Clustering contd…

Aarti Singh

Machine Learning 10-701
Apr 26, 2023

Some slides courtesy of Eric Xing, Carlos Guestrin

K-means Recap ...

given

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

K-means algorithm

- Optimize potential function:

$$\sum_{j=1}^{m} ||\mu_i - x_j||^2$$

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

$\mu_1 \ldots \mu_K$

- **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

latent states $\equiv$ latent cluster assignment

Computational Complexity

- At each iteration,
  - Computing distance between each of the n objects and the K cluster centers is O(Kn).
  - Computing cluster centers: Each object gets added once to some cluster: O(n).

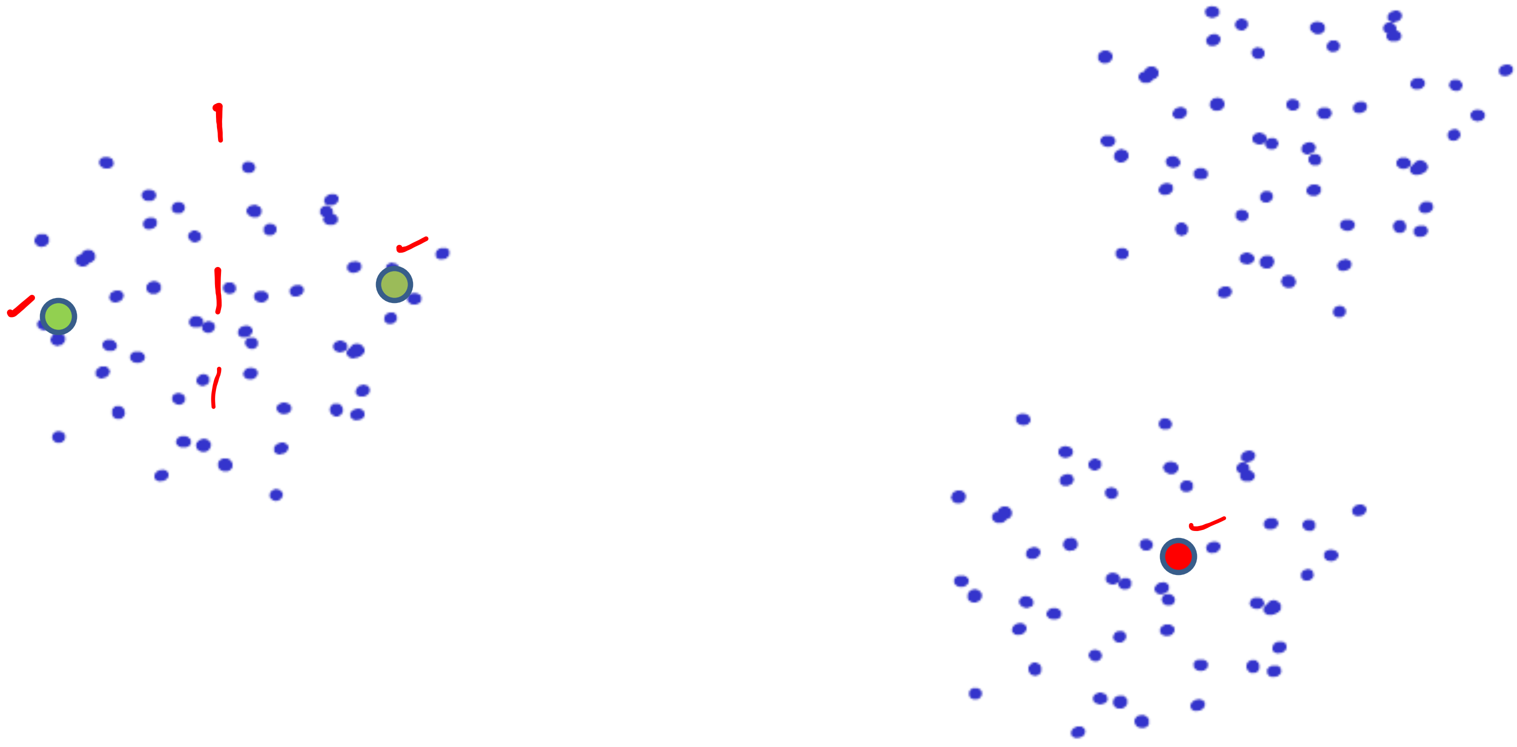
- Assume these two steps are each done once for l iterations: O(lKn).

Seed Choice

- Results are quite sensitive to seed selection.

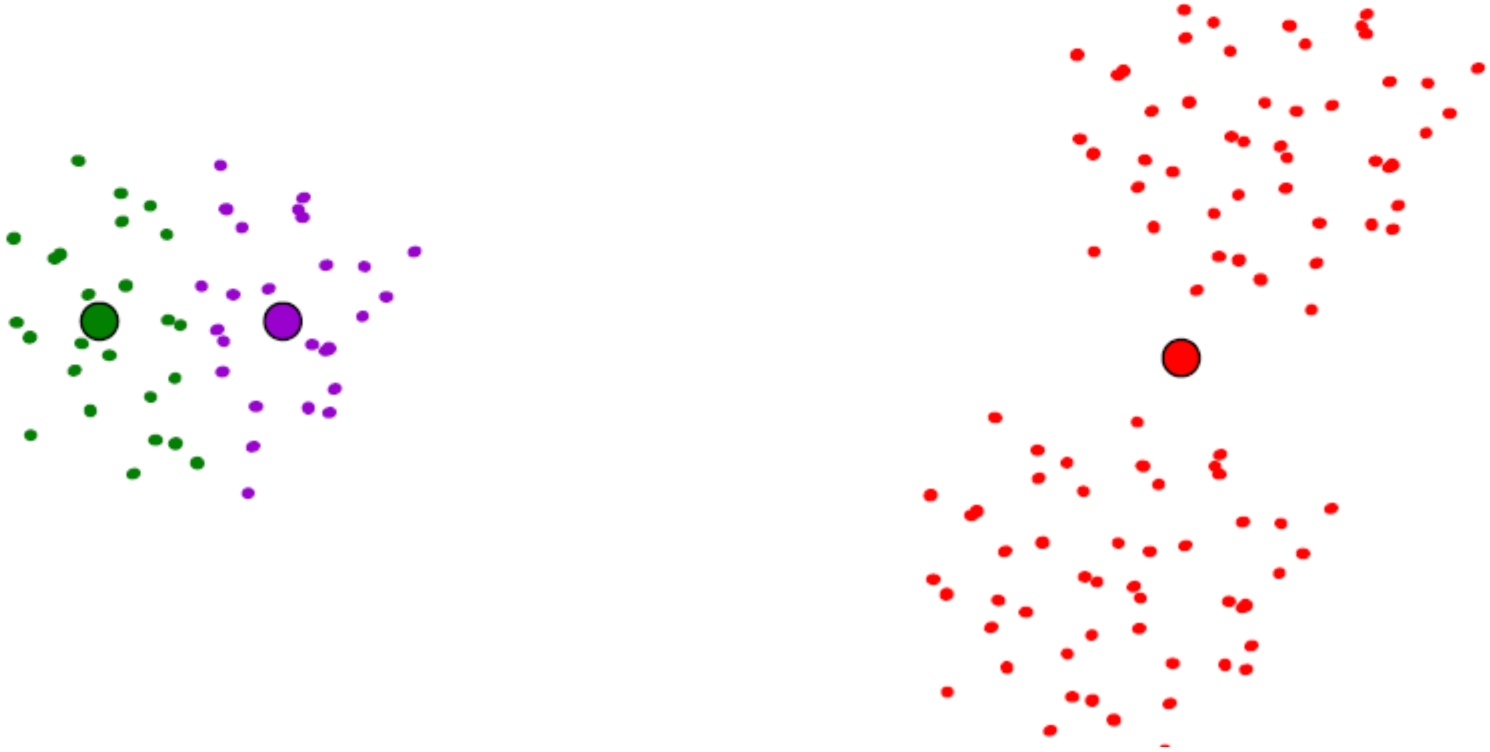
Seed Choice

- 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

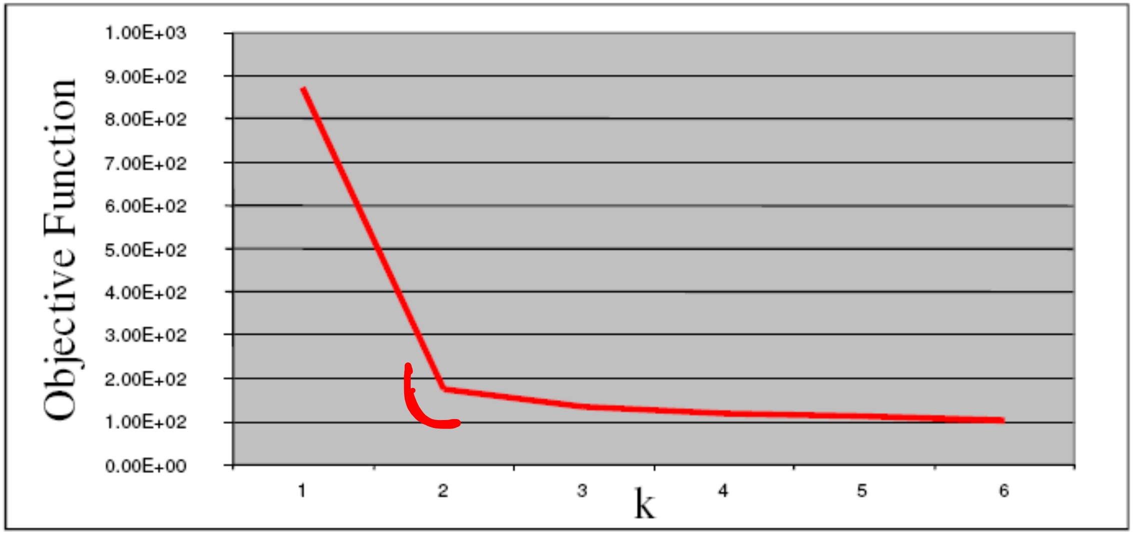
- Number of clusters K
  - Objective function

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

? $\min_K$ $\min_{\mu, C}$

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

  - Look for "Knee" in objective function



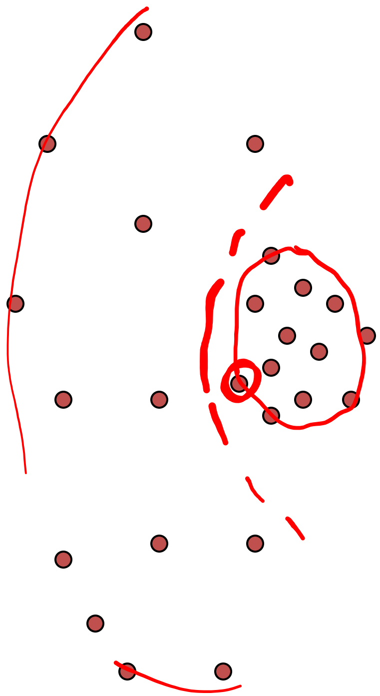
Other Issues

- Sensitive to Outliers
  - use K-medoids

- Shape of clusters

  Assumes isotropic, equal variance, convex clusters

K-means limitations



- Clusters may overlap "soft" assignment
- Some clusters may be "wider" than others
- Clusters may not be linearly separable

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

$p(x)$

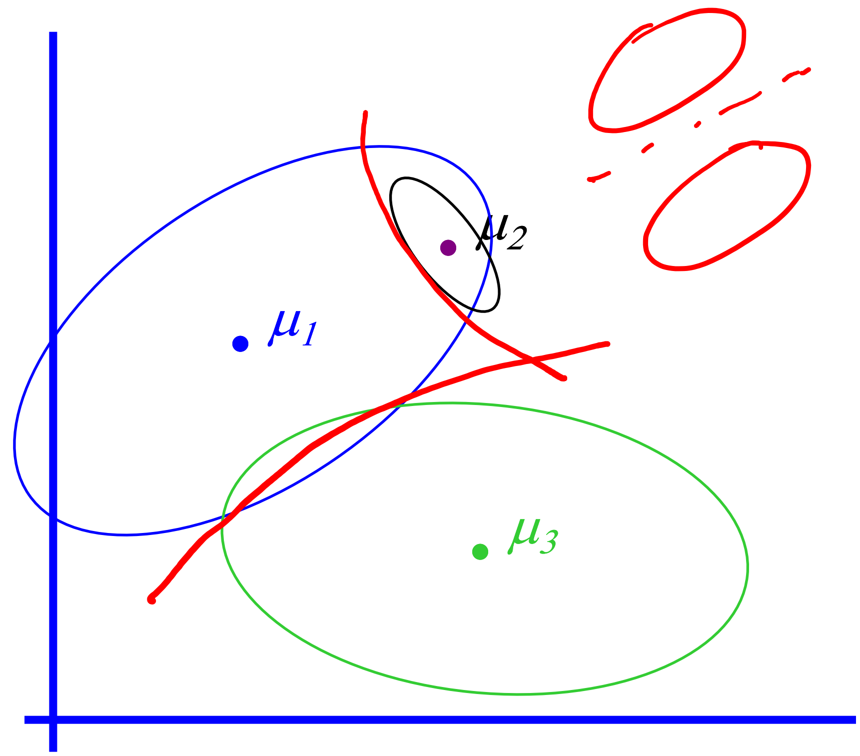
Generative approach

Mixture models

GMM – Gaussian Mixture Model (Multi-modal distribution)

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

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

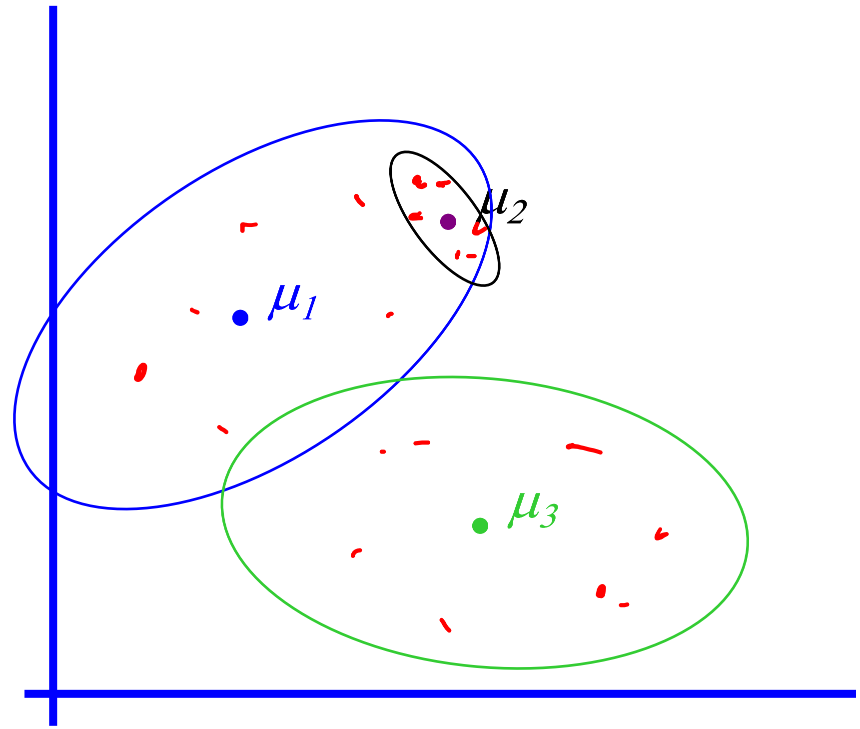


$p(y=i|x) \propto p(y=i)\, p(x|y=i)$

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 $P(y=i)$ $= p_i$
2) Datapoint $x \sim \mathrm{N}(\mu_i, \Sigma_i)$

Learning GMMs via EM algorithm

Iterate. On iteration t let our estimates be $p(y=i) \equiv p_i$

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

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

E-step

Compute "expected" classes of all datapoints for each class

$$\underbrace{\mathrm{P}\left(y = i \mid x_j, \lambda_t\right)}_{w_{ji}} \propto p_i^{(t)} \mathrm{p}\left(x_j \mid \mu_i^{(t)}, \Sigma_i^{(t)}\right)$$

Just evaluate a Gaussian at x_j

M-step

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

$$\mu_i^{(t+1)} = \frac{\sum_j \overbrace{\mathrm{P}\left(y = i \mid x_j, \lambda_t\right)}^{w_{ji}} x_j}{\sum_j \mathrm{P}\left(y = i \mid x_j, \lambda_t\right)}$$

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

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

m = #data points

EM for general GMMs: Example

$k=3$

μ_1, μ_2, μ_3

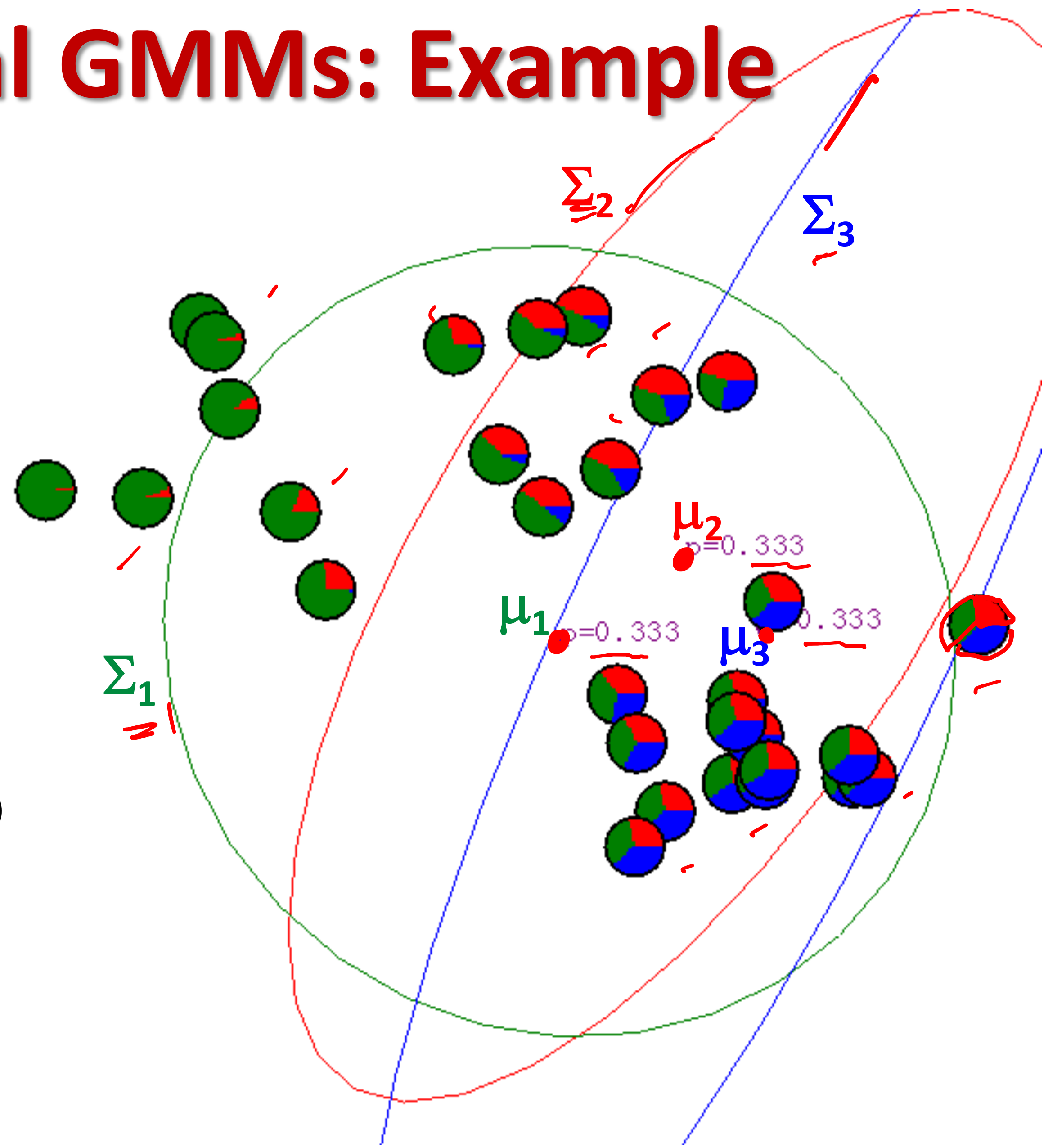
$\Sigma_1, \Sigma_2, \Sigma_3$

$p_1, p_2, p_3 = \frac{1}{3}$

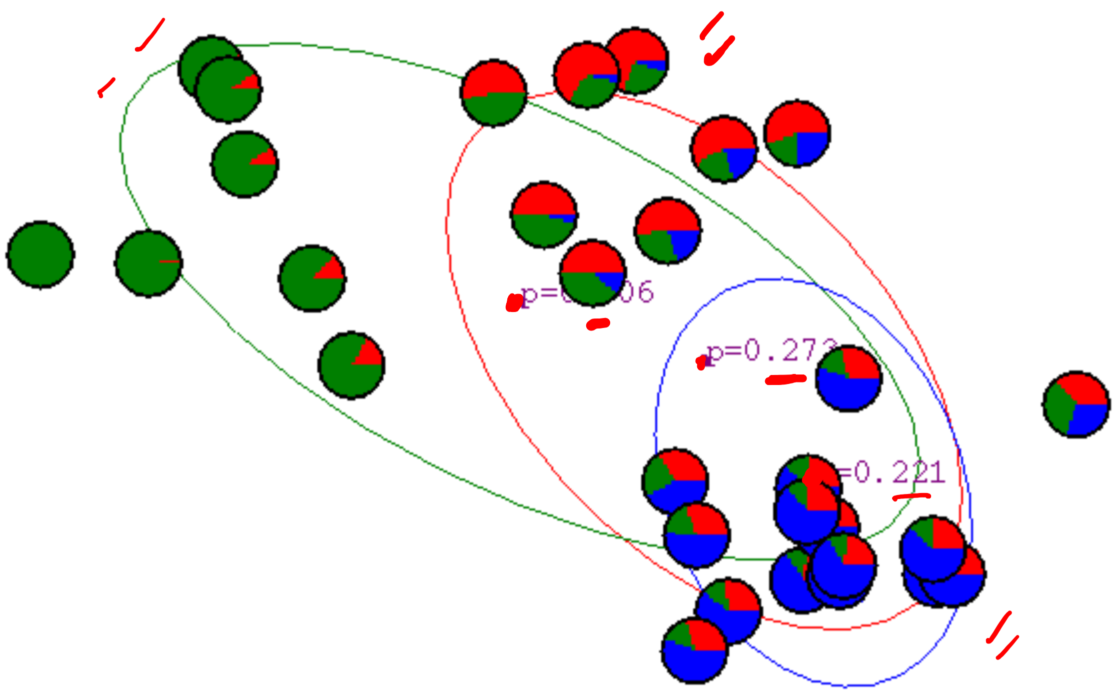
$P(y = \bullet \mid x_j, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, p_1, p_2, p_3)$

$\rightarrow W_{j\,\text{red}}$

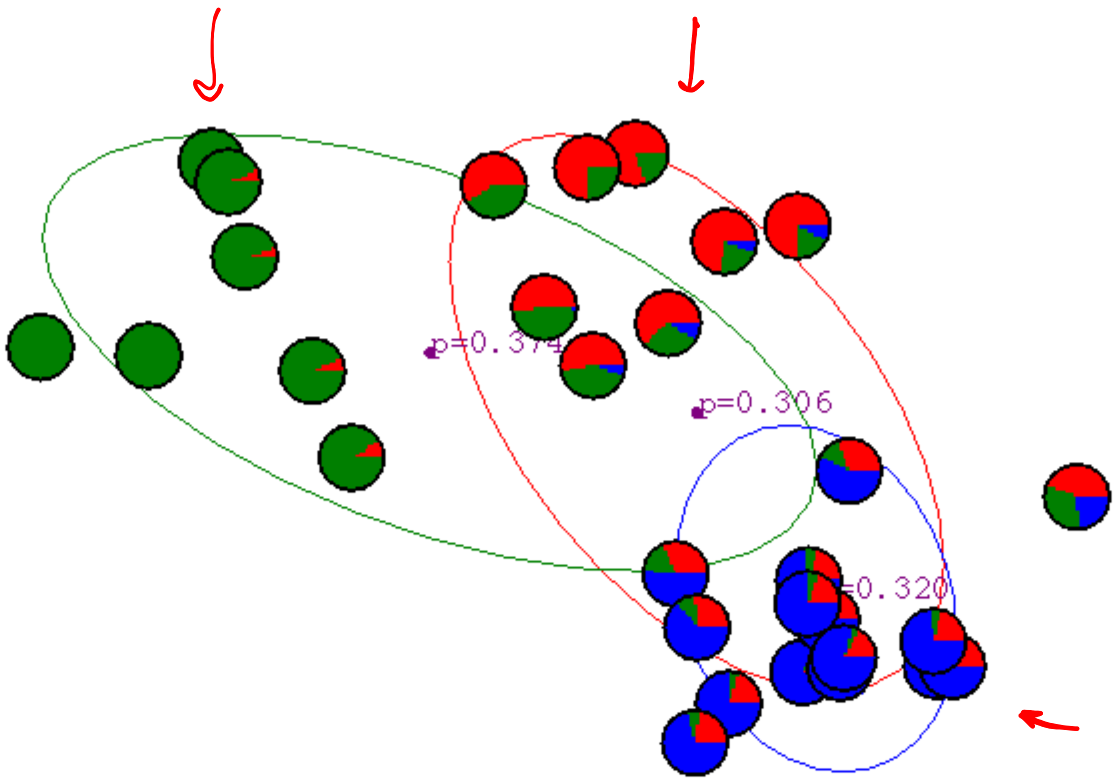
datapoint



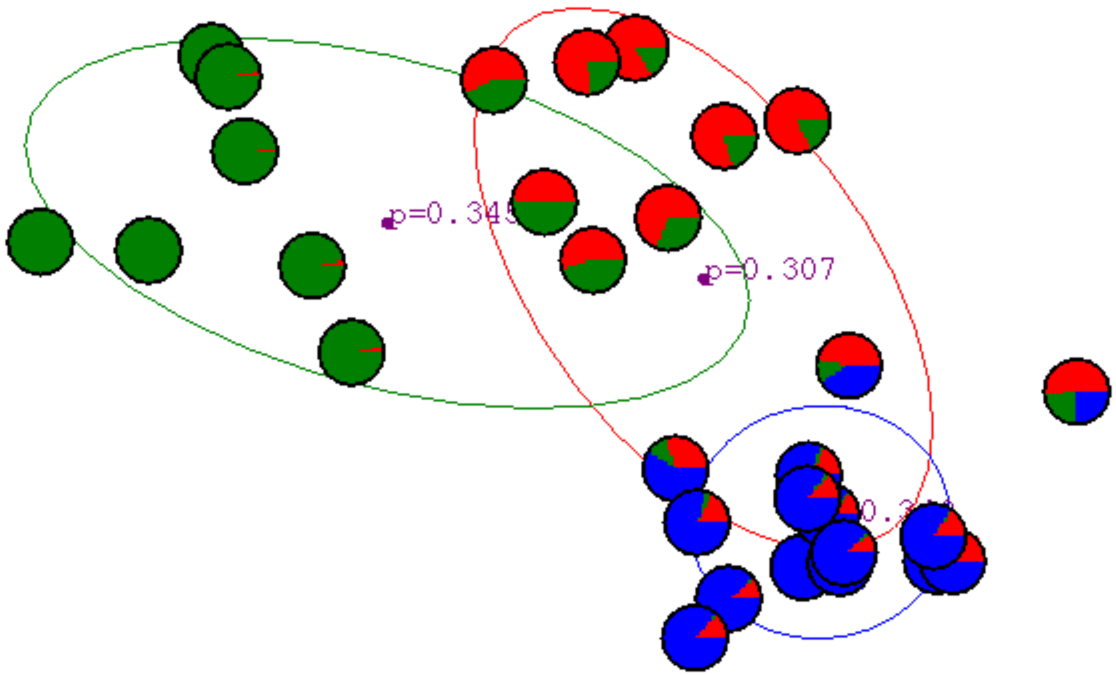
After 1st iteration



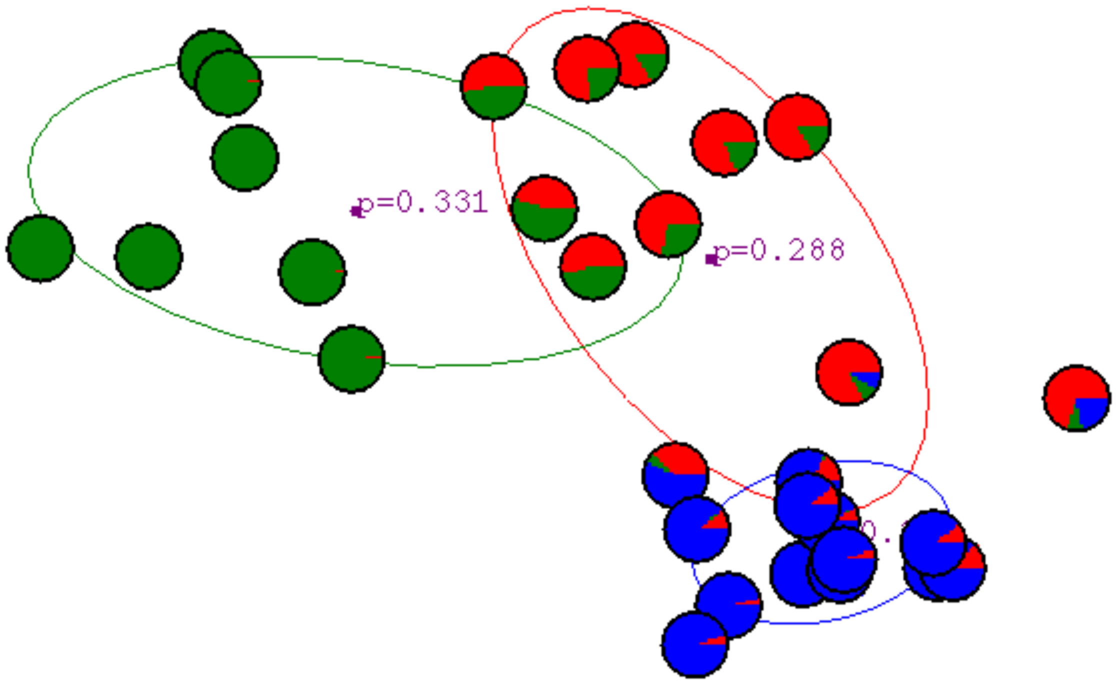
After 2nd iteration



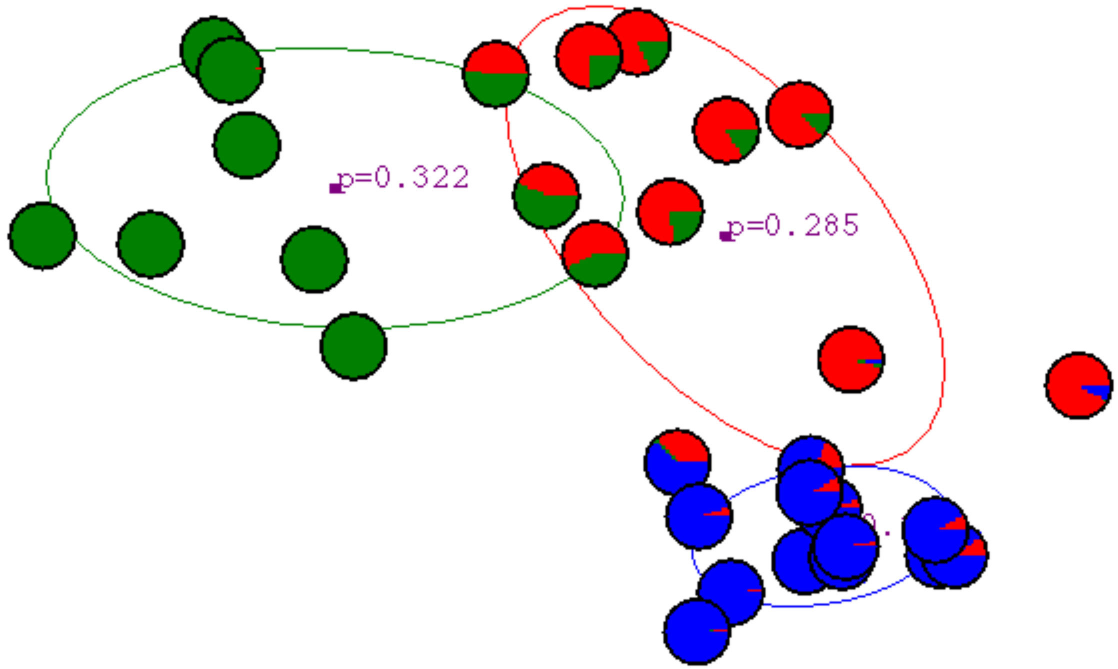
After 3rd iteration



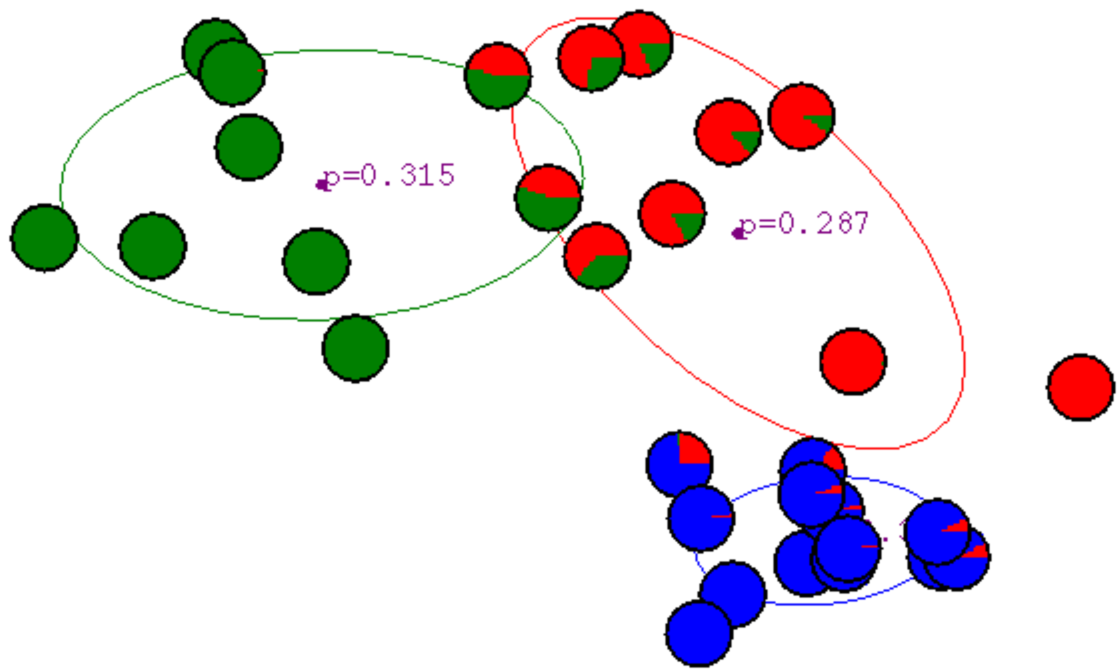
After 4th iteration



After 5th iteration



After 6^{th} iteration

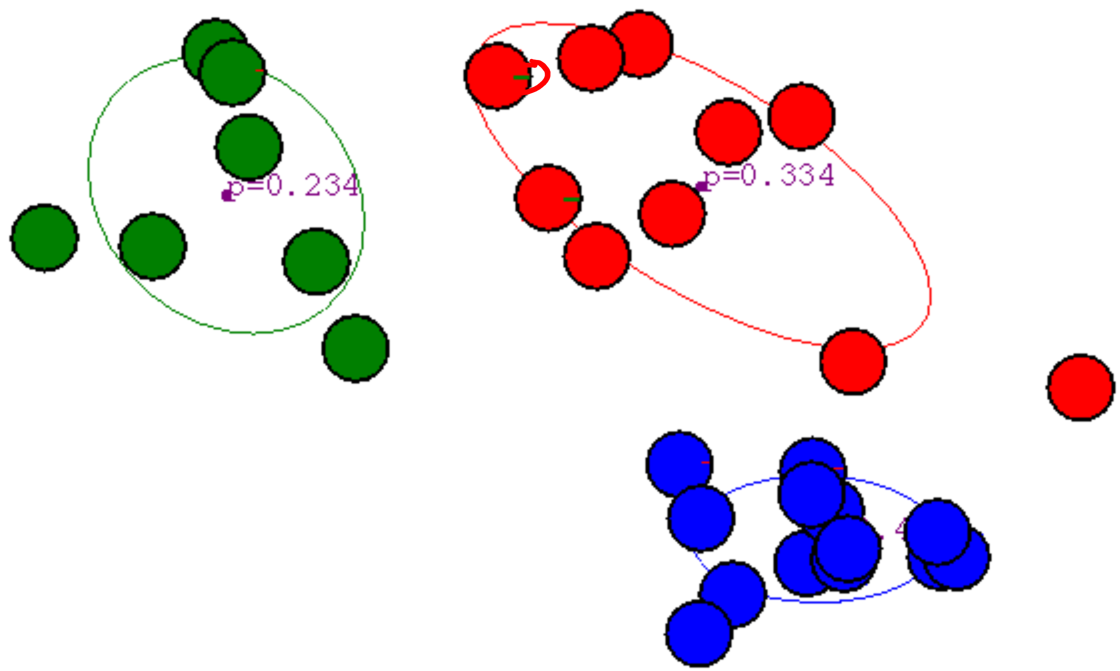


After 20th iteration

$p(x)$

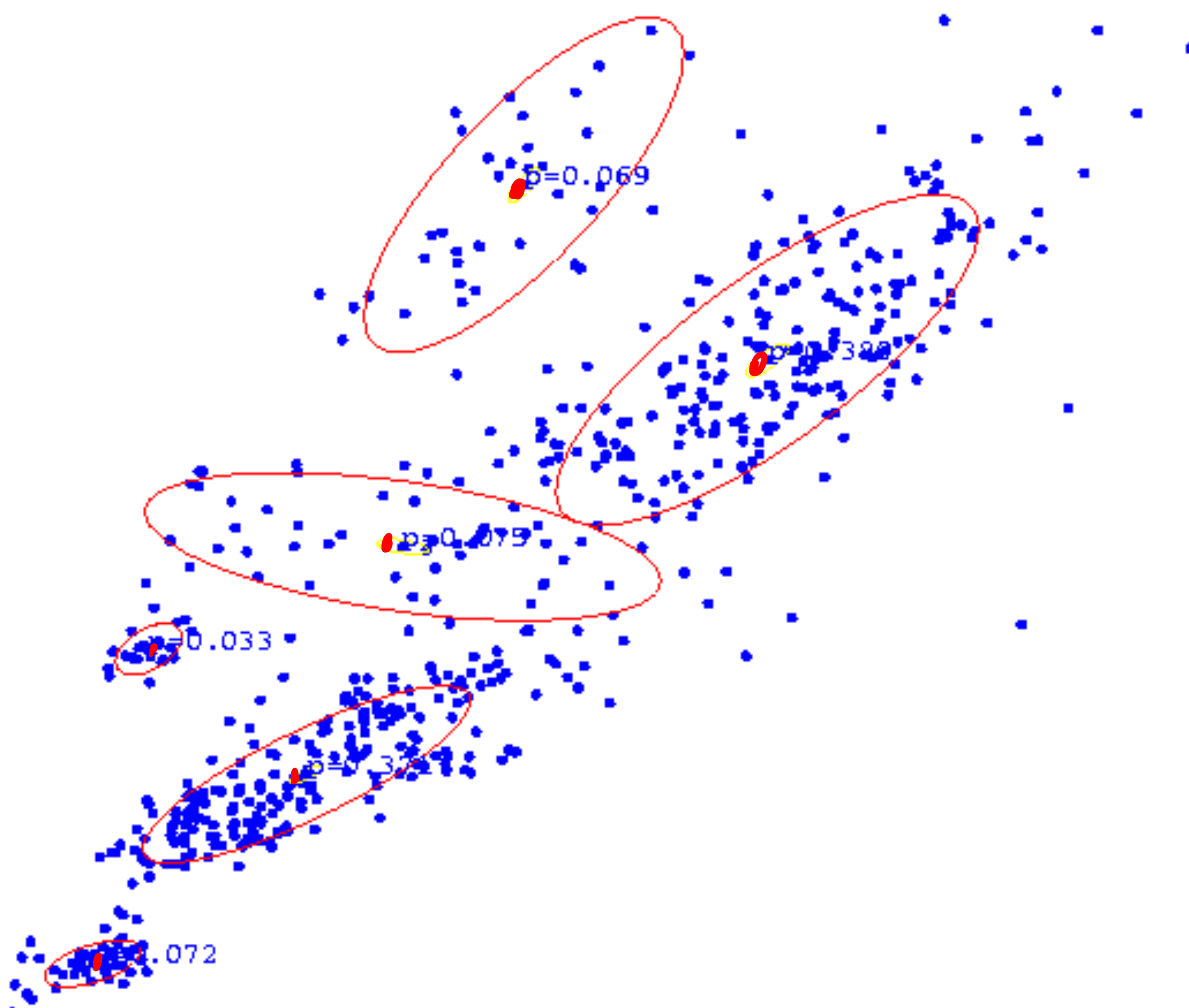
$\uparrow$

$p(y=i|x)$

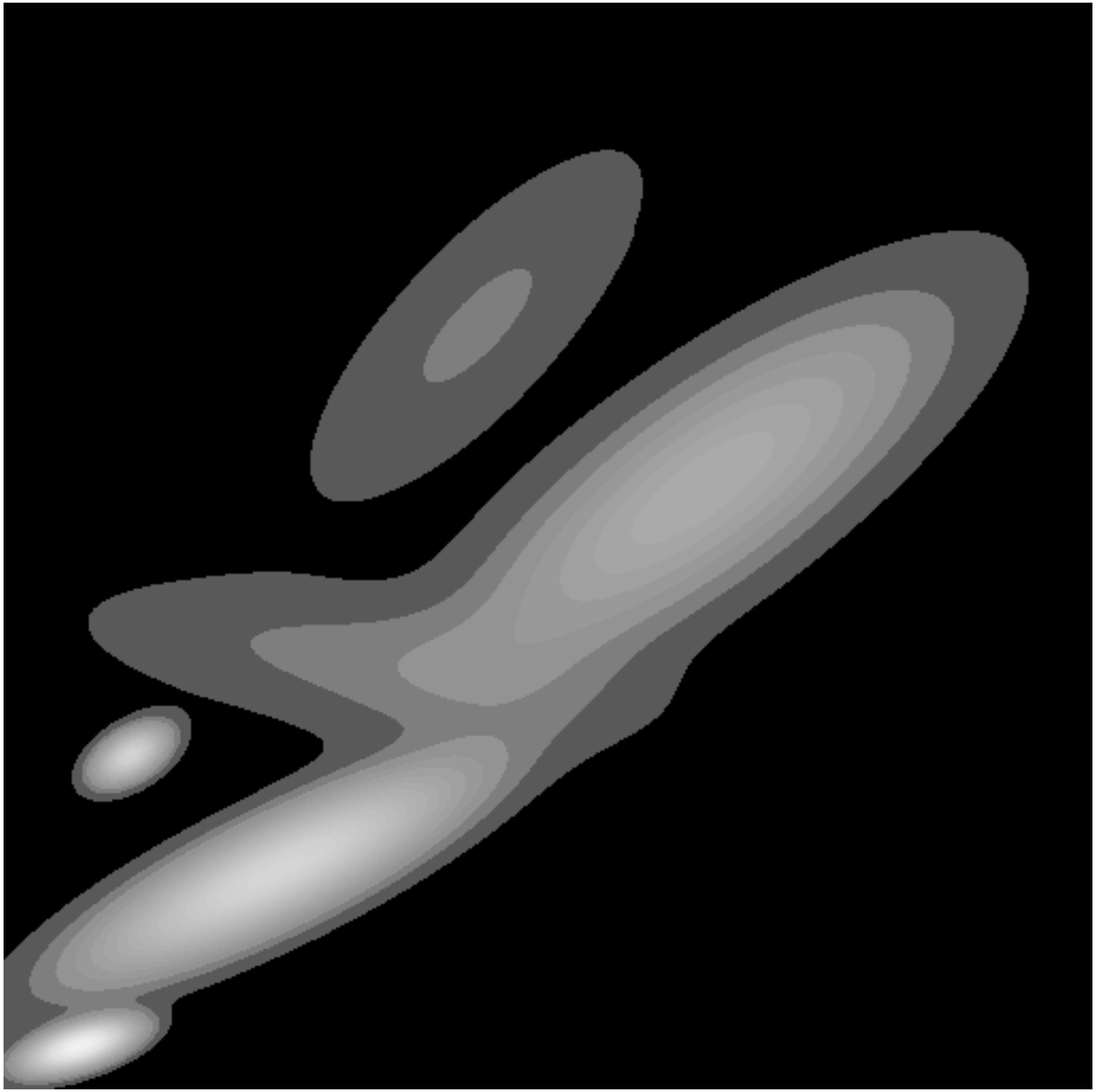


GMM clustering of assay data

k=6

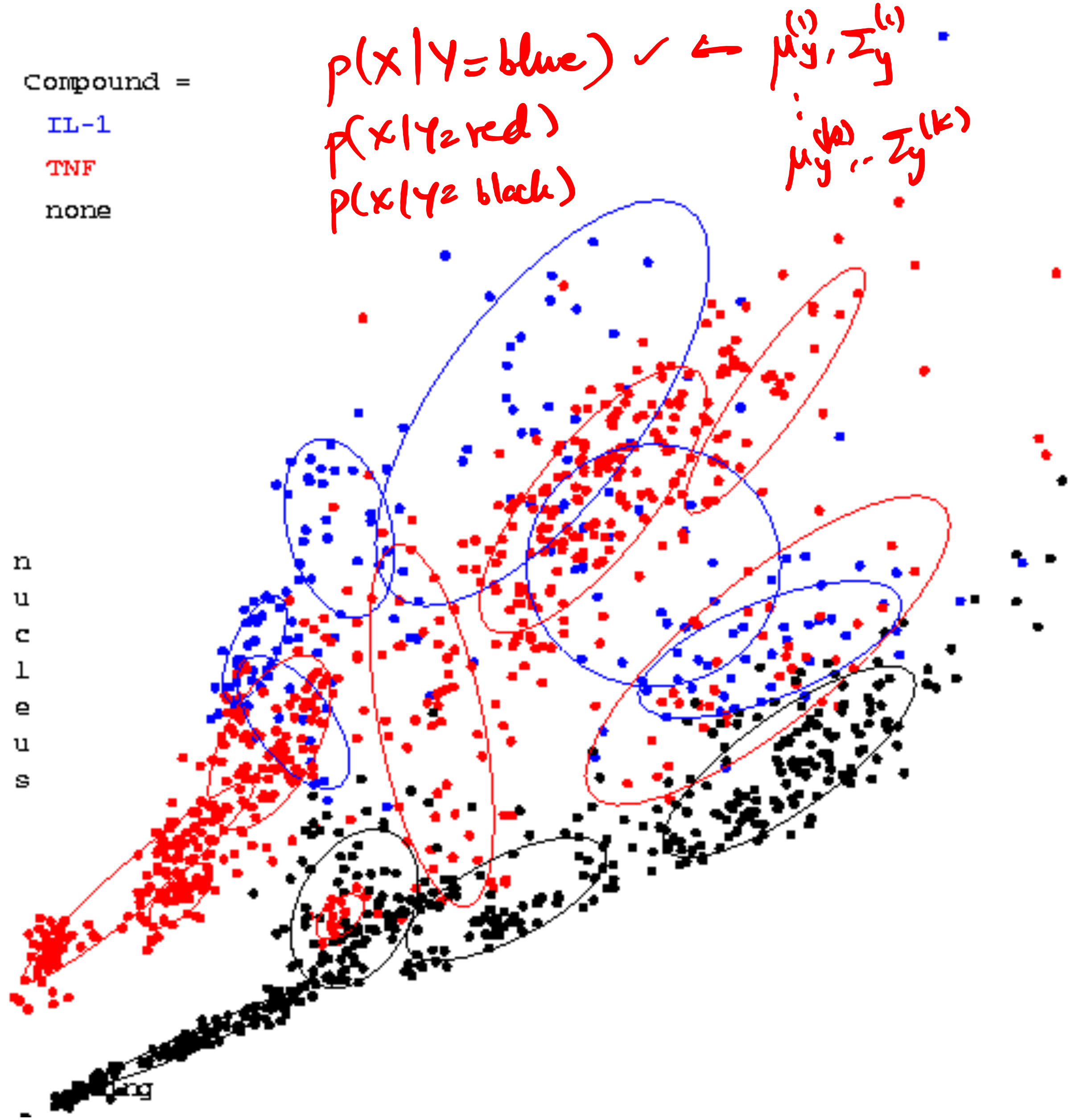


Resulting Density Estimator

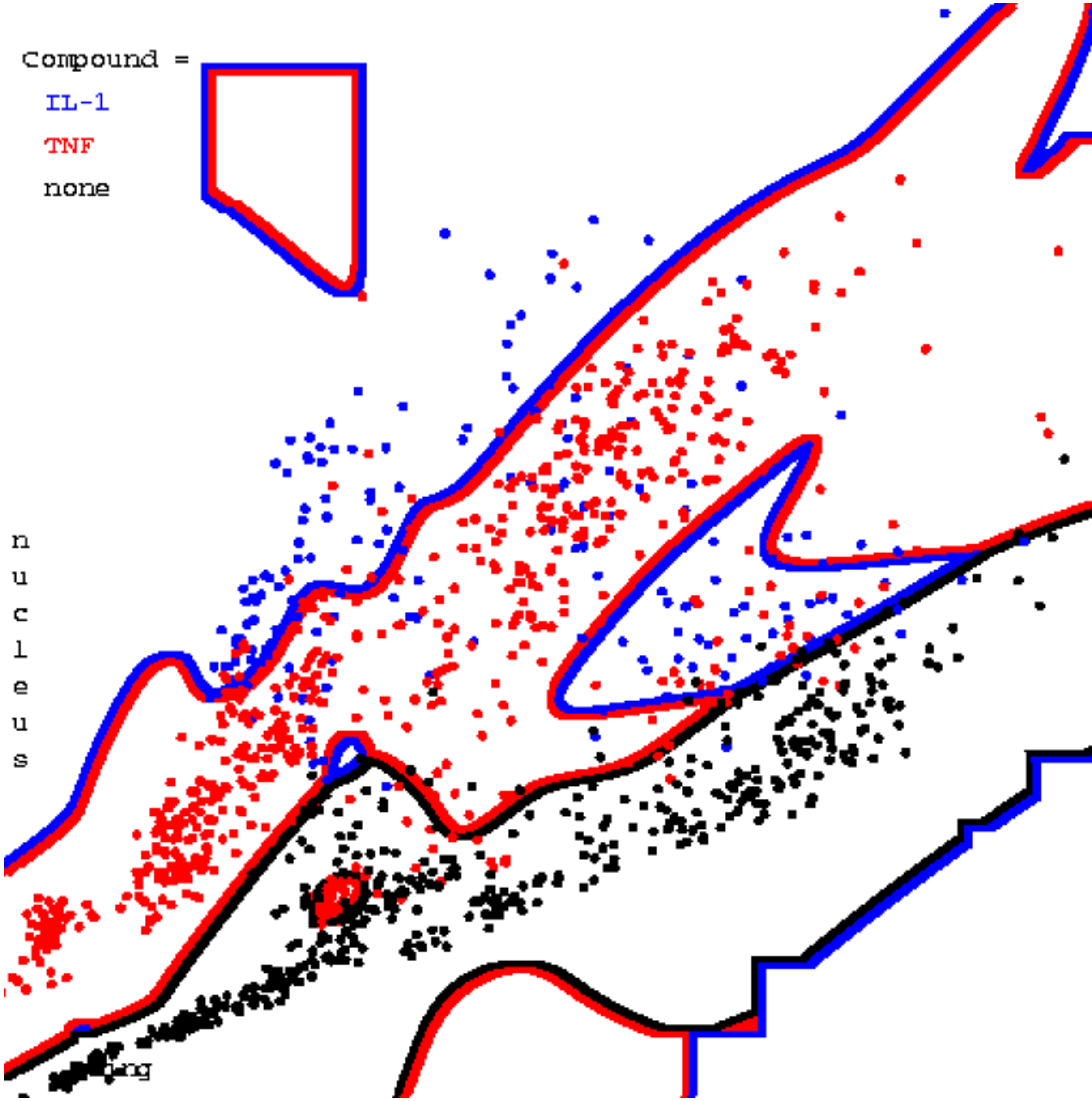


Three classes of assay

(each learned with it's own mixture model)



Resulting Bayes Classifier



Expectation-Maximization (EM)

A general algorithm to deal with hidden data

- No need to choose step size as in Gradient methods.
- EM is an Iterative algorithm with two linked steps:

  E-step: fill-in hidden data (Y) using inference

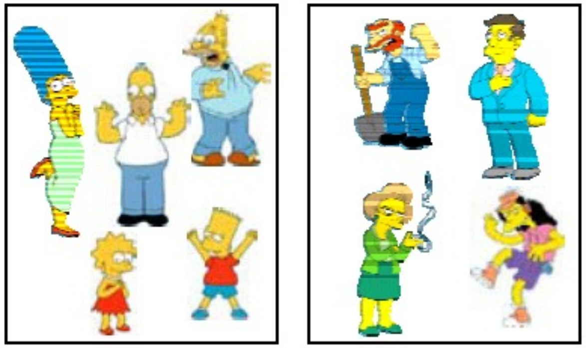
  M-step: apply standard MLE/MAP method to estimate parameters

  $\{p_i, \mu_i, \Sigma_i\}^k_{i=1}$

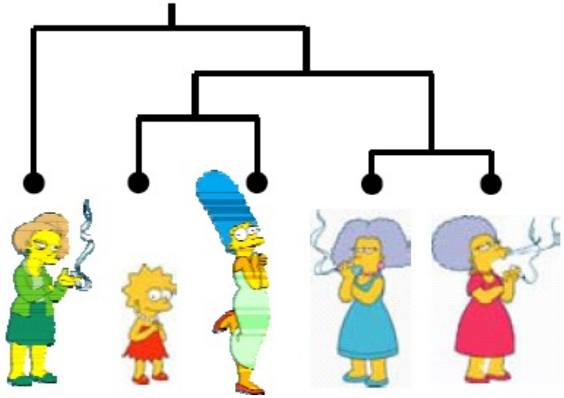
- This procedure monotonically improves the likelihood (or leaves it unchanged). Thus it always converges to a local optimum of the likelihood.

Clustering Algorithms

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



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



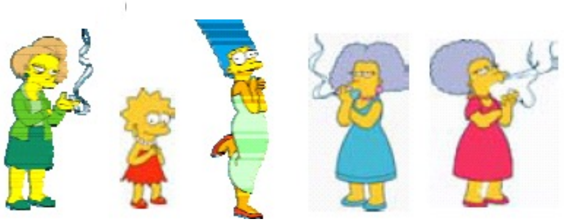
Hierarchical Clustering

- Bottom-Up Agglomerative Clustering

  Starts with each object in a separate cluster, and repeat:
  - Joins the most similar pair of clusters,
  - Update the similarity of the new cluster to others

  until there is only one cluster.

  Greedy - less accurate but simple to implement

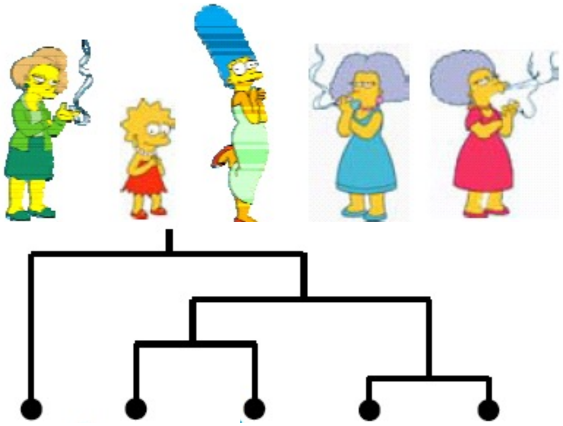


- Top-Down divisive

  Starts with all the data in a single cluster, and repeat:
  - Split each cluster into two using a partition algorithm

  Until each object is a separate cluster.

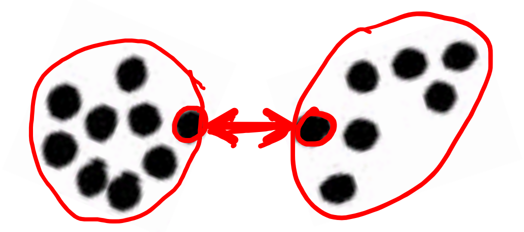
  More accurate but complex to implement



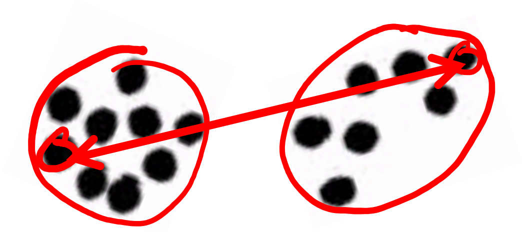
Bottom-up Agglomerative clustering

Different algorithms differ in how the similarities are defined (and hence updated) between two clusters

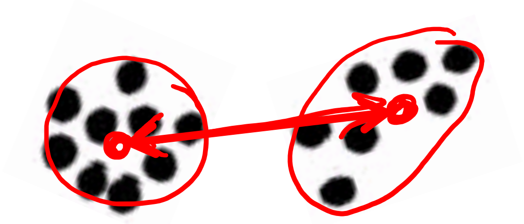
- Single-Linkage
  - Nearest Neighbor: similarity between their closest members.



- Complete-Linkage
  - Furthest Neighbor: similarity between their furthest members.



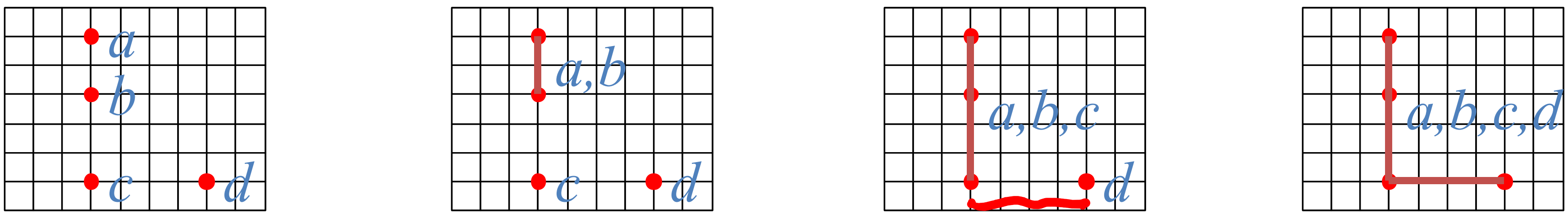
- Centroid
  - Similarity between the centers of gravity



- Average-Linkage
  - Average similarity of all cross-cluster pairs.

Single-Linkage Method

Euclidean Distance



(1) (2) (3)

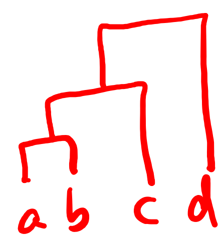
	b	c	d
a	2	5	6
b		3	5
c			4

	b	c	d
a	2	5	6
b		3	5
c			4

	c	d
a,b	3	5
c		4

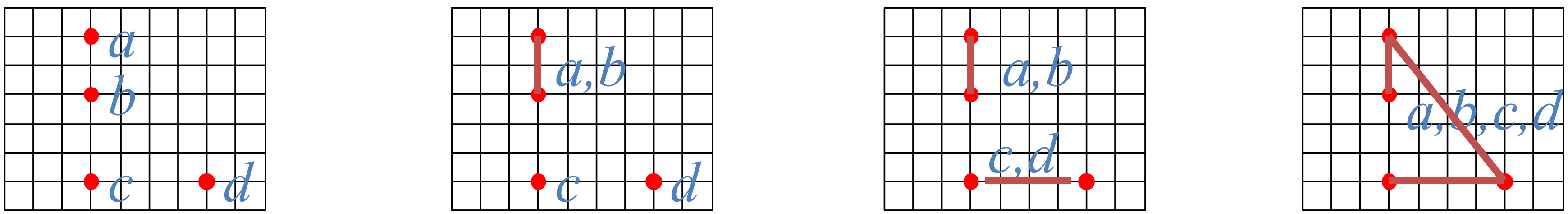
	d
a,b,c	4

Distance Matrix



Complete-Linkage Method

Euclidean Distance



(1) (2) (3)

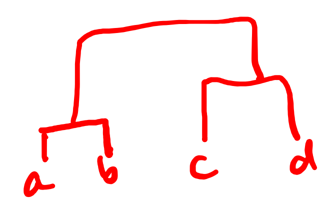
	b	*c*	*d*
a	2	5	6
b		3	5
c			4

	b	*c*	*d*
a	2	5	6
b		3	5
c			4

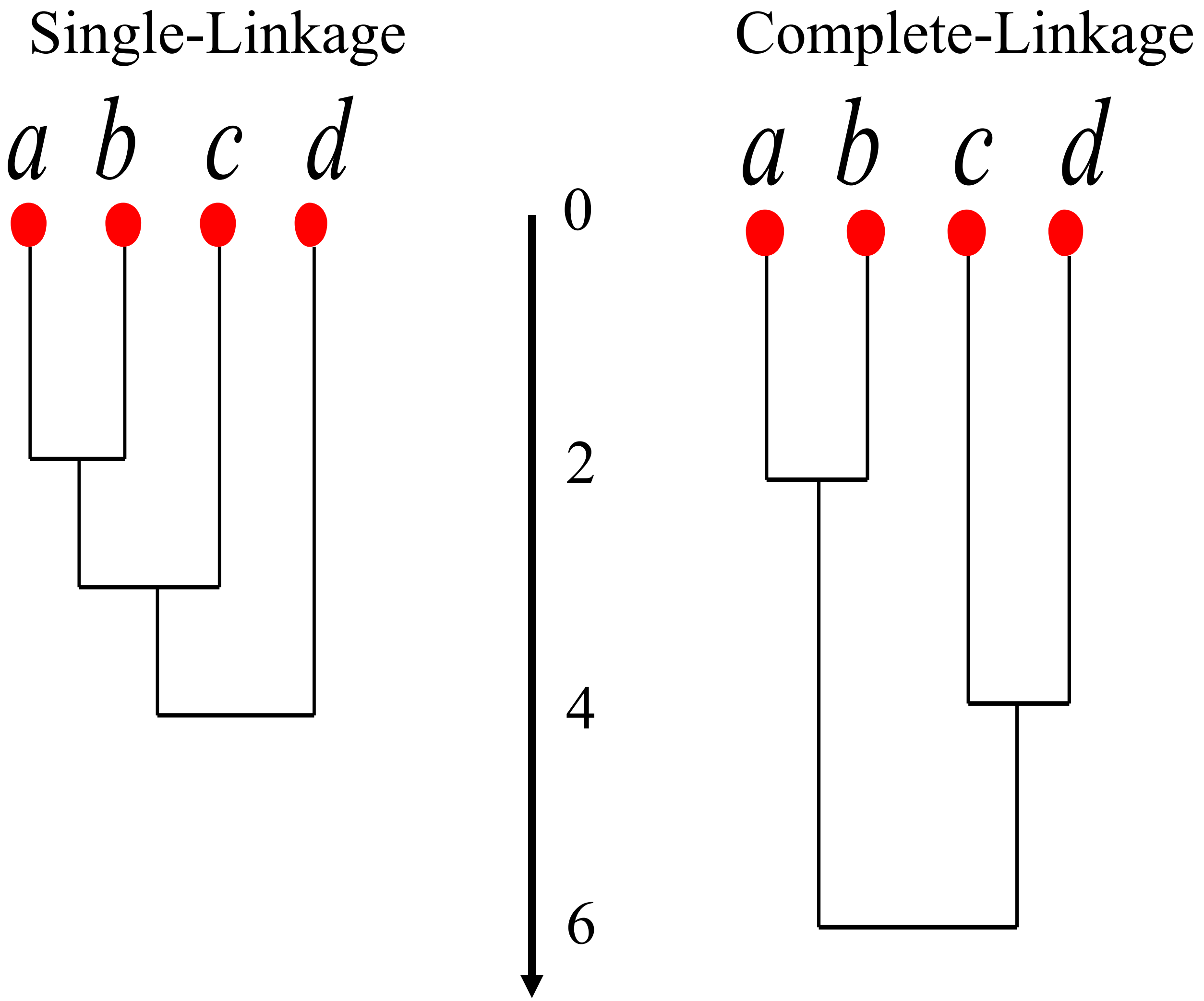
	c	*d*
a,b	5	6
c		4

	c,d
a,b	6

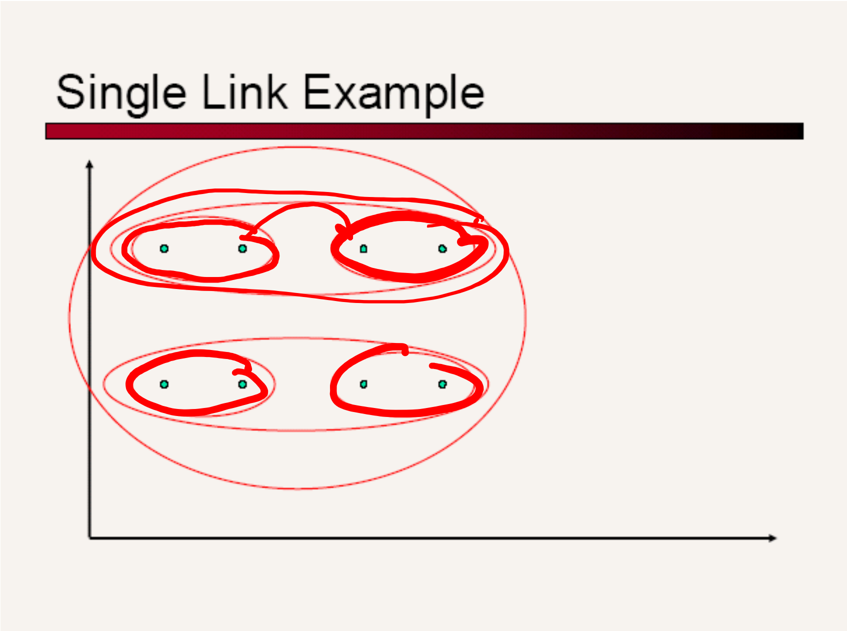
Distance Matrix

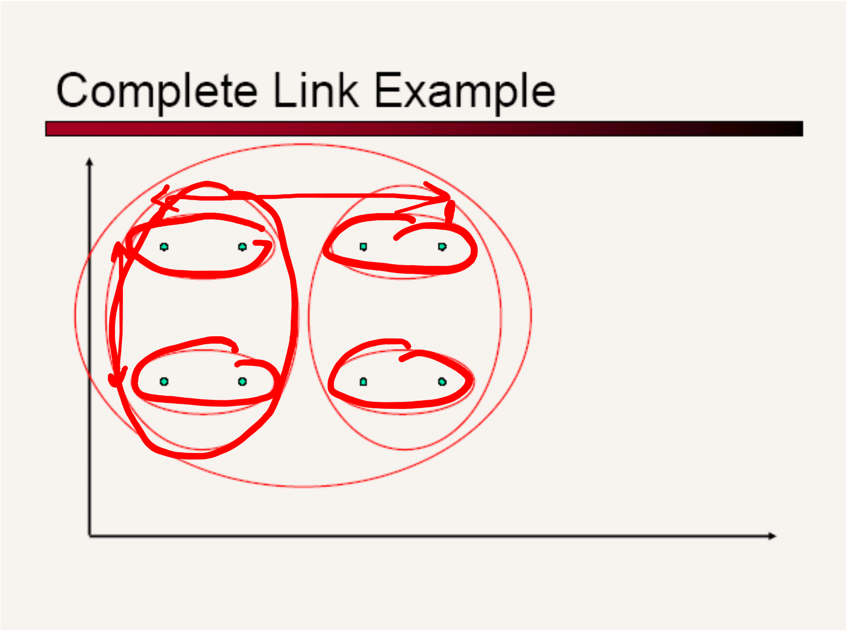


Dendrograms



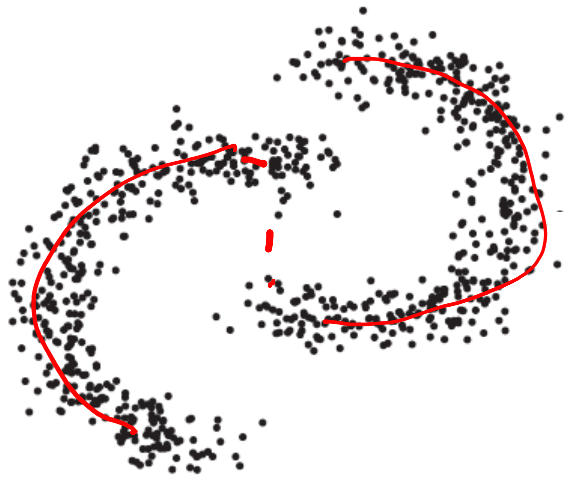
Another Example





Single vs. Complete Linkage

	Shape of clusters
Single-linkage	allows anisotropic and non-convex shapes
Complete-linkage	assumes isotopic, convex shapes



Computational Complexity

- All hierarchical clustering methods need to compute similarity of all pairs of *n* individual instances which is $O(n^2)$.

- At each iteration,
  - Sort similarities to find largest one $O(n^2 \log n)$.
  - Update similarity between merged cluster and other clusters.

    Computing similarity to each other cluster can be done in constant time.

- So we get $O(n^2 \log n)$ or $O(n^3)$ (if naïvely implemented)

nK x iteration

What you need to know…

- Partition based clustering algorithms
  - K-means
    - Coordinate descent
    - Seeding
    - Choosing K
  - Mixture models

    EM algorithm

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