CLUSTERING

DATA/MSML 603: Principles of Machine Learning

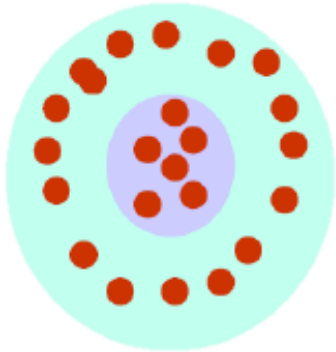
Last Time

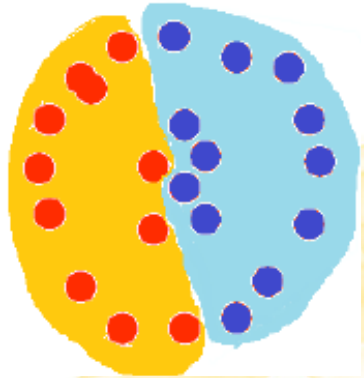
- Autoencoders
  - Often used for representation learning or dimensionality reduction
- Generative Adversarial Networks (GANs)
  - Consists of two neural networks – one for Generator and the other for Discriminator
  - Generative model used to produce new data points/samples
- Variational Autoencoders
  - Encoder produces a probability distribution (Gaussian)
  - Used to generate new data points/samples
- Transformer Networks
  - Employs self attention, multi-head attention and positional encoding

Clustering

What Is Clustering?

- Organization of unlabeled data into similar groups called **clusters**
  - **Unsupervised** learning
  - A cluster is a collection of data items which are “similar” to each other, and “dissimilar” to data items in other clusters
  - Helps discover patterns, outliers and structure in data and reduces complexity and dimensionality of data
    - Pre-process data before applying a dimensionality reduction technique
    - Clustering-based dimensionality reduction technique

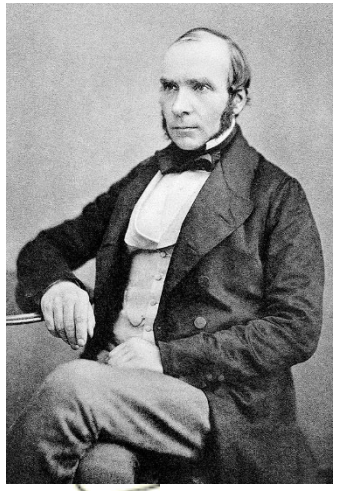




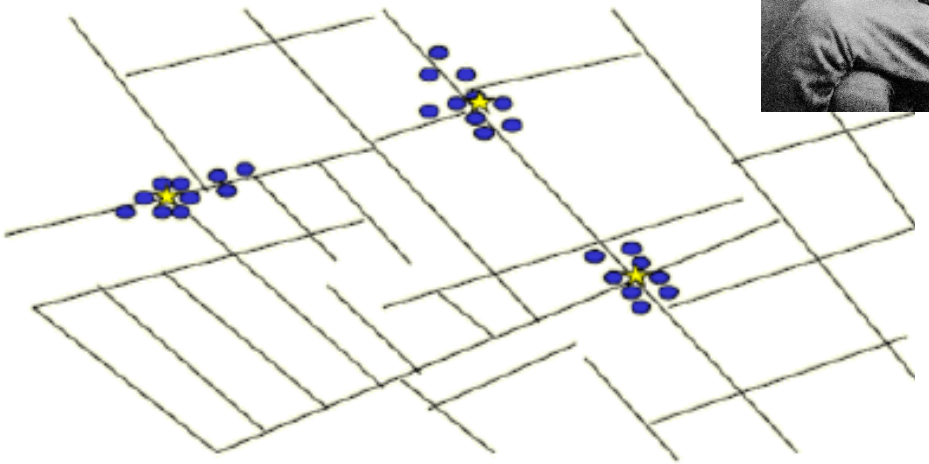
How is Clustering used in Dimensionality Reduction?

- Pre-processing data with clustering
  - Can help remove outliers, noise and irrelevant features
  - Reduce the number of data points by using cluster representatives or centroids (centers of clusters)
    - Can improve the efficiency and effectiveness of dimensionality reduction technique as well as scalability of algorithms
  - Example: Cluster data points into k clusters (for instance, using k-means clustering) and use the k cluster centroids as input for Principal Component Analysis (PCA)
- Clustering-based dimensionality reduction techniques (topic of next lecture)
  - Combine clustering and dimensionality reduction in one step
  - Find a low(er)-dimensional representation of data which preserves cluster structure
  - Example: Spectral clustering uses eigenvectors of a similarity matrix to cluster data

Historic Application of Clustering



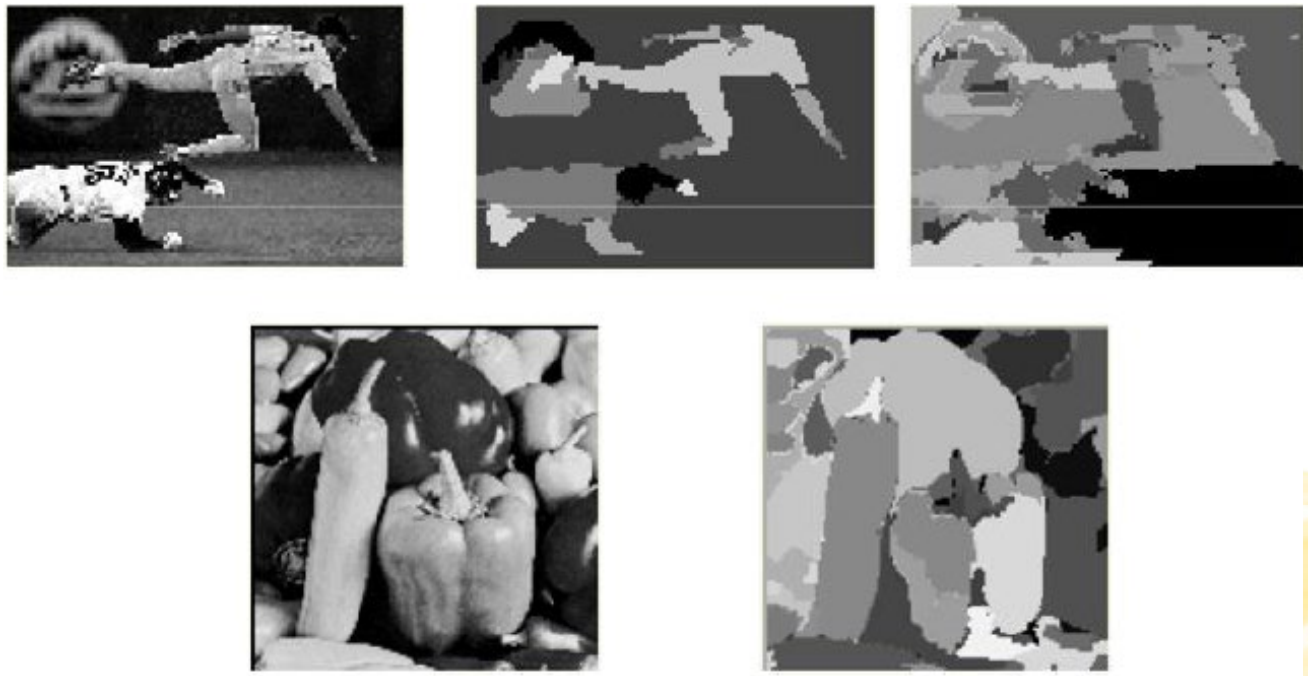
- John Snow, a British physician plotted the location of cholera deaths on a map during an outbreak in the 1850s
- The locations indicated that cases were clustered around certain intersections where there were polluted wells—thus exposing both the (source of the) problem and the solution



From: Nina Mishra HP Labs

Computer Vision Application: Image Segmentation

- Pixels with similar characteristics (e.g., color, intensity) grouped into clusters
  - Distinct segments of an image (e.g., objects or regions) represented as clusters
  - Boundary detection



From: Image Segmentation by Nested Cuts, O. Veksler, CVPR2000

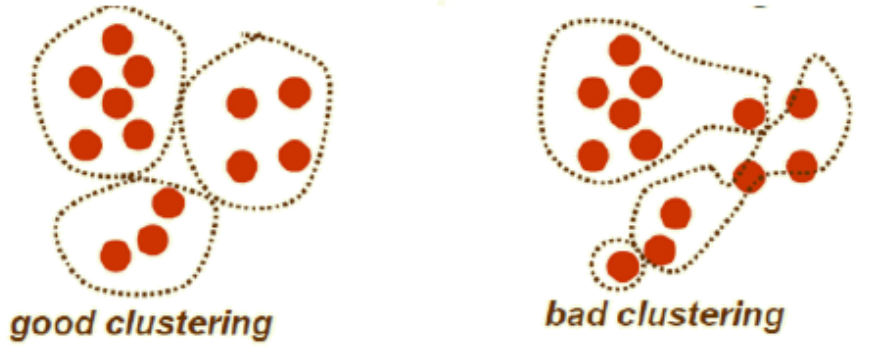
Other Applications of Clustering

- Customer segmentation and targeted advertising
  - Study purchase patterns and content recommendation (e.g., streaming)
- Epidemiological analysis
  - Identify clusters of regions or populations with high incidence
- Social media
  - Content personalization
- City planning
  - Resource allocation (playgrounds, schools, parks)
  - Transportation planning based on traffic patterns
- Biology and genetic research

What Do We Need for Clustering?

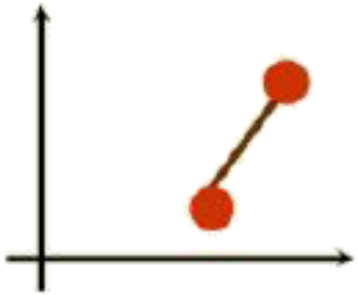
1. Proximity measure, either:
   - Similarity measure $s(x_i, x_k)$: large if x_i and x_k similar
   - Dissimilarity (or distance) measure $d(x_i, x_k)$: small if x_i and x_k are similar

2. Criterion function to evaluate clustering
3. Algorithm to compute clustering
   - For example, by optimizing the criterion function



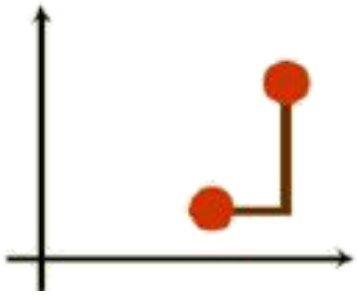
1. Distance (Dissimilarity) Measures

- Euclidean distance
  - Translation invariant: $$d_E(\mathbf{x}^1, \mathbf{x}^2) = \sqrt{\sum_{k=1}^{d} (x_k^1 - x_k^2)^2} = \|\mathbf{x}^1 - \mathbf{x}^2\|_2$$



- Manhanttan (city block) distance:

$$d_M(\mathbf{x}^1, \mathbf{x}^2) = \sum_{k=1}^{d} \left|x_k^1 - x_k^2\right| = \|\mathbf{x}^1 - \mathbf{x}^2\|_1$$



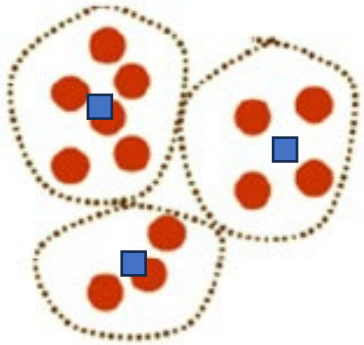
  - Approximation to Euclidean distance, cheaper to compute
- They are special cases of Minkowski distance

$$d_{MI}(\mathbf{x}^1, \mathbf{x}^2) = \left(\sum_{k=1}^{d} |x_k^1 - x_k^2|^p\right)^{1/p} = \|\mathbf{x}^1 - \mathbf{x}^2\|_p$$

  - p is a positive integer:

2. Cluster Evaluation (A Hard Problem)

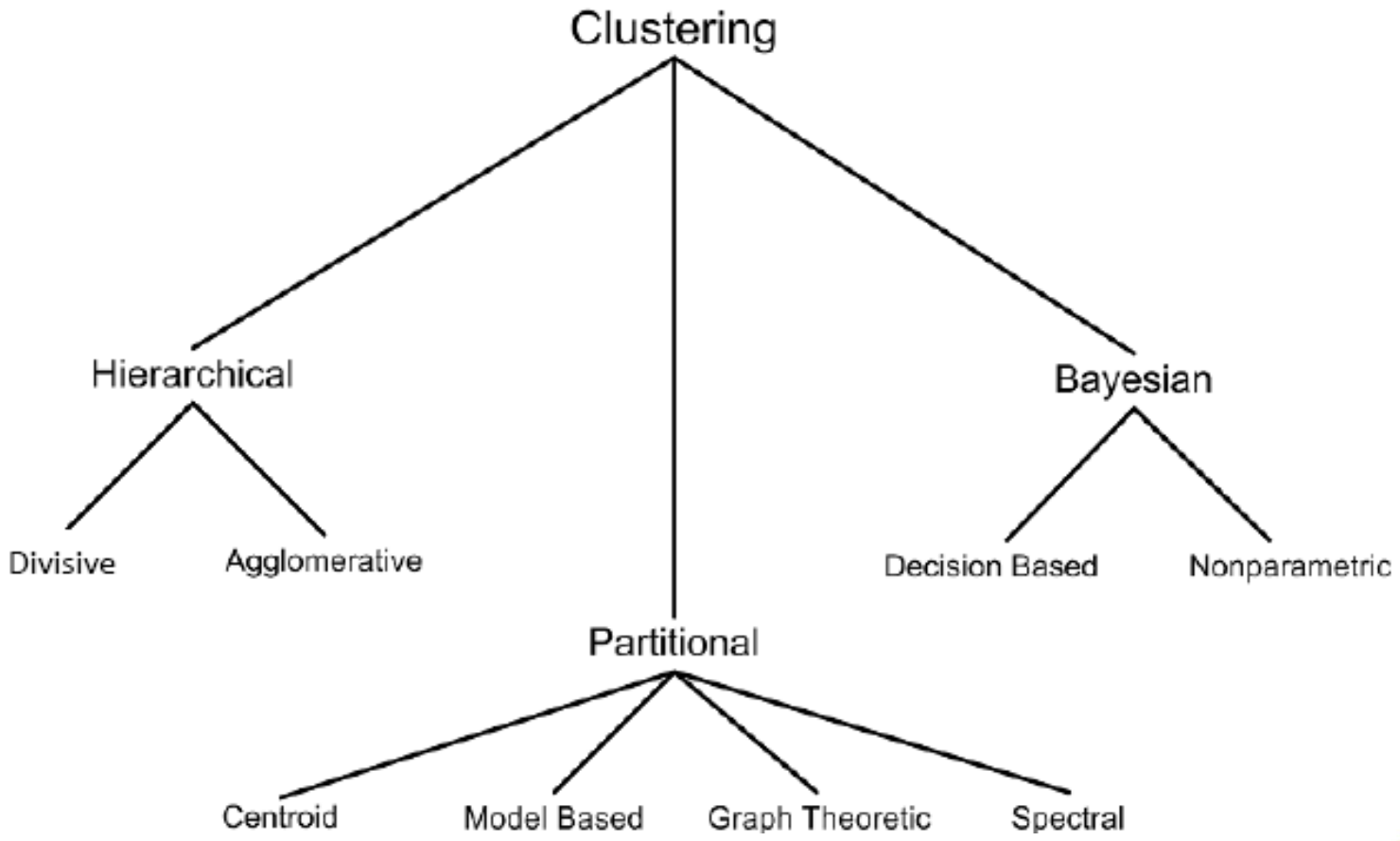
- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how close the data points in a cluster are to the cluster centroid
  - Sum of squared error (SSE) is a commonly used measure
- **Inter-cluster separation** (isolation):
  - Different cluster centroids should be far away from one another
- In most applications, expert judgments are still the key



2. How Many Clusters?

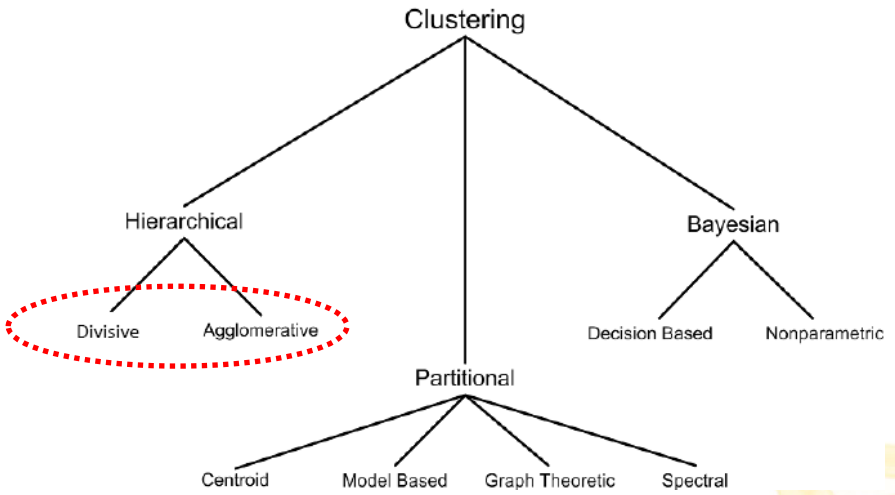
- Possible approaches
  1. Fix the number of clusters to k
  2. Find the best clustering according to the criterion function, e.g., information criterion, such as Akaike information criterion
     (number of clusters may vary)

3. Clustering Techniques (1/4)



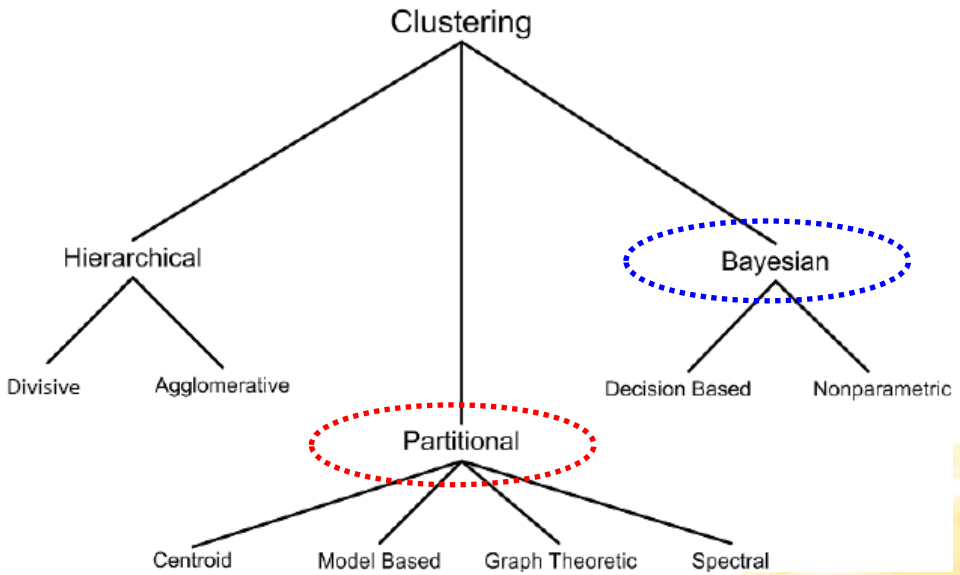
3. Clustering Techniques (2/4)

- **Hierarchical algorithms** find successive clusters using previously established clusters. These algorithms can be either agglomerative ("bottom-up") or divisive ("top-down"):
  - Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters
  - Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters

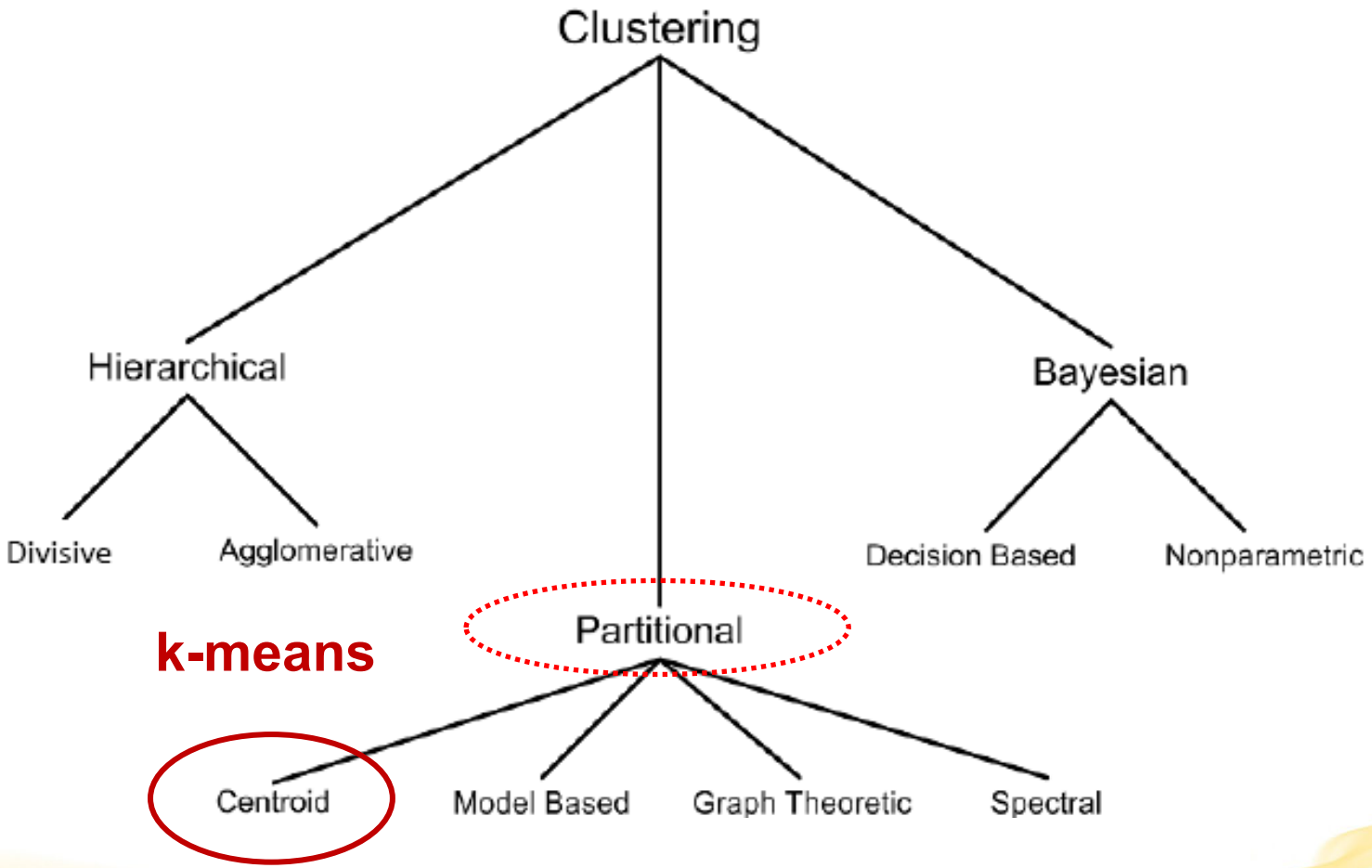


3. Clustering Techniques (3/4)

- **Partitional algorithms** typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering
- **Bayesian algorithms** try to generate a posterior distribution over the collection of all partitions of the data



3. Clustering Techniques (4/4)



k-Means Clustering

k-Means Clustering

- ***k*-means clustering** (MacQueen, 1967) is a **partitional clustering algorithm**
  - Idea itself goes back further to Steinhaus (1956)
- Let $D = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n\}$ be the set of data points
  - where $\mathbf{x}^i = (x_1^i, x_2^i, \ldots, x_r^i)$ is a vector in $X \subset \mathbb{R}^r$, and r is the dimension of data points
- *k*-means algorithm partitions the given data points into *k* clusters:
  - *k* is specified by the user
  - Each cluster has a **cluster center**, called the **centroid**

k-Means Algorithm

- Given *k*, the ***k-means* algorithm** works as follows:
  - Choose *k* (random) data points (seeds) to be the initial centroids, i.e., cluster centers
  - Assign each data point to the closest centroid
  - Re-compute the centroids using the current cluster memberships
  - If a convergence criterion is not met, repeat steps 2 and 3

k-Means Convergence (Stopping) Criterion

- No (or minimum) re-assignments of data points to different clusters, *or*
- No (or minimum) change of centroids, *or*
- minimum decrease in the **sum of squared error (SSE)**,

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mathbf{m}_j)^2$$

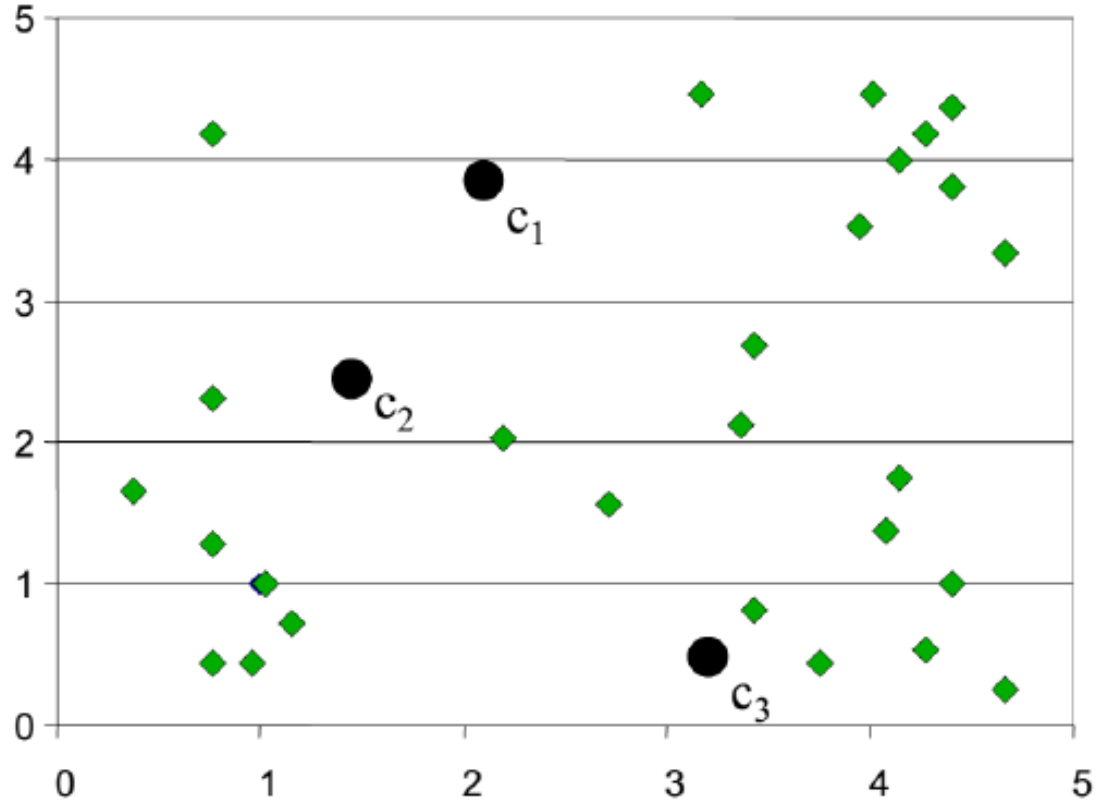
$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

- C_j is the jth cluster,
- $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j),
- $d(\mathbf{x}, \mathbf{m}_j)$ is the (Eucledian) distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

k-Means Clustering Example (1/6)

- Step 1

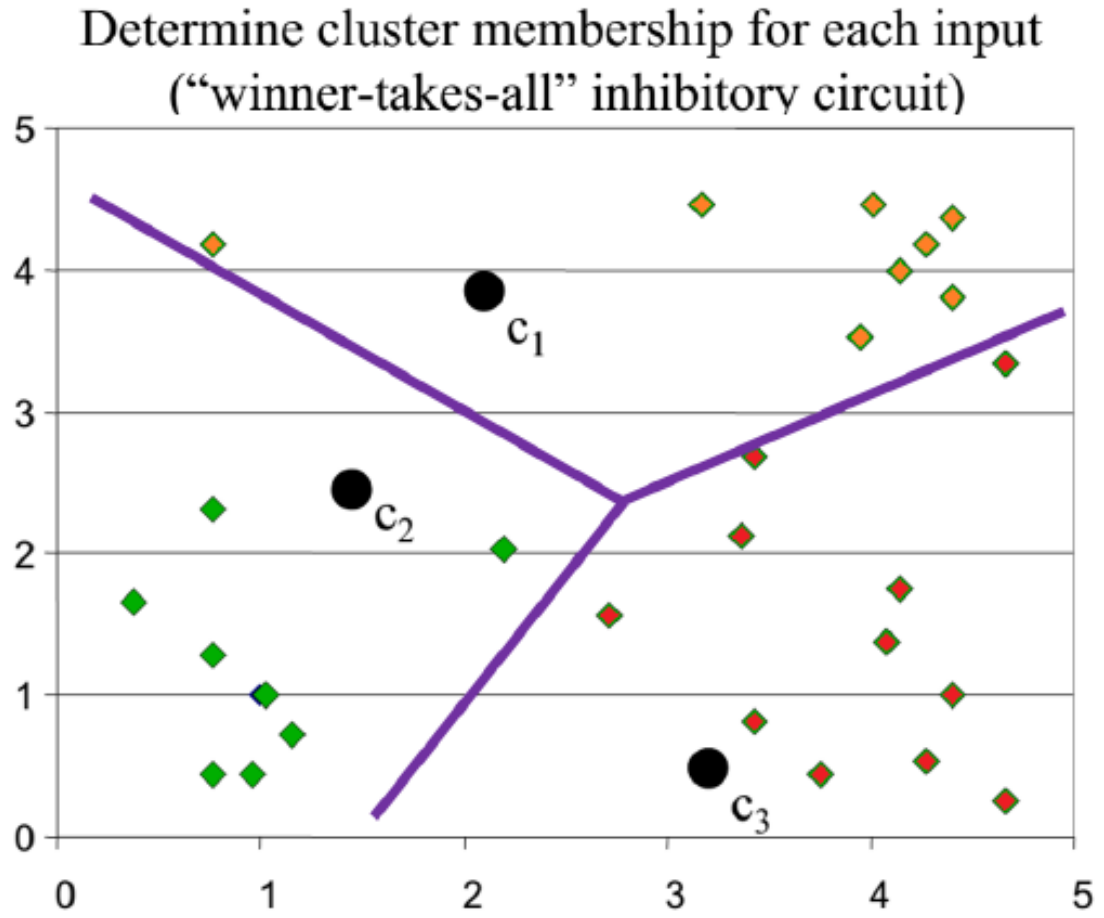
Randomly initialize the cluster centers (synaptic weights)



(k = 3)

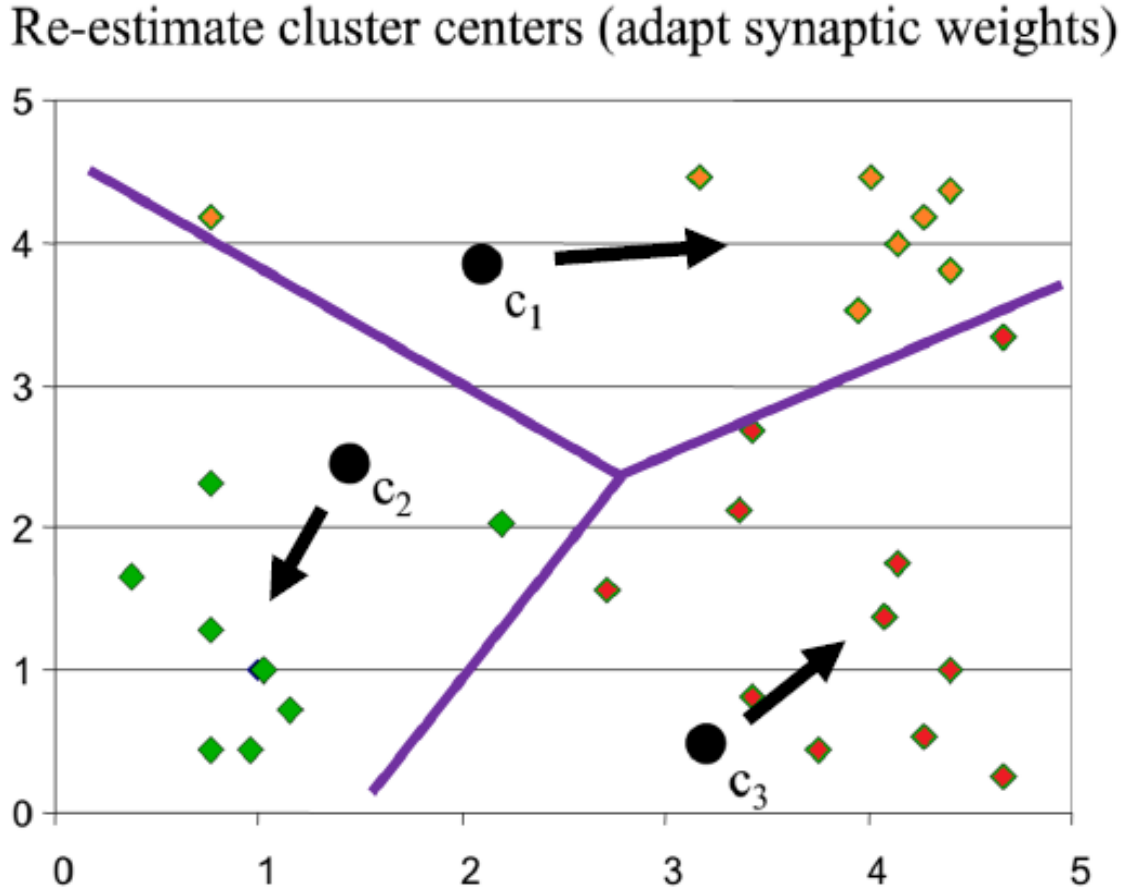
k-Means Clustering Example (2/6)

- Step 2



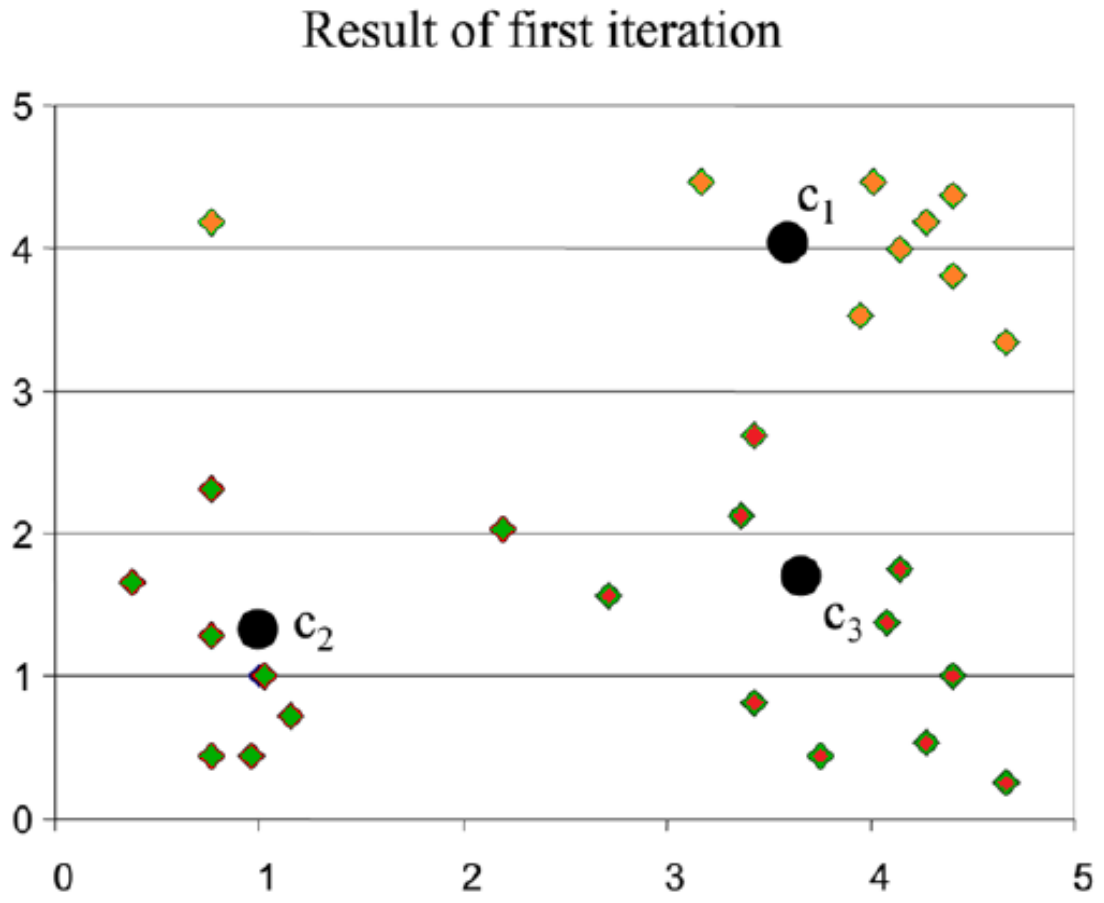
k-Means Clustering Example (3/6)

- Step 3

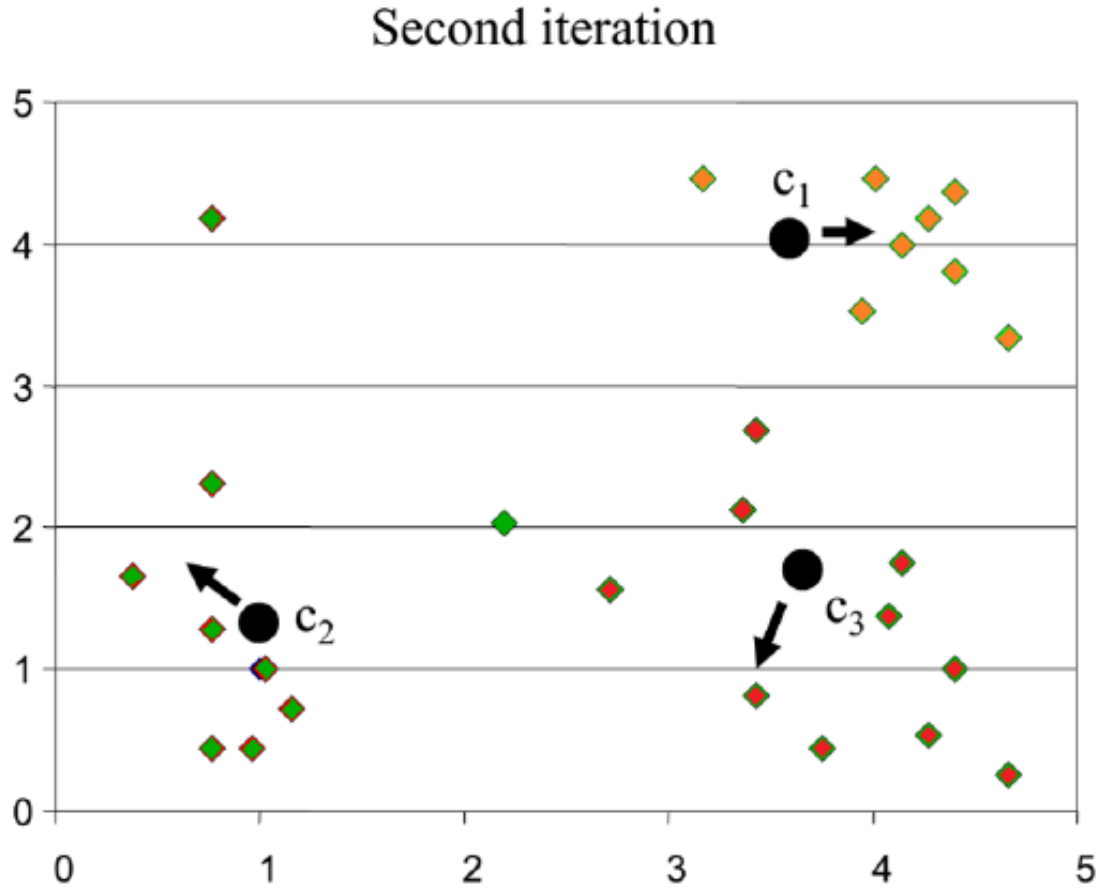


$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

k-Means Clustering Example (4/6)

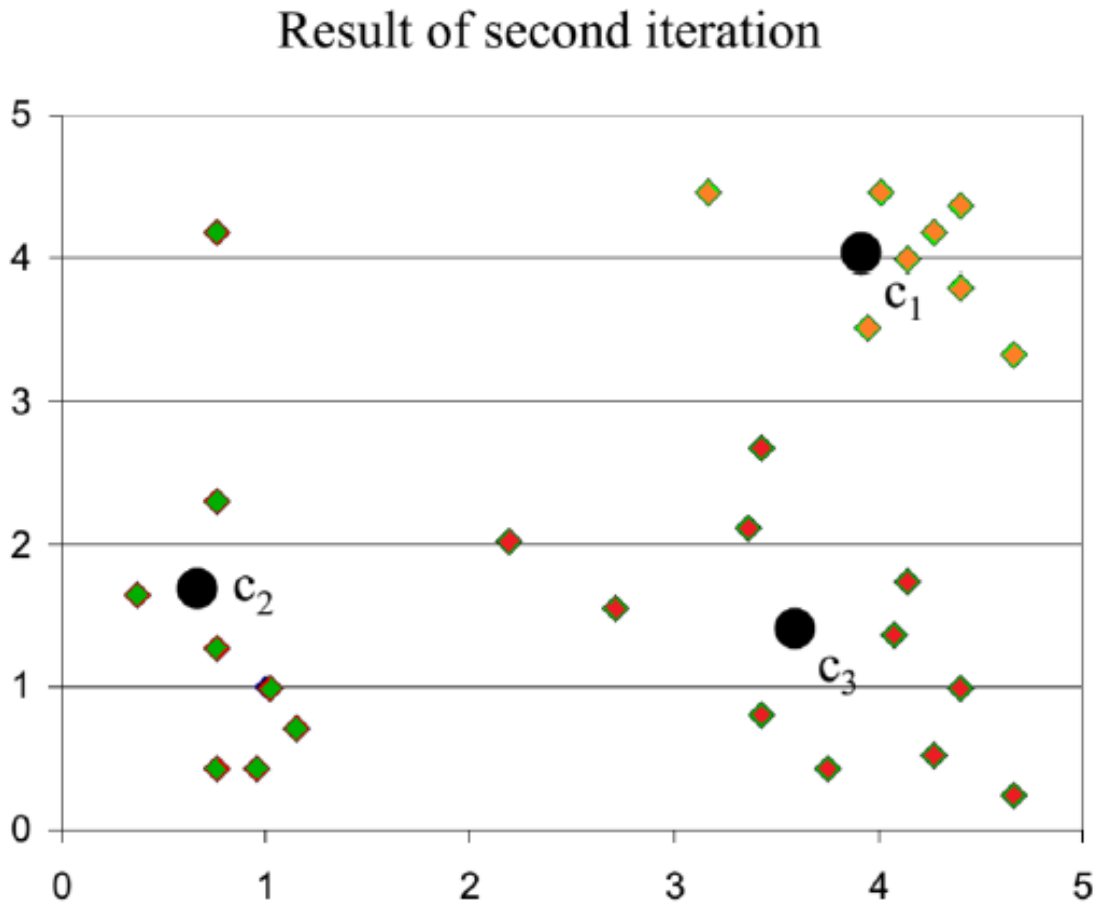


k-Means Clustering Example (5/6)



$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

k-Means Clustering Example (6/6)



Why Use k-Means?

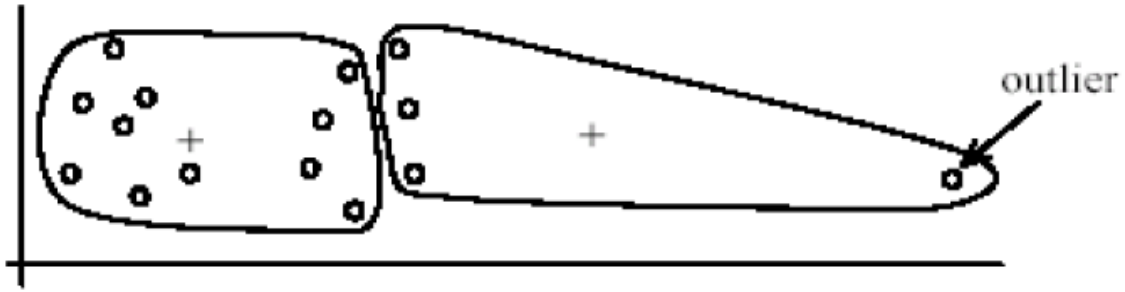
- Advantages:
  - **Simple**: easy to understand and to implement
  - **Efficient**: Time complexity: $O(tkn)$, where n is the number of data points, k is the number of clusters, and t is the number of iterations
  - Since both k and t are small, k-means is considered a linear algorithm
- k-means is the most popular clustering algorithm
- Comment:
  - It terminates at a local optimum if SSE is used
  - The global optimum is hard to find due to complexity

Weaknesses of k-Means

- The algorithm is only applicable if the mean is defined
  - For categorical data, *k*-modes clustering is used - the centroid is represented by the most frequent values (called the mode of cluster)
- The user needs to specify *k*
  - Need to know approximately how many clusters to look for
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points
  - Outliers could be errors in the data recording or some special data points with very different values

Outliers

- Undesirable clusters



$k = 2$

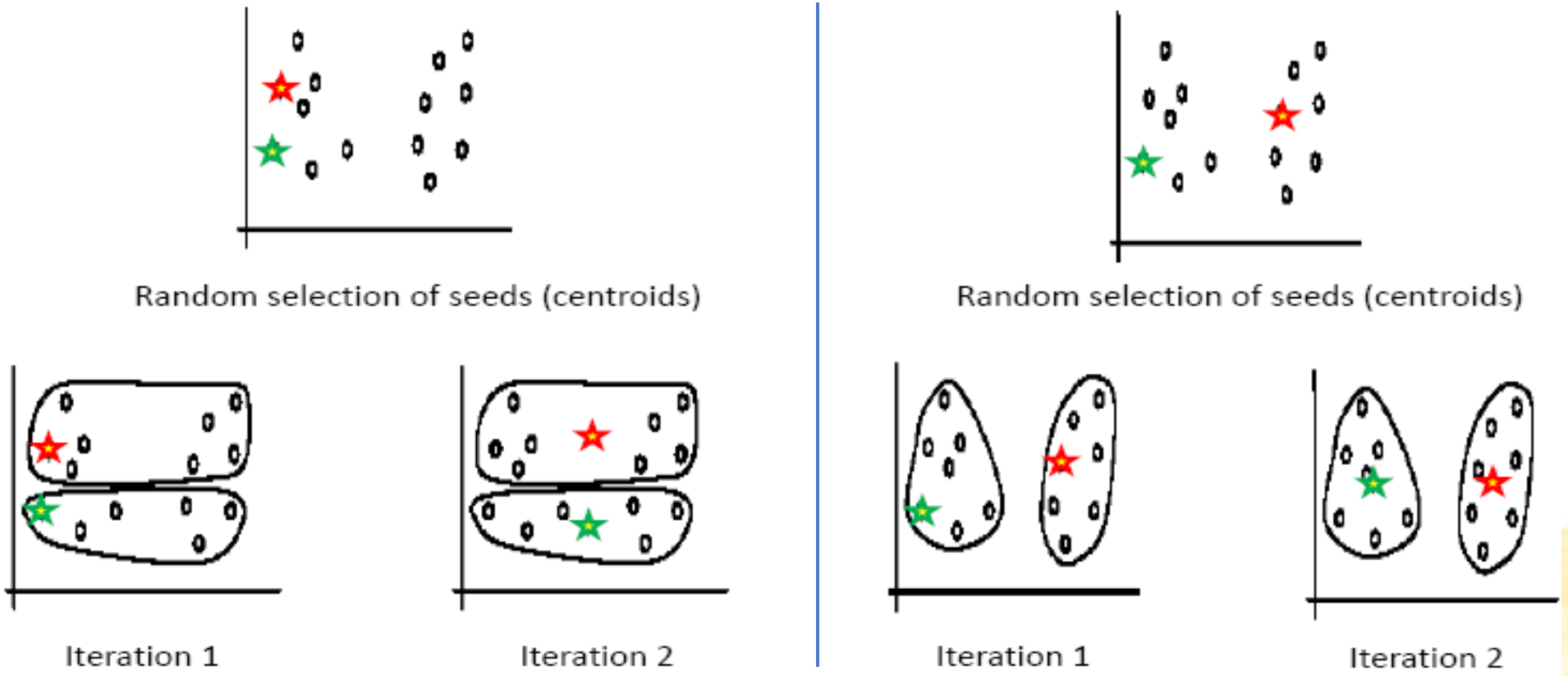
- Ideal clusters

Dealing With Outliers

- Remove some data points that are much further away from the centroids than other data points
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them
- Perform random sampling: by choosing a small subset of the data points, the chance of selecting an outlier is much smaller
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

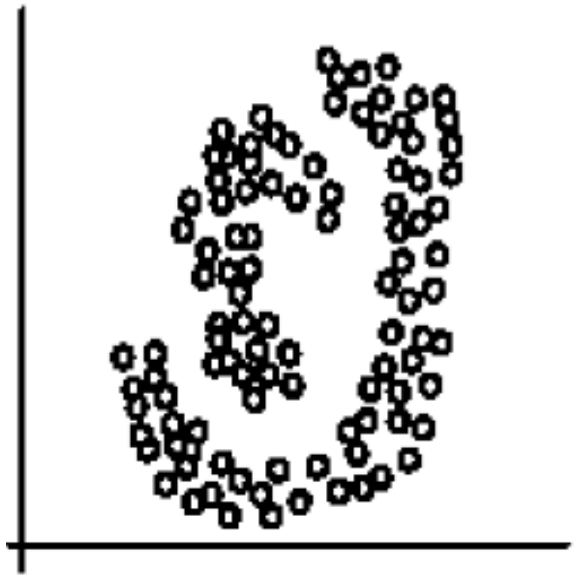
Sensitivity To Initial Seeds

- Sensitive to initial seeds

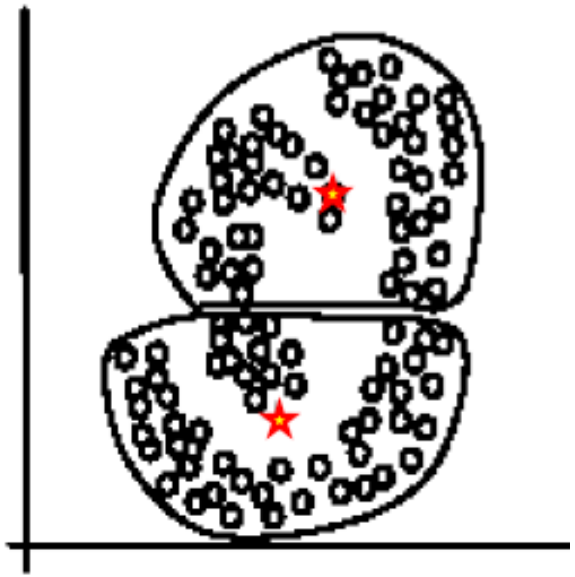


Special Data Structures

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres)



(A): Two natural clusters



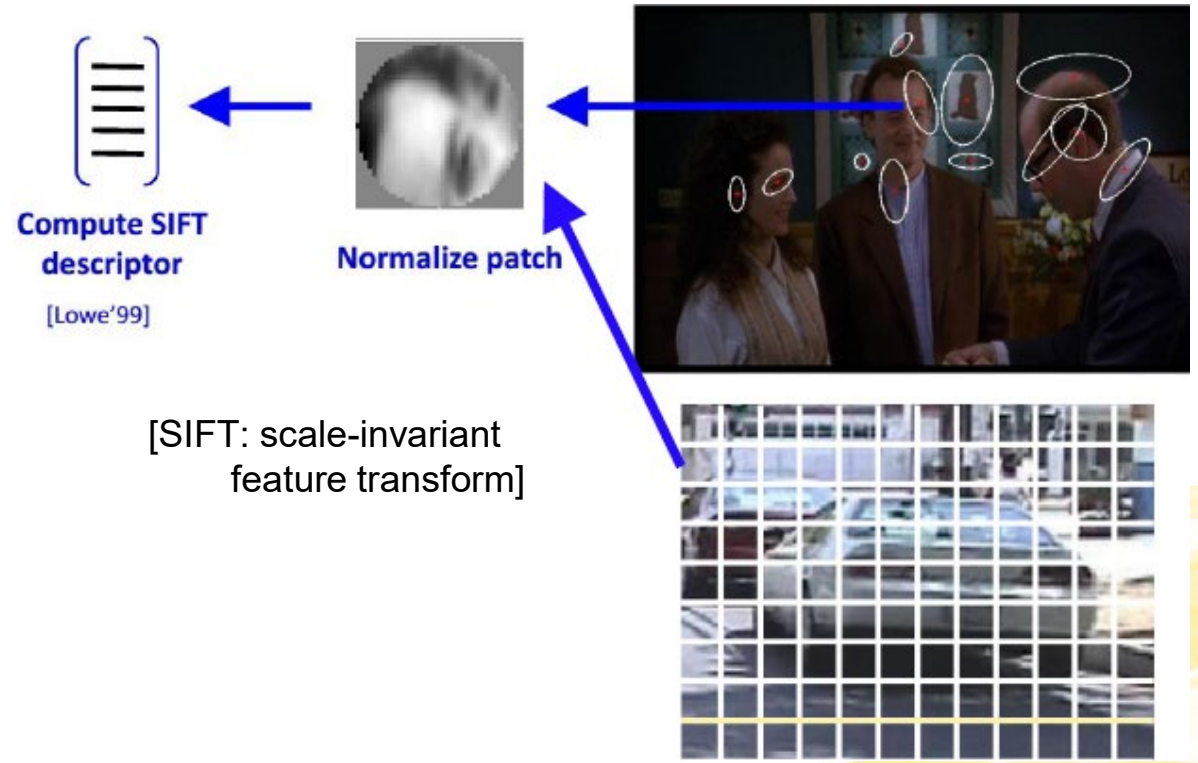
(B): *k*-means clusters

k-Means Summary

- Despite weaknesses, *k*-means clustering is still the most popular algorithm due to its simplicity and efficiency
- No clear evidence that any other clustering algorithm performs better in general
- Comparing different clustering algorithms is a difficult task
  - No one knows the correct clusters!

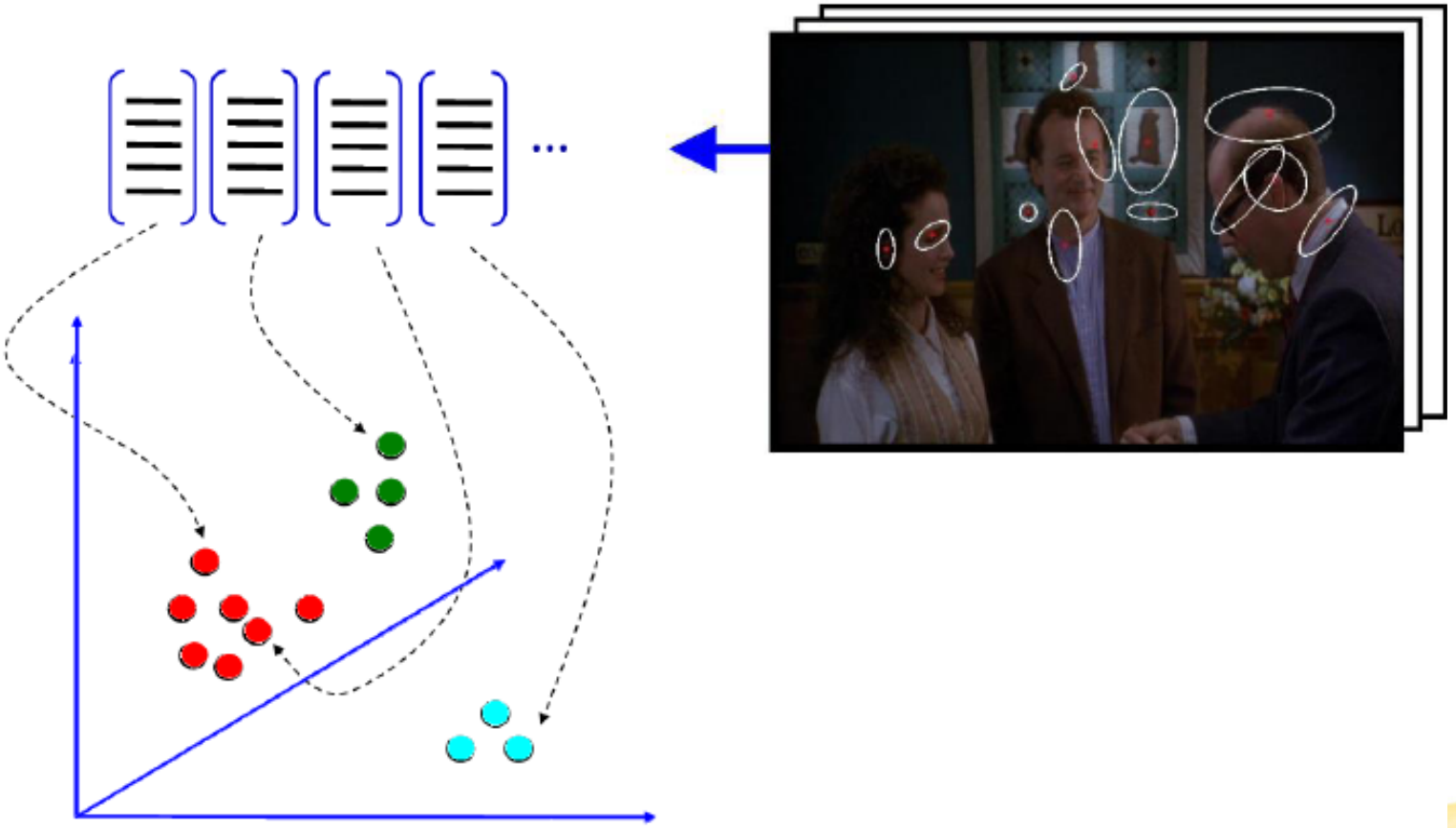
Application to Visual Object Recognition: Dictionary Learning (Bag of Words)

- Dictionary learning – find a set of basic image features (in **dictionary**)
  - Group similar image patches together into a cluster to capture unique characteristics of different image regions
  - Dictionary learns to represent features specific to each cluster using a set of dictionary **atoms**
  - Dictionary is a matrix containing atoms (basic features) as columns
  - An image written as a linear combination of atoms (usually interested in sparse representation)



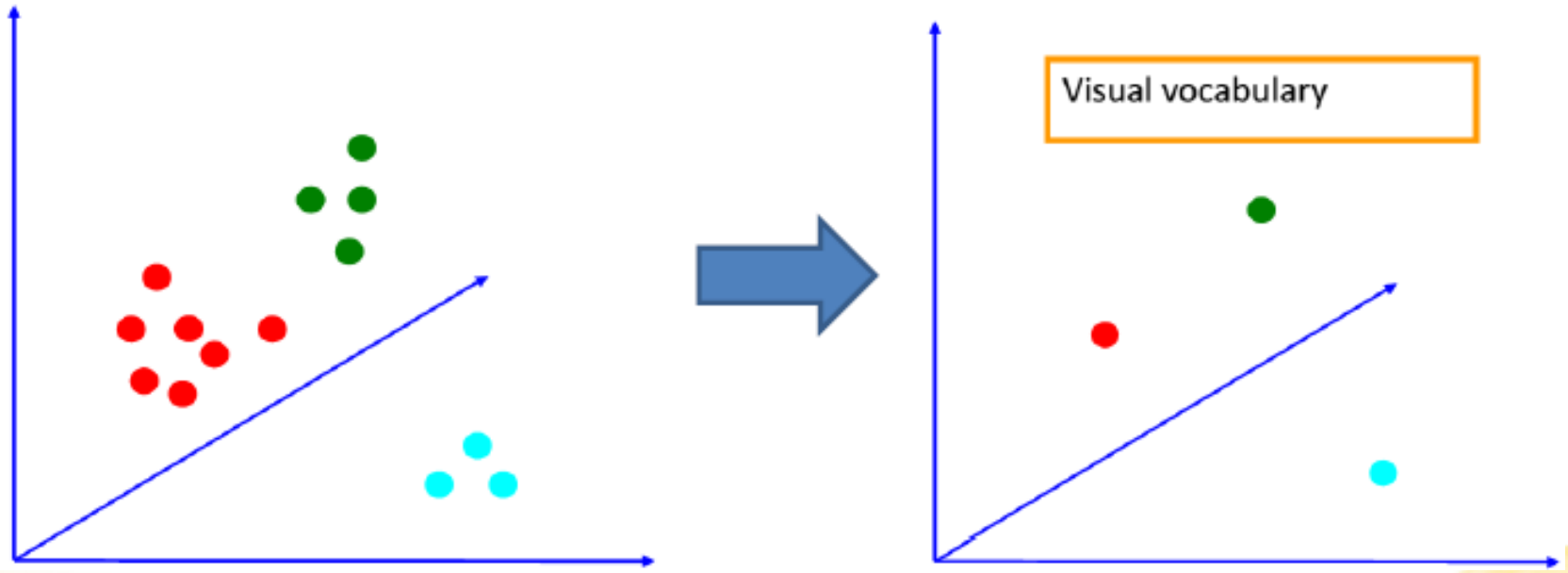
[SIFT: scale-invariant feature transform]

Learning the Visual Vocabulary (1/2)



Learning the Visual Vocabulary (2/2)

- Extract features from different patches of images
- Extracted features clustered into a set of “visual words” (common visual patterns)



k-Medoids Clustering

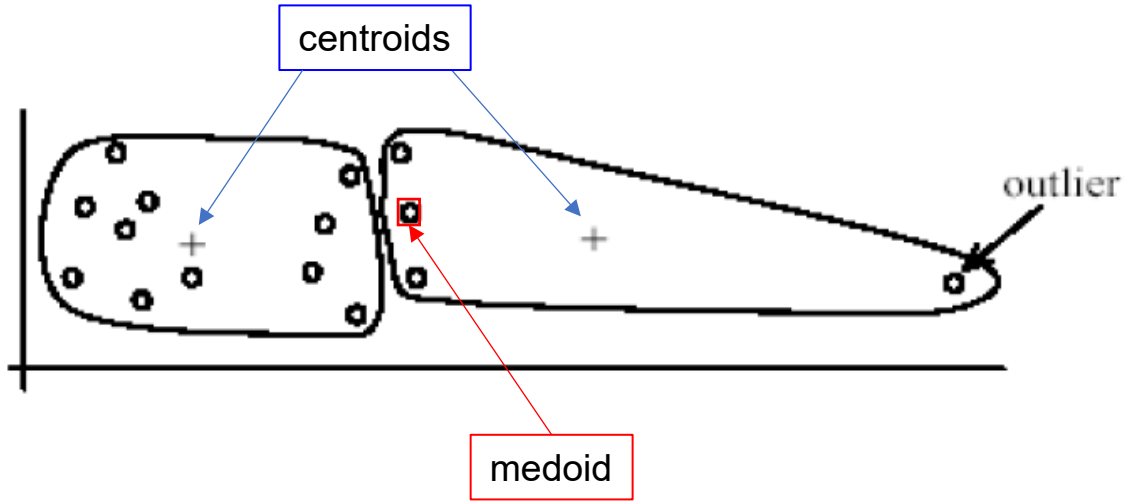
k-Medoids

- Almost the same as *k*-means
- Difference
  - *k*-means: centers are calculated (centroids)
  - *k*-medoids: centers (medoids) are selected from samples
    - Data point that minimizes the sum distances to all other points in the cluster
  - Distance metric: *k*-means (L2 norm), *k*-medoids (L1 norm)

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

Outliers

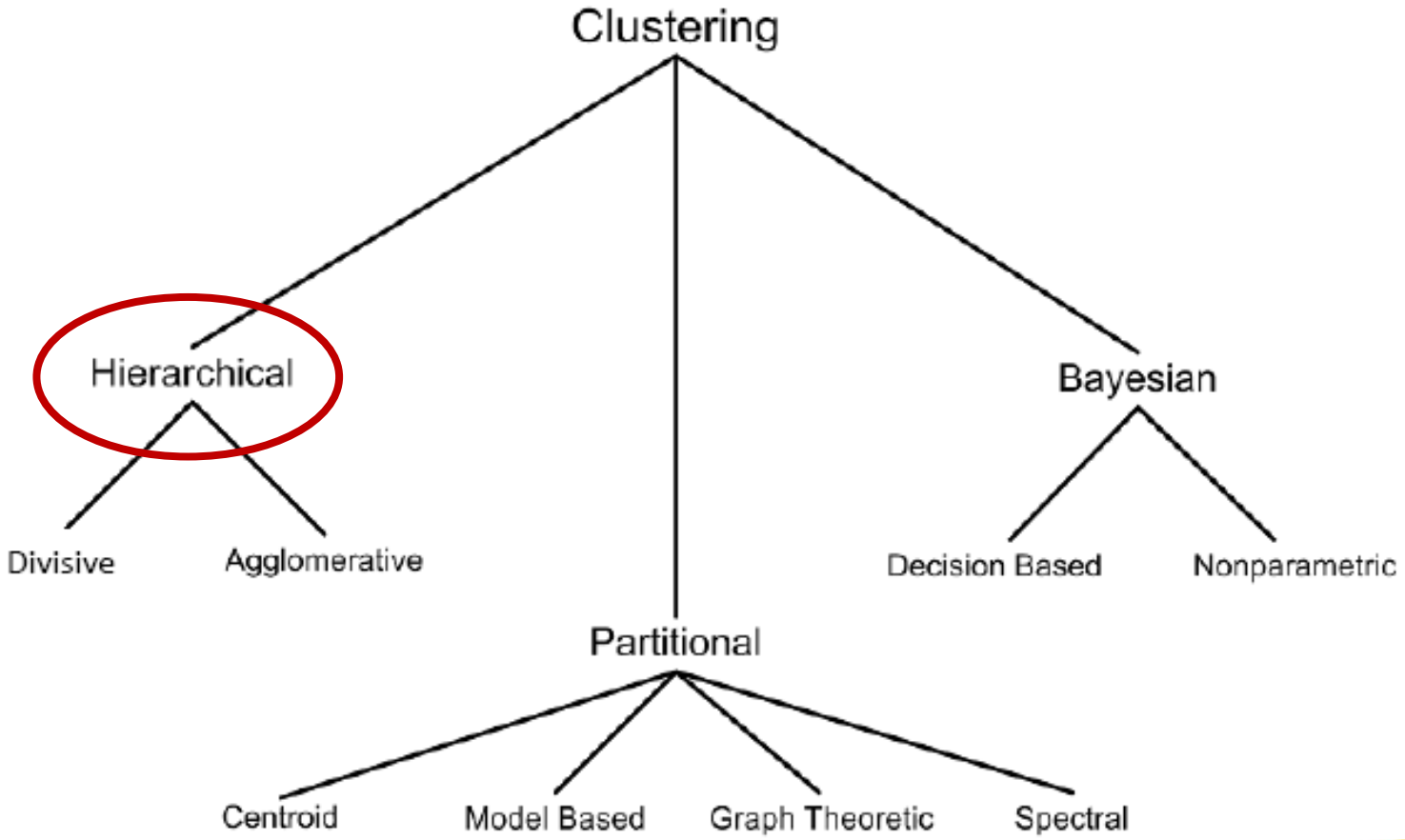
- Advantage
  - *k*-medoids is more robust to noise and outliers than *k*-means



$k = 2$

Hierarchical Clustering

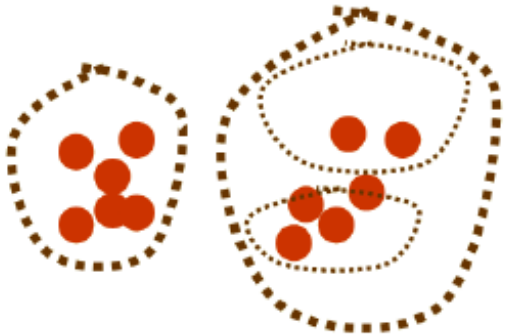
Clustering Techniques



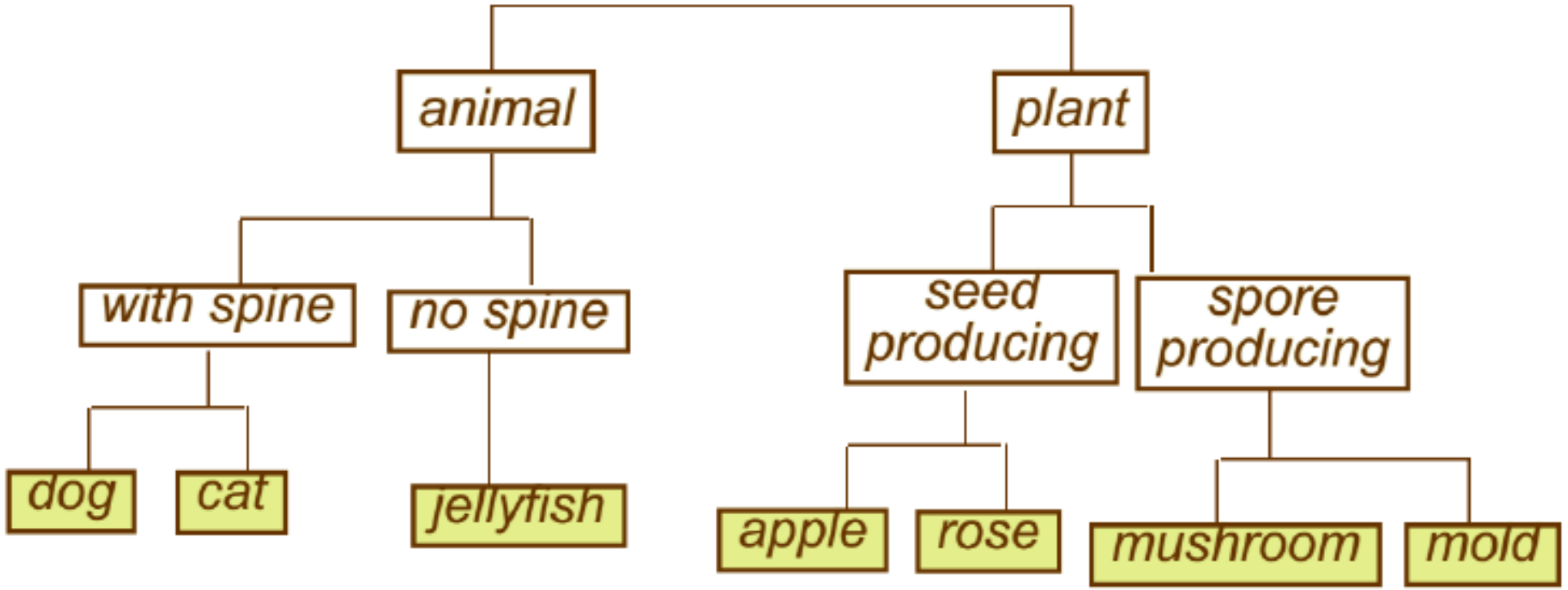
Hierarchical Clustering

- Up to now, we considered “flat” clustering:

- For some data, hierarchical clustering is more appropriate than “flat” clustering
- Hierarchical clustering



Example: Biological Taxonomy



Dendrogram

- Preferred way to represent hierarchical clustering (a tree diagram)

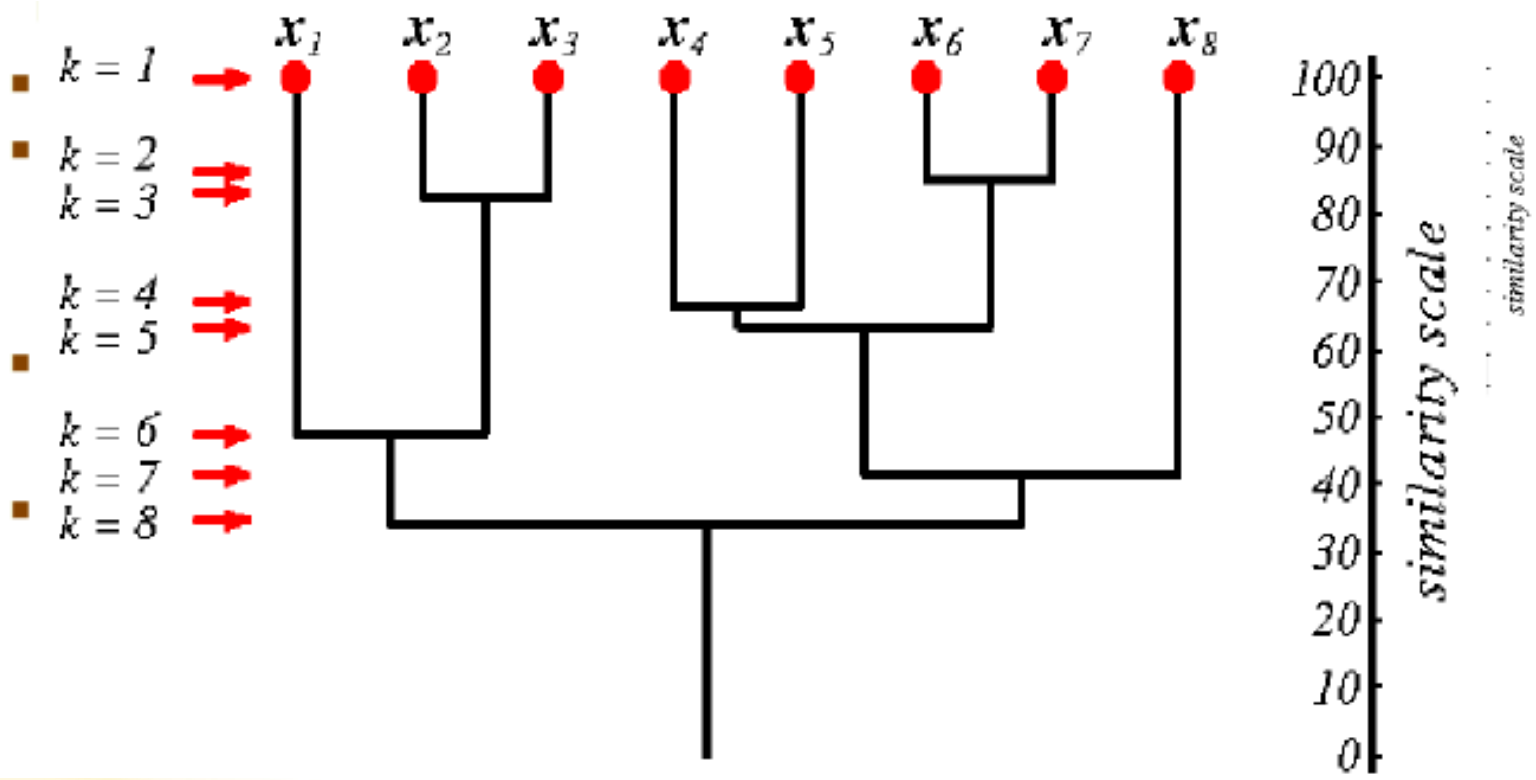
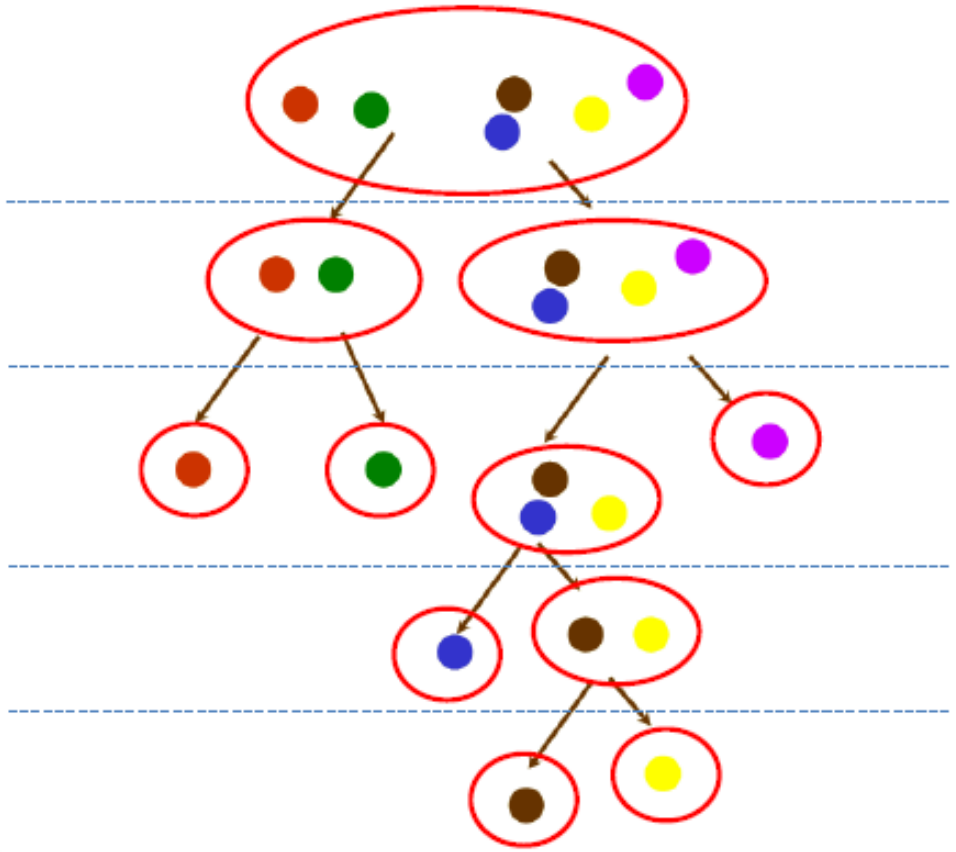


Diagram that shows the arrangements of the clusters

Types of Hierarchical Clustering

- **Divisive (top down) clustering**
  - Start with all data points in one cluster, the root, then
    - Split the root into a set of child clusters
    - Each child cluster is recursively divided further
    - Stop when only singleton clusters of individual data points remain, i.e., each cluster with only a single point
- **Agglomerative (bottom up) clustering**
  - The dendrogram is built from the bottom level by
    - merging the most similar (or nearest) pair of clusters
    - stopping when all the data points are merged into a single cluster (i.e., the root cluster).

Divisive Hierarchical Clustering



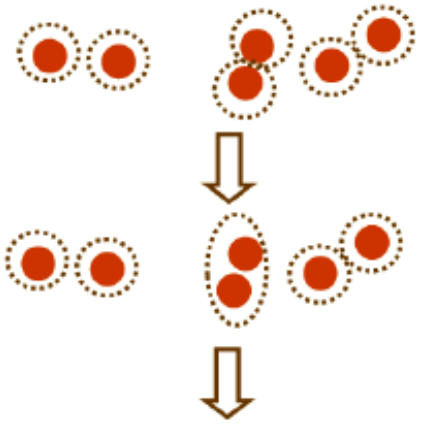
- Any “flat” algorithm which produces a fixed number of clusters can be used
  - Set c=2

Agglomerative Hierarchical Clustering

initialize with each example in singleton cluster

while there is more than ***1*** cluster

1. find 2 nearest clusters
2. merge them



Four common ways to measure cluster distance

1. minimum distance $d_{\min}(D_i, D_j) = \min_{x \in D_i, y \in D_j} \| x - y \|$
2. maximum distance $d_{\max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \| x - y \|$
3. average distance $d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{x \in D_i} \sum_{y \in D_j} \| x - y \|$
4. mean distance $d_{mean}(D_i, D_j) = \| \mu_i - \mu_j \|$

Divisive vs. Agglomerative

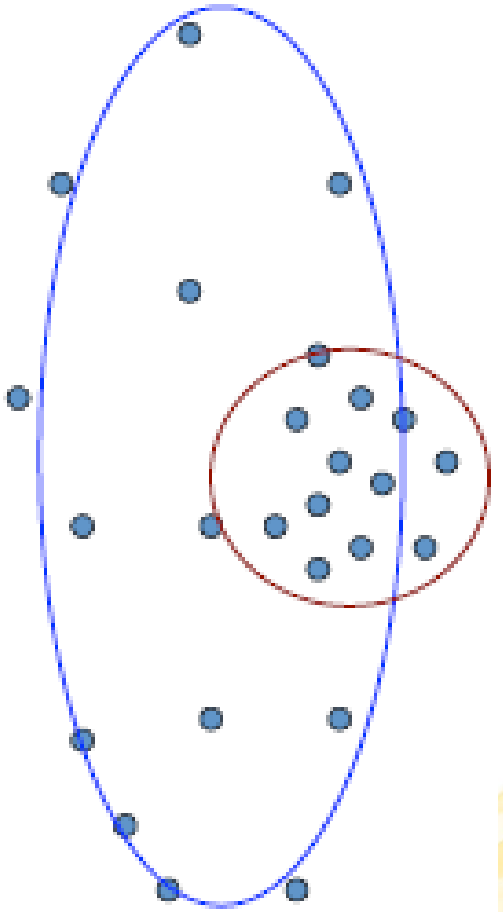
- **Agglomerative** clustering is faster to compute, in general
- **Divisive** clustering may be less “blind” to the global structure of the data

Divisive	***Agglomerative***
when taking the first step (split), have access to all the data; can find the best possible split in 2 parts	when taking the first step merging, do not consider the global structure of the data, only look at pairwise structure
	

Gaussian Mixture Model

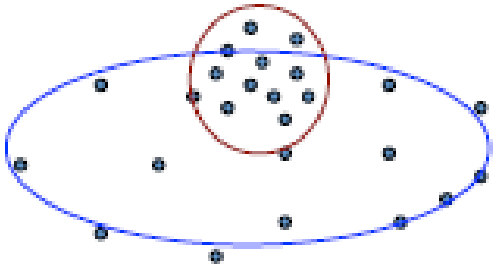
The Evils of “Hard Assignments”

- Clusters may overlap
- Some clusters may be “wider” than others
- Distances can be deceiving!
- Aggregate model much richer than component models (for individual clusters)



Probabilistic Clustering

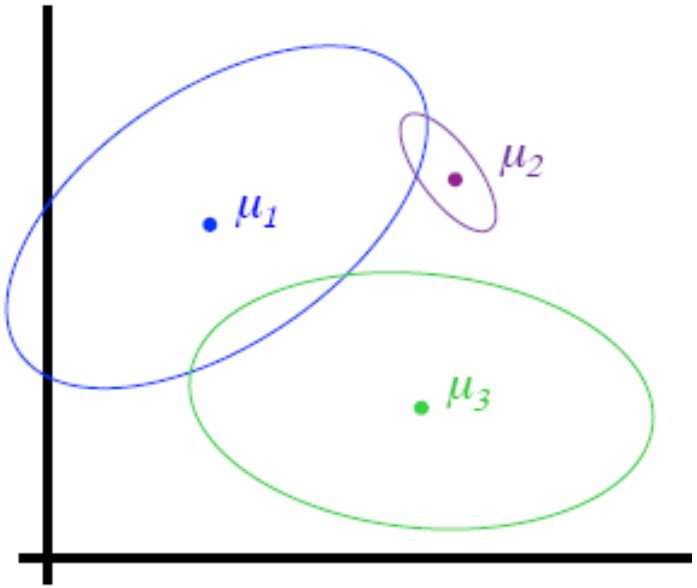
- Try a **probabilistic model**!
  - Soft clustering
  - Allows overlaps, clusters of different sizes, etc.
- Can tell a generative story for data
  - $\mathbf{P}(X, Y) = \mathbf{P}(X \mid Y)\mathbf{P}(Y)$
- **Challenge**: We need to estimate model parameters **without labels**



Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
...	...	...

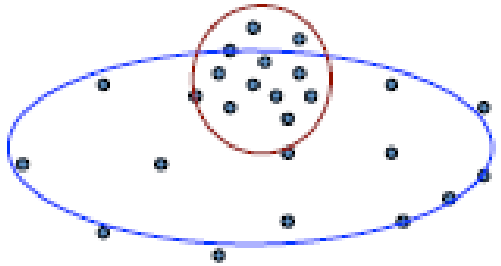
General Gaussian Mixture Model Assumption

- $\mathbf{P}(\mathrm{Y})$: There are *K* components (*K* > 1)
- $\mathbf{P}(\mathrm{X}|\mathrm{Y})$: Each component *k* generates data from a multivariate Gaussian with mean $\boldsymbol{\mu}_k$ and covariance matrix Σ_k
- Each data point is sampled from a two-step generative process (assuming known parameters):
  1. Choose component k with probability $\mathbf{P}(\mathrm{Y} = k)$
  2. Generate datapoint from distribution $N(\mu_k, \Sigma_k)$



Gaussian mixture model (GMM)

What Model Should We Use?



- In general, choice of model depends on X!
- Here, maybe Gaussian Naïve Bayes?
  - Numbers of samples from different components given by a multinomial random variable
  - Given fixed components, X_i (independent) Gaussian

Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
...	...	...

Component probability: $\mathbf{P}(Y_i = k) = \pi_k, \ k = 1, \ldots, K$

Conditional PDF (scalar case):

$$p(x \mid Y = k) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right)$$

conditional probability density

Could We Make Fewer Assumptions?

- Vector case – multivariate Gaussian distribution

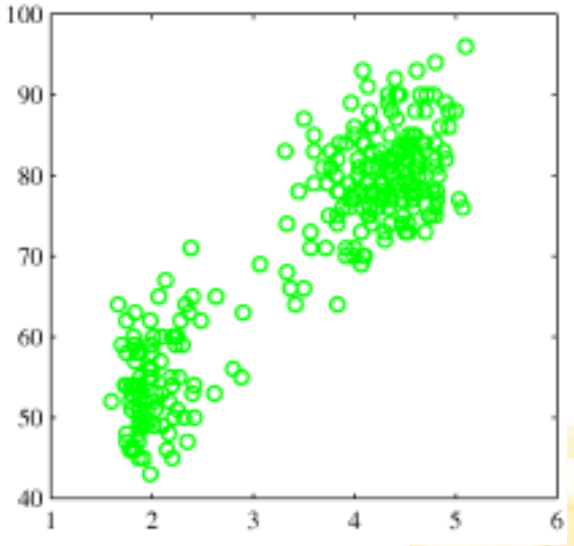
Conditional PDF (vector case):

m = dimension of $\mathbf{x}_j$

$$p(\mathbf{x}_j \mid Y = k) = \frac{1}{(2\pi)^{m/2}|\Sigma_k|^{0.5}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_k)\right),$$

$$\mathbf{x}_j \in \mathbb{R}^m$$

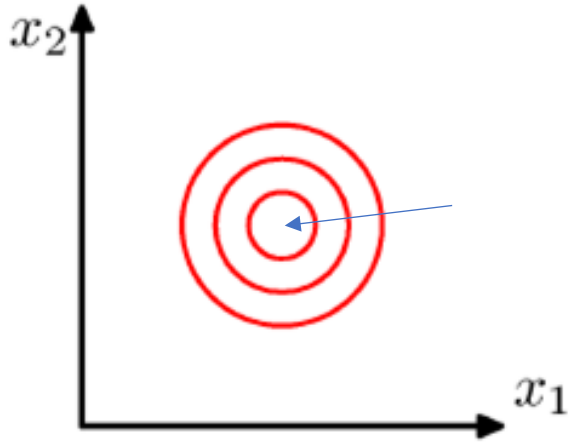
conditional probability density

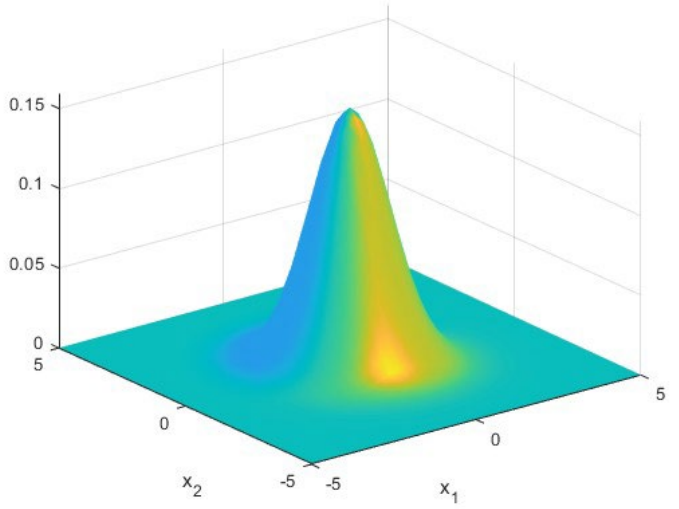


Multivariate Gaussians (1/4)

$$X \sim N(\boldsymbol{\mu}, \Sigma): \quad p(\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{0.5}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}_j - \boldsymbol{\mu})\right), \qquad |\Sigma| = \det(\Sigma)$$

m = dimension of $\mathbf{x}$

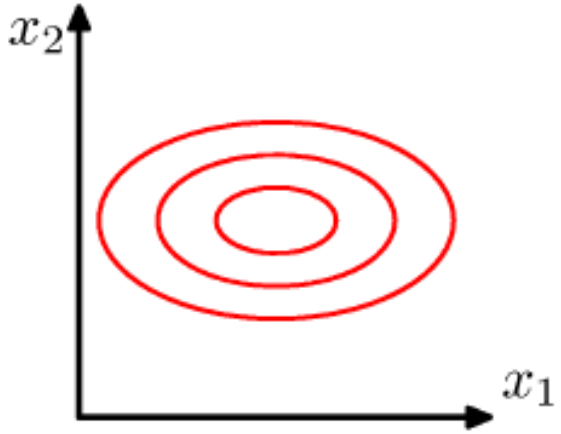


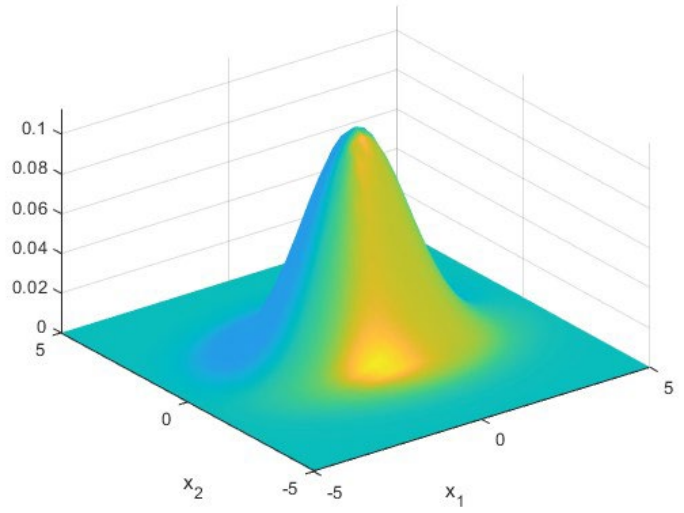


$\Sigma \propto$ identity matrix

Multivariate Gaussians (2/4)

$$p(\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{0.5}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}_j - \boldsymbol{\mu})\right)$$



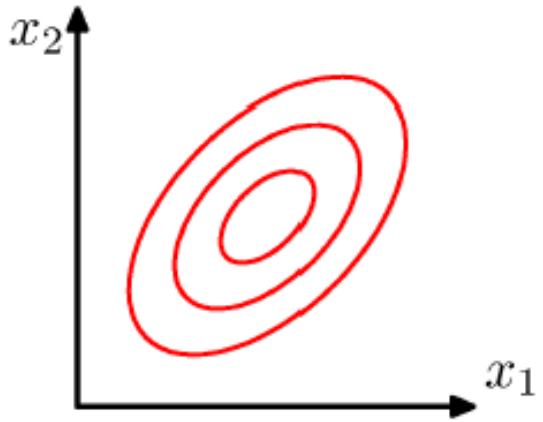


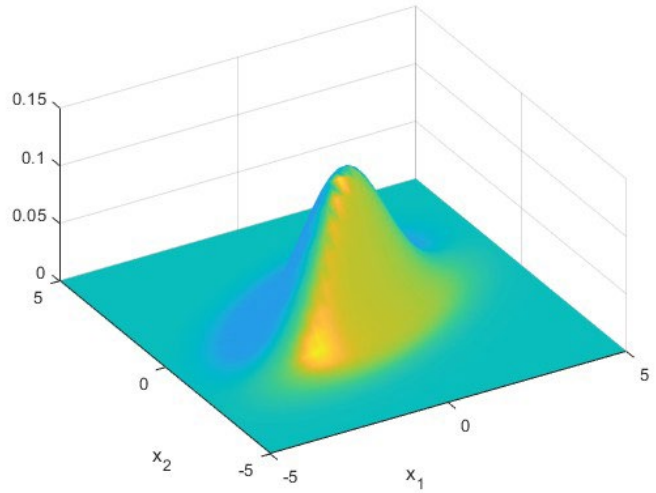
Σ = diagonal matrix

X_i are independent *ala* Gaussian NB

Multivariate Gaussians (3/4)

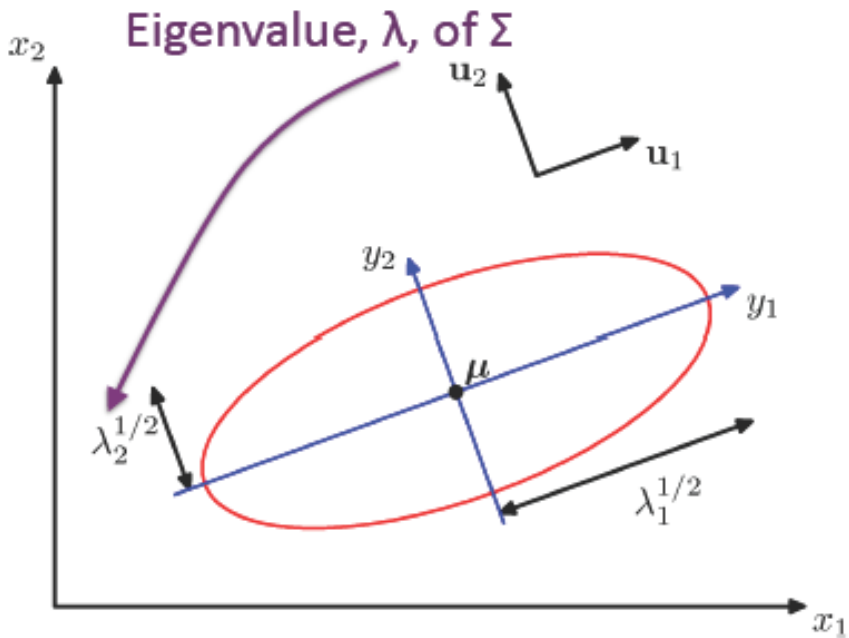
$$p(\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{0.5}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}_j - \boldsymbol{\mu})\right) =: N(\mathbf{x}_j; \boldsymbol{\mu}, \Sigma)$$





Σ = arbitrary (semidefinite) matrix:
- specifies rotation (change of basis)
- eigenvalues specify relative elongation

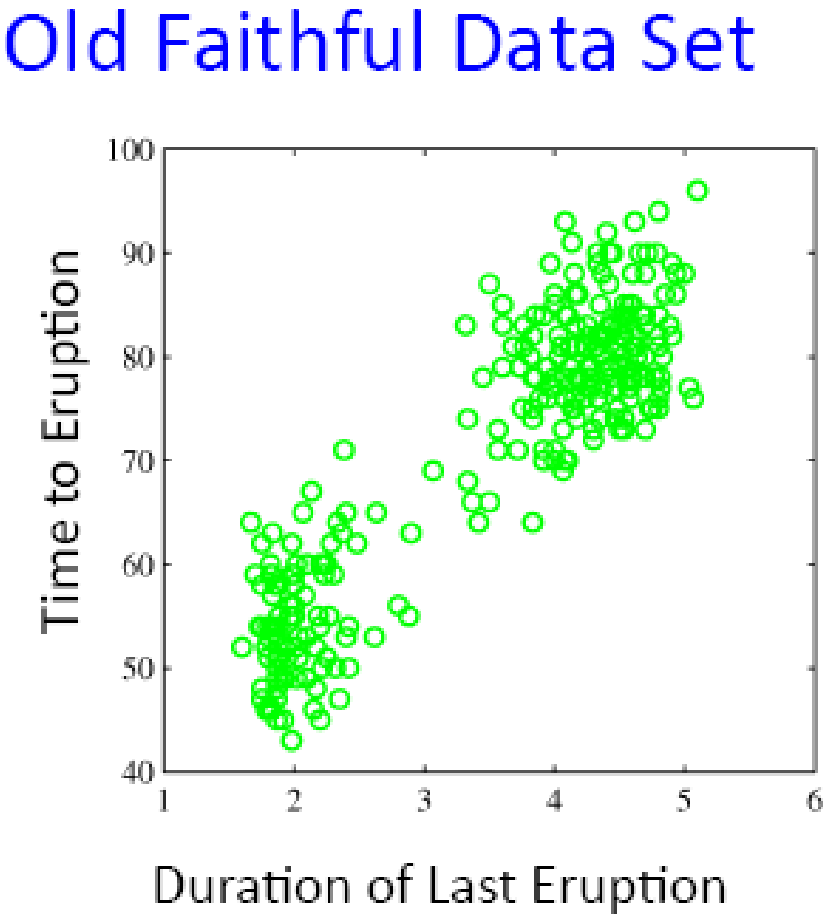
Multivariate Gaussians (4/4)



Covariance matrix, Σ, = degree to which x_i vary together

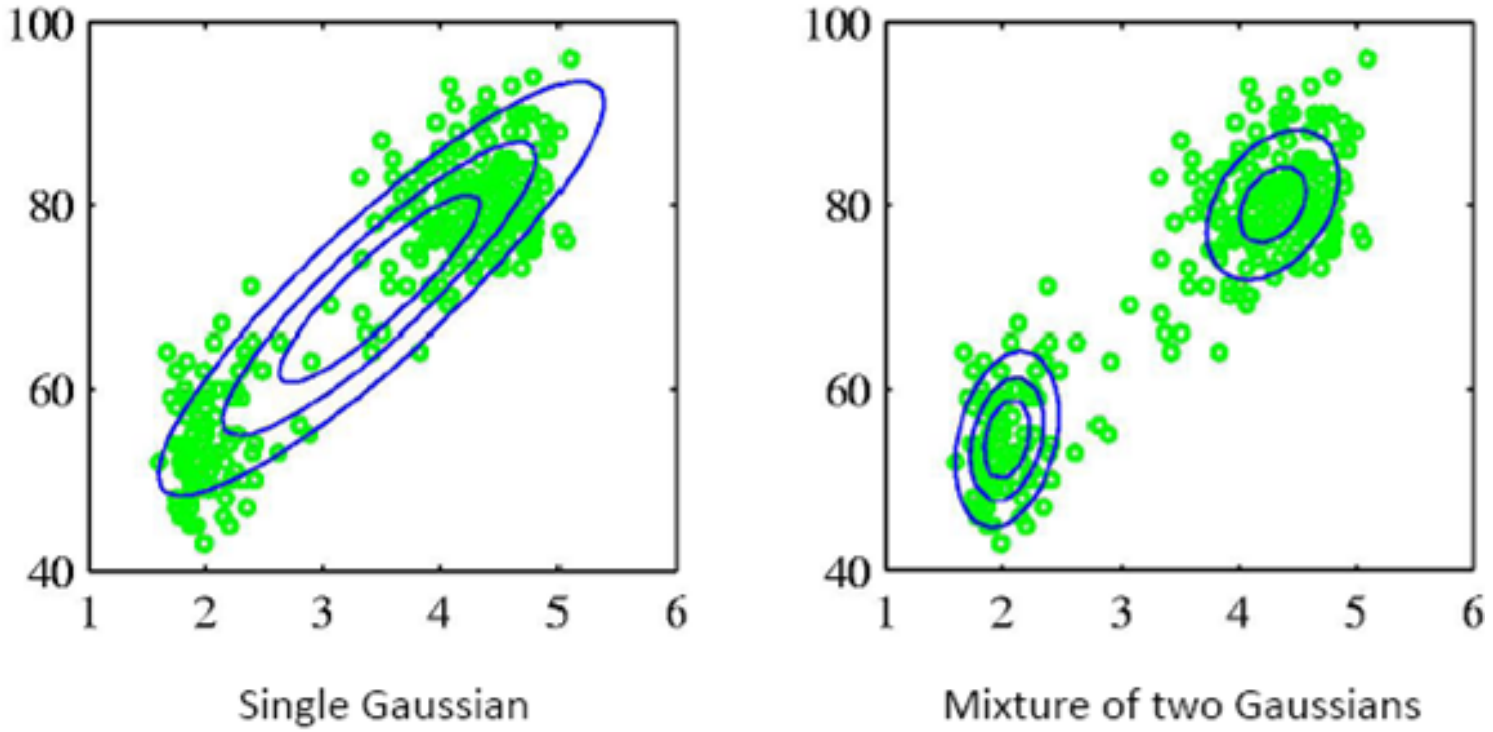
$$p(\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{0.5}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_j - \boldsymbol{\mu})\right)$$

Mixtures of Gaussians (1/4)



Mixtures of Gaussians (2/4)

Old Faithful Data Set



Mixtures of Gaussians (3/4)

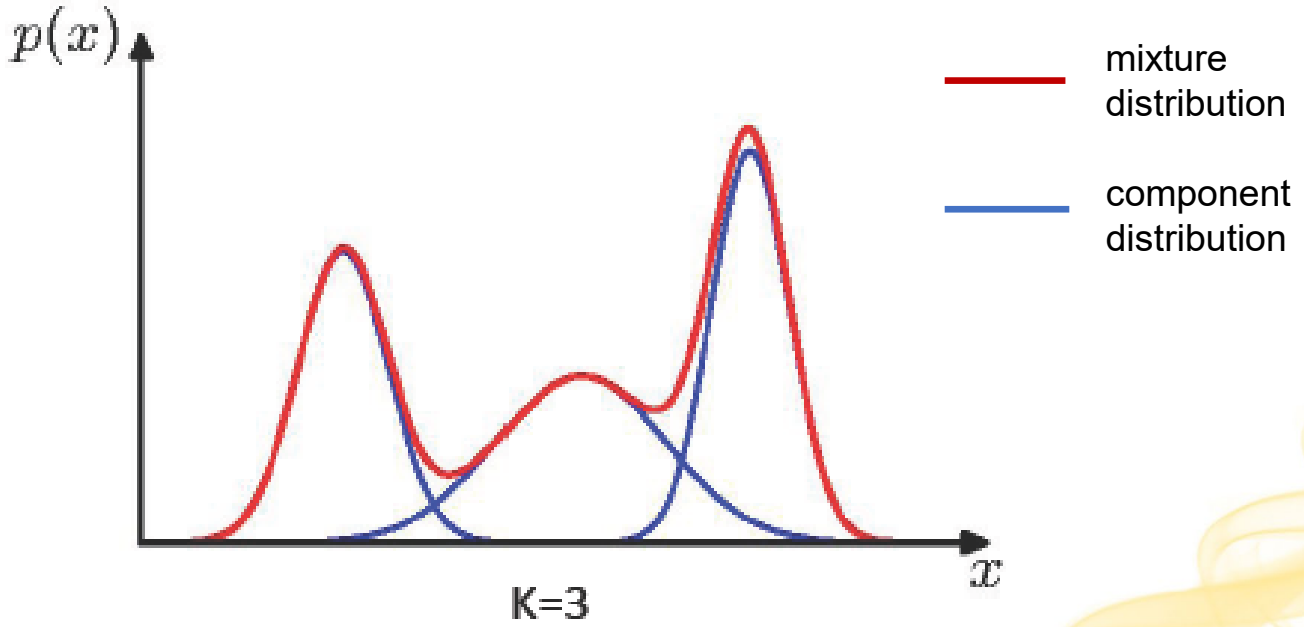
- Allows us to combine simple models into a complex model

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

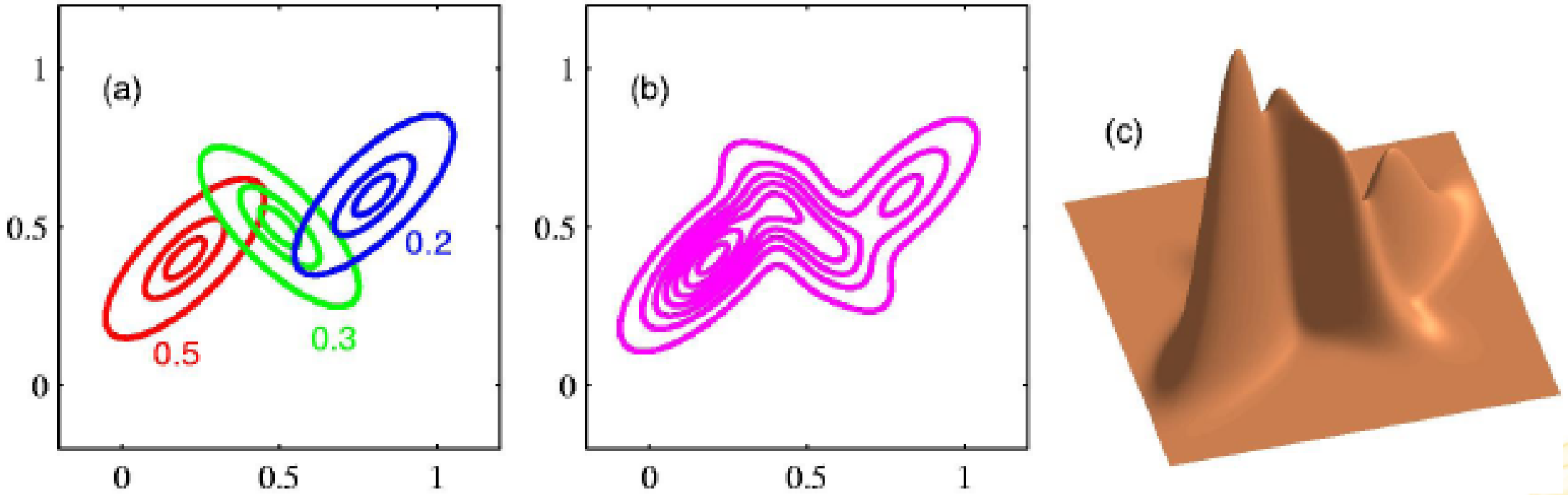
Component

Mixing coefficient

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$

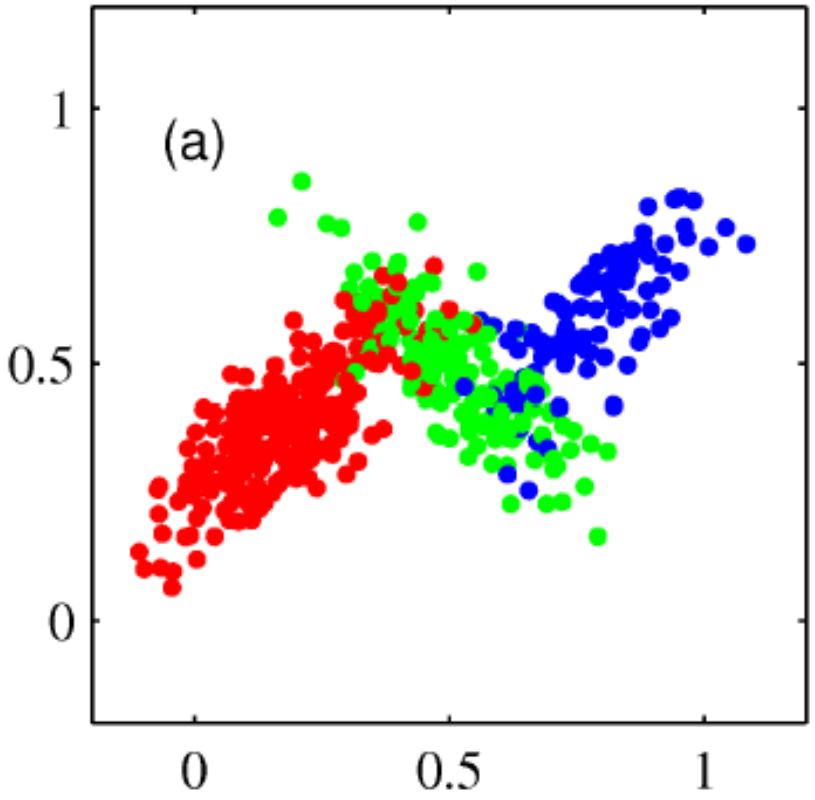


Mixtures of Gaussians (4/4)



Eliminating Hard Assignments to Clusters (1/2)

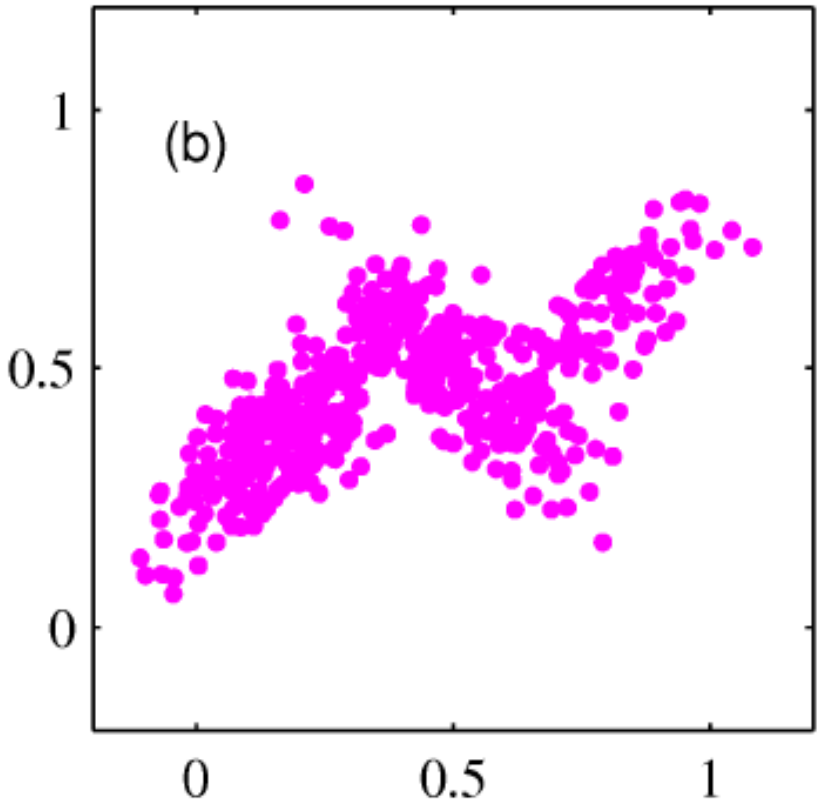
- Model data as mixture of multivariate Gaussians



(with known components)

Eliminating Hard Assignments to Clusters (2/2)

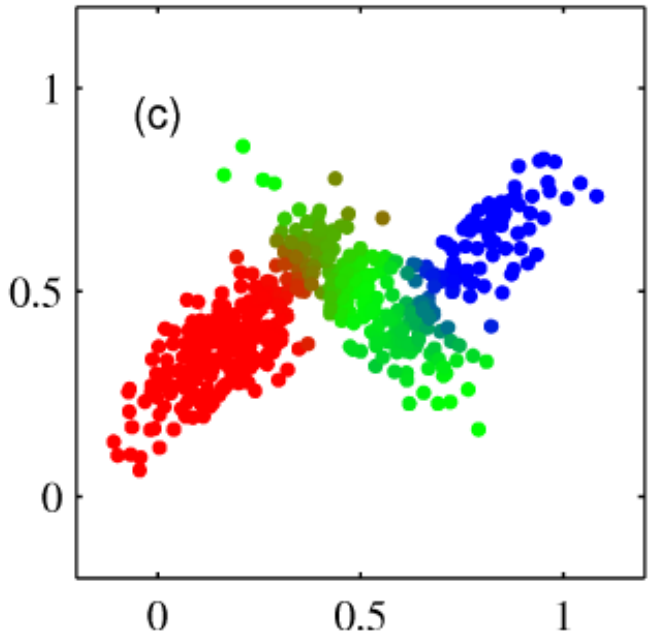
- Model data as mixture of multivariate Gaussians



(with unknown components)

Eliminating Hard Assignments to Clusters

- Model data as mixture of multivariate Gaussians



Shown is the *posterior probability* that a point was generated from ith Gaussian: $\mathbf{P}(Y = i \mid \mathbf{x})$

ML Estimation in Supervised Settings

- Recall: **Univariate Gaussian**

$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

- **Mixture of multivariate Gaussians:** for each component distribution

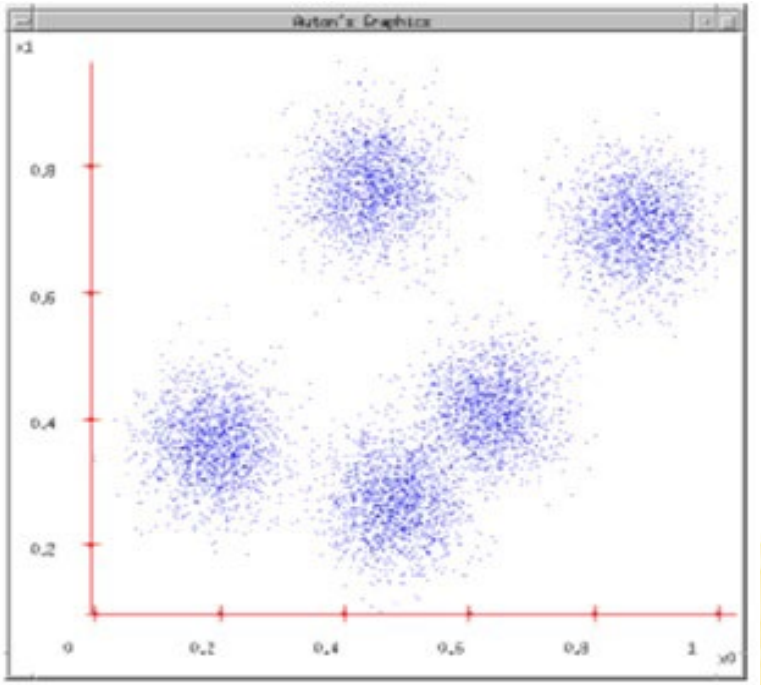
ML estimate for each of the Multivariate Gaussians is given by:

$$\mu_{ML}^k = \frac{1}{n}\sum_{j=1}^{n} x_n \qquad \Sigma_{ML}^k = \frac{1}{n}\sum_{j=1}^{n}\left(\mathbf{x}_j - \mu_{ML}^k\right)\left(\mathbf{x}_j - \mu_{ML}^k\right)^T$$

Just sums over *x* generated from the *k*'th Gaussian

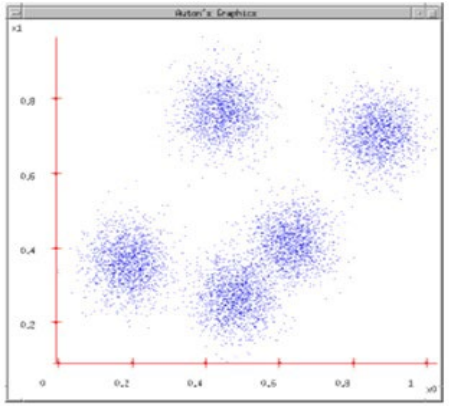
That was easy!
But what if there are unobserved data?

- MLE:
  - $\arg\max_{\boldsymbol{\theta}} \prod_{j=1}^{n} \overbrace{p(\mathbf{x}_j \mid Y_j = y_j)\mathbf{P}(Y_j = y_j)}^{=: p(\mathbf{x}_j, y_j)}$
  - $\boldsymbol{\theta} = \{(\pi_k, \boldsymbol{\mu}_k, \Sigma_k), k = 1, \ldots, K\}$: all model parameters including, class probabilities, means and variances
- Problem: We do not know y_j, i.e., which component is responsible for generating each data point
- Maximize ***marginal likelihood***



$$\arg\max_{\boldsymbol{\theta}} \prod_{j=1}^{n} p(\mathbf{x}_j) = \arg\max_{\boldsymbol{\theta}} \prod_{j=1}^{n} \left(\sum_{k=1}^{K} p(\mathbf{x}_j \mid Y_j = k)\mathbf{P}(Y_j = k) \right)$$

How do we optimize?



- Maximize ***marginal likelihood***

$$\arg\max_{\boldsymbol{\theta}} \prod_{j=1}^{n} p(\mathbf{x}_j) = \arg\max_{\boldsymbol{\theta}} \prod_{j=1}^{n} \Big(\sum_{k=1}^{K} \underbrace{p(\mathbf{x}_j \mid Y_j = k)\mathbf{P}(Y_j = k)}_{= p(\mathbf{x}_j, k)} \Big)$$

- Almost always a hard problem
  - Usually no closed form solution
  - Even when $\log(p(X, Y))$ is convex, $\log(p(X))$ generally is not
  - For all but the simplest $p(X)$, we will have to use gradient ascent, in a big messy space with many local optima

Learning General Mixtures of Gaussian

- Marginal likelihood with multivariate Gaussian:

$$\prod_{j=1}^{n} p(\mathbf{x}_j) = \prod_{j=1}^{n} \left(\sum_{k=1}^{K} \underbrace{\frac{1}{(2\pi)^{m/2}|\Sigma_k|^{0.5}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_k)\right)}_{= p(\mathbf{x}_j \mid Y_j = k)} \cdot \mathbf{P}(Y_j = k) \right)$$

- Need to differentiate and solve for the parameters $\boldsymbol{\theta} = \{(\pi_k, \boldsymbol{\mu}_k, \Sigma_k), k = 1, \ldots, K\}$
- There is no closed-form solution, gradient is complex, and many local optima

- ***How do we solve it then? Is there a better way?***

Expectation Maximization

- Recall (for one data point)

$$p(\mathbf{x}_i) = \sum_{k=1}^{K} \underbrace{\frac{1}{(2\pi)^{m/2}|\Sigma_k|^{0.5}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right)}_{= N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)} \cdot \underbrace{\mathbf{P}(Y_i = k)}_{= \pi_k}$$

$$= \sum_{k=1}^{K} \pi_k \cdot N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)$$

- Log-likelihood: $\ell(\mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\theta}) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k \cdot N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)\right)$
- Differentiate w.r.t. each parameter and set the derivative to 0 (assume scalar instead of vector for simplicity)

$$E.g., \text{Derivative w.r.t. } \mu_l: \quad \sum_{i=1}^{n} \frac{1}{\sum_{k=1}^{K} \pi_k \cdot N(x_i; \boldsymbol{\mu}_k, \Sigma_k)} \pi_l N(x_i; \mu_l, \sigma_l^2) \frac{(x_i - \mu_l)}{\sigma_l^2} = 0$$

  - Cannot analytically solve for μ_l

Expectation Maximization

- Key idea: Break the optimization w.r.t. parameters into two steps and alternate between them

1. Expectation (E) step: Evaluate the ***posterior probabilities*** using the current estimates of component Gaussian distributions and mixing components $\{(\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k), k = 1, \ldots, K\}$

$$\mathbf{P}(Y_i = l \mid X_i = \mathbf{x}_i) = \frac{p(\mathbf{x}_i \mid Y_i = l)\mathbf{P}(Y_i = l)}{p(\mathbf{x}_i)} = \frac{\hat{\pi}_l \cdot N(\hat{\boldsymbol{\mu}}_l, \hat{\Sigma}_l)}{\sum_{k=1}^{K} \hat{\pi}_k \cdot N(\hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k)} = \gamma_i(l), \;\; l = 1, \ldots, K$$

2. Maximization (M) step: Update new parameters $\hat{\boldsymbol{\theta}} = \{(\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k), k = 1, \ldots, K\}$ with fixed posterior probabilities

$$N_l = \sum_{i=1}^{n} \gamma_i(l), \;\; \hat{\boldsymbol{\mu}}_l = \frac{1}{N_l} \sum_{i=1}^{n} \gamma_i(l) \; \mathbf{x}_i,$$

$$\hat{\Sigma}_l = \frac{1}{N_l} \sum_{i=1}^{n} \gamma_i(l)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)^T \quad \text{and} \quad \hat{\pi}_l = \frac{N_l}{n}$$

N_l = effective # of data points assigned to component l

Expectation Maximization

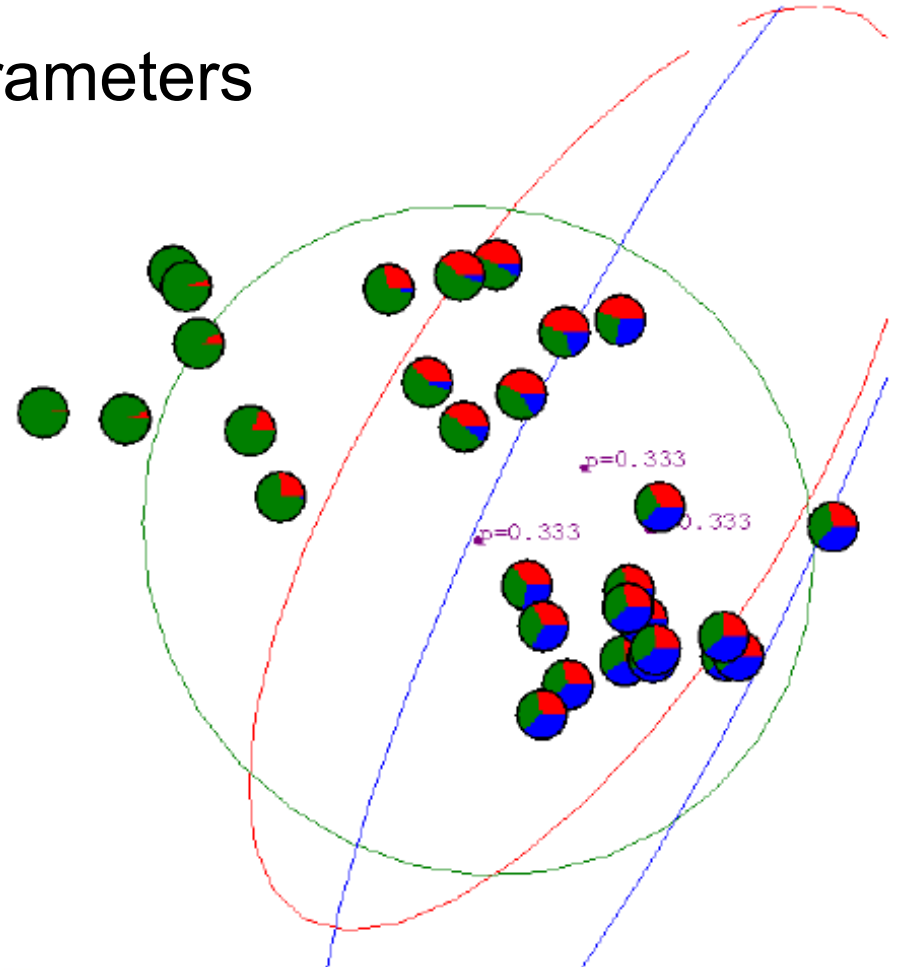
Expectation-Maximization (EM) algorithm

1. Initialize parameters $\hat{\boldsymbol{\theta}} = \{(\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k), k = 1, \ldots, K\}$ and evaluate the log-likelihood
2. Expectation (E) step: Evaluate the ***posterior probabilities*** $\gamma_i(l) = \mathbf{P}(Y_i = l \mid X_i = \mathbf{x}_i)$ using the current estimates of parameters
3. Maximization (M) step: Update new parameters $\hat{\boldsymbol{\theta}} = \{(\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k), k = 1, \ldots, K\}$ with fixed posterior probabilities
4. Evaluate the log-likelihood with the new parameter estimates. Stop if stopping criterion is met. Otherwise, go back to Step 2

- Useful fact: log-likelihood increases after each iteration

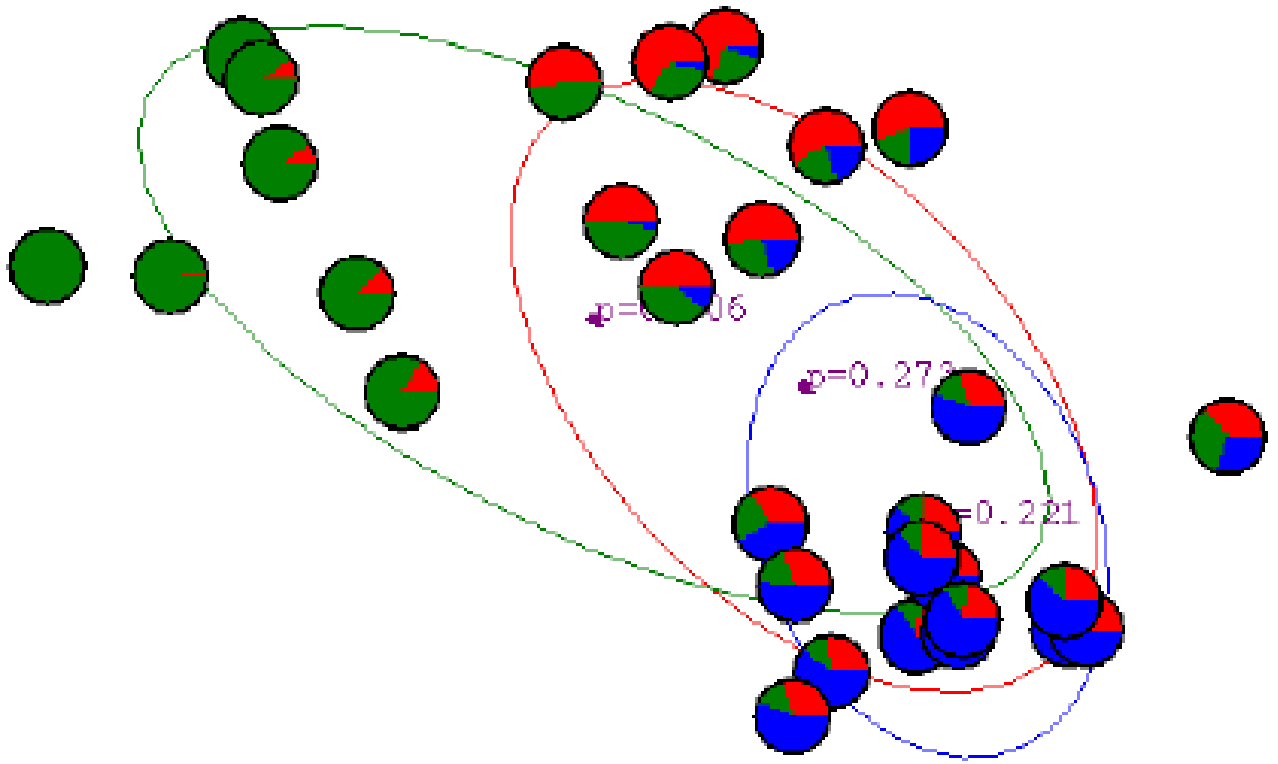
Gaussian Mixture Example (1/8)

- Start with initial parameters



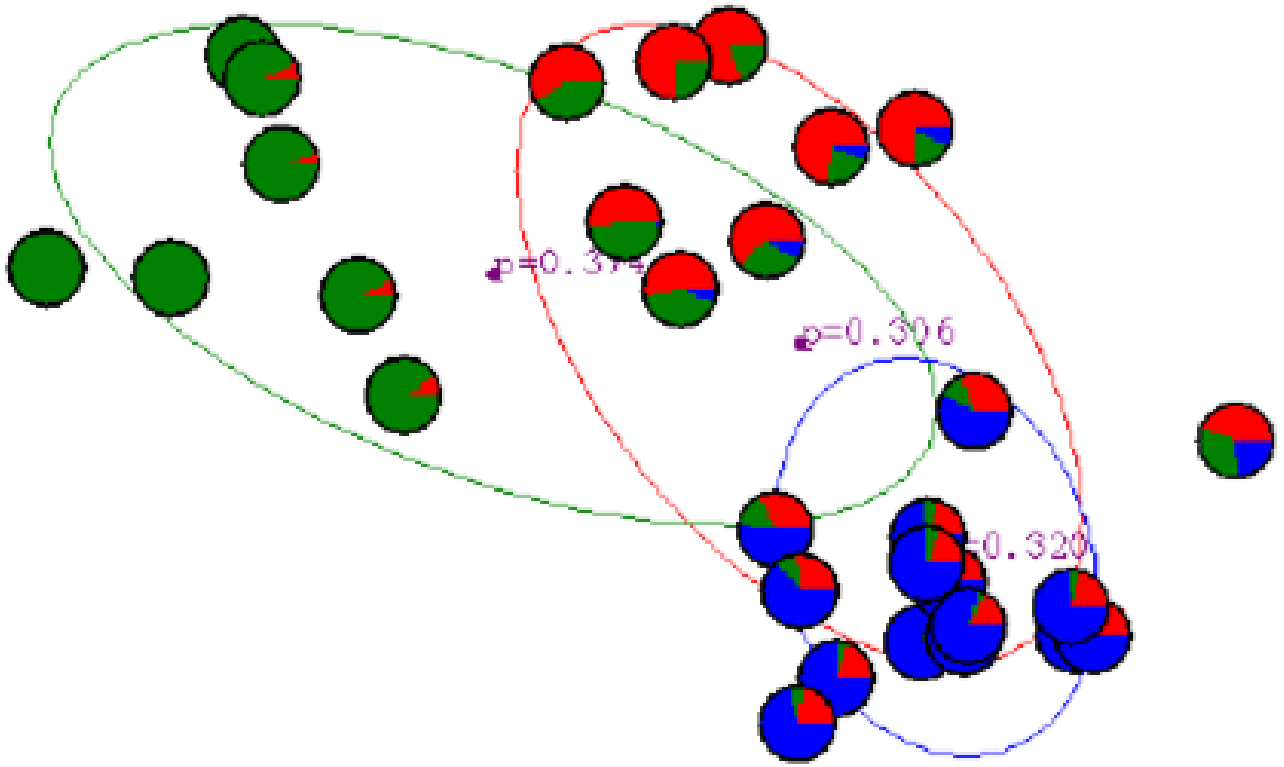
Gaussian Mixture Example (2/8)

- After first iteration



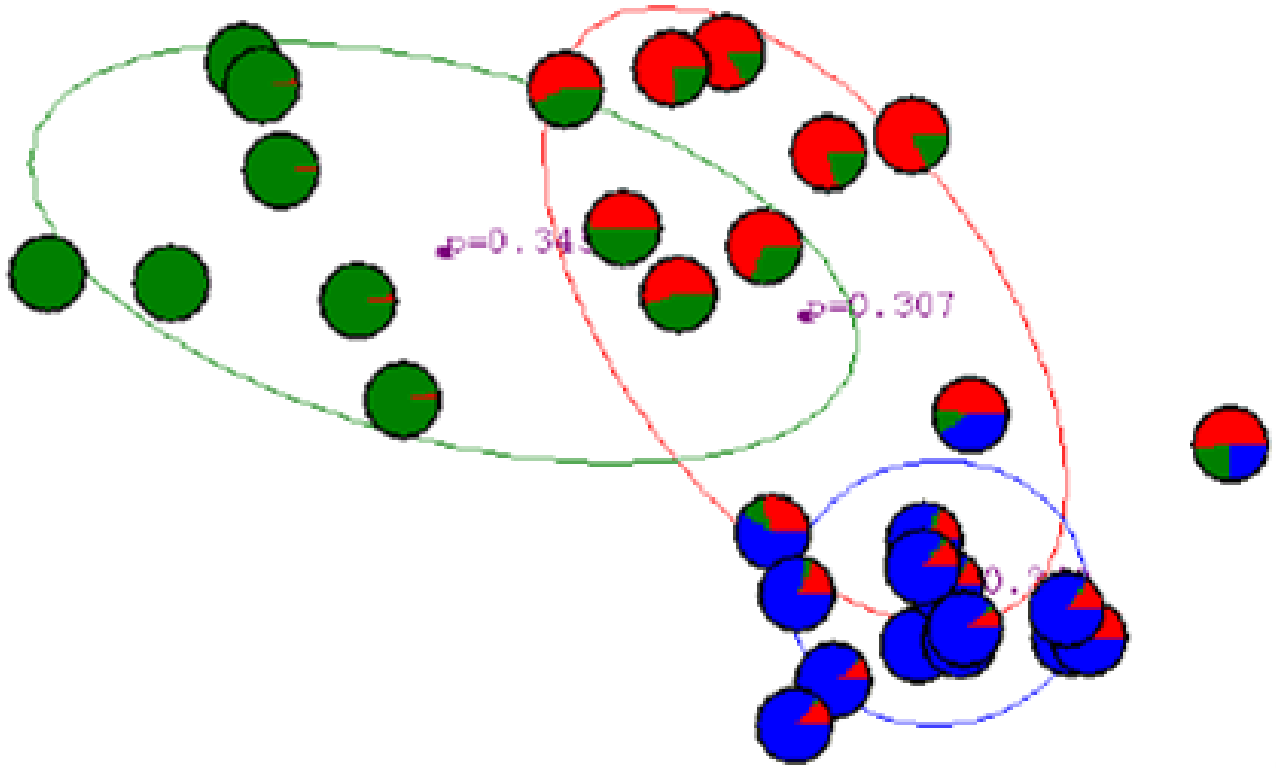
Gaussian Mixture Example (3/8)

- After 2nd iteration



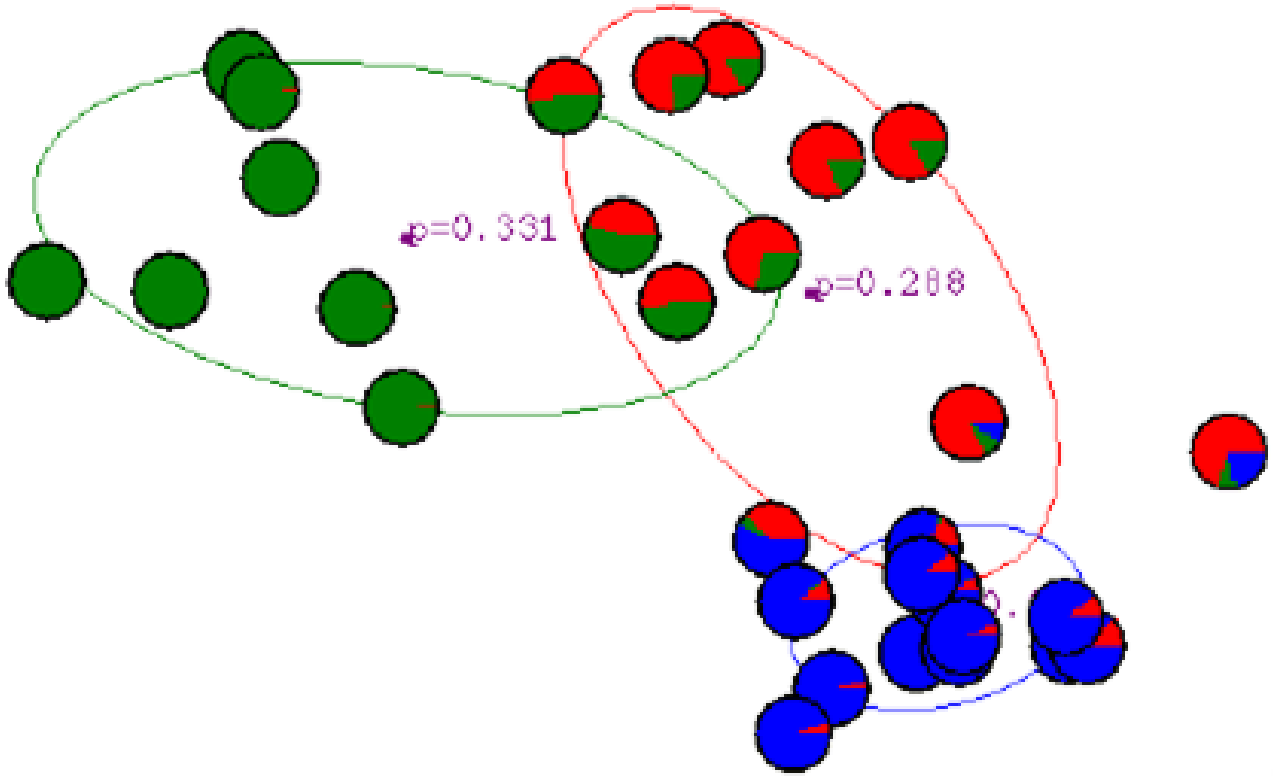
Gaussian Mixture Example (4/8)

- After 3rd iteration



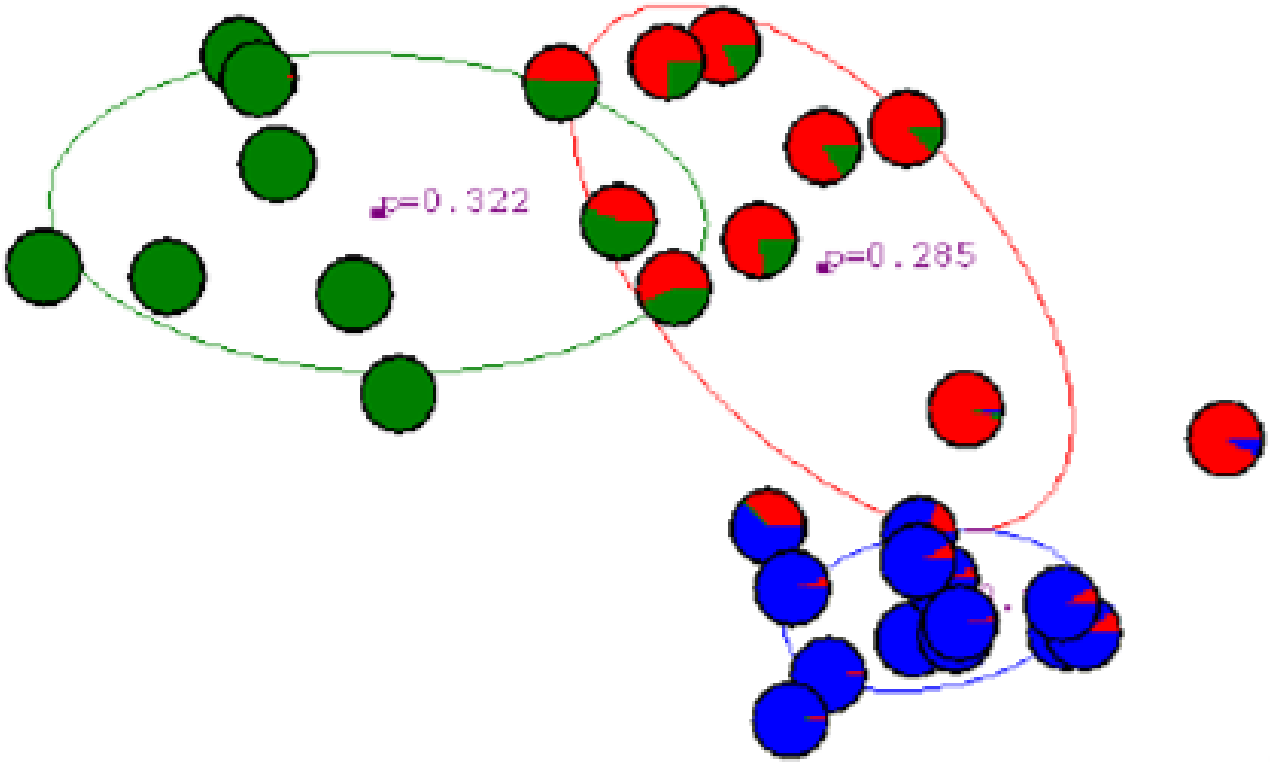
Gaussian Mixture Example (5/8)

- After 4th iteration



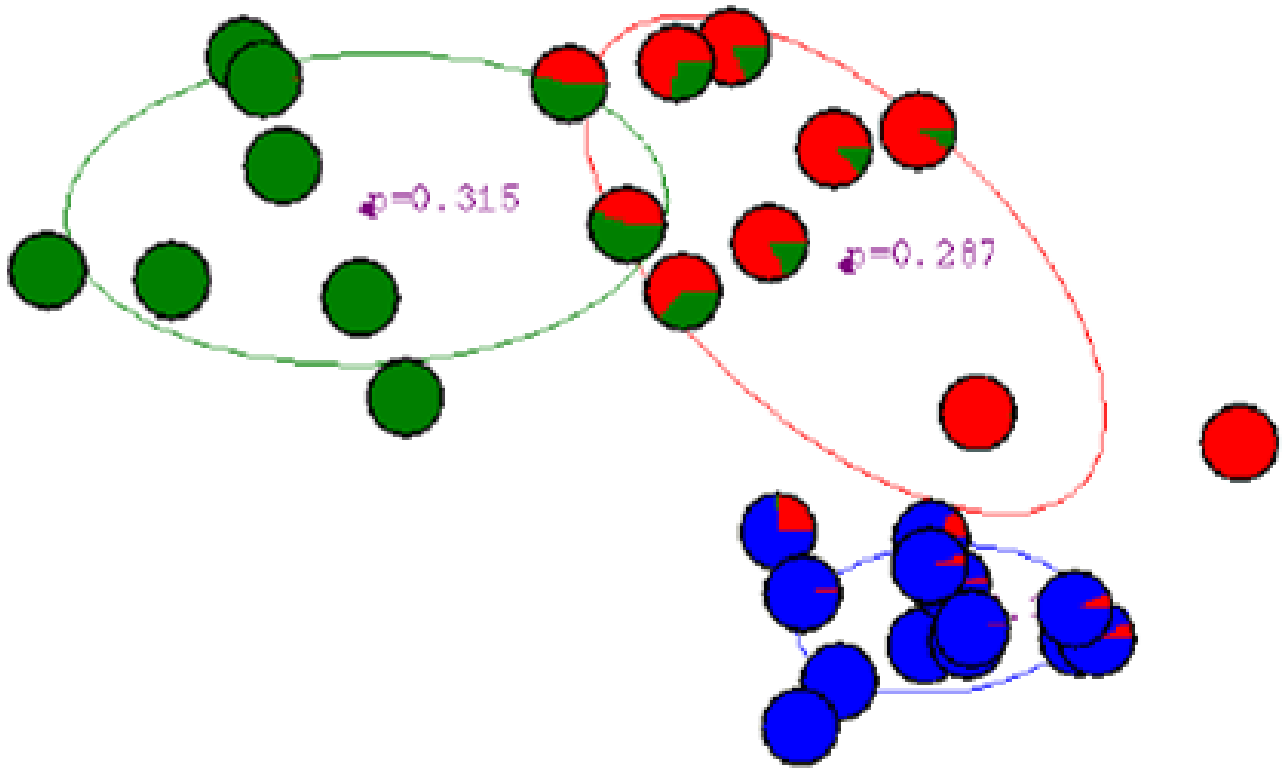
Gaussian Mixture Example (6/8)

- After 5th iteration



Gaussian Mixture Example (7/8)

- After 6th iteration



Gaussian Mixture Example (8/8)

- After 20th iteration