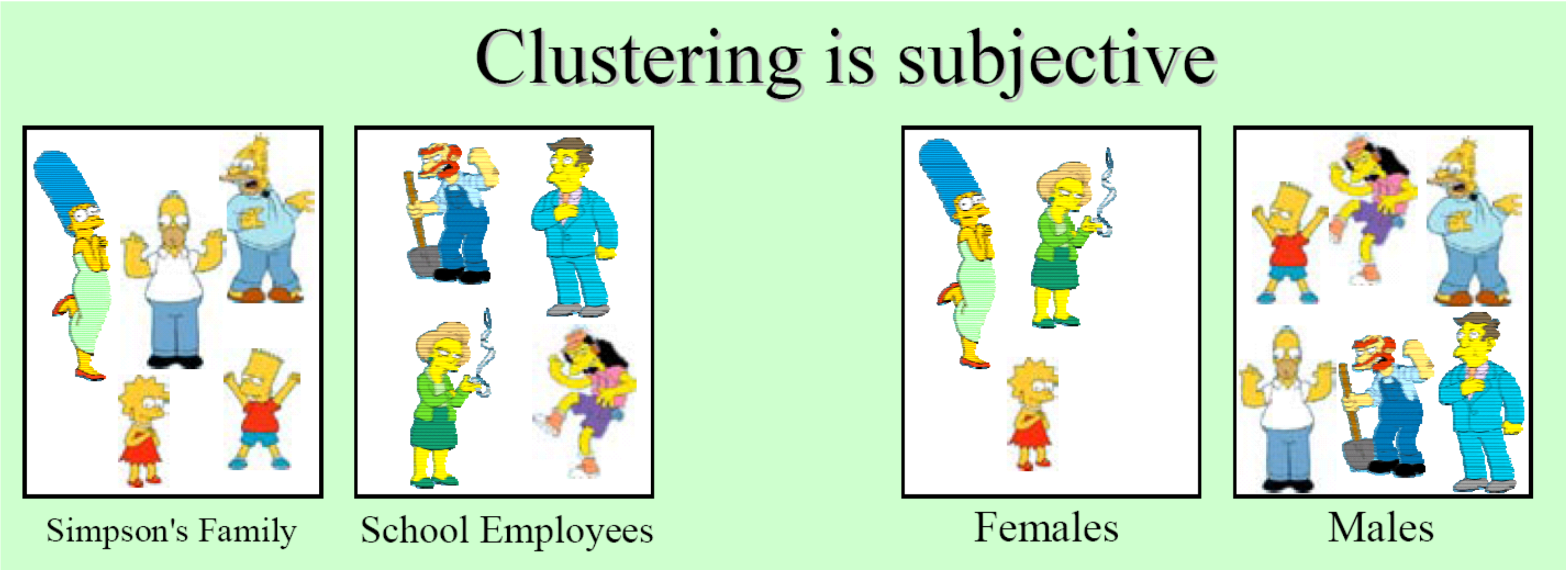
Aarti Singh & Geoff Gordon

Machine Learning 10-701
May 5, 2021

Some slides courtesy of Eric Xing, Carlos Guestrin

What is clustering?

- Clustering: the process of grouping a set of objects into classes of similar objects
  - high intra-class similarity
  - low inter-class similarity
  - It is the most common form of unsupervised learning



What is Similarity?

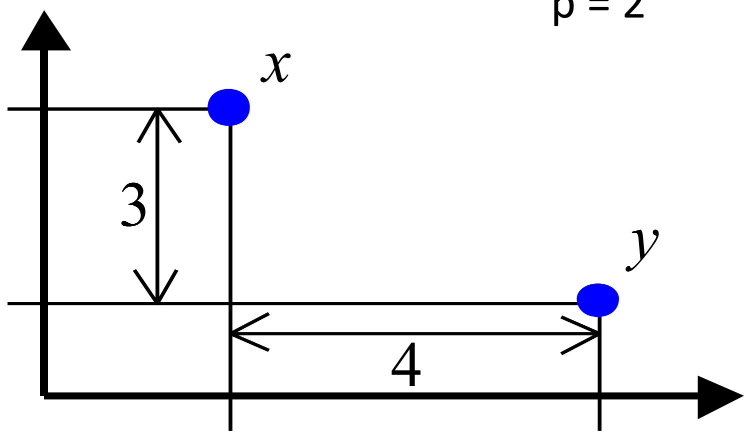
Hard to define! But *we know it when we see it*

- The real meaning of similarity is a philosophical question. We will take a more pragmatic approach - think in terms of a distance (rather than similarity) between vectors or correlations between random variables.

Distance metrics

$x = (x_1, x_2, \ldots, x_p)$
$y = (y_1, y_2, \ldots, y_p)$

p = 2



Euclidean distance $\quad d(x,y) = \sqrt[2]{\sum_{i=1}^{p} |x_i - y_i|^2}$ **5**

Manhattan distance $\quad d(x,y) = \sum_{i=1}^{p} |x_i - y_i|$ **7**

Sup-distance $\quad d(x,y) = \max_{1 \le i \le p} |x_i - y_i|$ **4**

Correlation coefficient

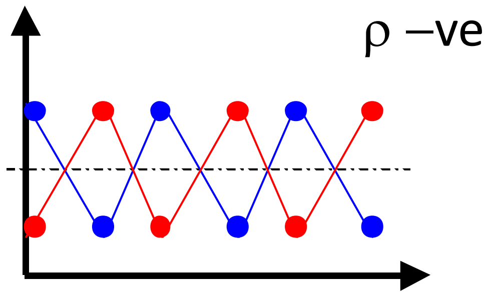
$x = (x_1, x_2, \ldots, x_p)$
$y = (y_1, y_2, \ldots, y_p)$

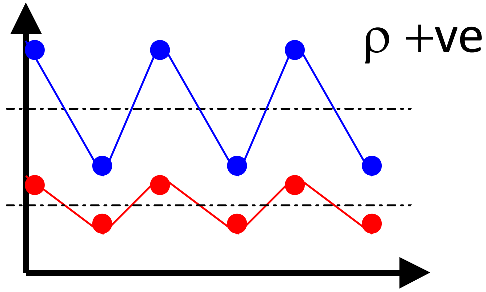
Random vectors (e.g. expression levels of two genes under various drugs)

Pearson correlation coefficient

$$\rho(x, y) = \frac{\sum_{i=1}^{p}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{p}(x_i - \bar{x})^2 \times \sum_{i=1}^{p}(y_i - \bar{y})^2}}$$

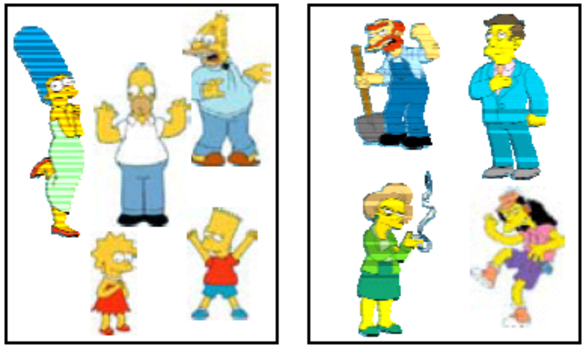
where $\bar{x} = \frac{1}{p}\sum_{i=1}^{p} x_i$ and $\bar{y} = \frac{1}{p}\sum_{i=1}^{p} y_i$.



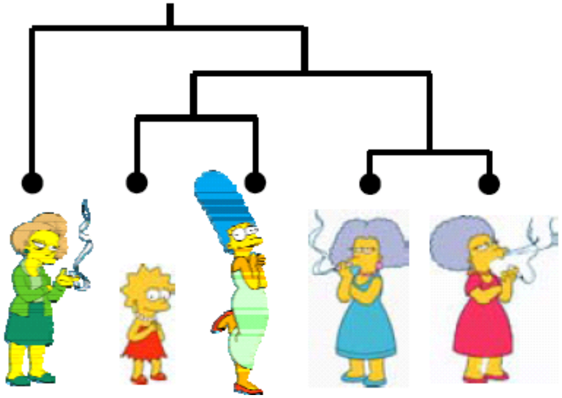


Clustering Algorithms

- **Partition algorithms**
  - K means clustering
  - Mixture-Model based clustering



- Hierarchical algorithms
  - Single-linkage
  - Average-linkage
  - Complete-linkage
  - Centroid-based



Partitioning Algorithms

- Partitioning method: Construct a partition of n objects into a set of K clusters
- Given: a set of objects and the number K
- Find: a partition of K clusters that optimizes the chosen partitioning criterion
  - Globally optimal: exhaustively enumerate all partitions
  - Effective heuristic method: K-means algorithm

K-Means

Algorithm

Input – Desired number of clusters, *k*

Initialize – the *k* cluster centers (randomly if necessary)

Iterate –

1. Assign points to the nearest cluster centers

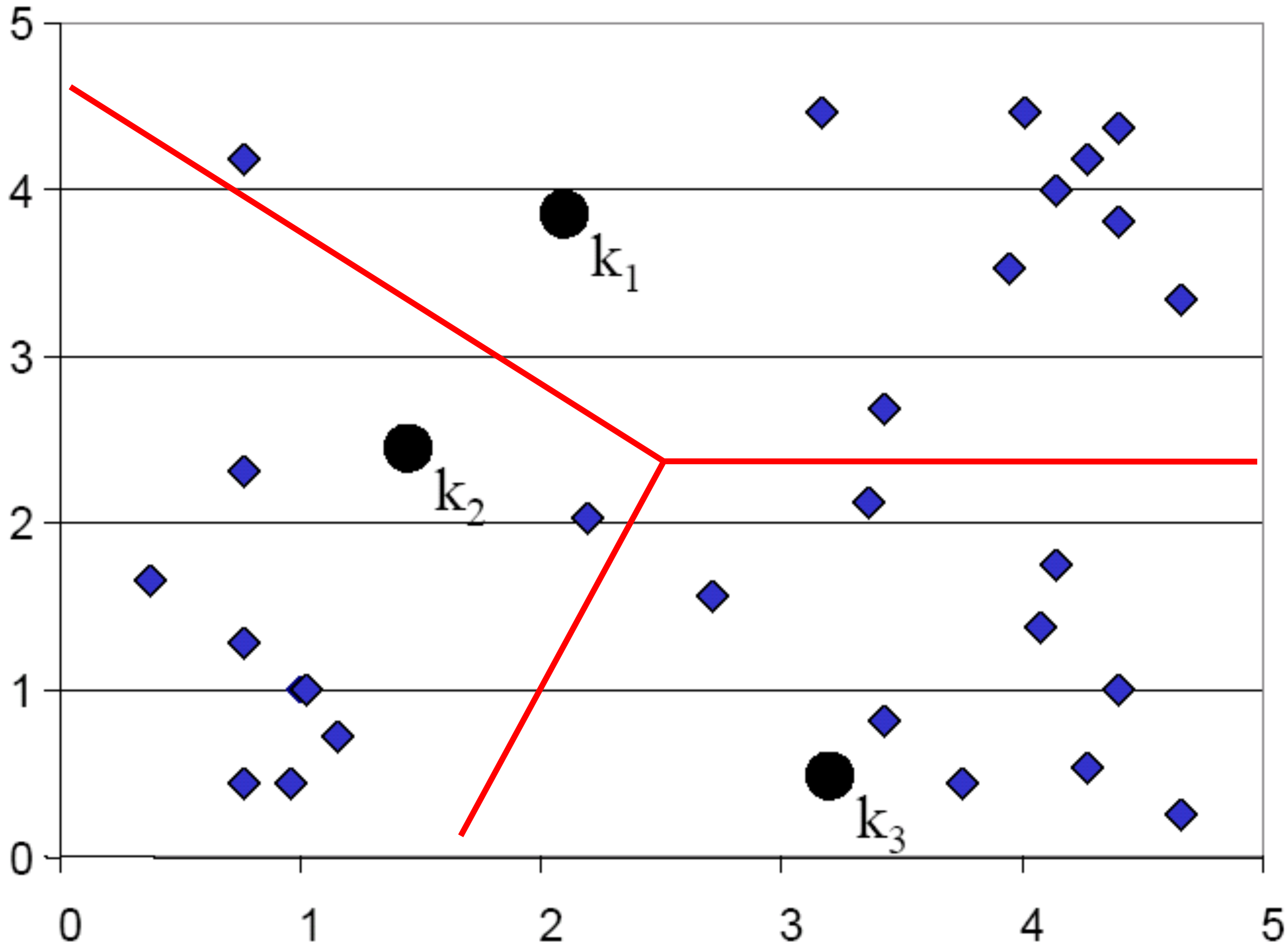
2. Re-estimate the *k* cluster centers (aka the centroid or mean), by assuming the memberships found above are correct.

$$\vec{\mu}_k = \frac{1}{\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \vec{x}_i$$

Termination –

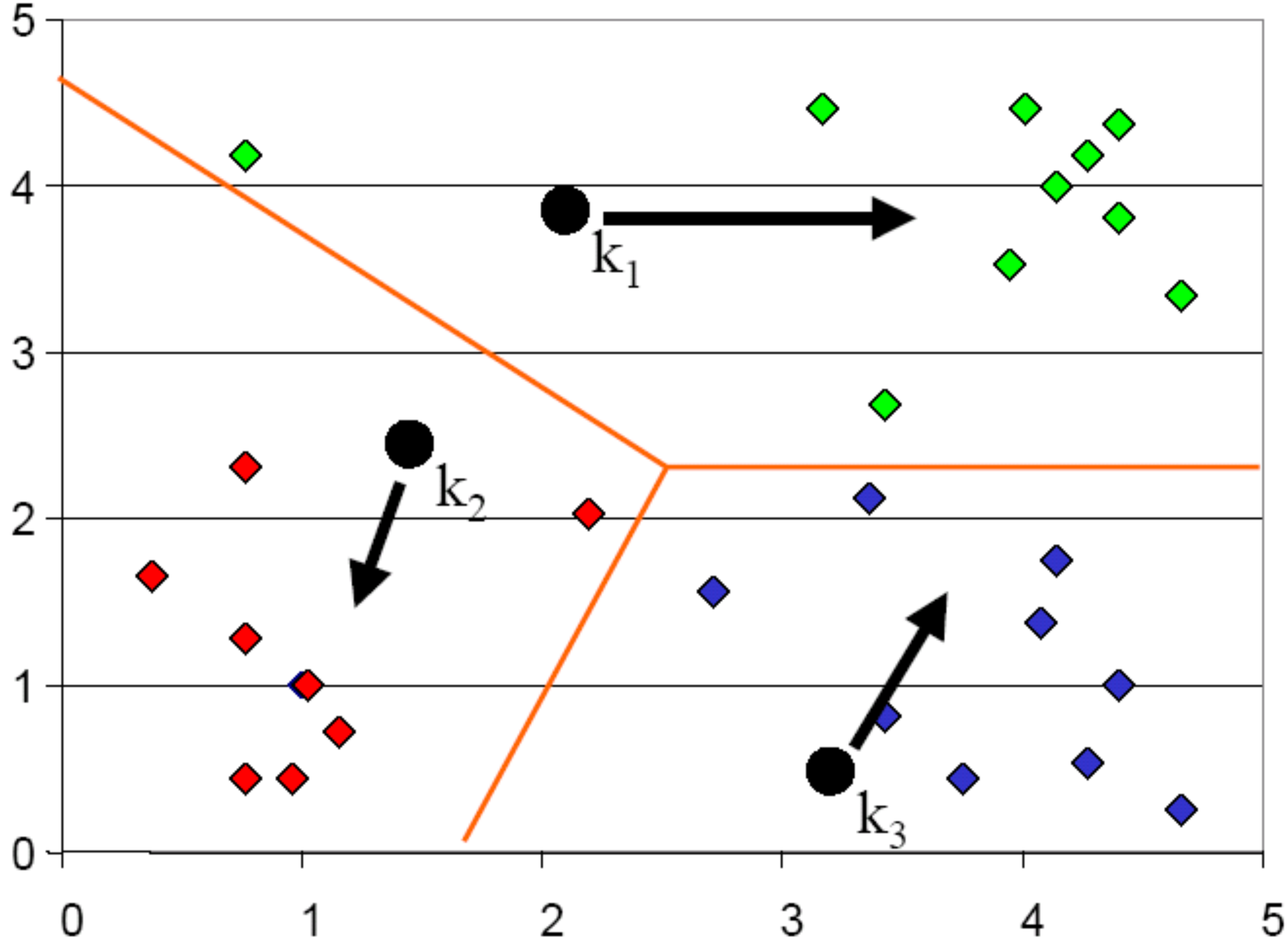
If none of the objects changed membership in the last iteration, exit. Otherwise go to 1.

K-means Clustering: Step 1

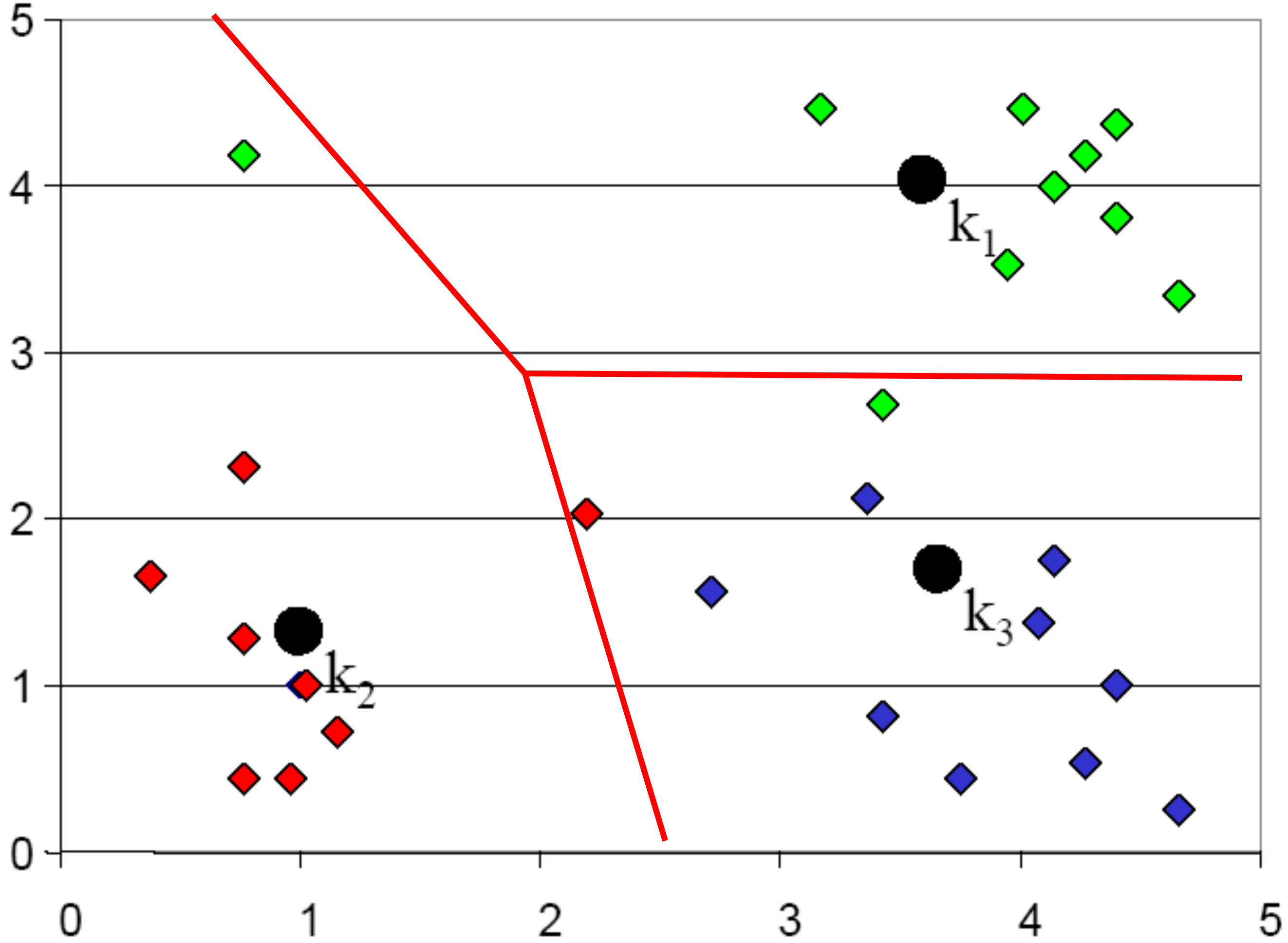


Voronoi diagram

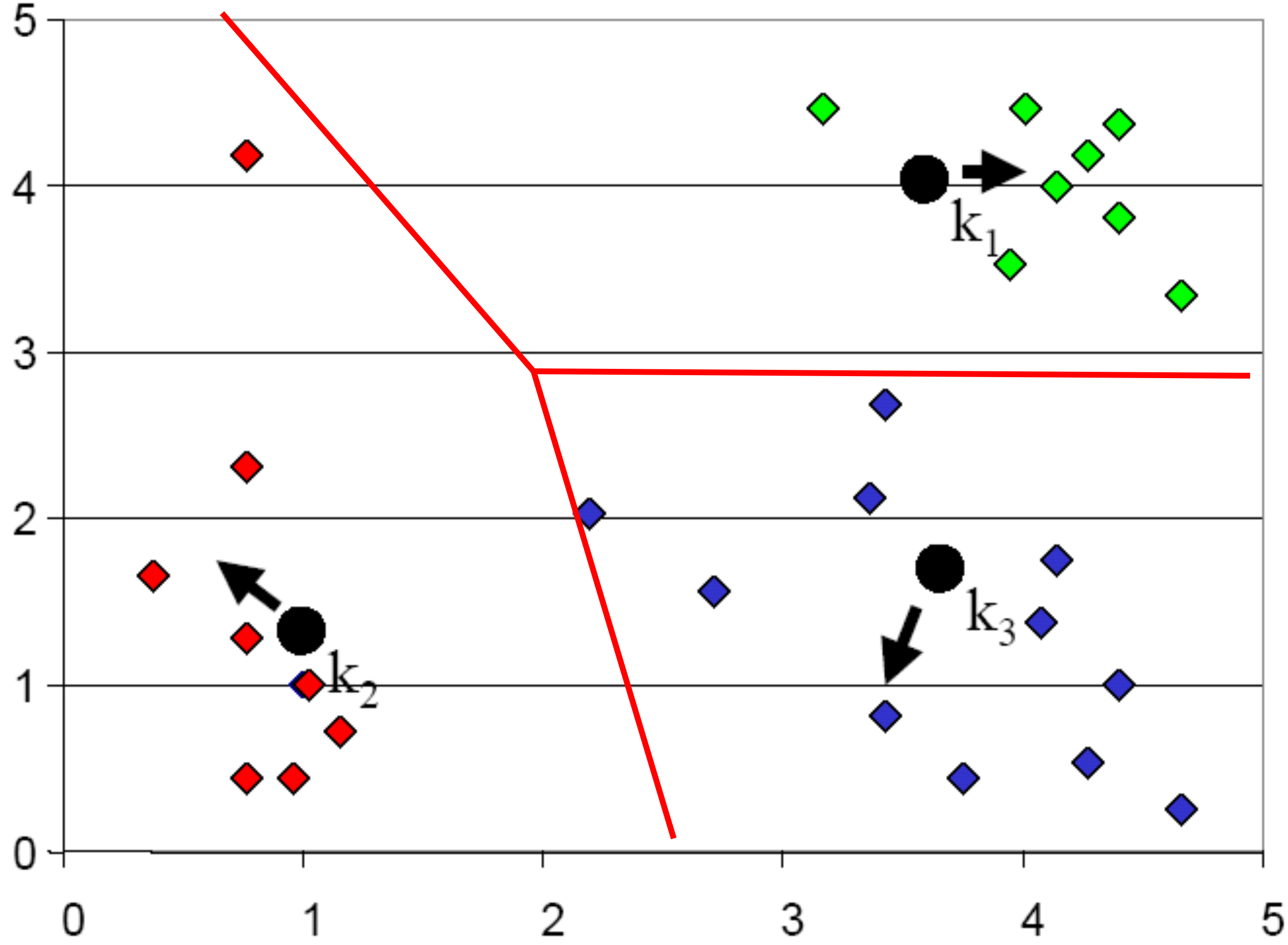
K-means Clustering: Step 2



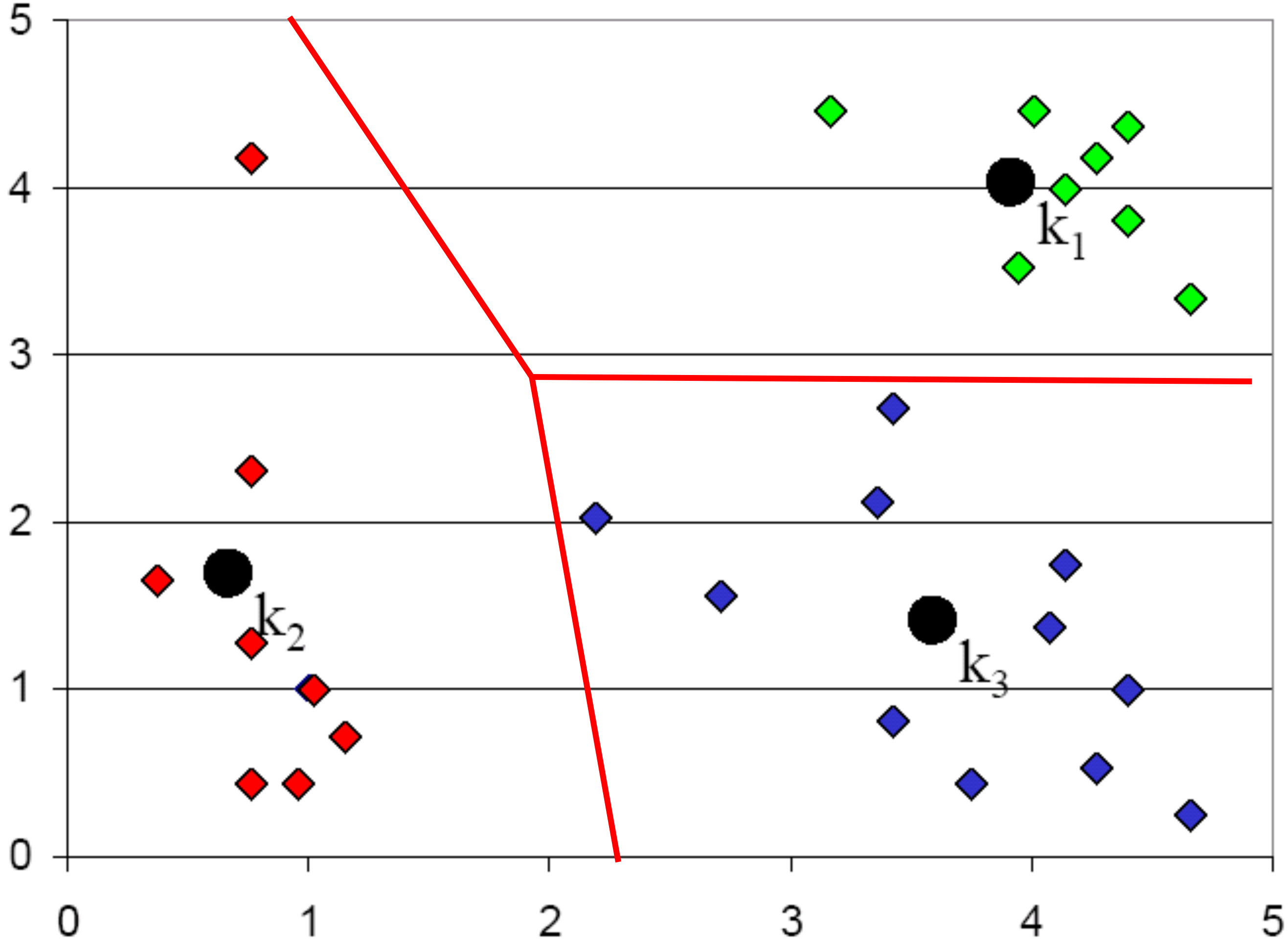
K-means Clustering: Step 3



K-means Clustering: Step 4



K-means Clustering: Step 5



K-means Recap …

- Randomly initialize *k* centers
  - $\mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)}$

K-means Recap ...

- Randomly initialize *k* centers
  - $\mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)}$

Iterate t = 0, 1, 2, ...

- **Classify**: Assign each point $j \in \{1, \ldots m\}$ to nearest center:
  - $C^{(t)}(j) \leftarrow \arg \min_{i=1,\ldots,k} \|\mu_i^{(t)} - x_j\|^2$

K-means Recap ...

- Randomly initialize *k* centers
  - $\mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)}$

Iterate t = 0, 1, 2, …

- **Classify**: Assign each point j∈{1,…m} to nearest center:
  - $C^{(t)}(j) \leftarrow \arg\min_{i=1,\ldots,k} \|\mu_i^{(t)} - x_j\|^2$
- **Recenter**: μ_i becomes centroid of its point**s:**
  - $\mu_i^{(t+1)} \leftarrow \arg\min_{\mu} \sum_{j:C^{(t)}(j)=i} \|\mu - x_j\|^2 \qquad i \in \{1, \ldots, k\}$
  - Equivalent to $\mu_i \leftarrow$ average of its points!

What is K-means optimizing?

- Potential function F(**μ**,C) of centers **μ** and point allocations C:

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

$$= \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Optimal K-means:
  - $\min_{\mu} \min_{C} F(\mu, C)$

➢ Is the K-means objective convex?

K-means algorithm

- Optimize potential function:

$$\min_{\mu}\min_{C} F(\mu, C) = \min_{\mu}\min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- **K-means algorithm:** (coordinate descent on F)

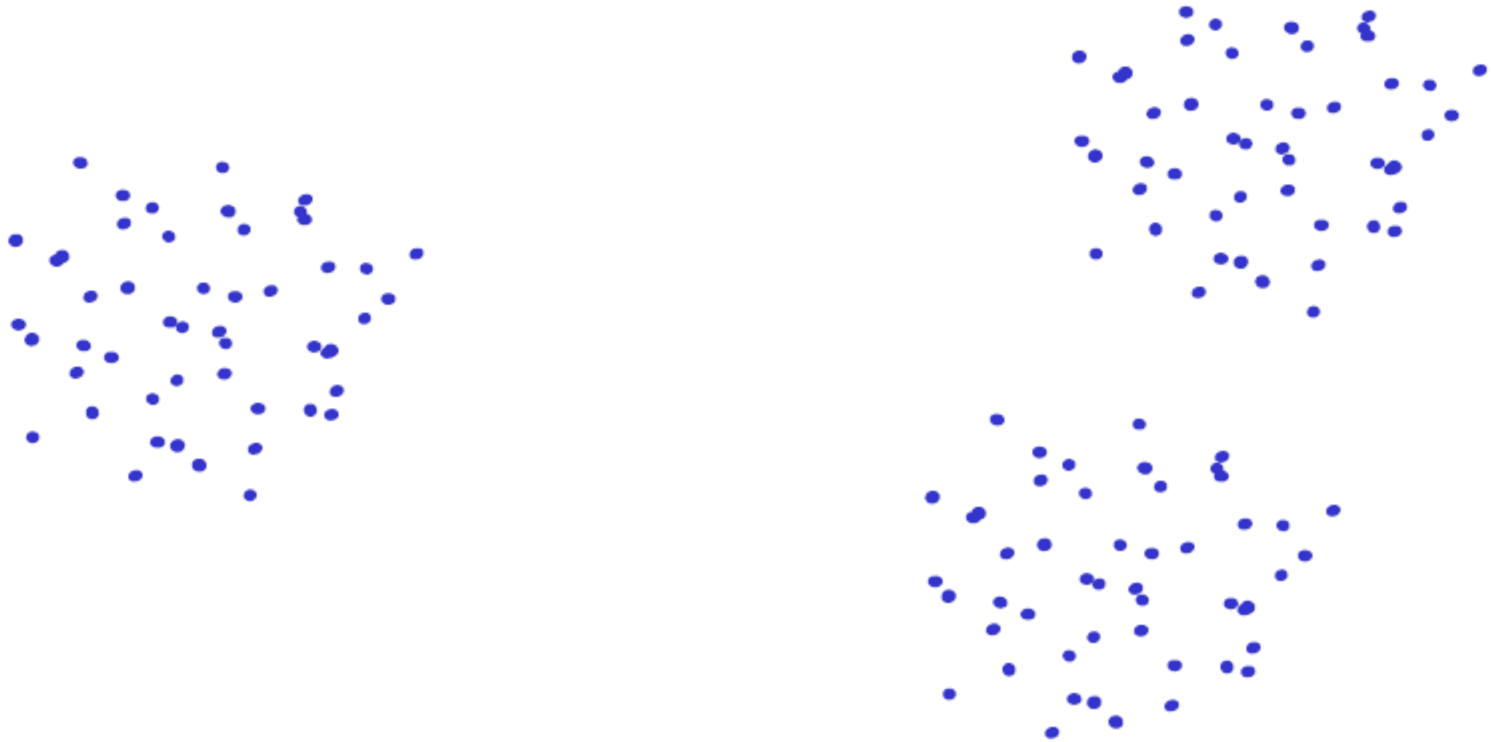
(1) Fix μ, optimize C — **Expected** cluster assignment

(2) Fix C, optimize μ — **Maximum** likelihood for center

Similar to EM/Baum Welch algorithm for learning HMM parameters

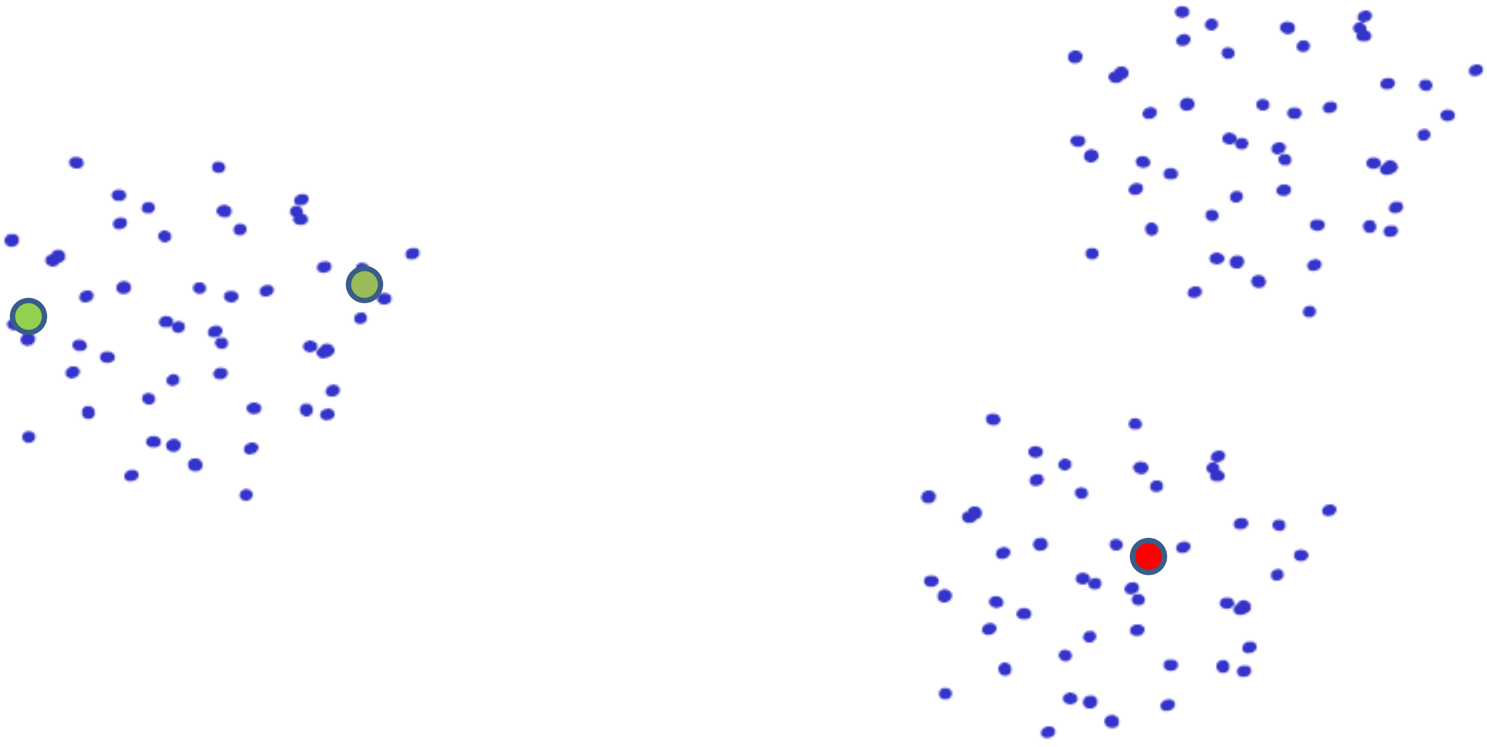
Seed Choice

- Results are quite sensitive to seed selection.



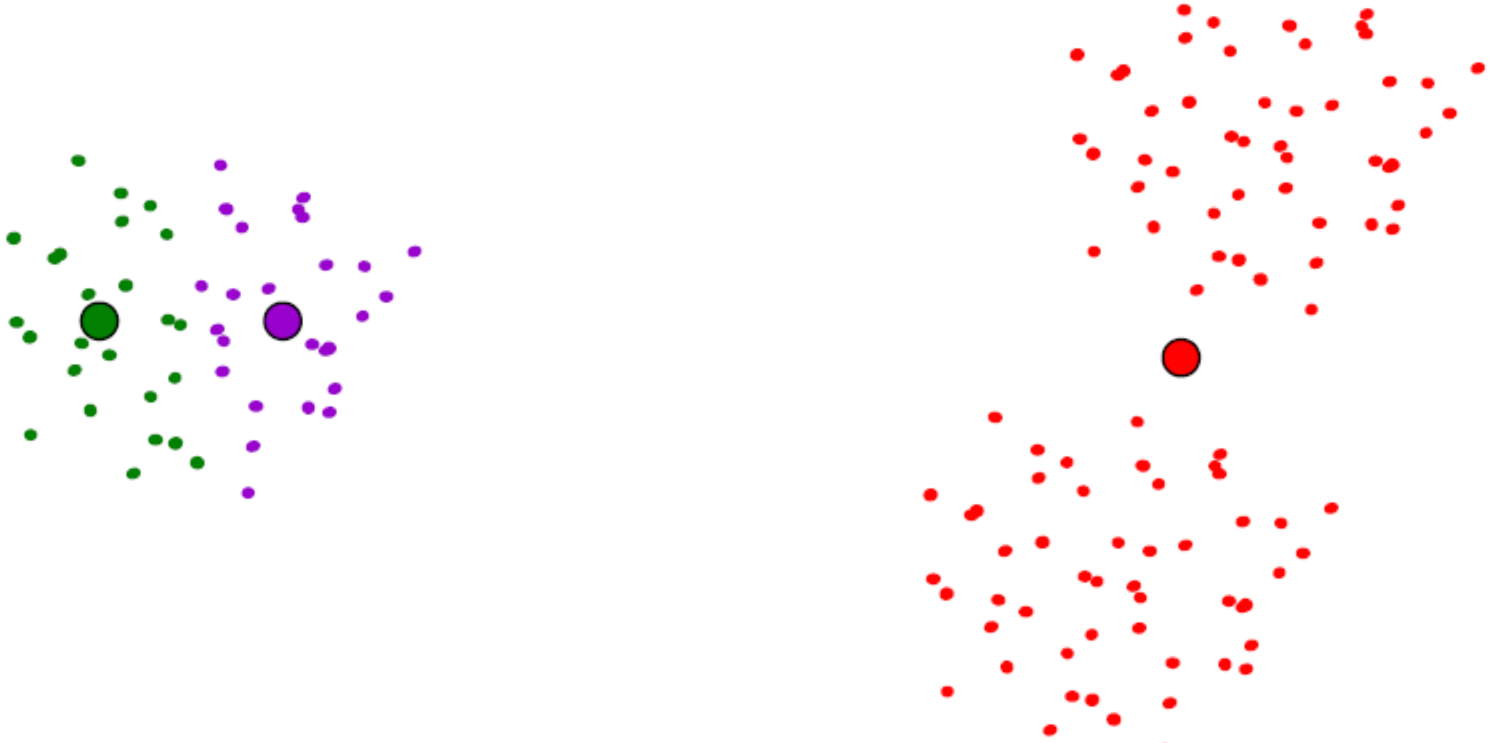
Seed Choice

- Results are quite sensitive to seed selection.



Seed Choice

- Results are quite sensitive to seed selection.



Seed Choice

- Results can vary based on random seed selection.

- Some seeds can result in poor convergence rate, or convergence to sub-optimal clustering.
  - Try out multiple starting points (very important!!!)
  - k-means ++ algorithm of Arthur and Vassilvitskii

    key idea: choose centers that are far apart

      (probability of picking a point as cluster center $\propto$ distance from nearest center picked so far)

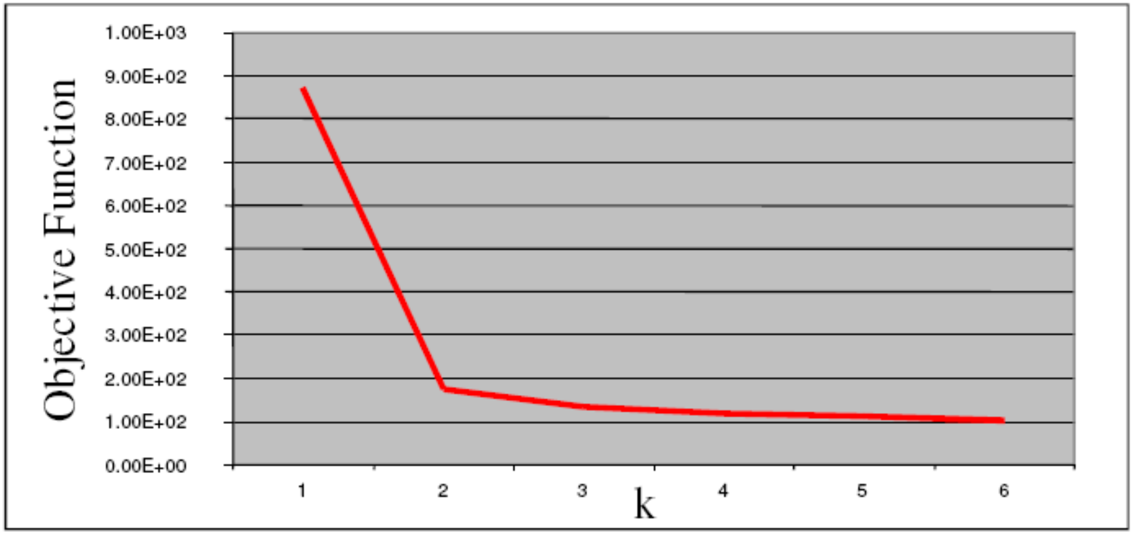
Other Issues

- Number of clusters K
  - Objective function

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

➢ Can you pick K by minimizing the objective over K?

  - Look for "Knee" in objective function



Other Issues

- Sensitive to Outliers
  - use K-medoids

- Shape of clusters
  Assumes isotropic, equal variance, convex clusters

Partitioning Algorithms

- K-means
  - **hard assignment**: each object belongs to only one cluster
- Mixture modeling
  - **soft assignment**: probability that an object belongs to a cluster

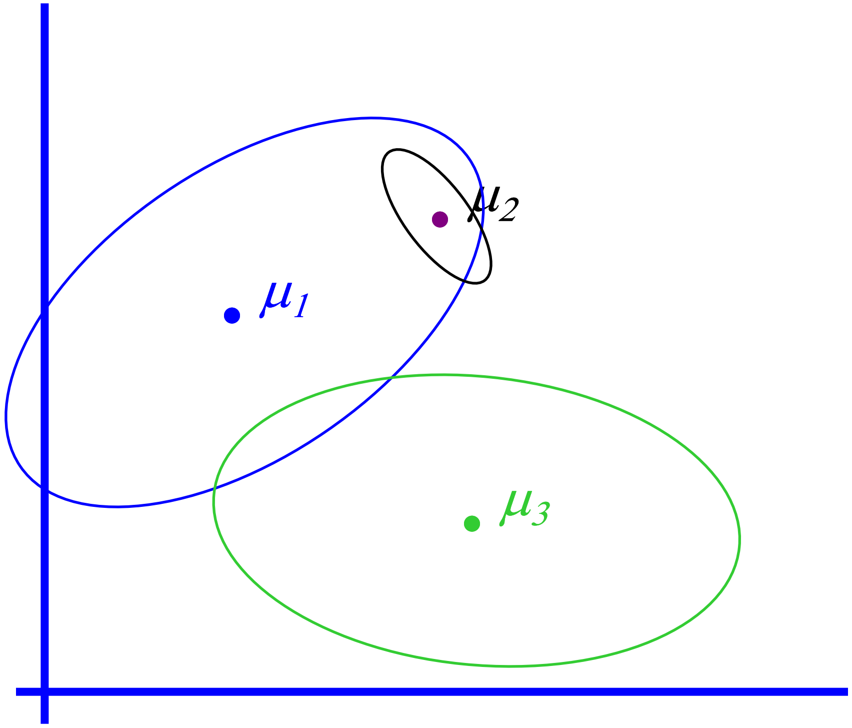
Generative approach

Mixture models

GMM – Gaussian Mixture Model (Multi-modal distribution)

$$p(x|y=i) \sim \mathrm{N}(\mu_i, \Sigma_i)$$

$$p(x) = \sum_i \underbrace{p(x|y=i)}_{\text{Mixture component}} \underbrace{P(y=i)}_{\text{Mixture proportion}}$$



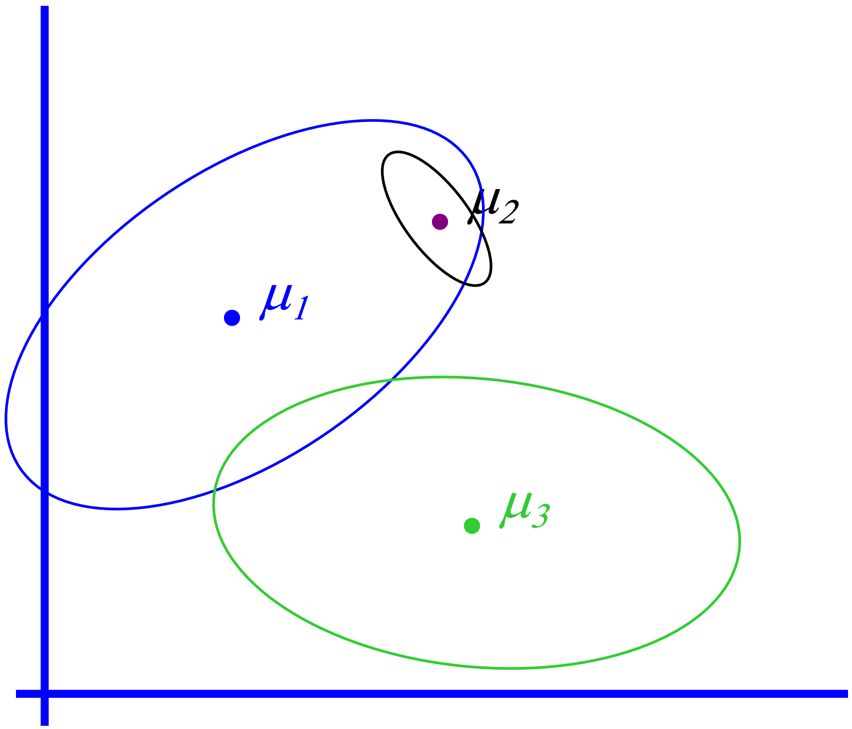
Mixture models

GMM – Gaussian Mixture Model (Multi-modal distribution)

- There are k components
- Component *i* has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i

Each data point is generated according to the following recipe:

1) Pick a component at random: Choose component i with probability *P(y=i)*
2) Datapoint x $\sim \mathrm{N}(\mu_i, \Sigma_i)$



Learning GMMs via EM algorithm

Iterate. On iteration t let our estimates be

$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_k^{(t)}, p_1^{(t)}, p_2^{(t)} \ldots p_k^{(t)} \}$

$p_i^{(t)}$ is shorthand for estimate of $P(y=i)$ on t'th iteration

E-step

Compute "expected" classes of all datapoints for each class

$$P\left(y=i \middle| x_j, \lambda_t\right) \propto p_i^{(t)} \, p\left(x_j \middle| \mu_i^{(t)}, \Sigma_i^{(t)}\right)$$

Just evaluate a Gaussian at x_j

M-step

Compute MLEs given our data's class membership distributions (weights)

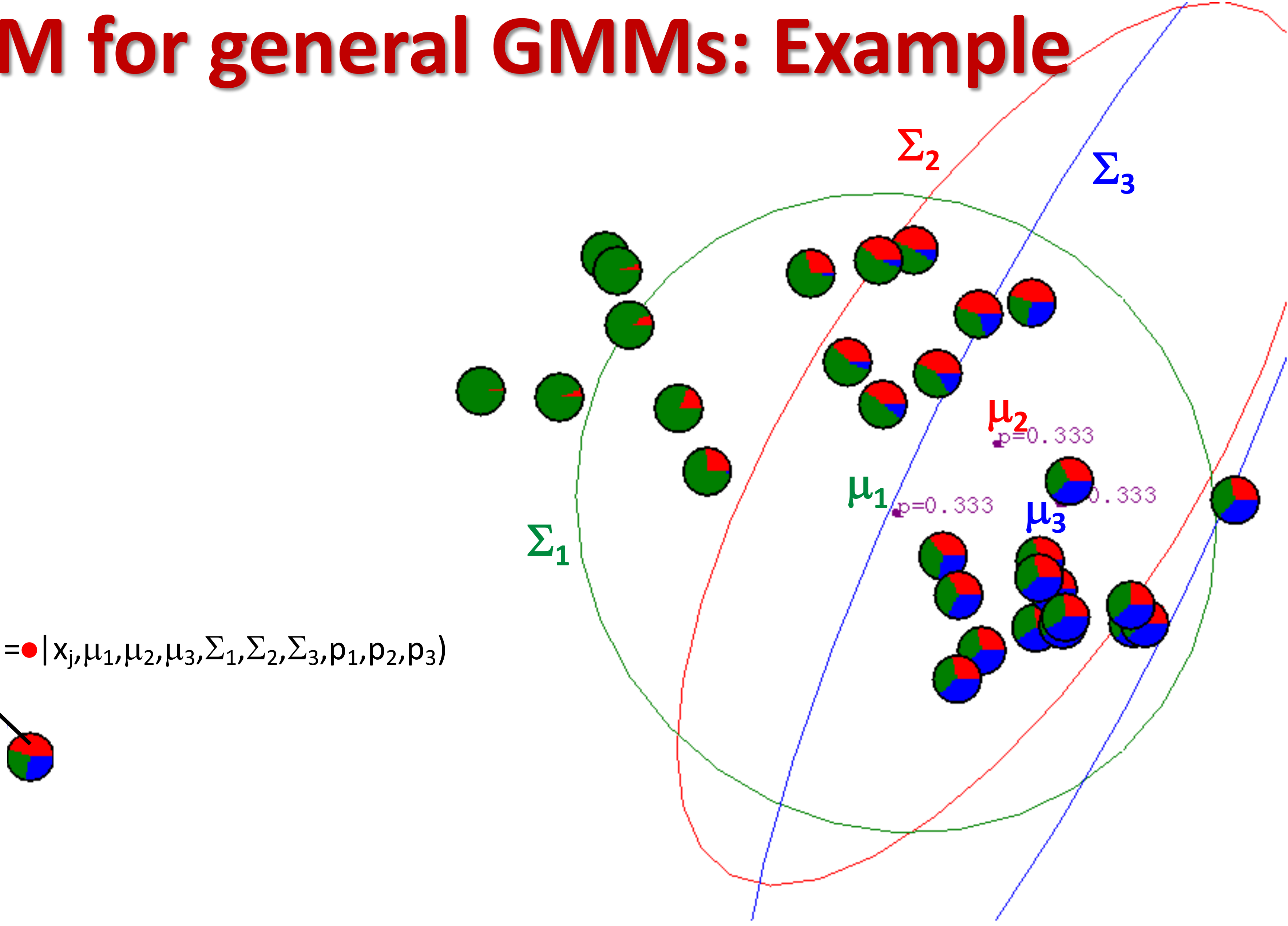
$$\mu_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x_j, \lambda_t\right) x_j}{\sum_j P\left(y=i \middle| x_j, \lambda_t\right)} \qquad \Sigma_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x_j, \lambda_t\right)\left(x_j - \mu_i^{(t+1)}\right)\left(x_j - \mu_i^{(t+1)}\right)^T}{\sum_j P\left(y=i \middle| x_j, \lambda_t\right)}$$

$$p_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x_j, \lambda_t\right)}{m}$$

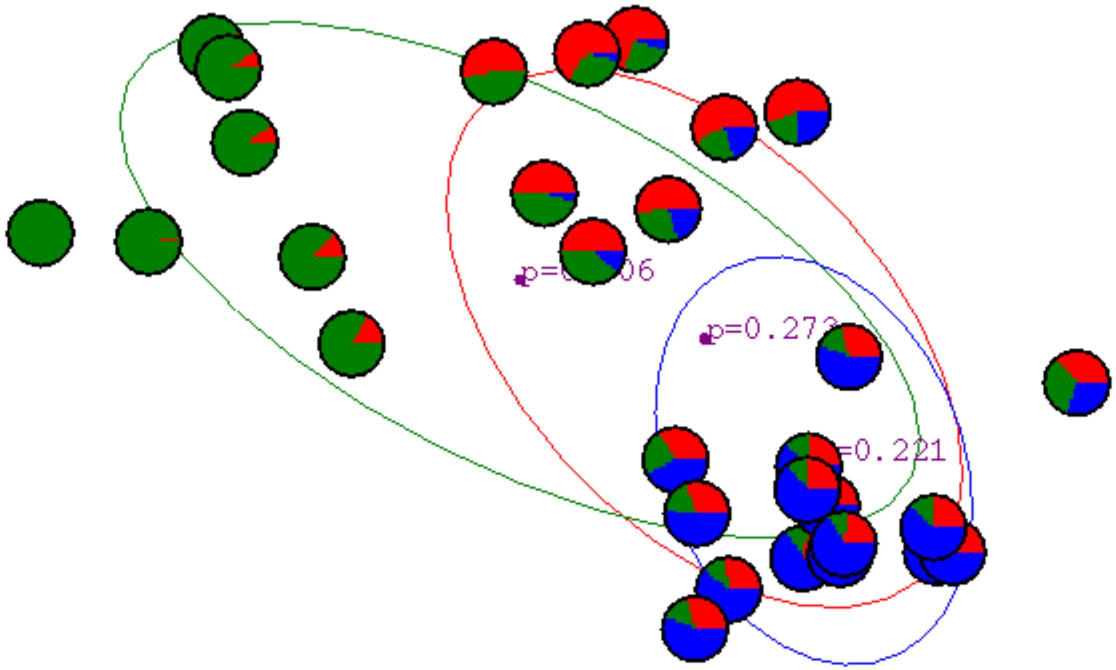
m = #data points

EM for general GMMs: Example

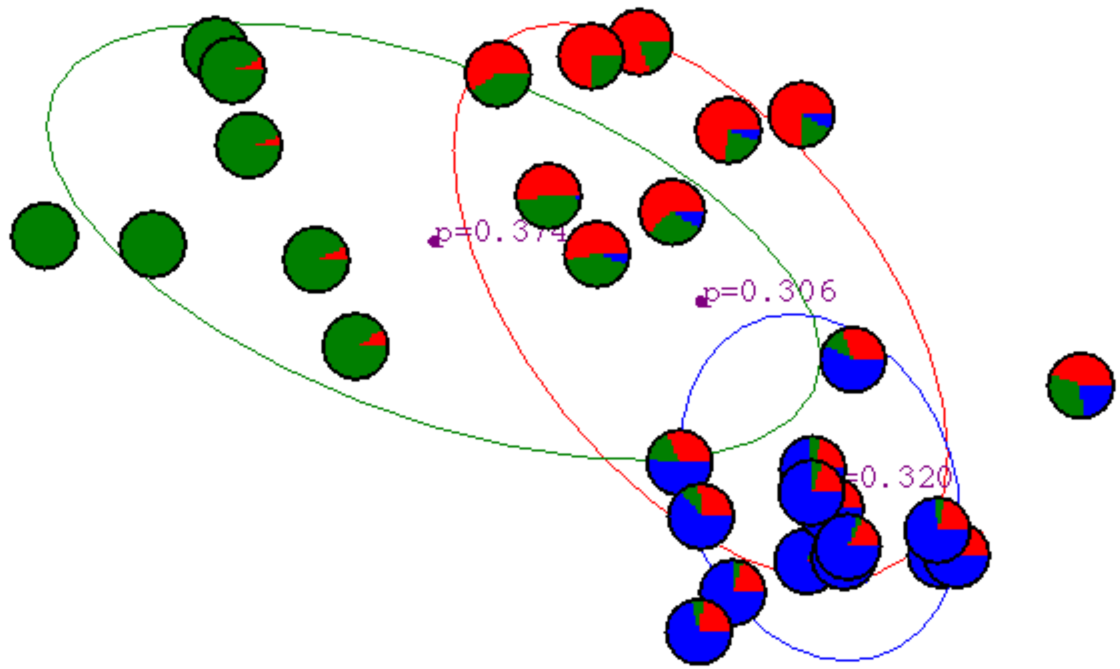
$P(y = \bullet \mid x_j, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, p_1, p_2, p_3)$



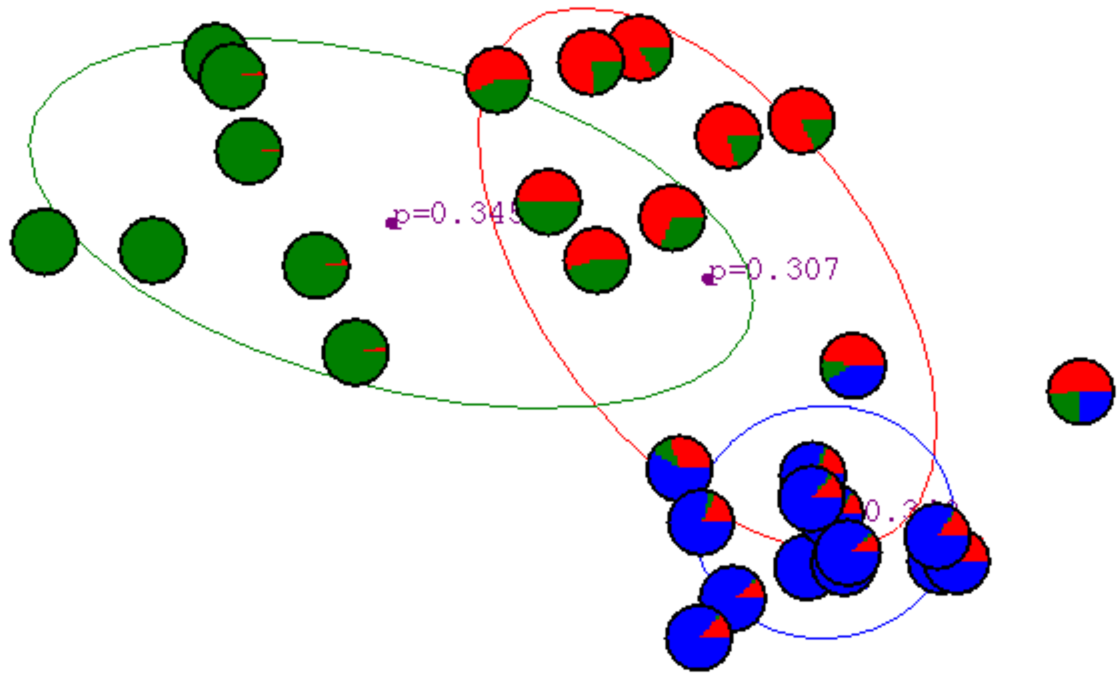
After 1st iteration



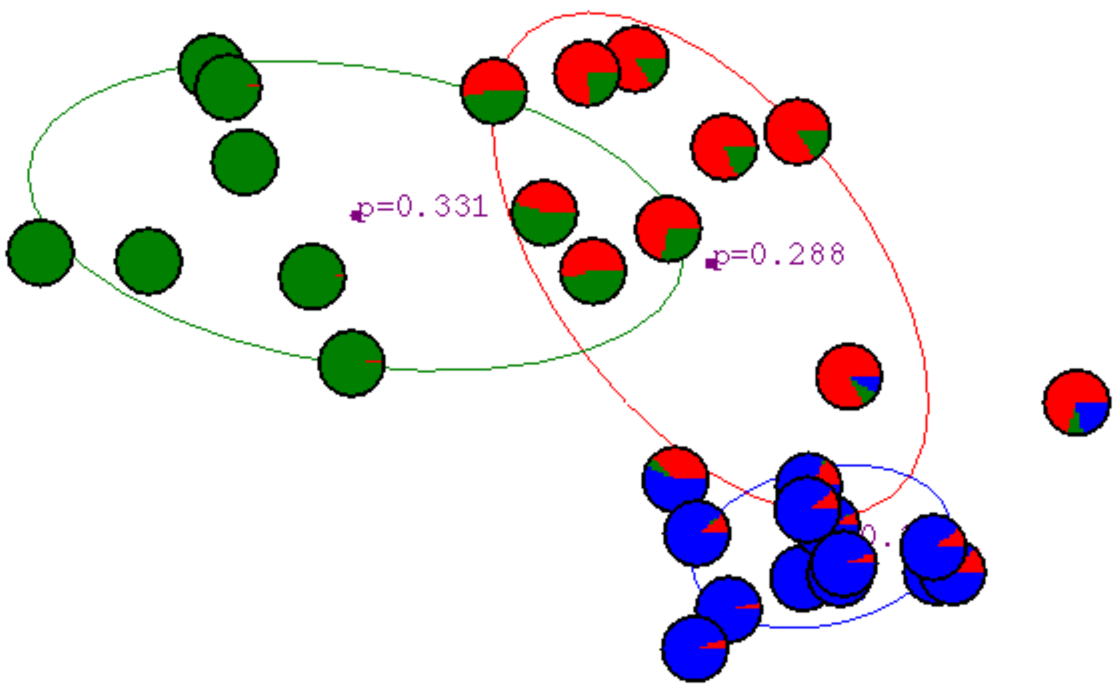
After 2nd iteration



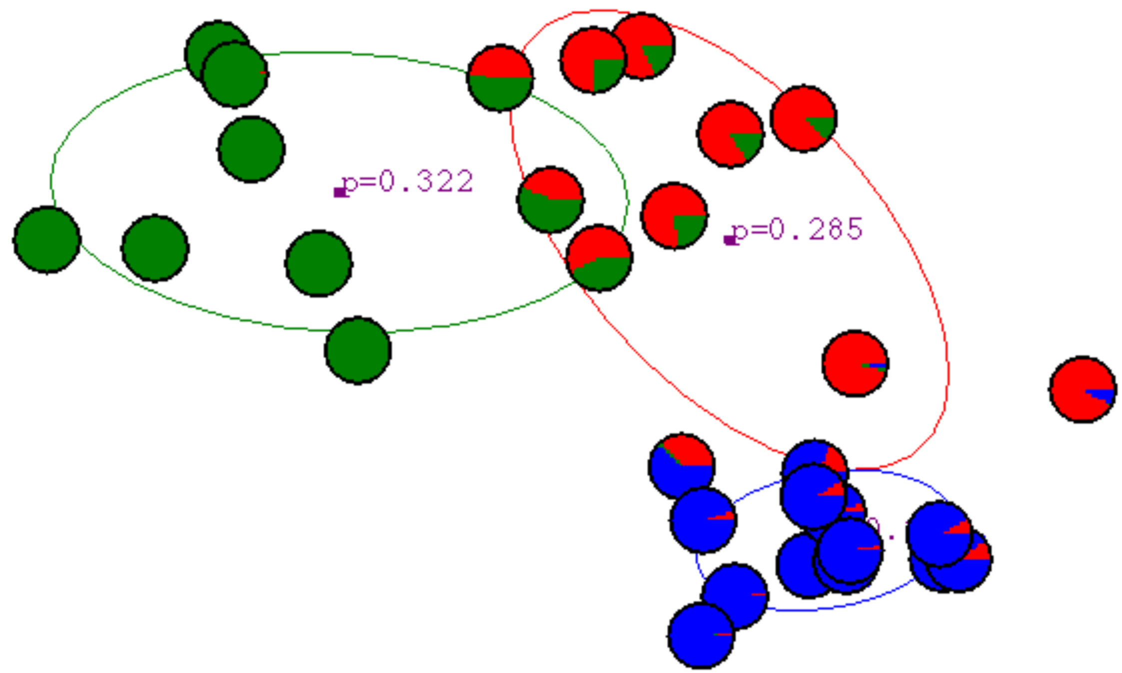
After 3rd iteration



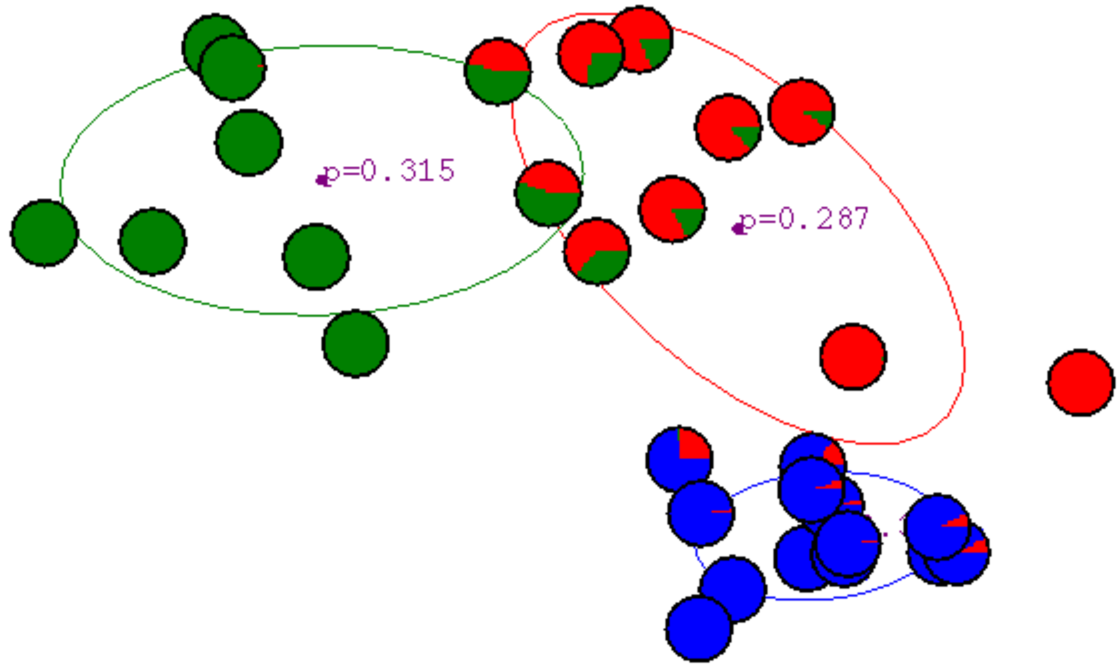
After 4th iteration



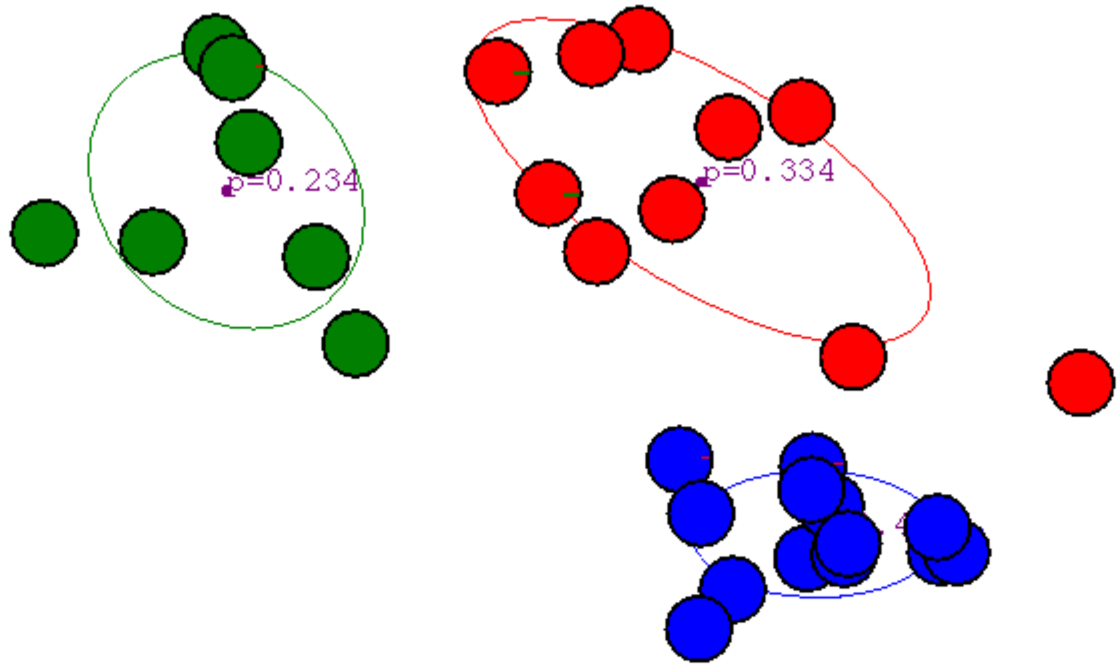
After 5th iteration



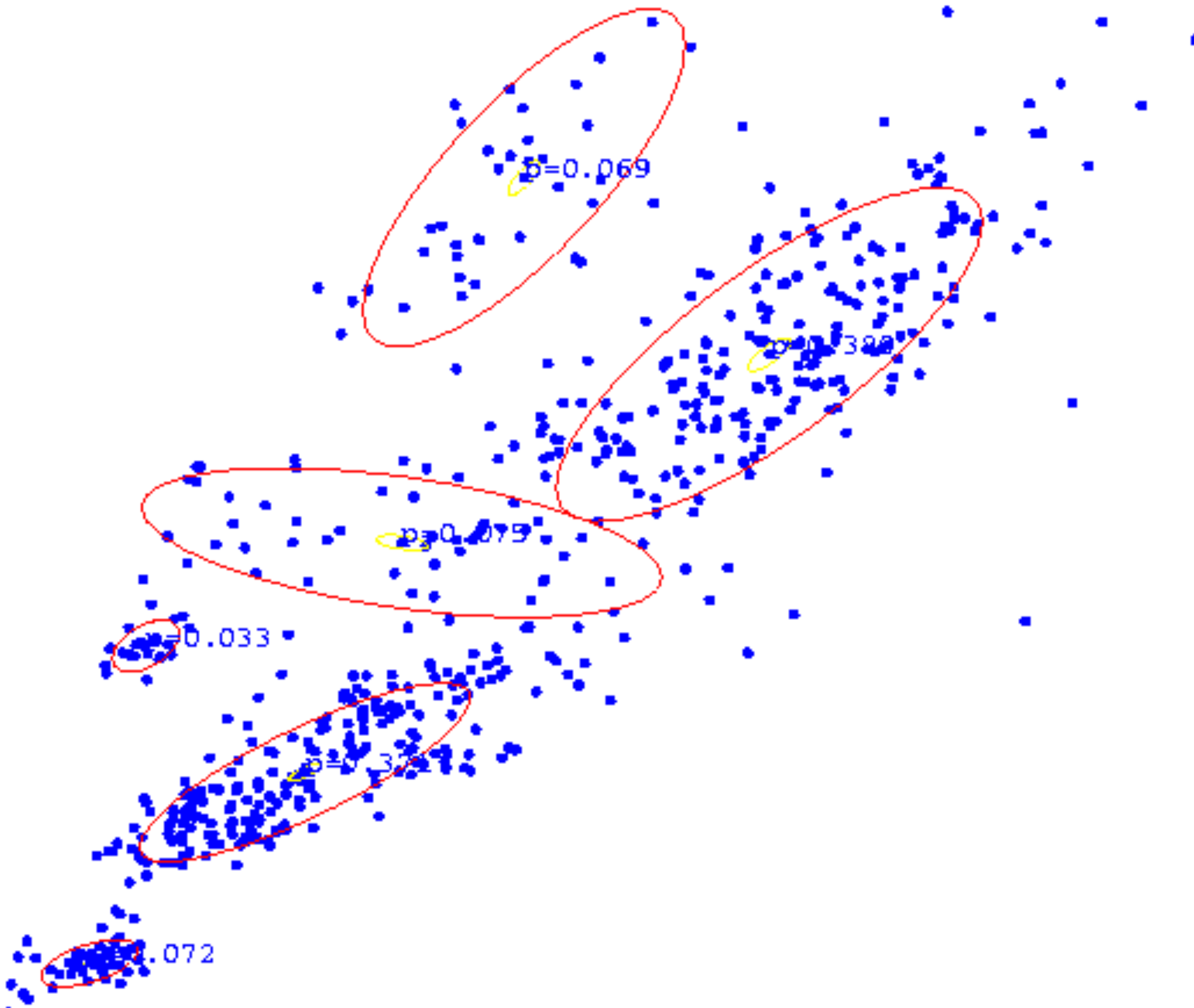
After 6th iteration



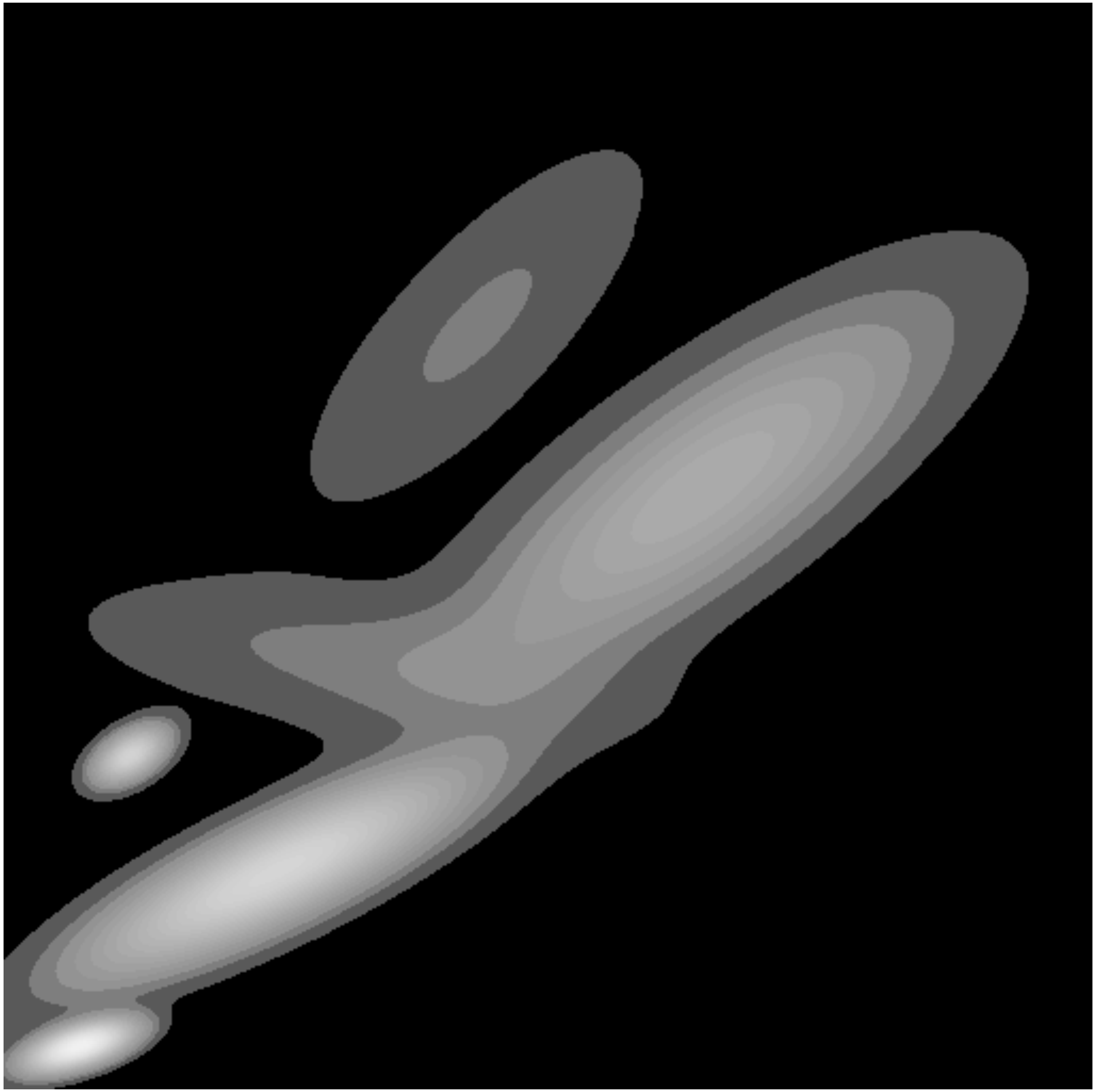
After 20th iteration



GMM clustering of assay data

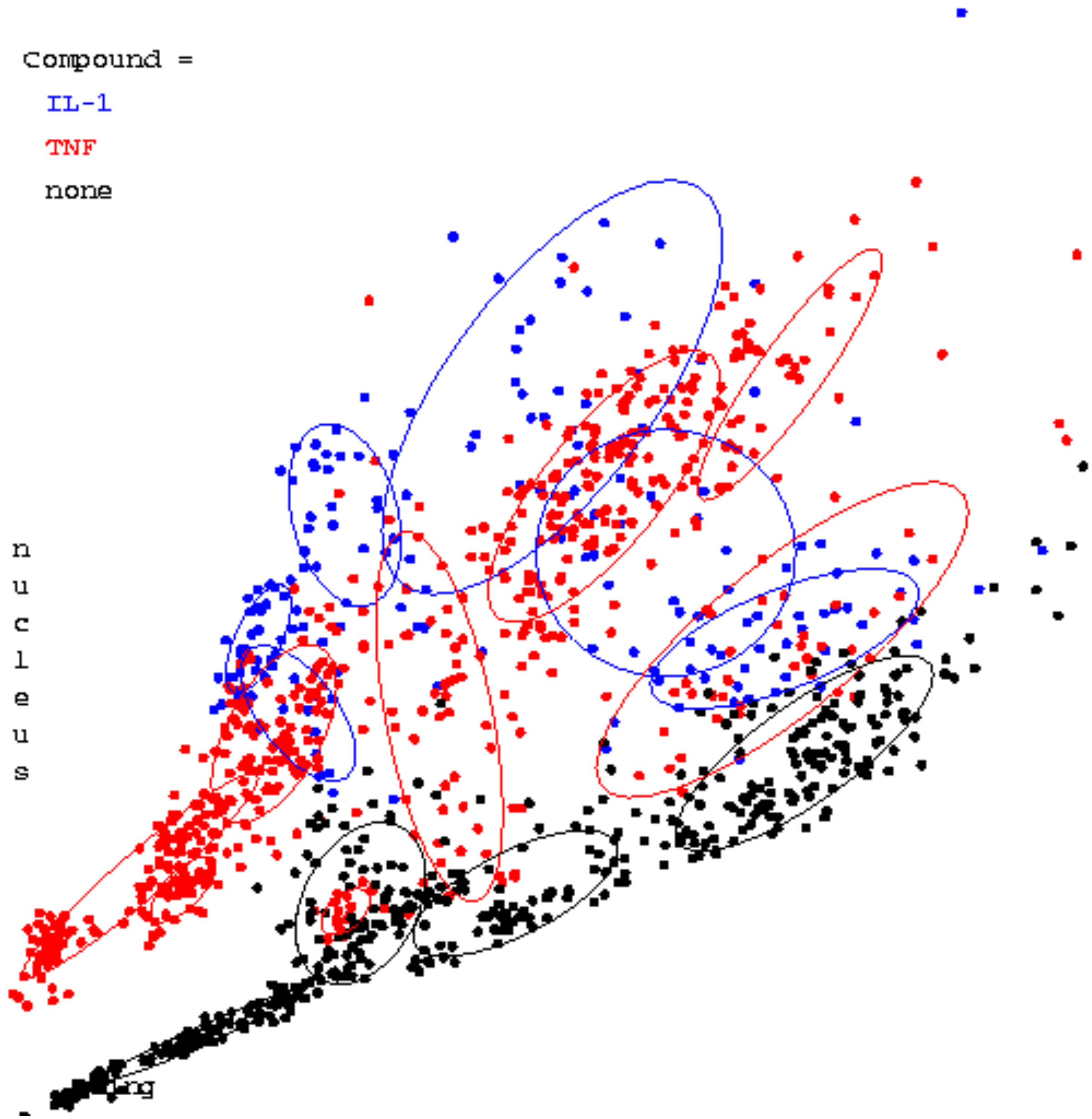


Resulting Density Estimator



Three classes of assay

(each learned with it's own mixture model)



Resulting Bayes Classifier