Machine Learning 2

DS 4420 - Spring 2018

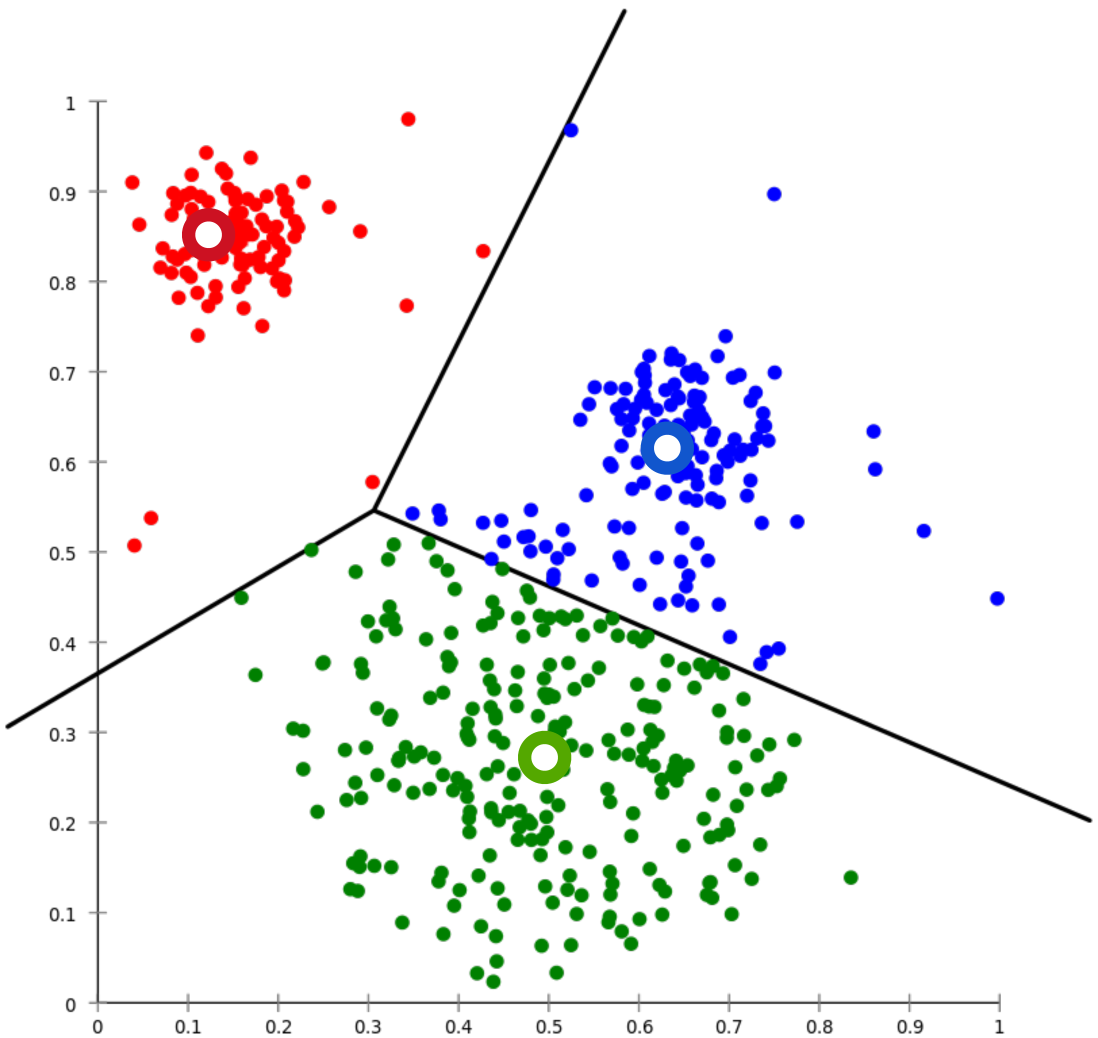
From clustering to EM

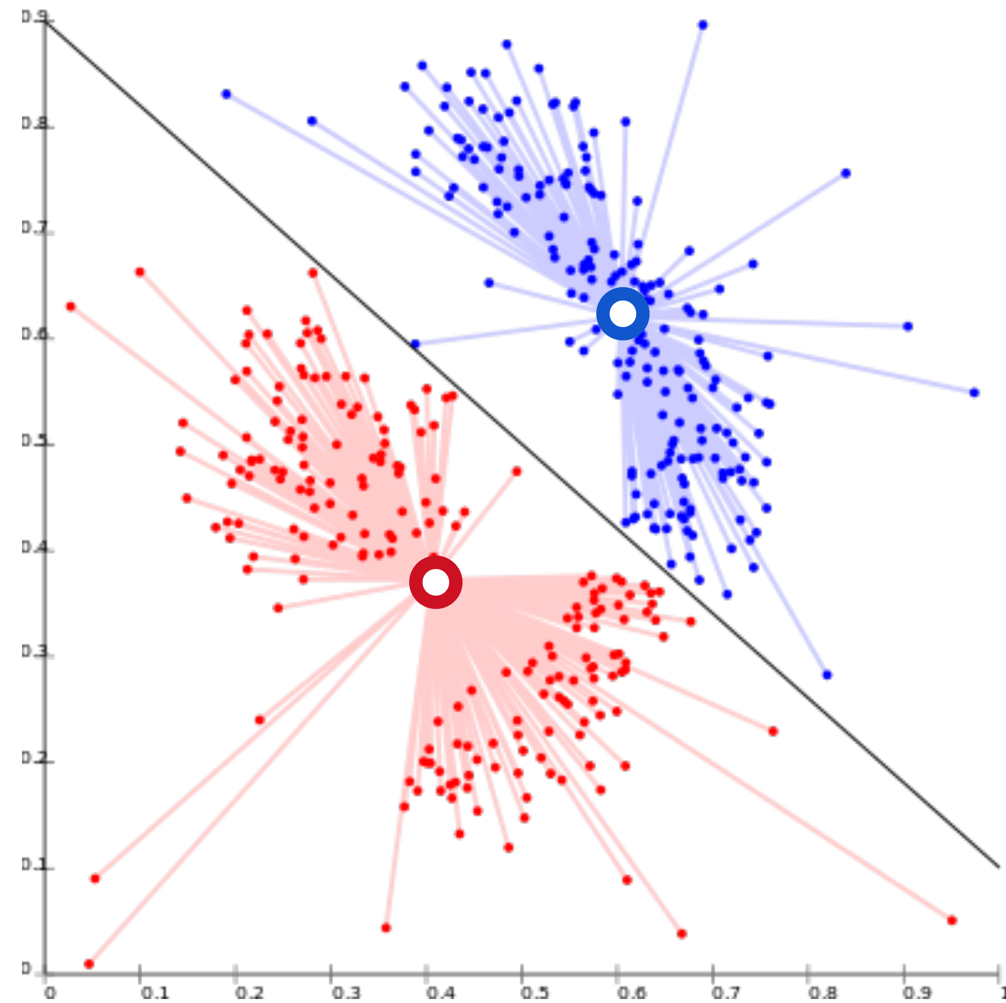
Byron C. Wallace

Clustering

Four Types of Clustering

1. *Centroid-based (K-means, K-medoids)*

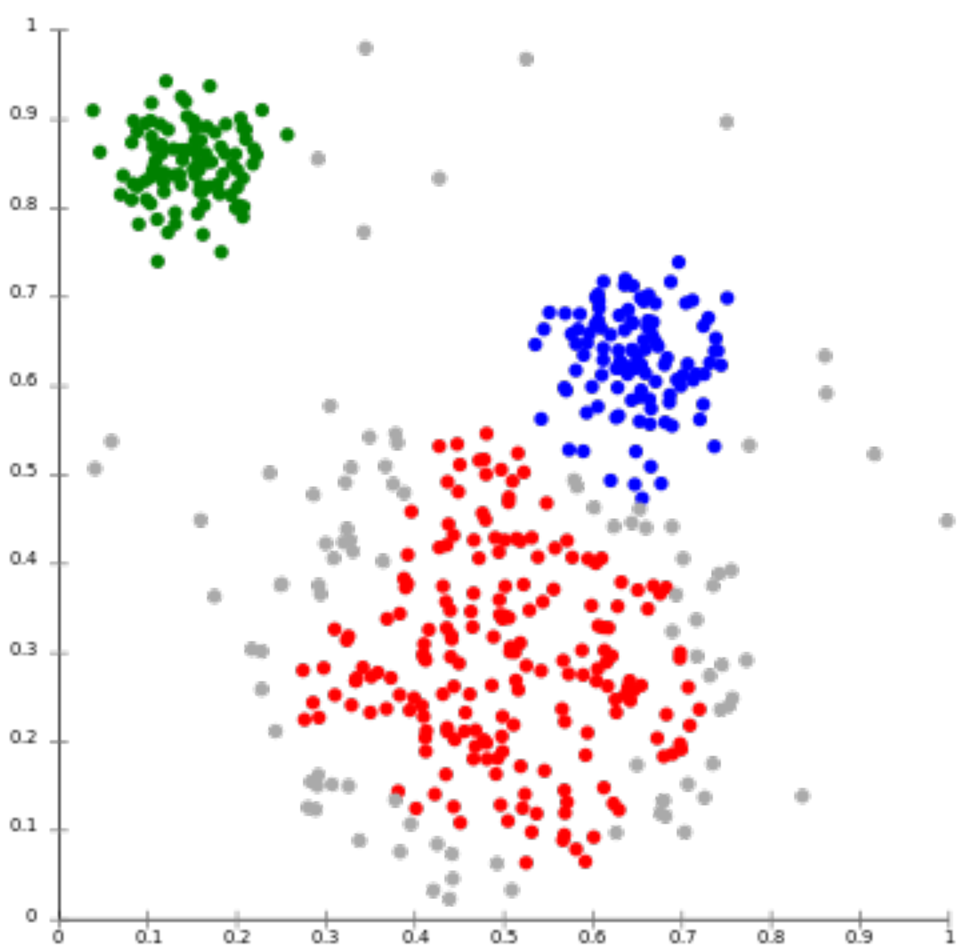


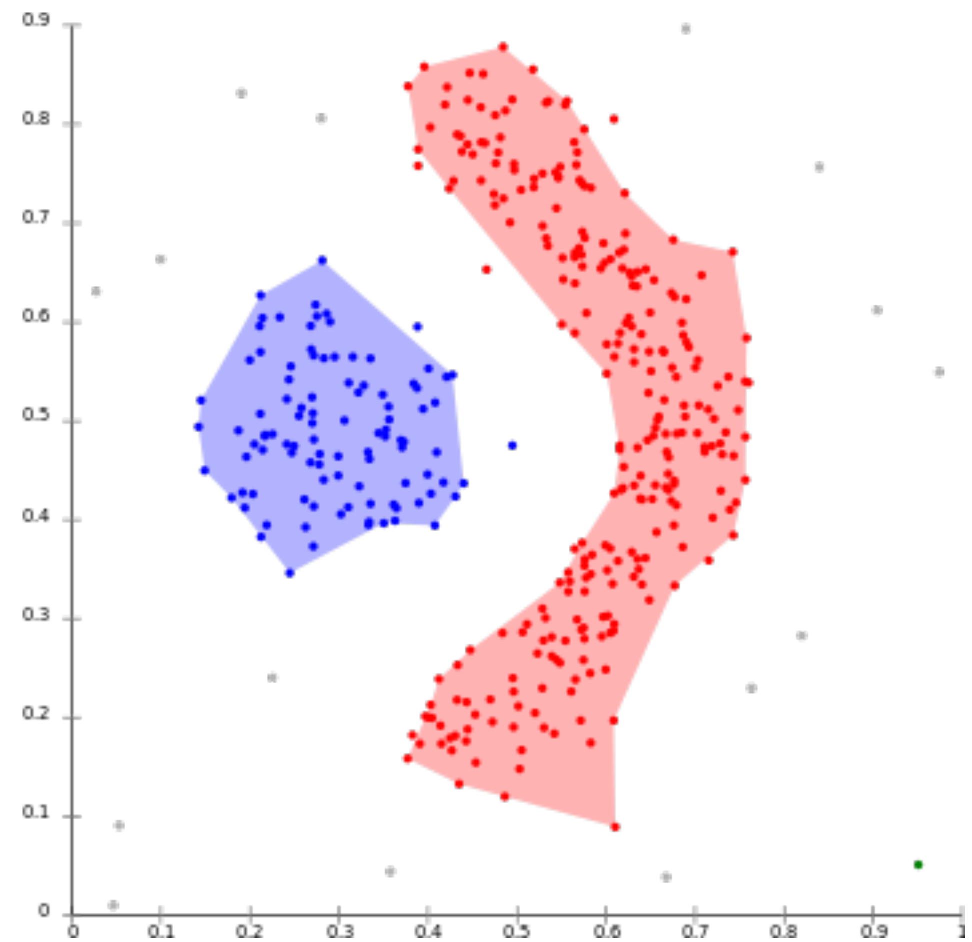


Notion of Clusters: Voronoi tesselation

Four Types of Clustering

2. *Density-based (DBSCAN, OPTICS)*

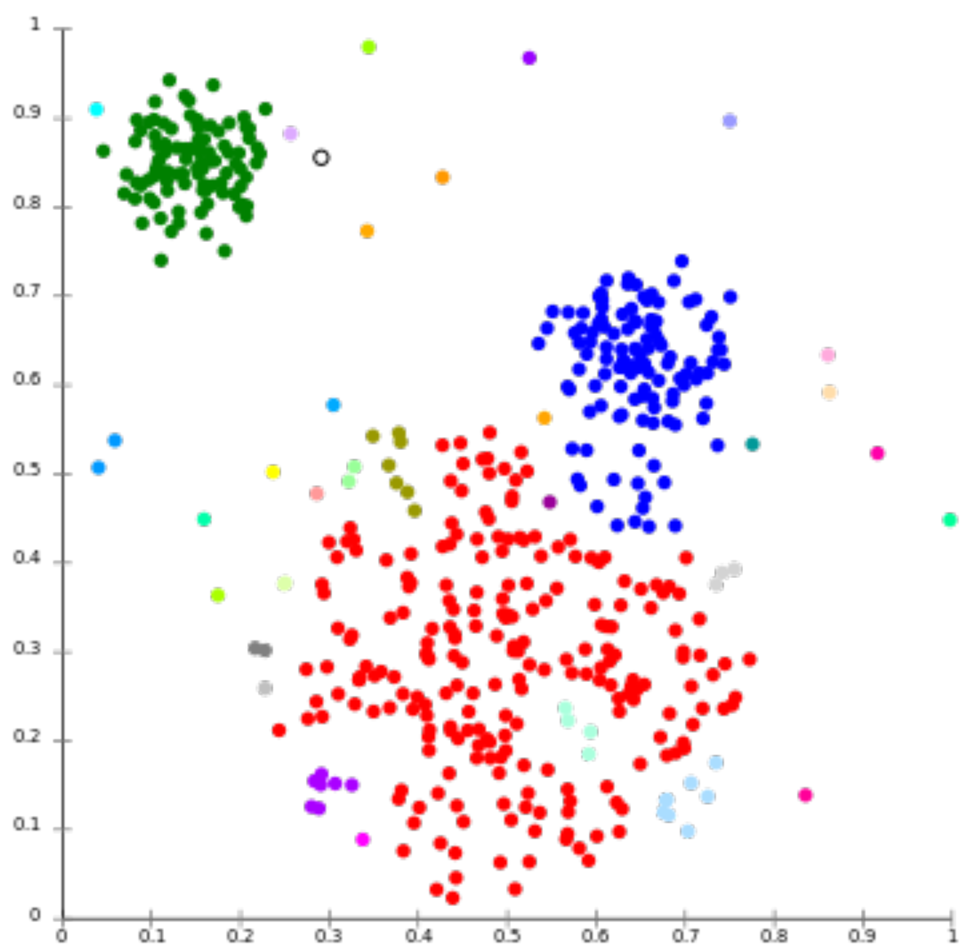


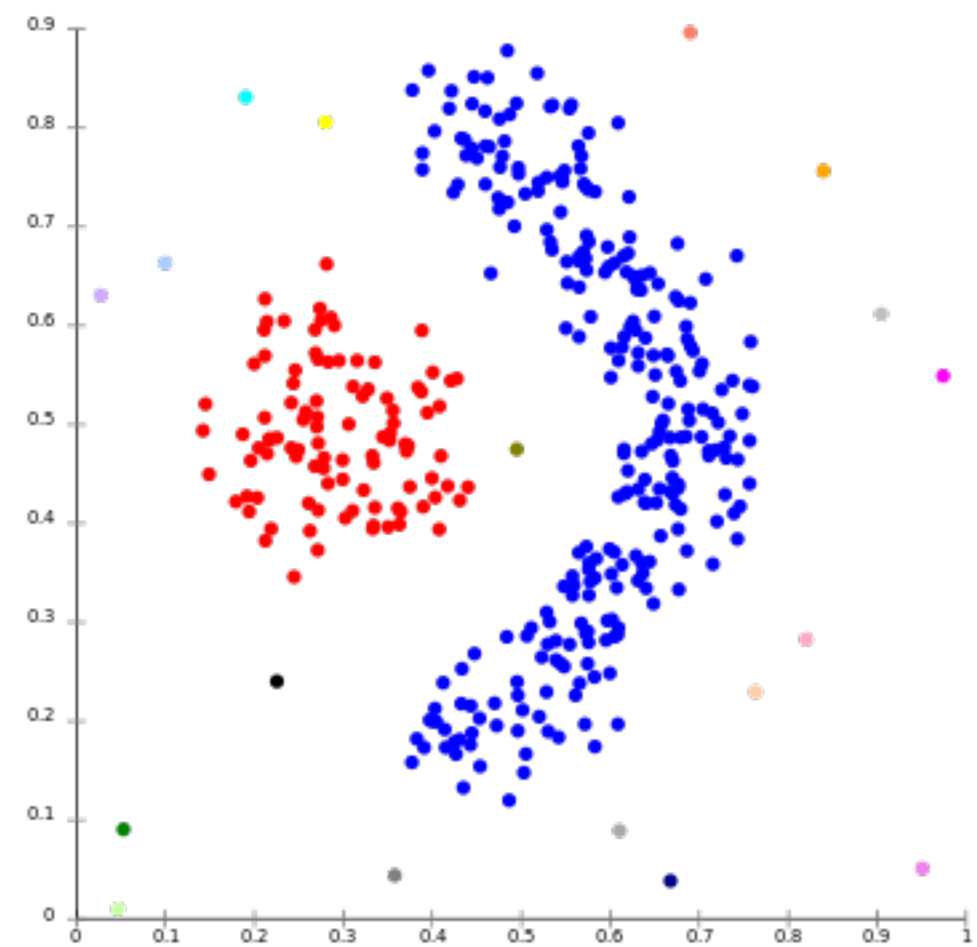


Notion of Clusters: Connected regions of high density

Four Types of Clustering

3. *Connectivity-based (Hierarchical)*

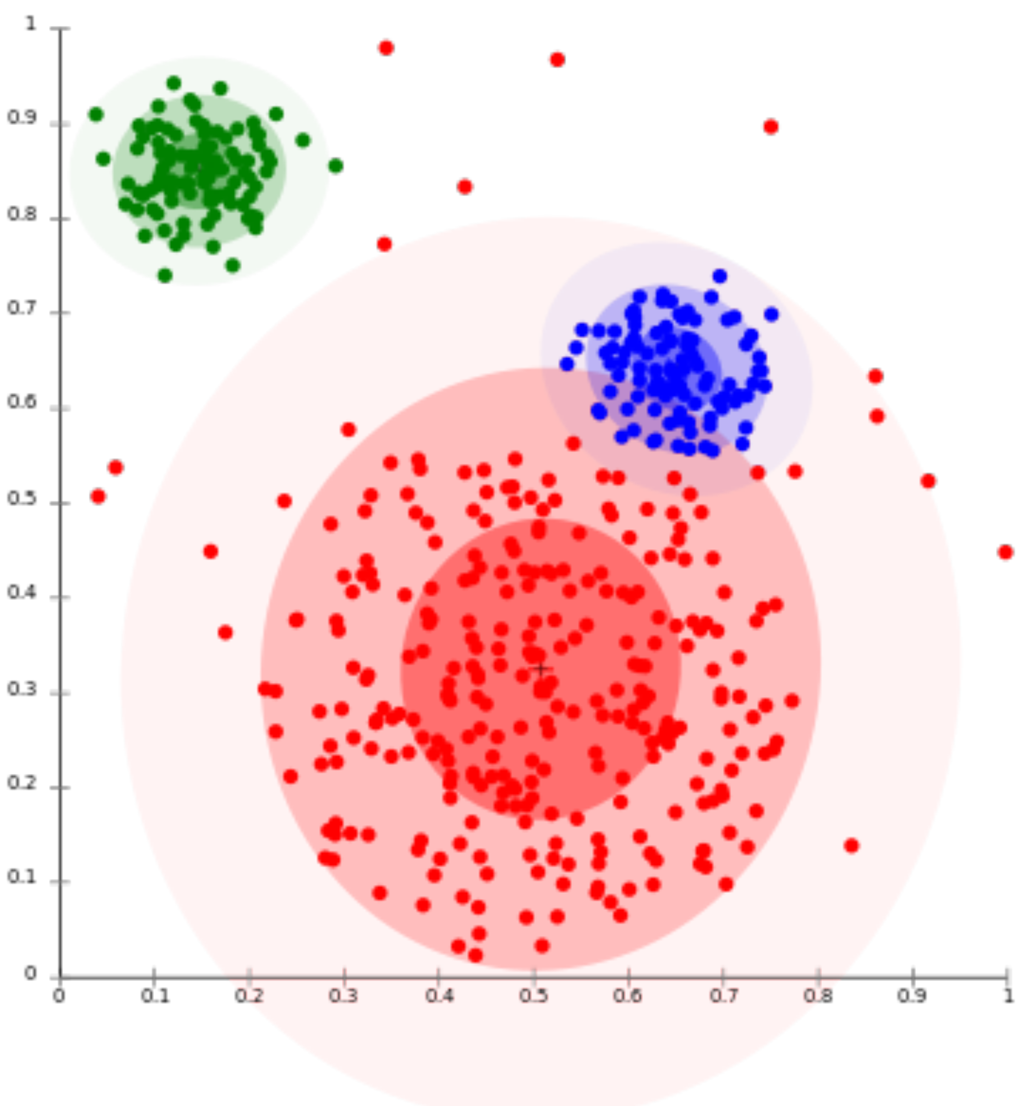


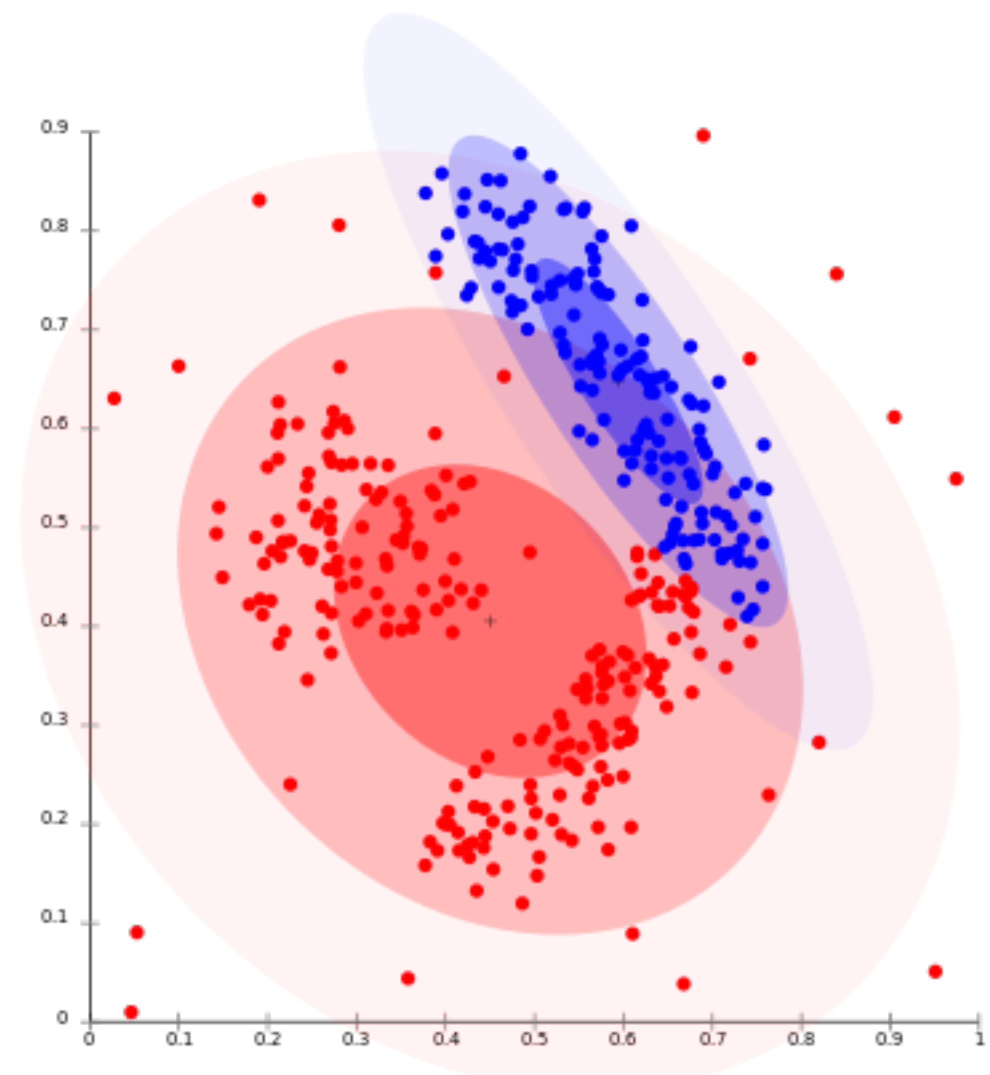


Notion of Clusters: Cut off dendrogram at some depth

Four Types of Clustering

4. *Distribution-based (Mixture Models)*



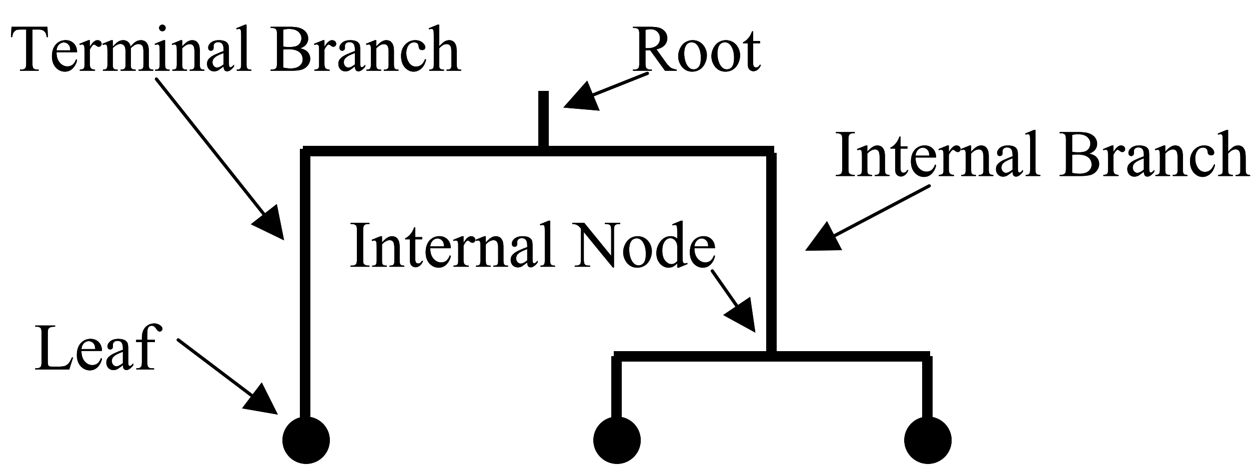


Notion of Clusters: Distributions on features

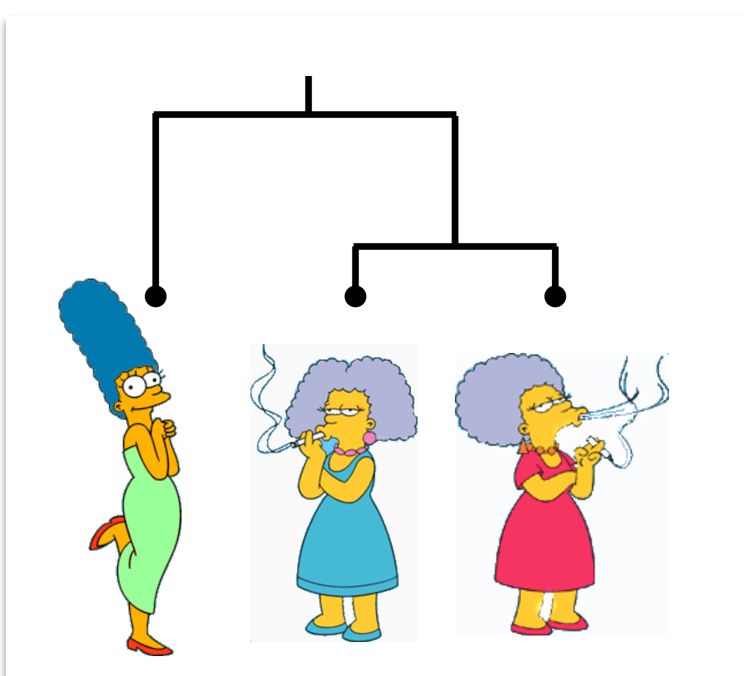
Hierarchical Clustering

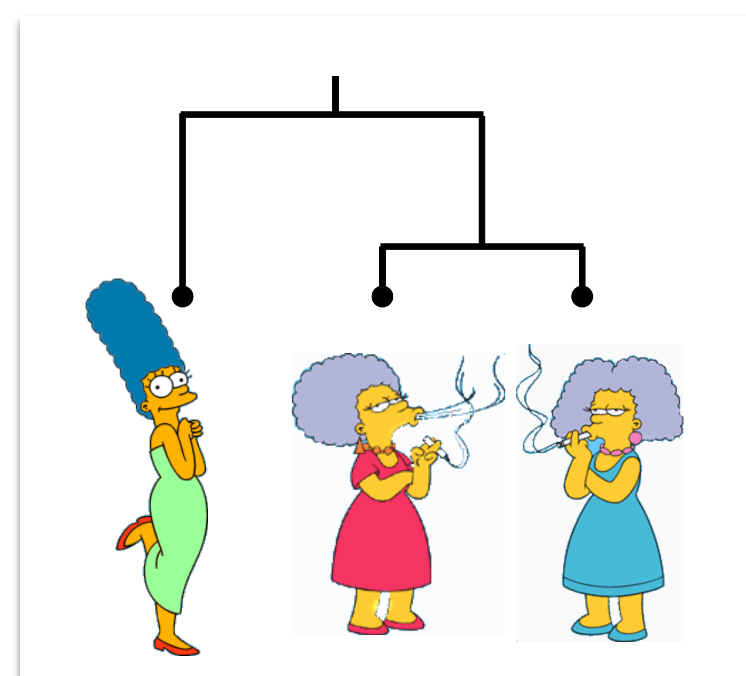
Dendrogram

(*a.k.a. a similarity tree*)



Similarity of A and B is represented as height of lowest shared internal node

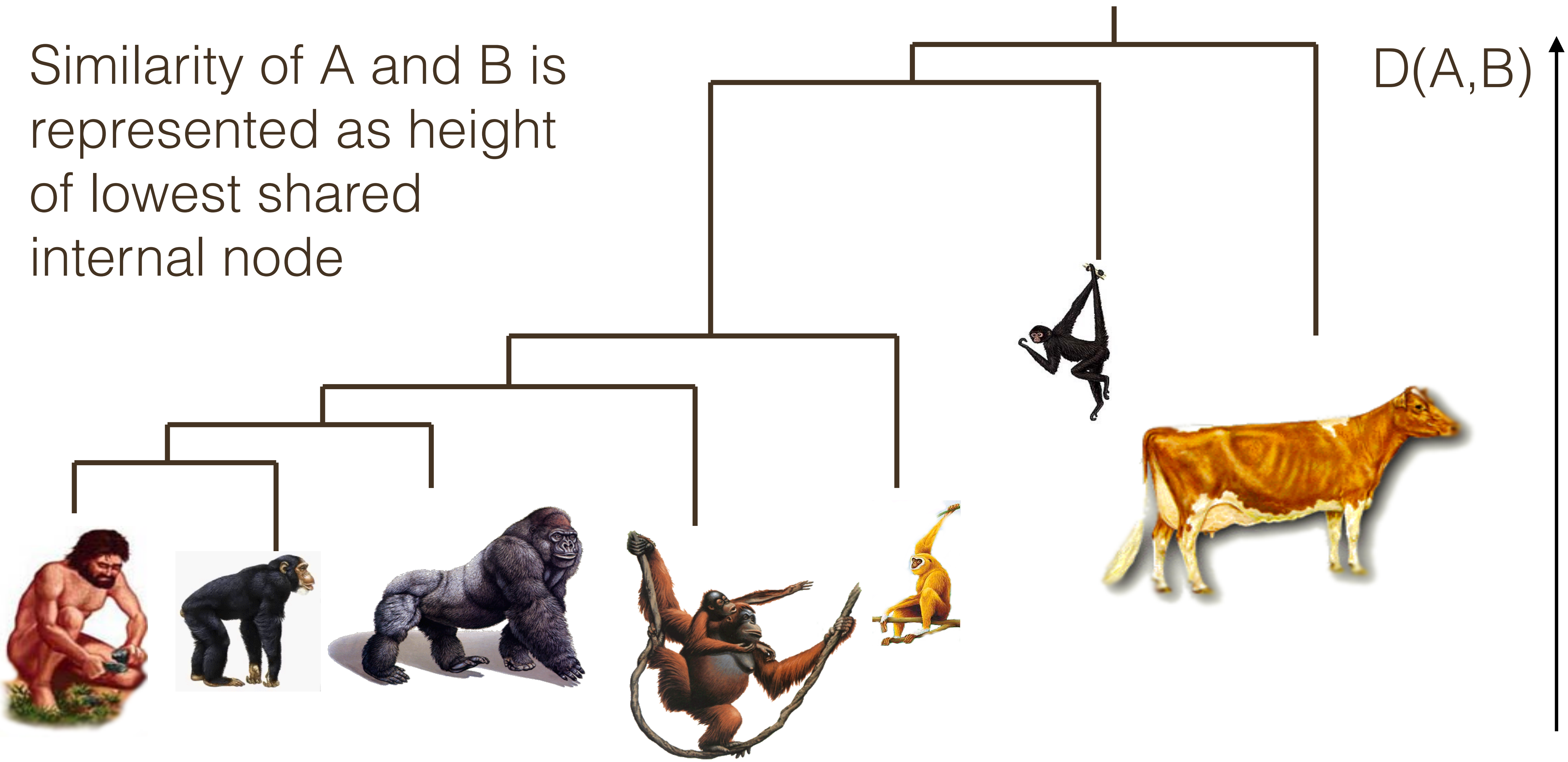




Dendrogram

(*a.k.a. a similarity tree*)

Similarity of A and B is represented as height of lowest shared internal node

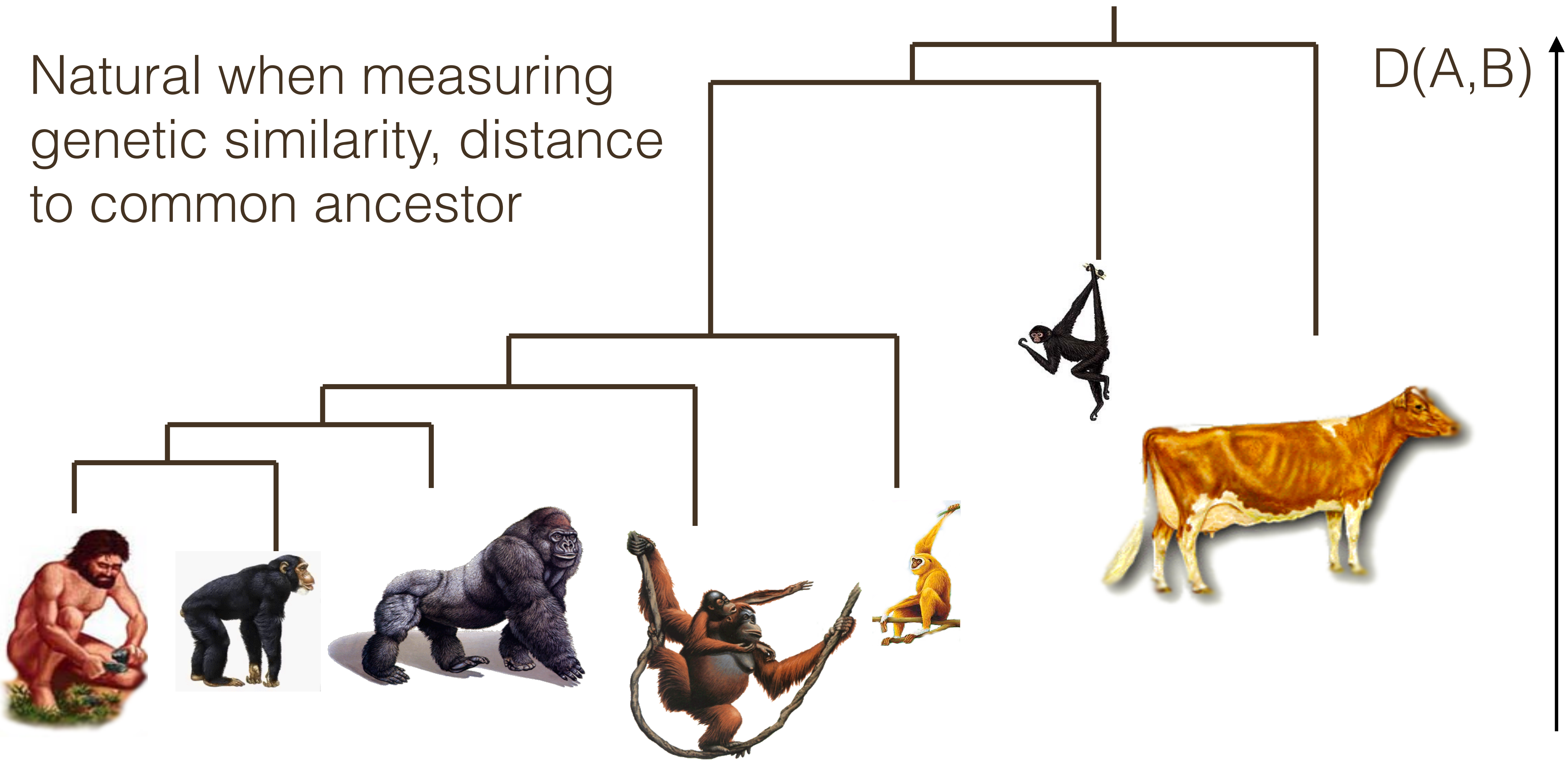


(Bovine: 0.69395, (Spider Monkey: 0.390, (Gibbon:0.36079,(Orang: 0.33636, (Gorilla: 0.17147, (Chimp: 0.19268, Human: 0.11927): 0.08386): 0.06124): 0.15057): 0.54939);

Dendrogram

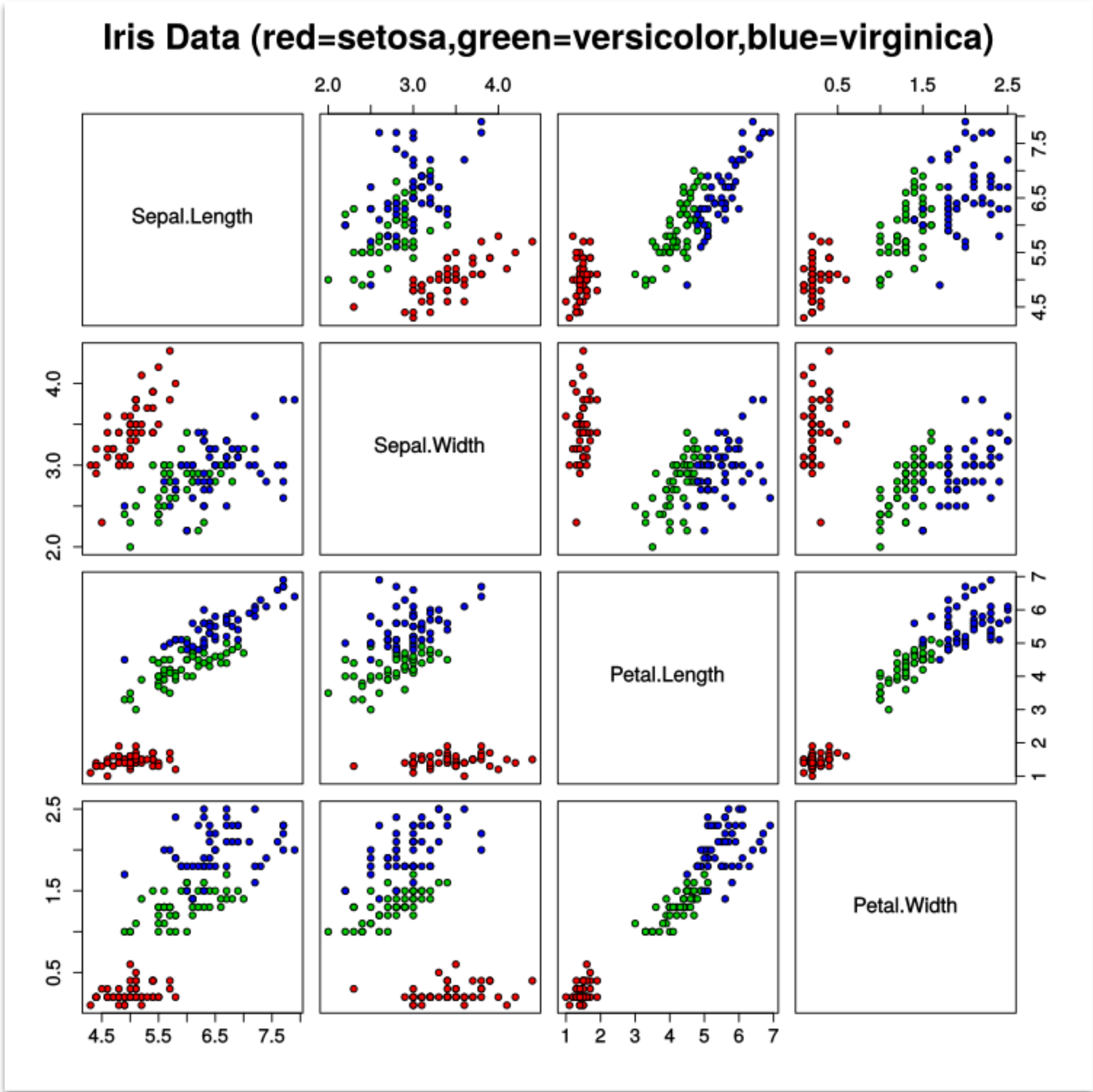
(*a.k.a. a similarity tree*)

Natural when measuring genetic similarity, distance to common ancestor



(Bovine: 0.69395, (Spider Monkey: 0.390, (Gibbon:0.36079,(Orang: 0.33636, (Gorilla: 0.17147, (Chimp: 0.19268, Human: 0.11927): 0.08386): 0.06124): 0.15057): 0.54939);

Example: Iris data



Iris Setosa

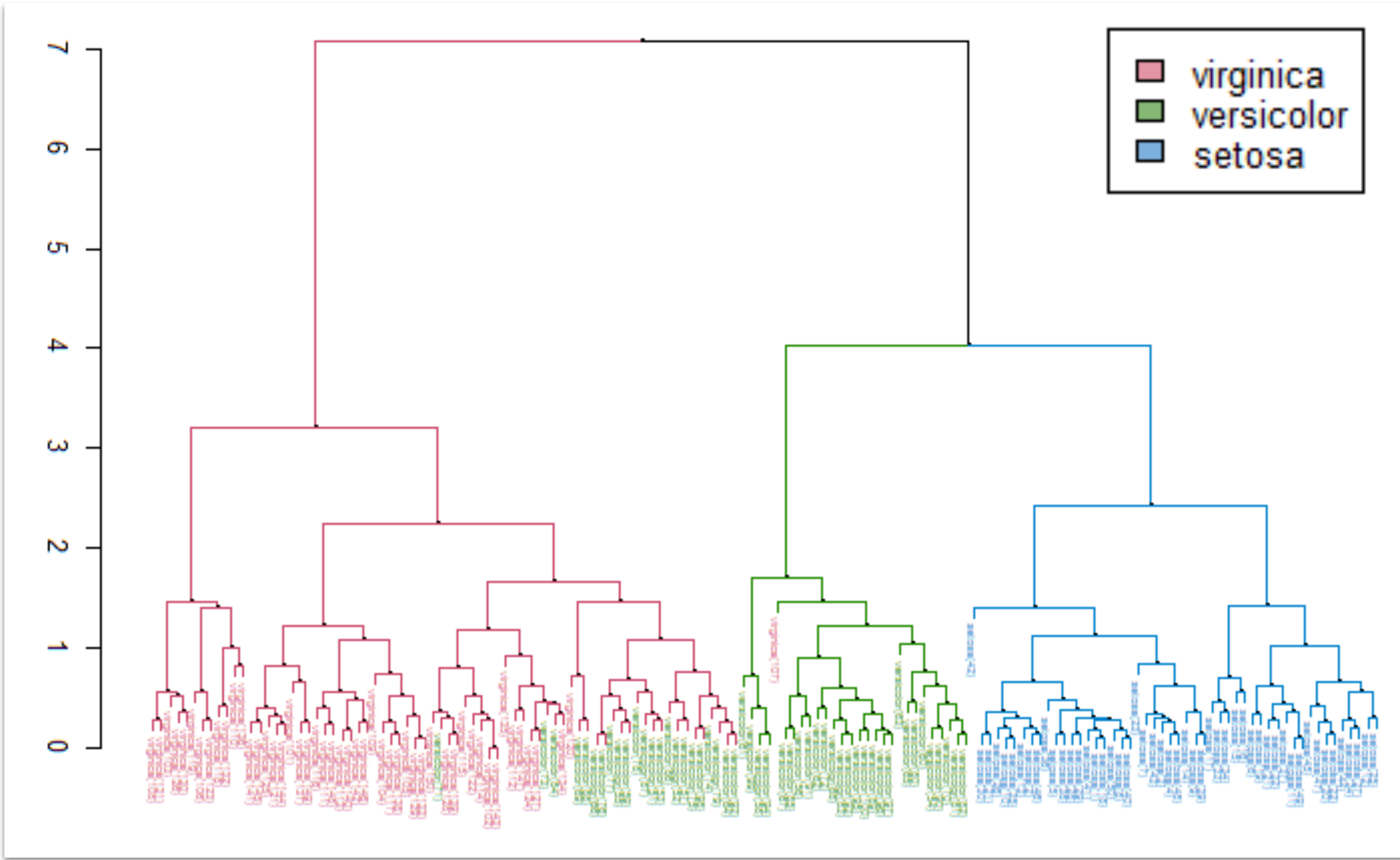
Iris versicolor

Iris virginica

https://en.wikipedia.org/wiki/Iris_flower_data_set

Hierarchical Clustering

(*Euclidian Distance*)



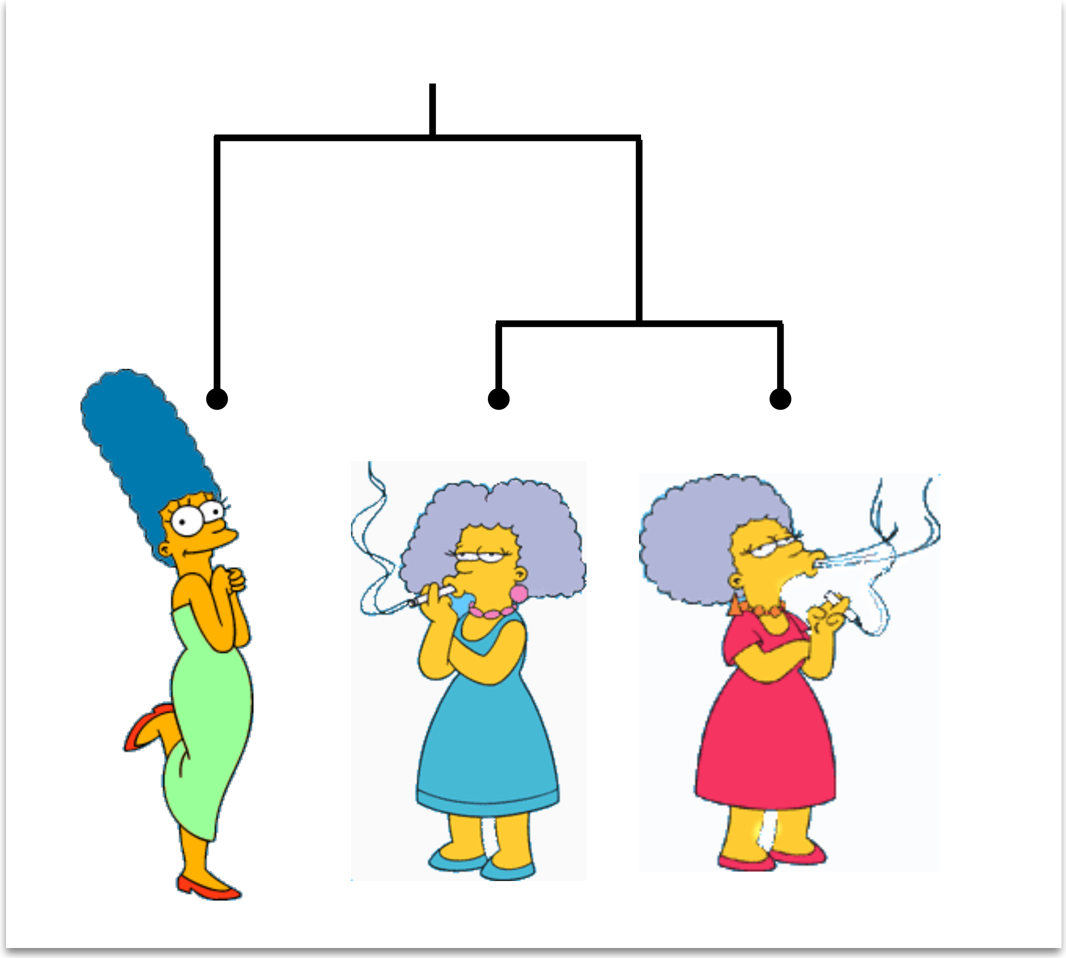
https://en.wikipedia.org/wiki/Iris_flower_data_set

Edit Distance

Distance Patty and Selma

```
Change dress color,    1 point
Change earring shape,  1 point
Change hair part,      1 point
```

D(Patty, Selma) = **3**



Distance Marge and Selma

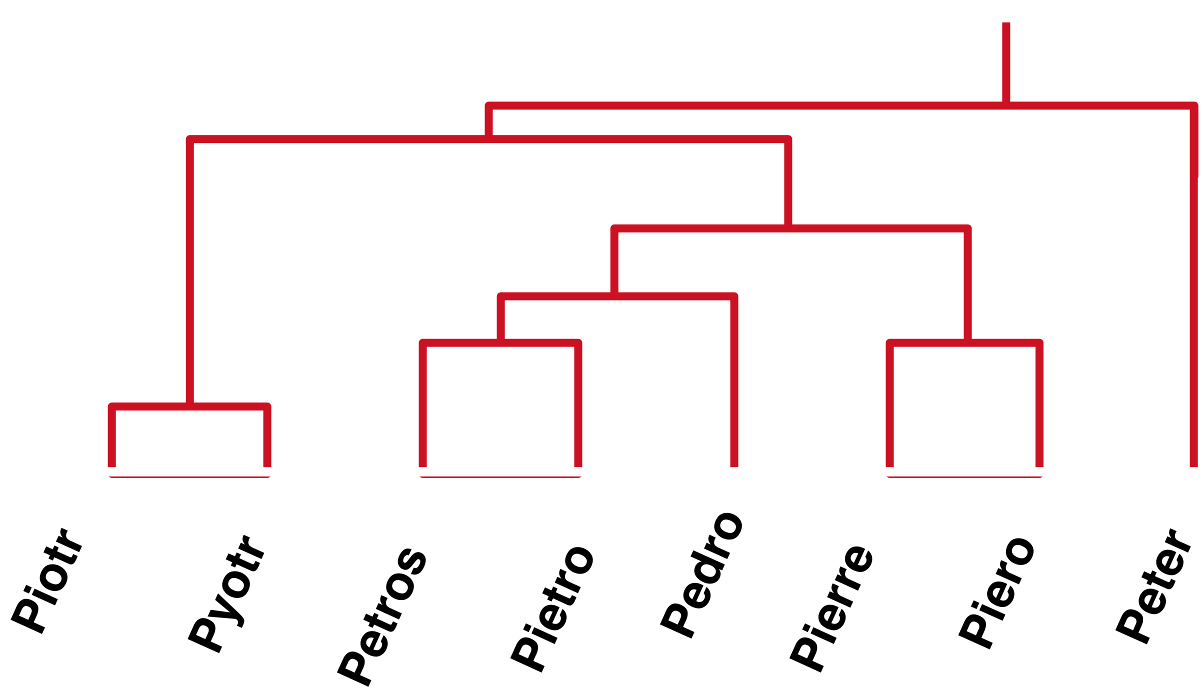
```
Change dress color,    1 point
Add earrings,          1 point
Decrease height,       1 point
Take up smoking,       1 point
Lose weight,           1 point
```

D(Marge,Selma) = **5**

Can be defined for any set of discrete features

Edit Distance for Strings

- Transform string *Q* into string *C*, using only ***Substitution***, ***Insertion*** and ***Deletion***.
- Assume that each of these operators has a **cost** associated with it.
- The similarity between two strings can be defined as the cost of the ***cheapest*** transformation from *Q* to *C*.



Similarity “Peter” and “Piotr”?

Substitution	1 Unit
Insertion	1 Unit
Deletion	1 Unit

D(**Peter**,**Piotr**) is 3

Peter

Substitution (i for e)

Piter

Insertion (o)

Pioter

Deletion (e)

Piotr

Hierarchical Clustering

(Edit Distance)

Pedro (Portuguese)

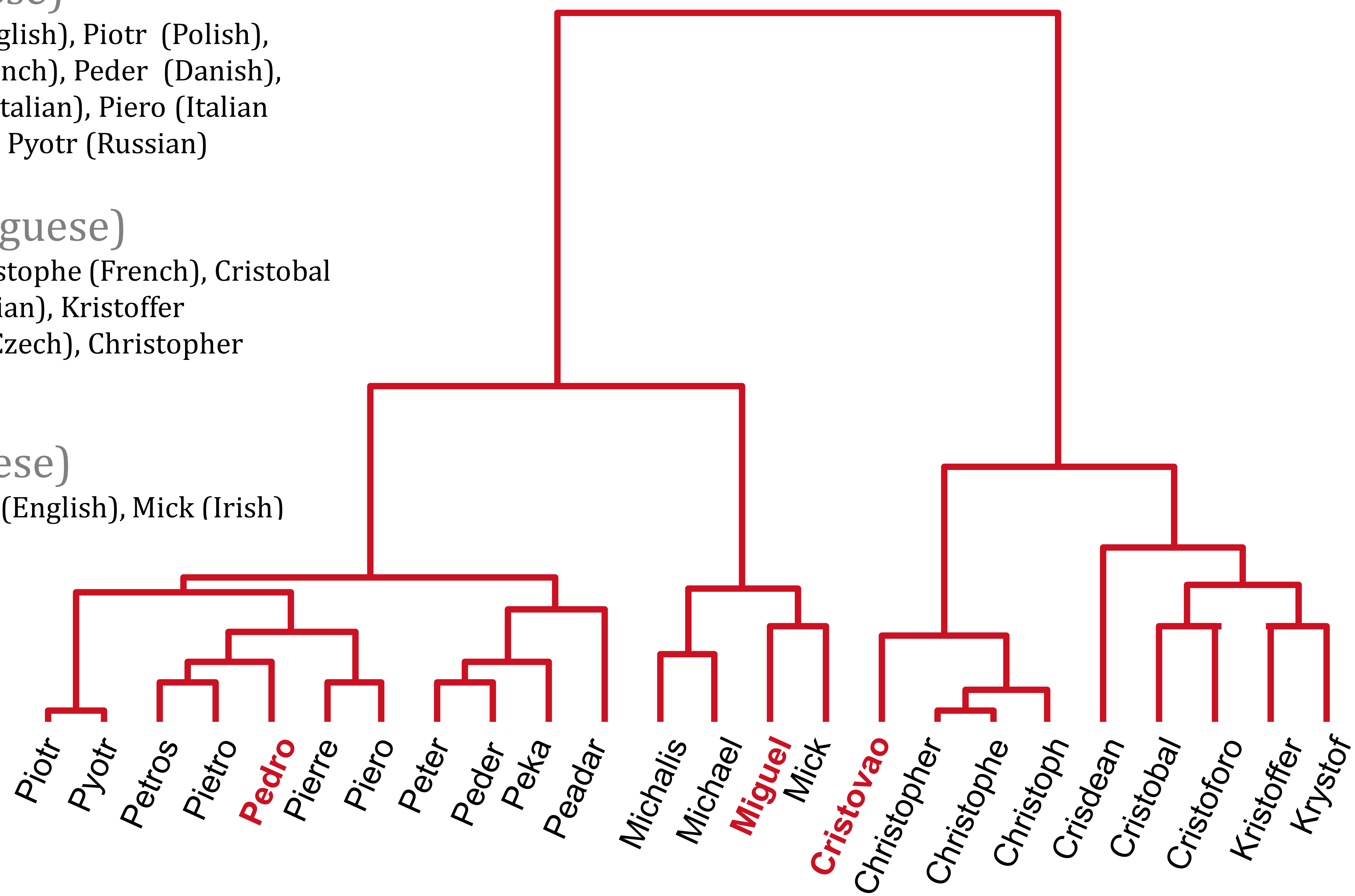
Petros (Greek), Peter (English), Piotr (Polish), Peadar (Irish), Pierre (French), Peder (Danish), Peka (Hawaiian), Pietro (Italian), Piero (Italian Alternative), Petr (Czech), Pyotr (Russian)

Cristovao (Portuguese)

Christoph (German), Christophe (French), Cristobal (Spanish), Cristoforo (Italian), Kristoffer (Scandinavian), Krystof (Czech), Christopher (English)

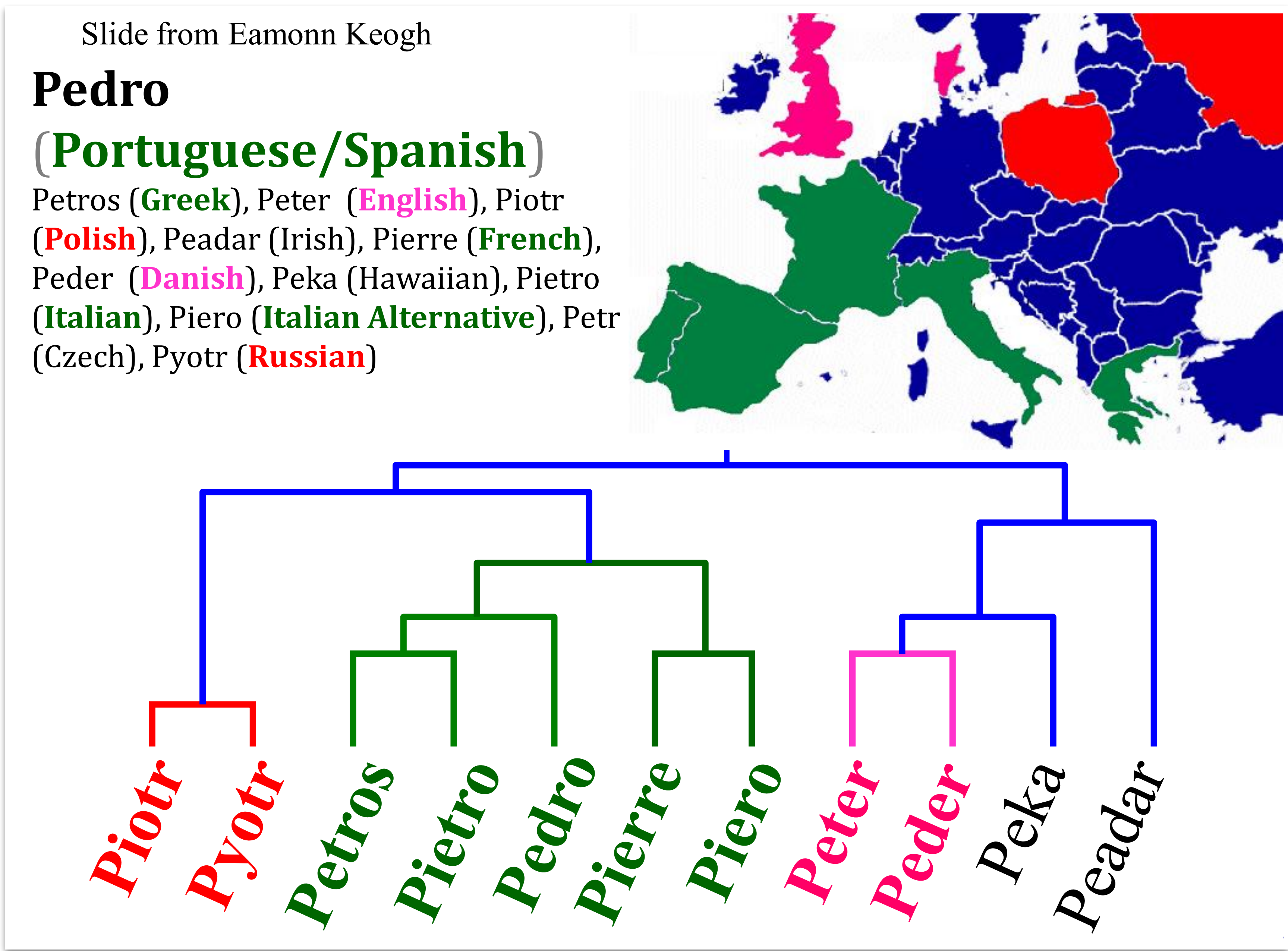
Miguel (Portuguese)

Michalis (Greek), Michael (English), Mick (Irish)



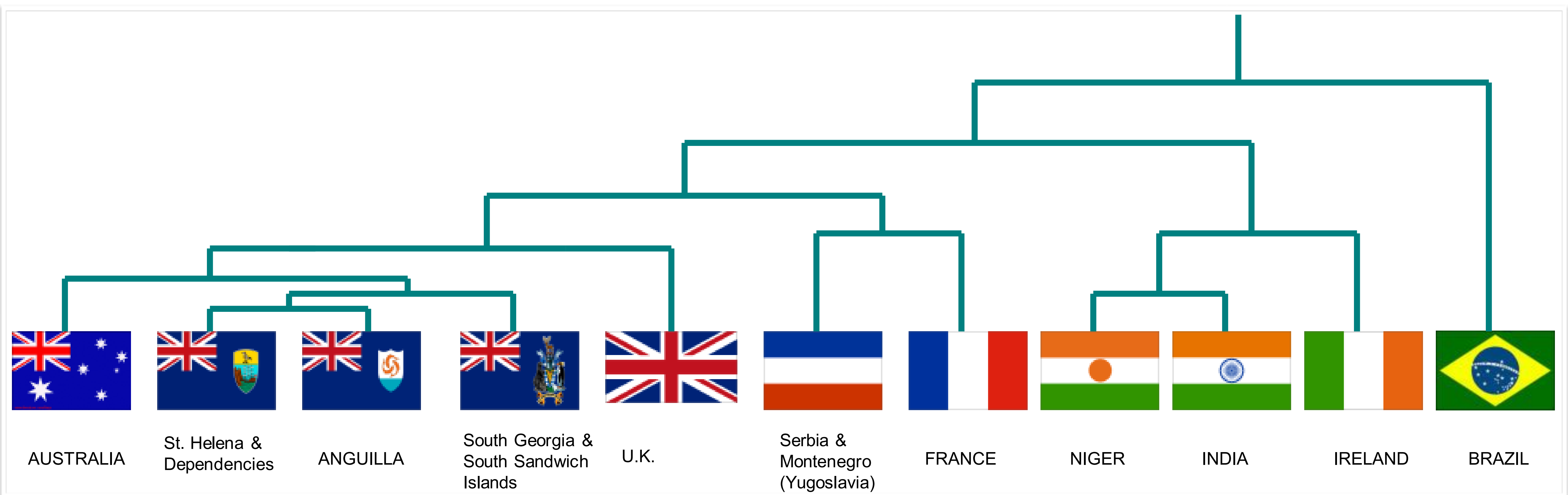
Meaningful Patterns

Edit distance yields clustering according to geography



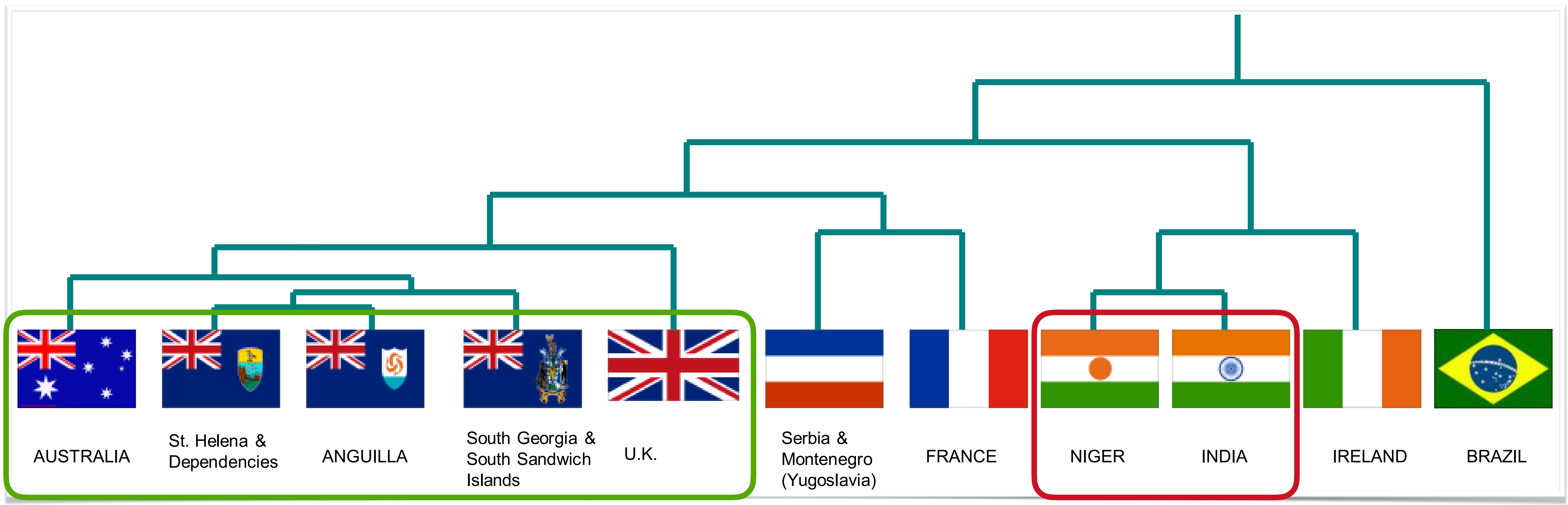
Spurious Patterns

In general clusterings will only be as meaningful as your distance metric



Spurious Patterns

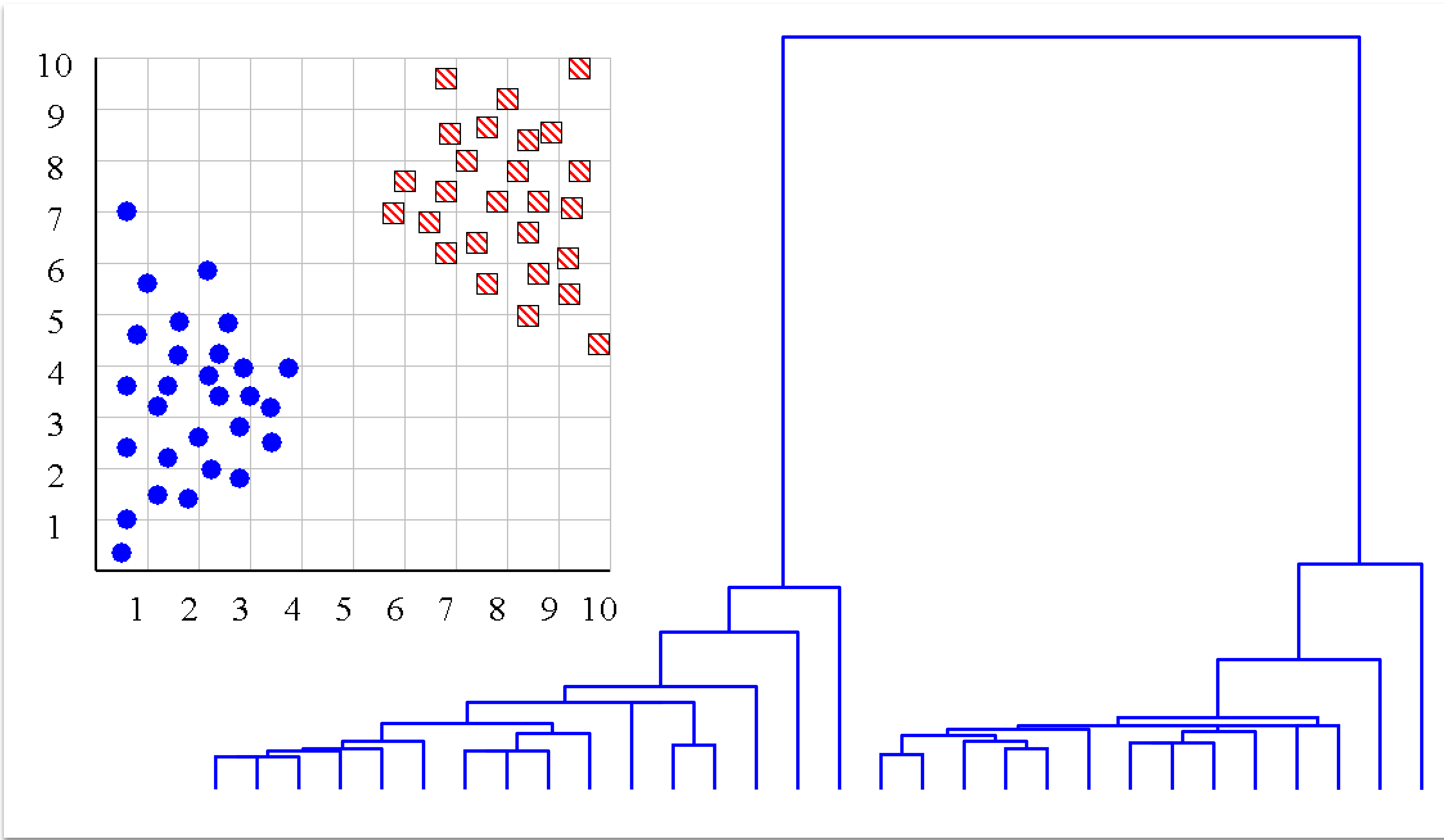
In general clusterings will only be
as meaningful as your distance metric



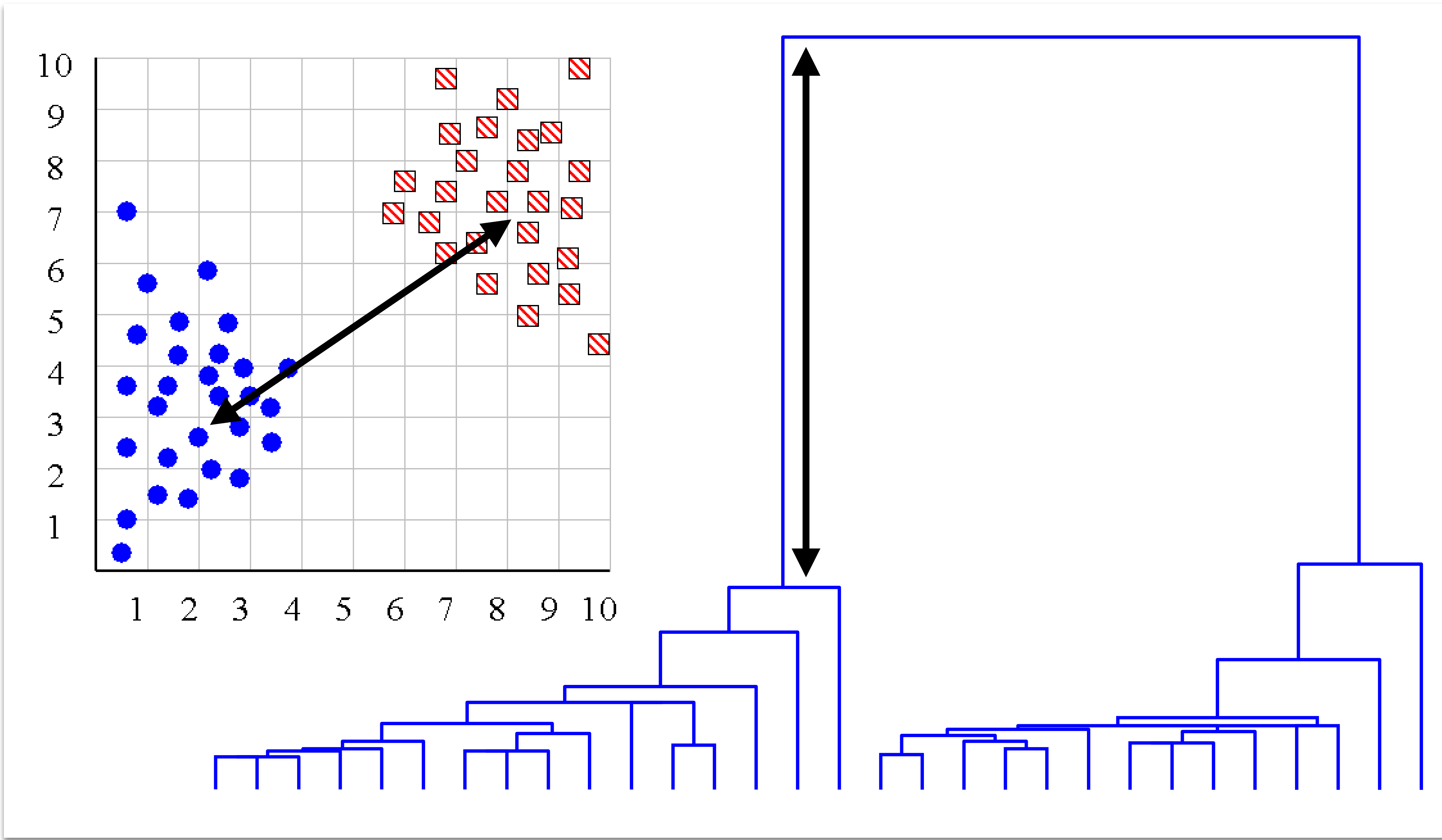
Former UK colonies

No relation

“Correct” Number of Clusters

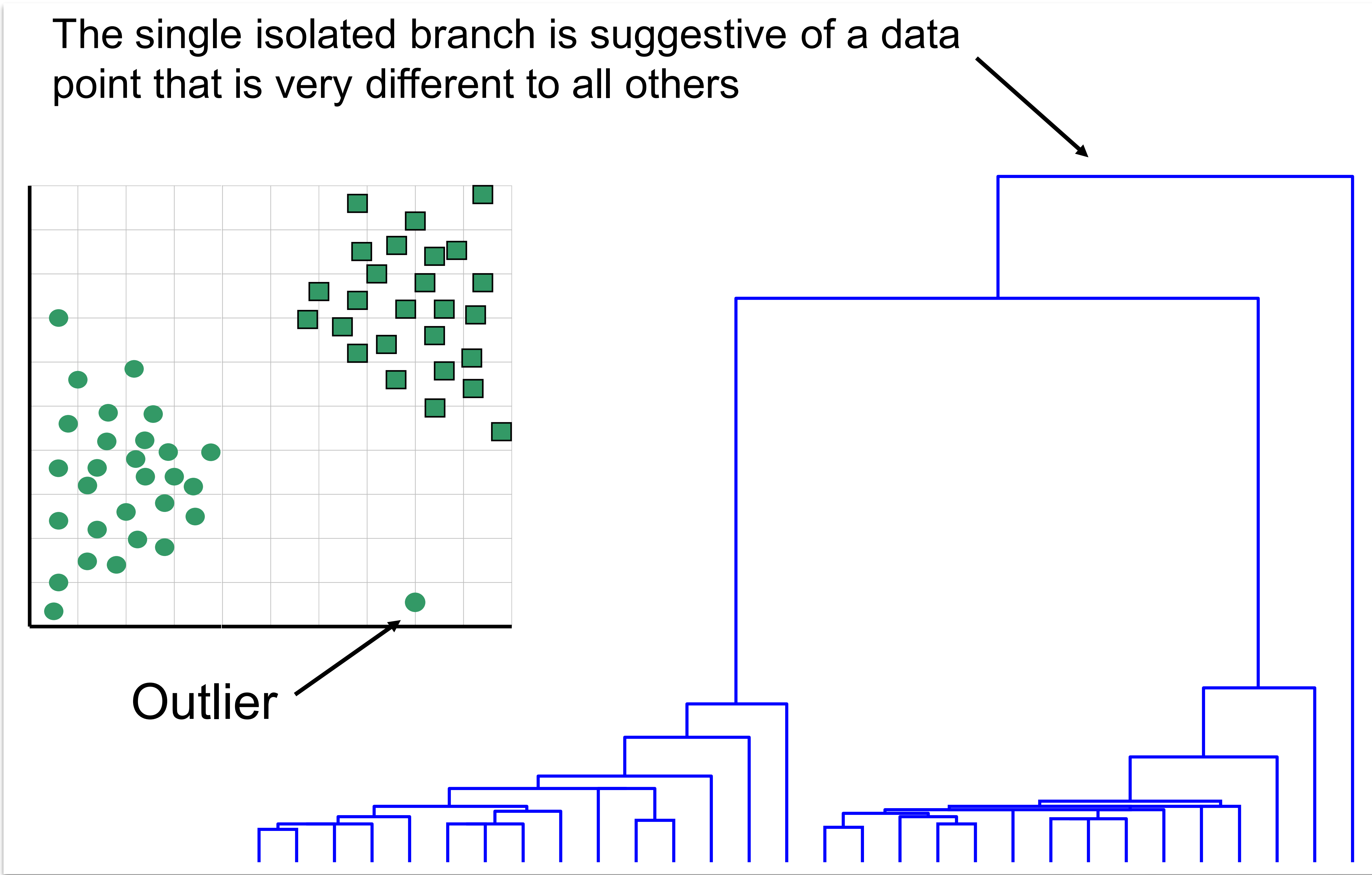


“Correct” Number of Clusters



Determine number of clusters by looking at distance

Detecting Outliers



Bottom up vs. Top down

Bottom-up (*agglomerative):* Each item starts as its own cluster; greedily merge

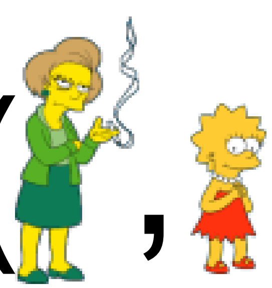
Bottom up vs. Top down

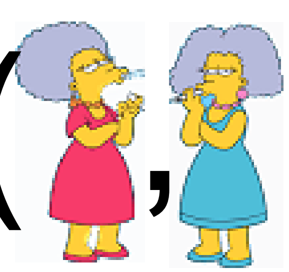
Bottom-up (*agglomerative):* Each item starts as its own cluster; greedily merge

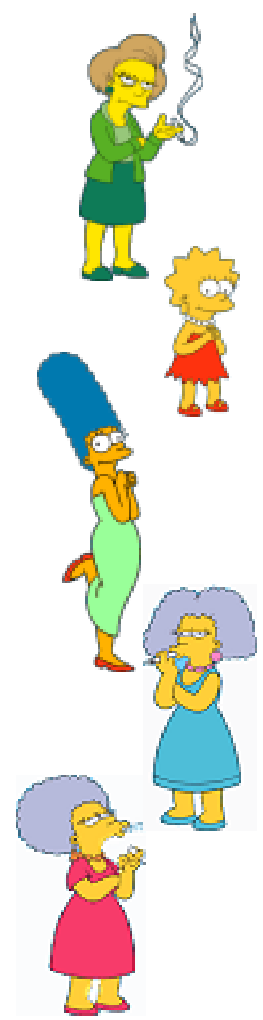
Top-down (*divisive*): Start with one big cluster (all data); recursively split

Distance Matrix

We begin with a distance matrix which contains the distances between every pair of objects in our database.

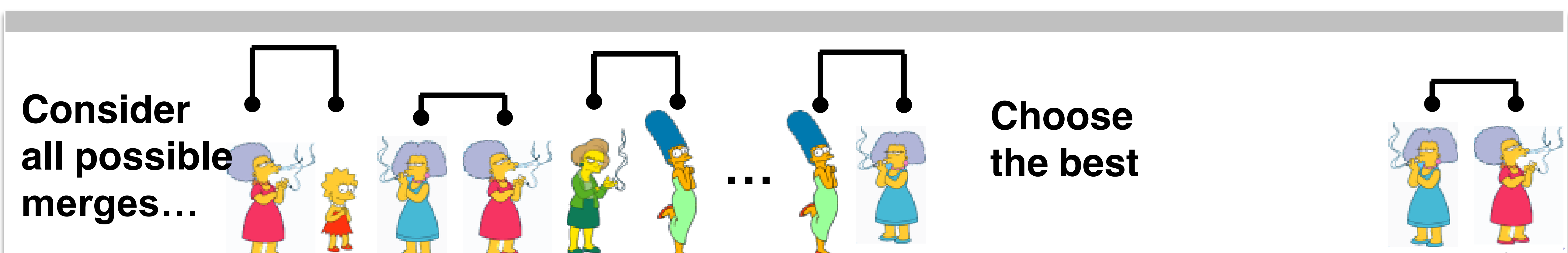
D() = 8

D() = 1

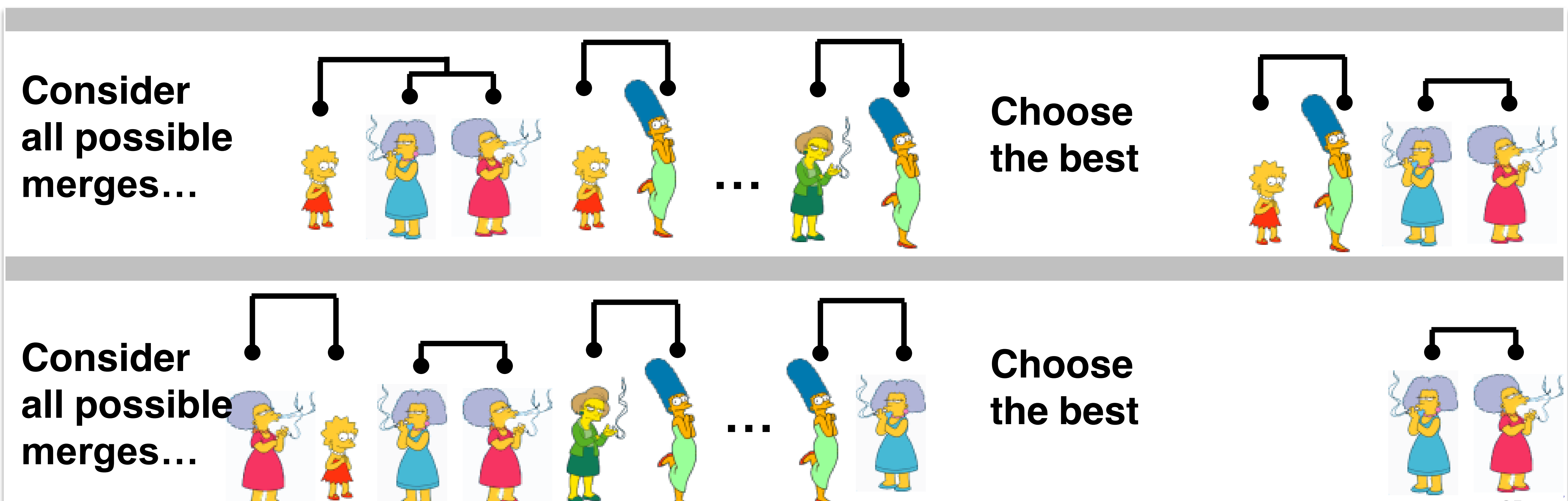


0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

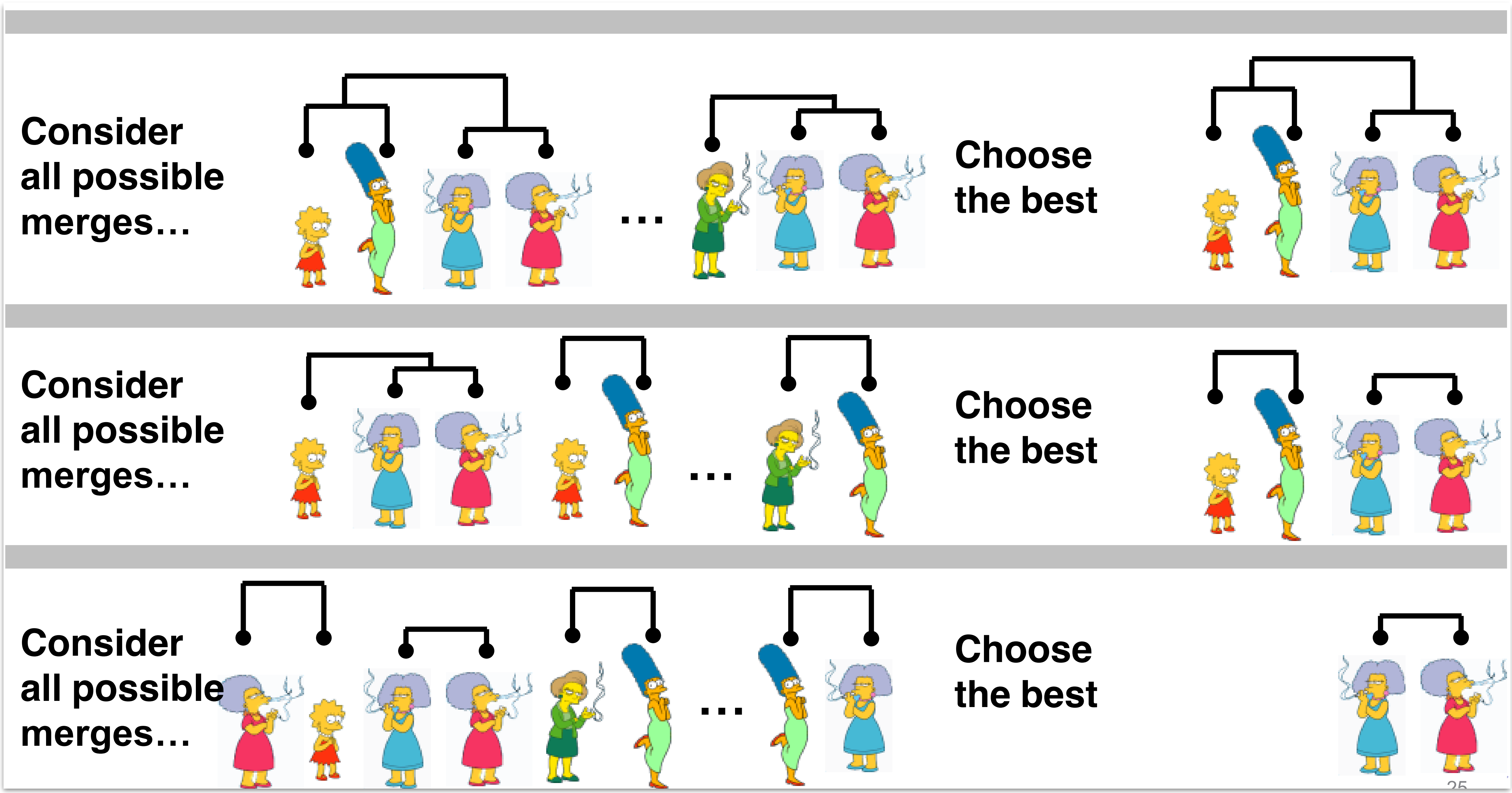
Bottom-up (Agglomerative Clustering)



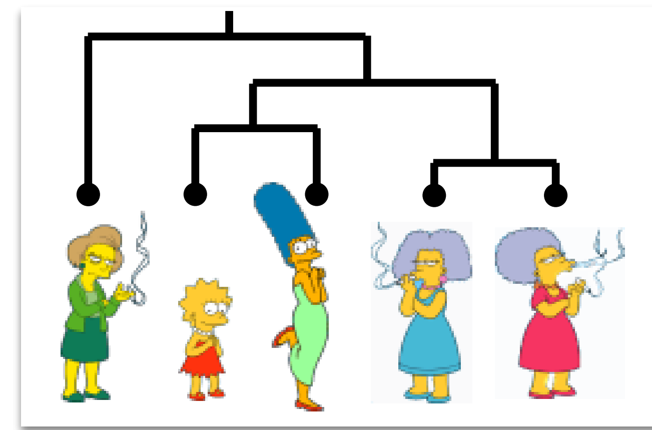
Bottom-up (Agglomerative Clustering)

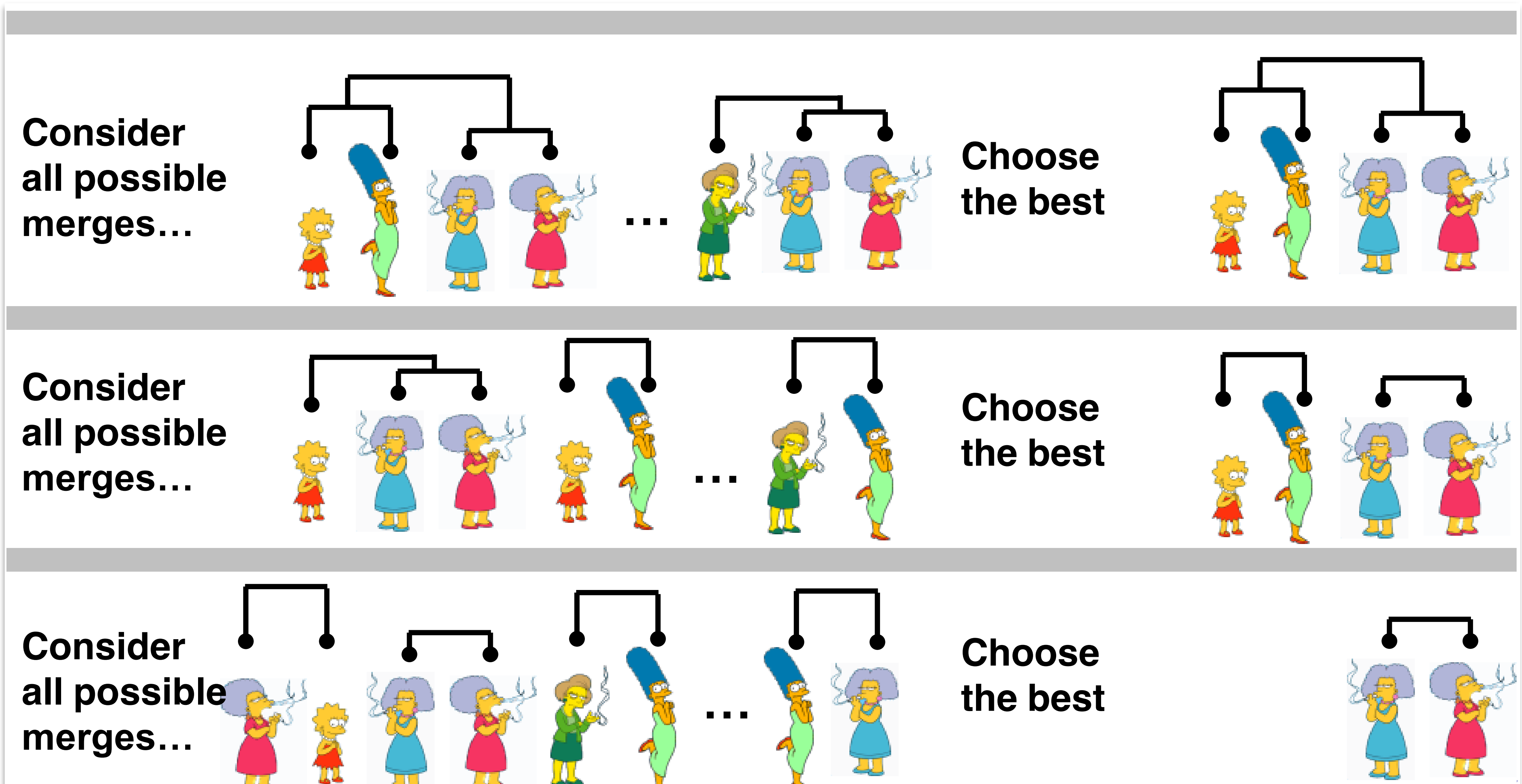


Bottom-up (Agglomerative Clustering)



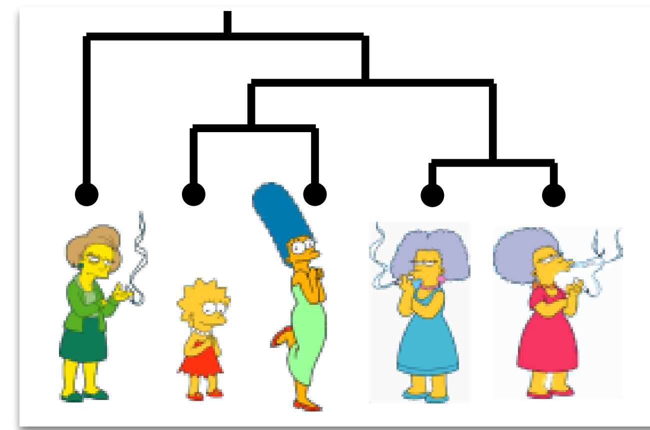
Bottom-up (Agglomerative Clustering)

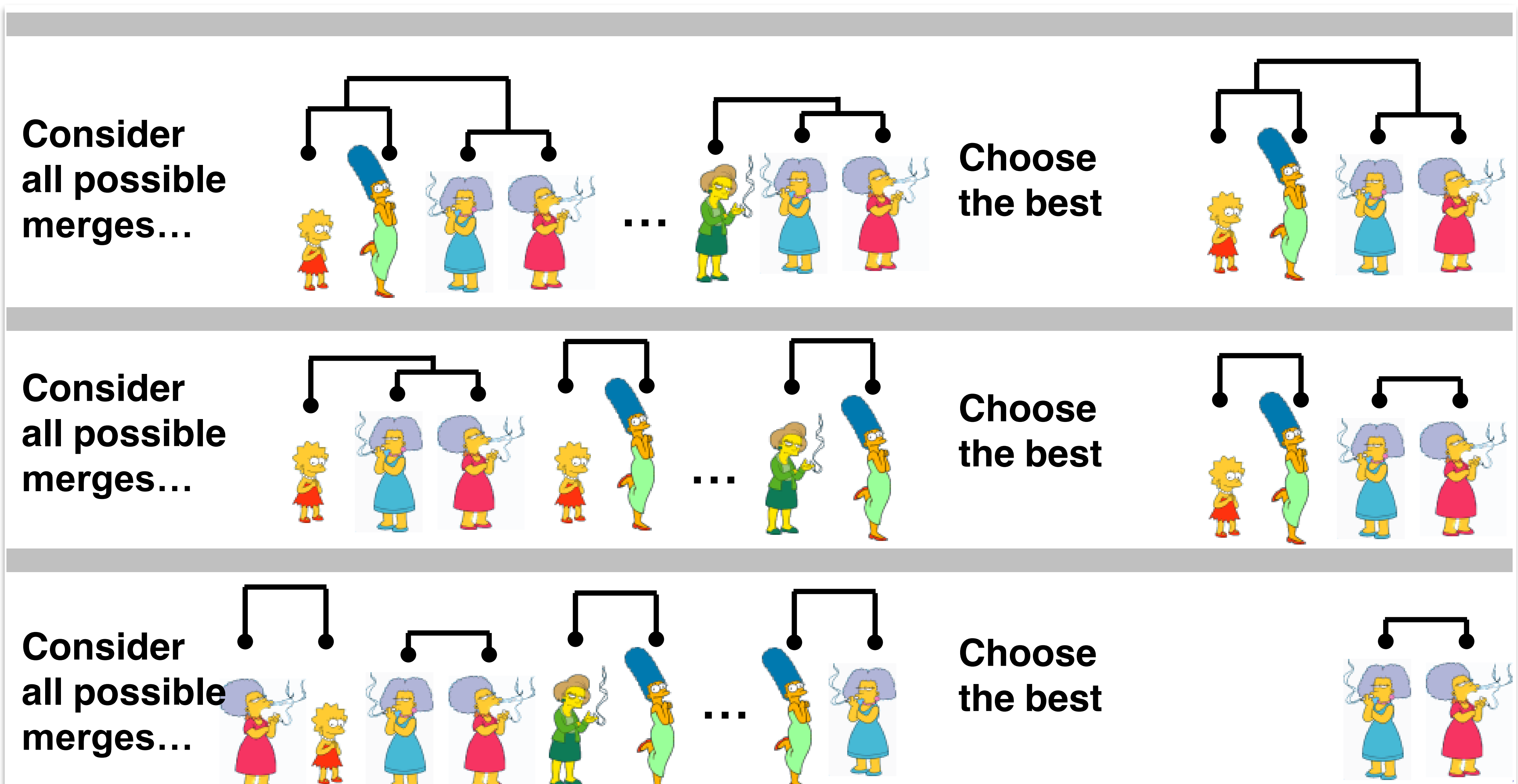




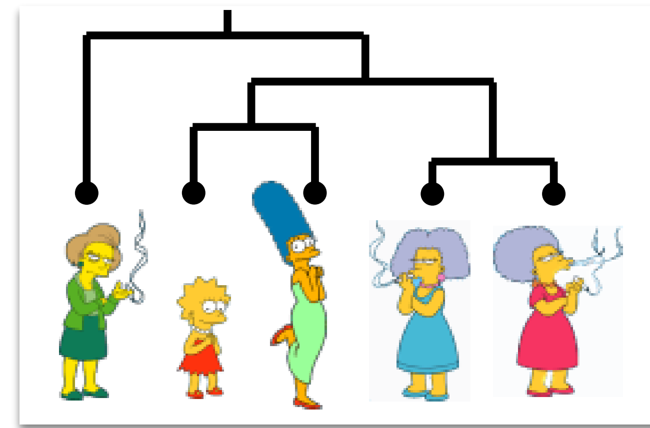
Bottom-up (Agglomerative Clustering)

Can you now implement this?

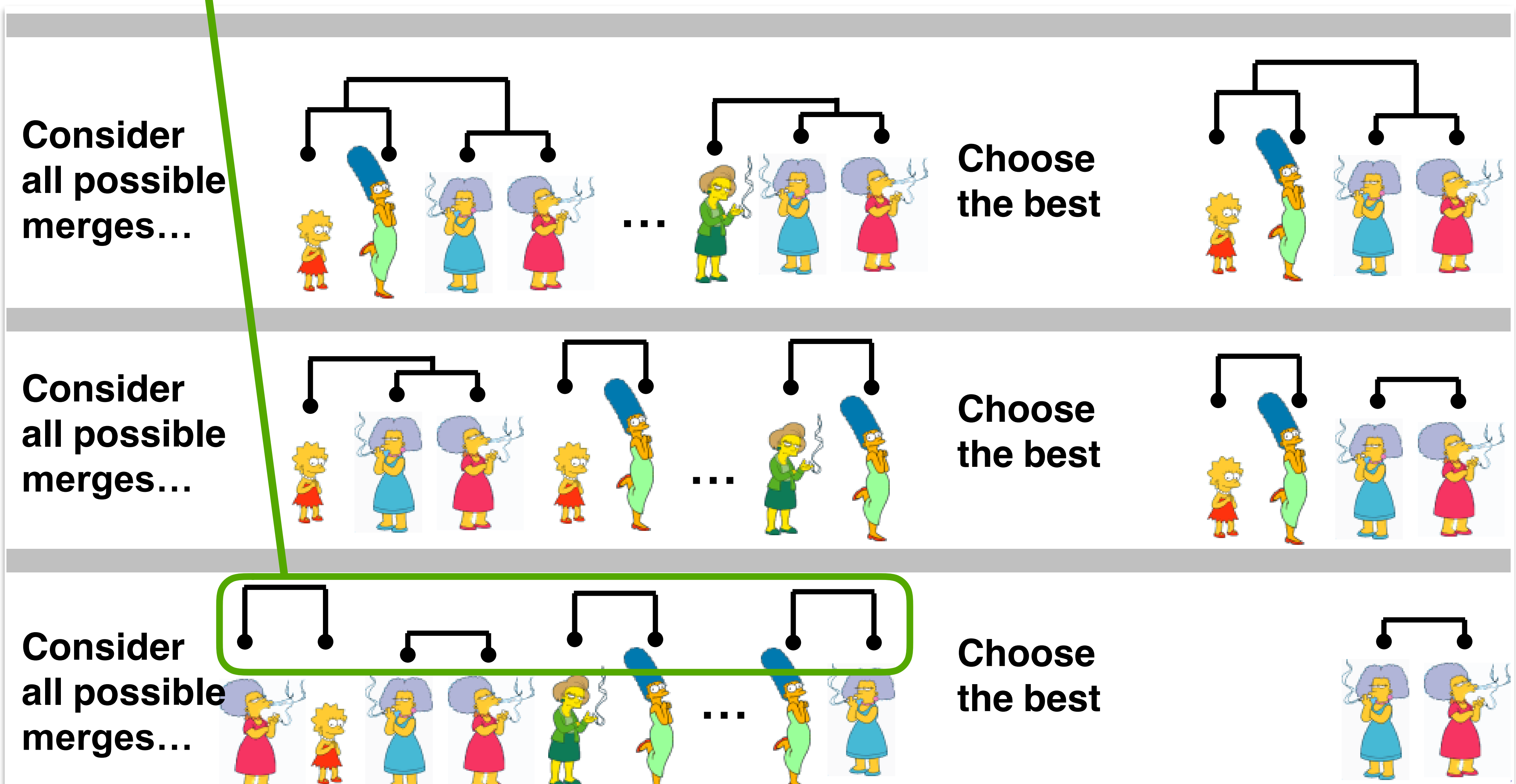




Bottom-up (Agglomerative Clustering)

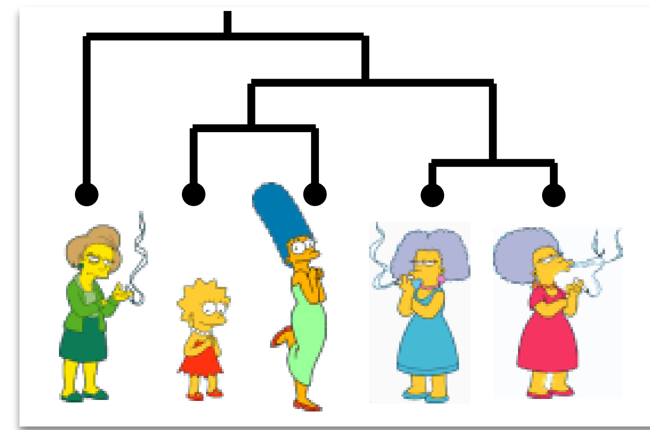


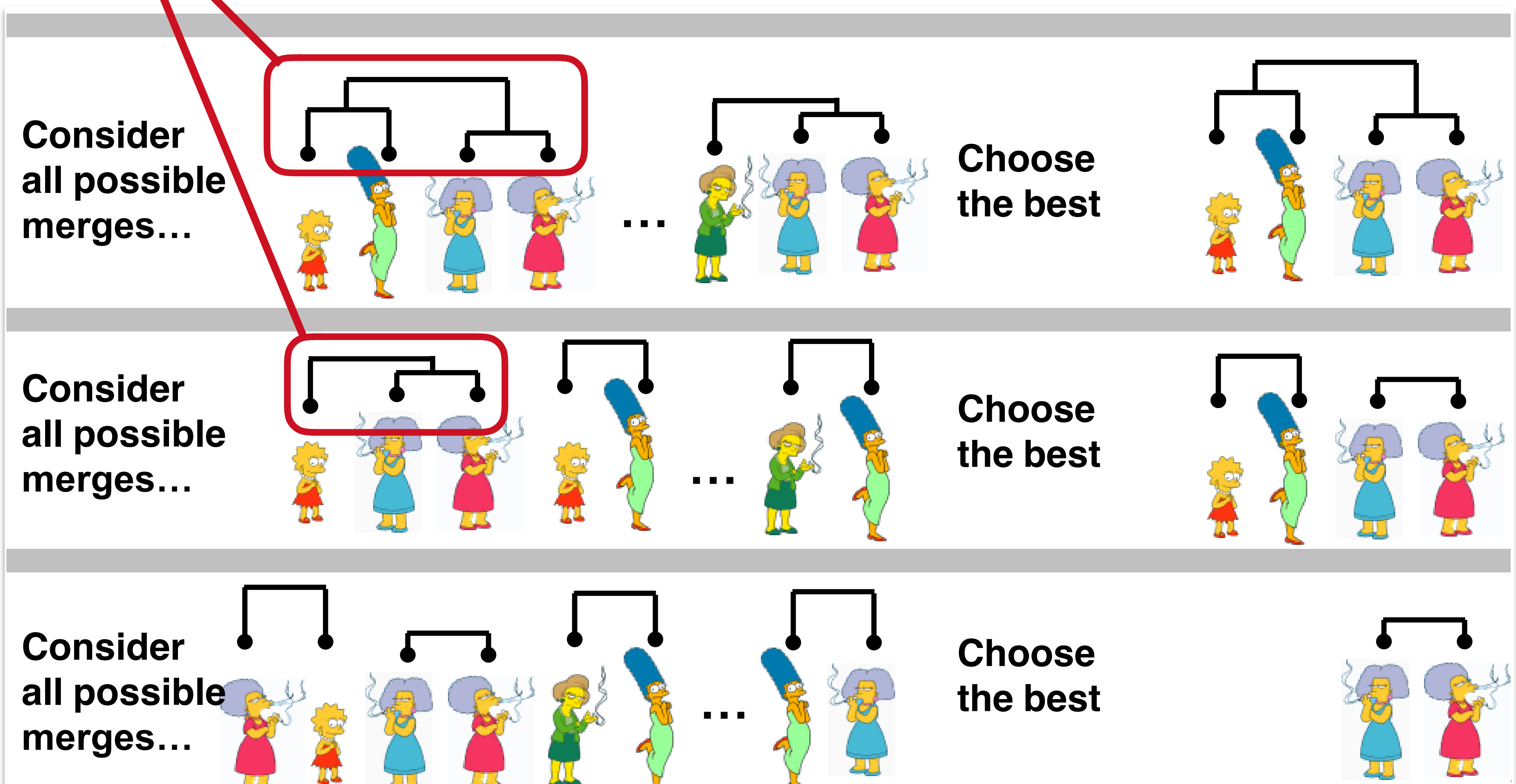
Distances between examples
(can calculate using metric)



Bottom-up (Agglomerative Clustering)

How do we calculate the distance to a cluster?

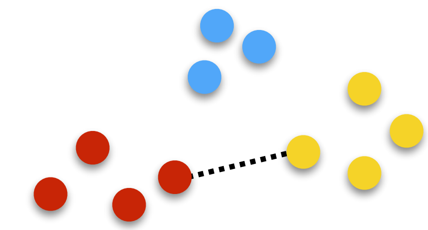




Clustering Criteria

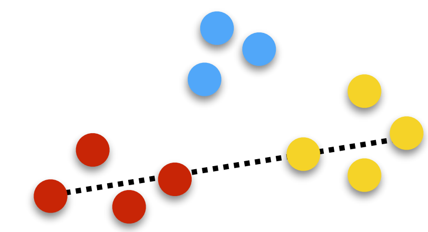
Single link:
(Closest point)

$$d(A,B) = \min_{a \in A, b \in B} d(a,b)$$



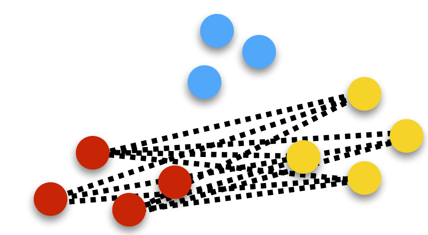
Complete link:
(Furthest point)

$$d(A,B) = \max_{a \in A, b \in B} d(a,b)$$



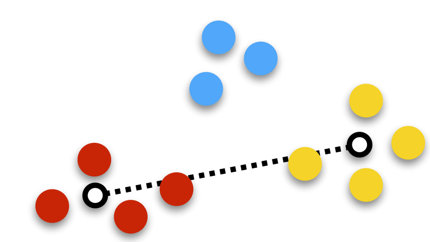
Group average:
(Average distance)

$$d(A,B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} d(a,b)$$



Centroid:
(Distance of average)

$$d(A,B) = d(\mu_A, \mu_B) \quad \mu_X = \frac{1}{|X|} \sum_{x \in X} x$$



Hierarchical Clustering Summary

+ No need to specify number of clusters

Hierarchical Clustering Summary

- + No need to specify number of clusters
- + Hierarchical structure maps nicely onto human intuition in some domains

Hierarchical Clustering Summary

- \+ No need to specify number of clusters
- \+ Hierarchical structure maps nicely onto human intuition in some domains
- \- *Scaling:* Time complexity at least $O(n^2)$ in number of examples

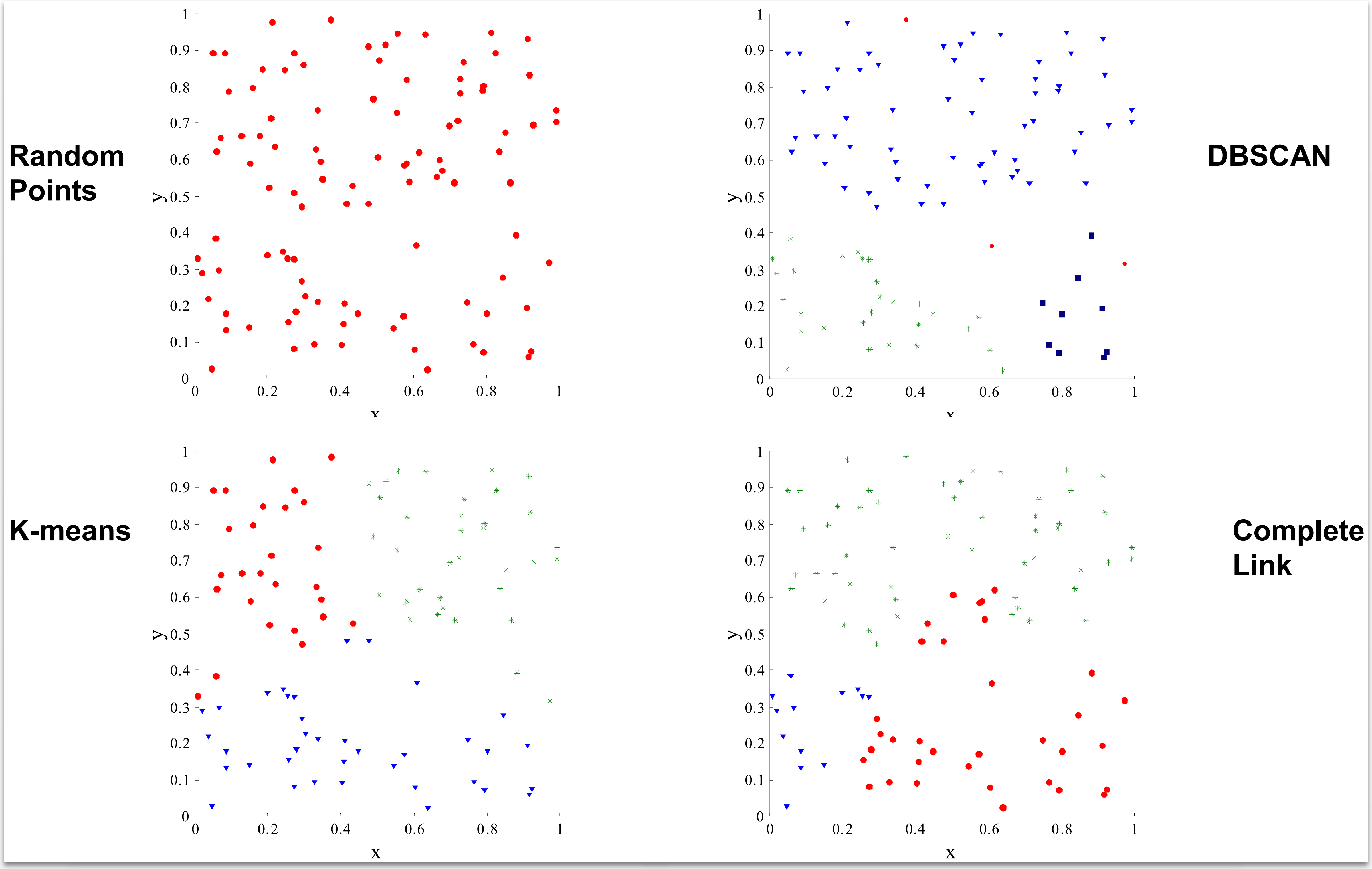
Hierarchical Clustering Summary

+ No need to specify number of clusters
+ Hierarchical structure maps nicely onto human intuition in some domains
- *Scaling:* Time complexity at least $O(n^2)$ in number of examples
- *Heuristic search method*: Local optima are a problem

Hierarchical Clustering Summary

+ No need to specify number of clusters
+ Hierarchical structure maps nicely onto human intuition in some domains
- *Scaling:* Time complexity at least $O(n^2)$ in number of examples
- *Heuristic search method*: Local optima are a problem
- *Interpretation* of results is (very) subjective

Evaluation?



Random Points
DBSCAN
K-means
Complete Link
x
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Clustering Criteria

Internal Quality Criteria
Measure compactness of clusters

- Sum of Squared Error (SSE)
- Scatter Criteria

Clustering Criteria

Internal Quality Criteria

Measure compactness of clusters

- Sum of Squared Error (SSE)
- Scatter Criteria

External Quality Criteria

- Precision-Recall Measure
- Mutual Information

From K-means to Mixture Models

From K-means to Mixture Models

Let's come back to K-means for a moment

Input: $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$

Number of clusters K

Initialize: K random centroids $\mu_1, \mu_2, \ldots, \mu_K$

Repeat Until Convergence

1. For $i = 1, \ldots, K$ do

   $C_i = \{\mathbf{x} \in X | i = \arg\min_{1 \le j \le K} \| \mathbf{x} - \mu_j \|^2\}$

2. For $i = 1, \ldots, K$ do

   $\mu_i = \arg\min_{\mathbf{z}} \sum_{\mathbf{x} \in C_i} \| \mathbf{z} - \mathbf{x} \|^2\}$

Output: $C_1, C_2, \ldots, C_K$

A probabilistic view

- K-means feels a bit heuristic

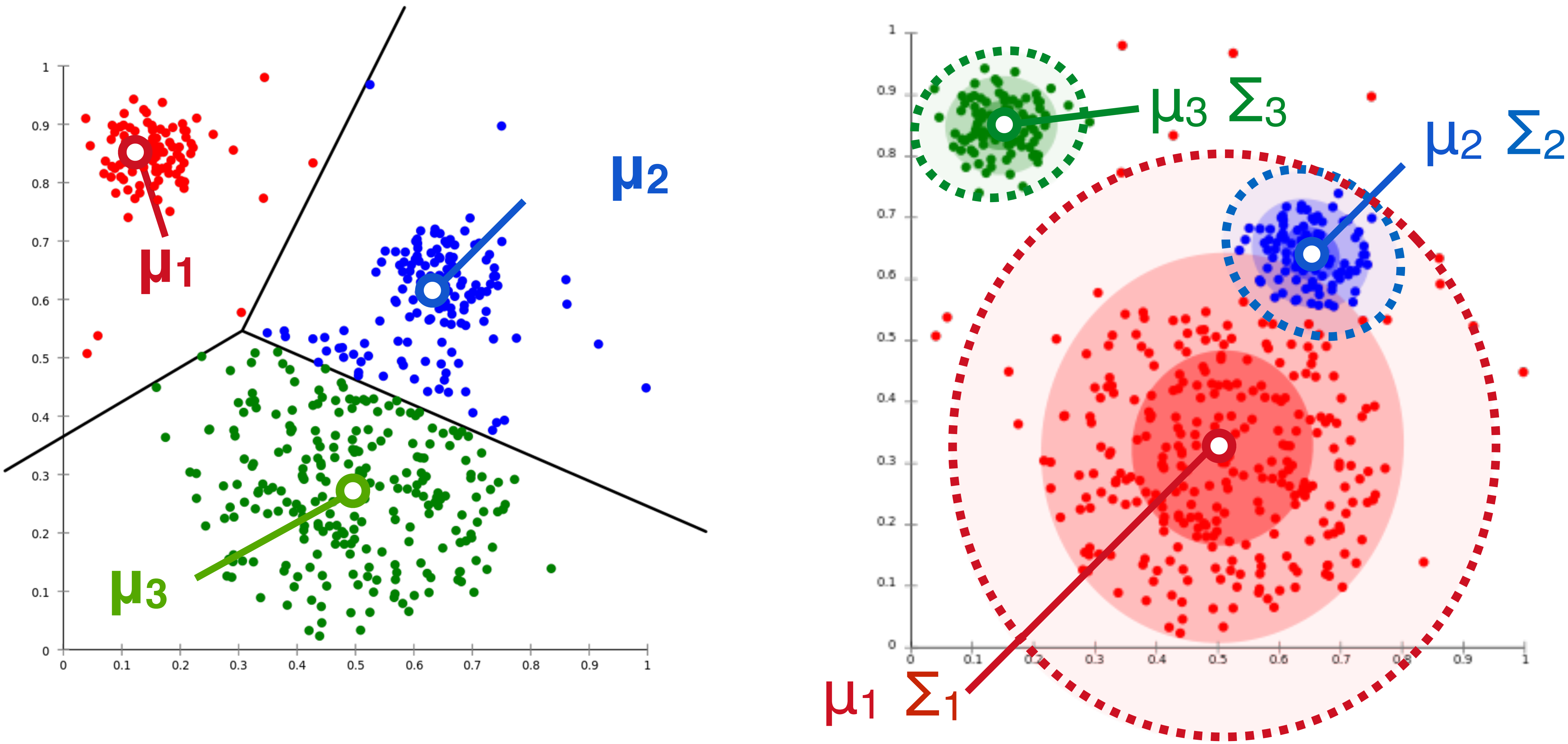
- What if we instead took a *probabilistic* view of clustering?

- *Mixture models* define a “generative story” for the data observed

Some slides derived from ***Matt Gormley and Eric Xing (CMU)***

K-Means vs *Gaussian Mixture Models*

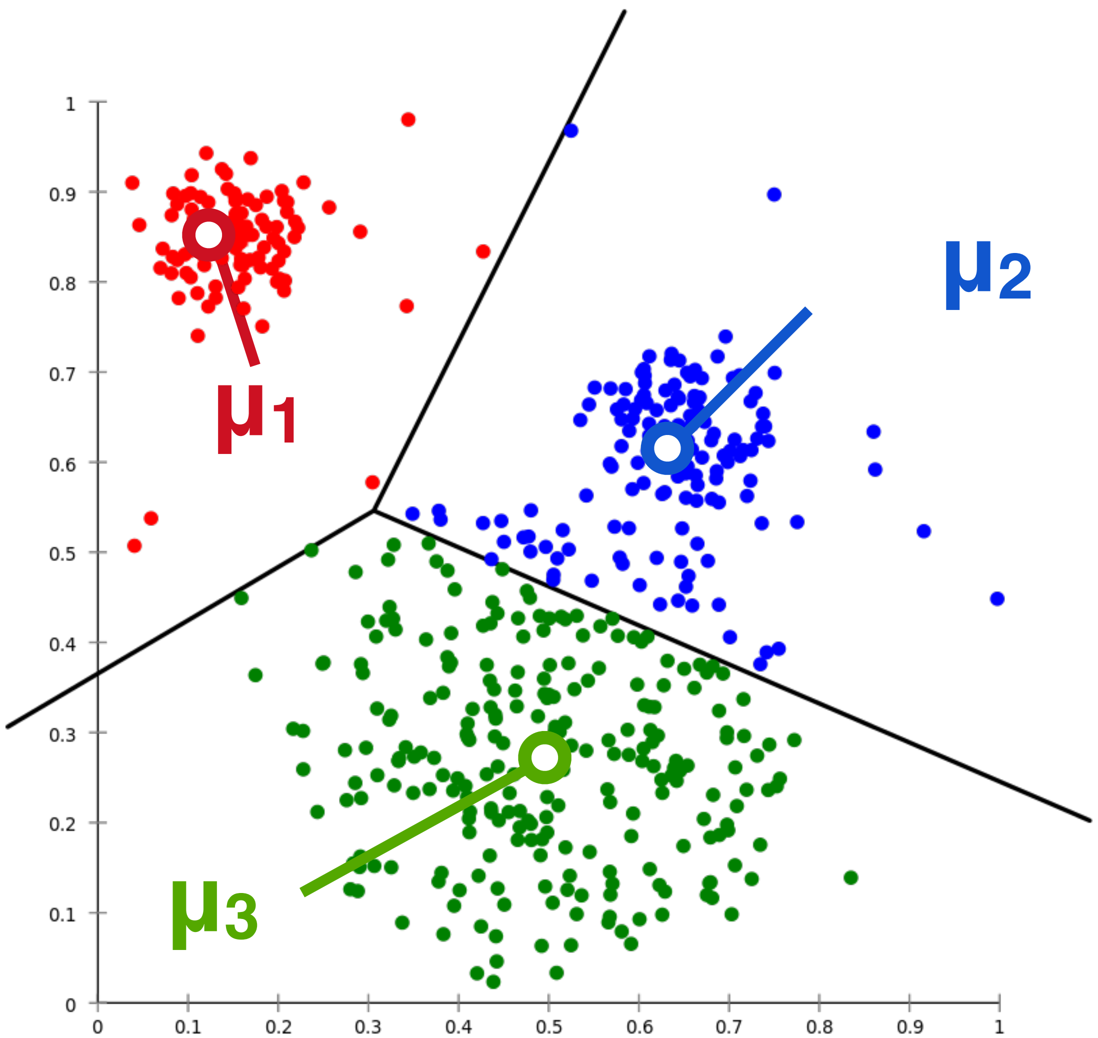
Idea: Learn both means μ_k and covariances Σ_k

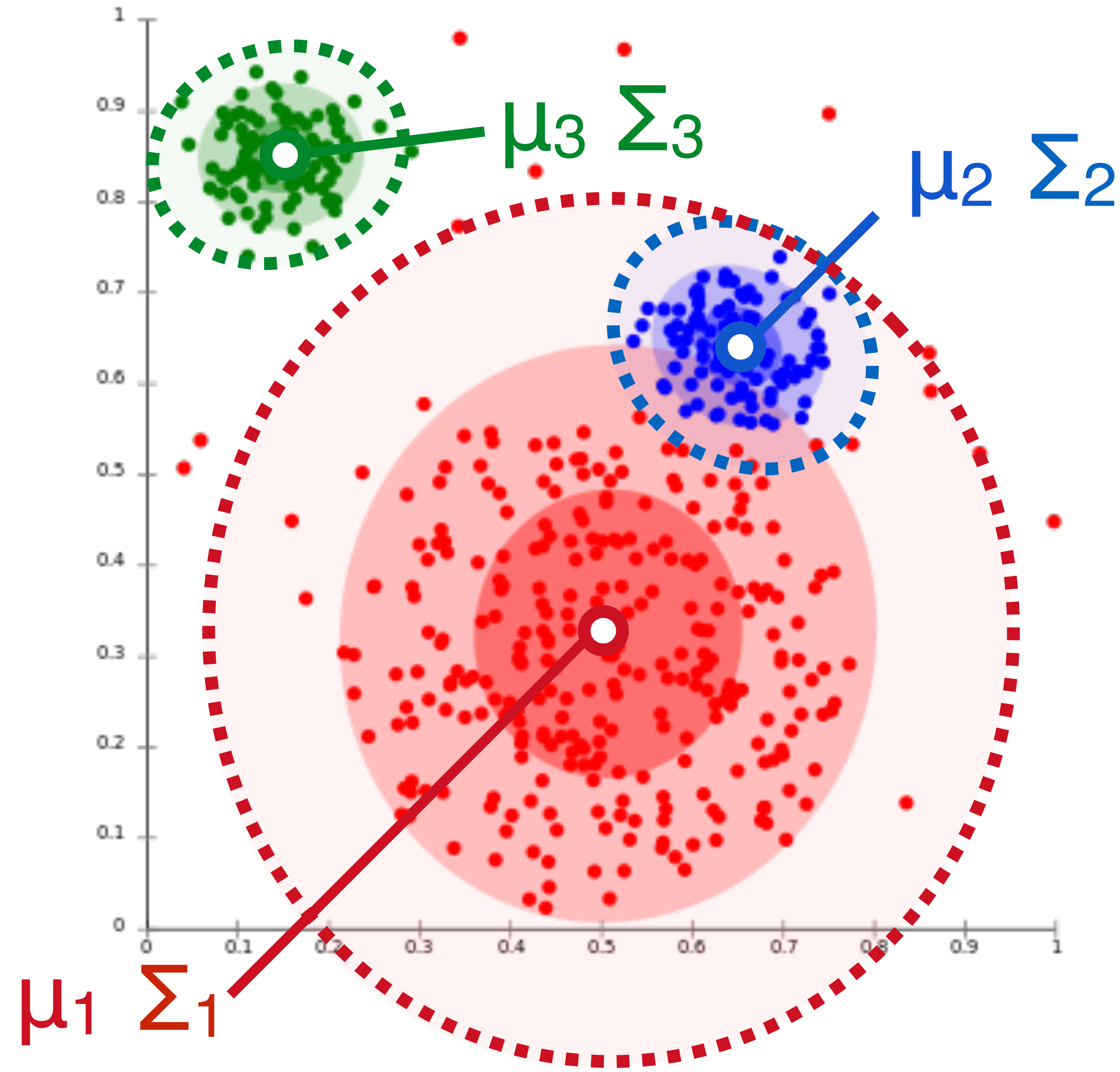


Don't just learn *where* the center of the cluster is, but also *how big it is*, and *what shape it has*.

K-Means vs Gaussian Mixture Models

Idea: Replace *hard* assignments with *soft* assignments





Hard assignments to clusters

$$\gamma_{nk} = I[z_n = k]$$

(one-hot vector)

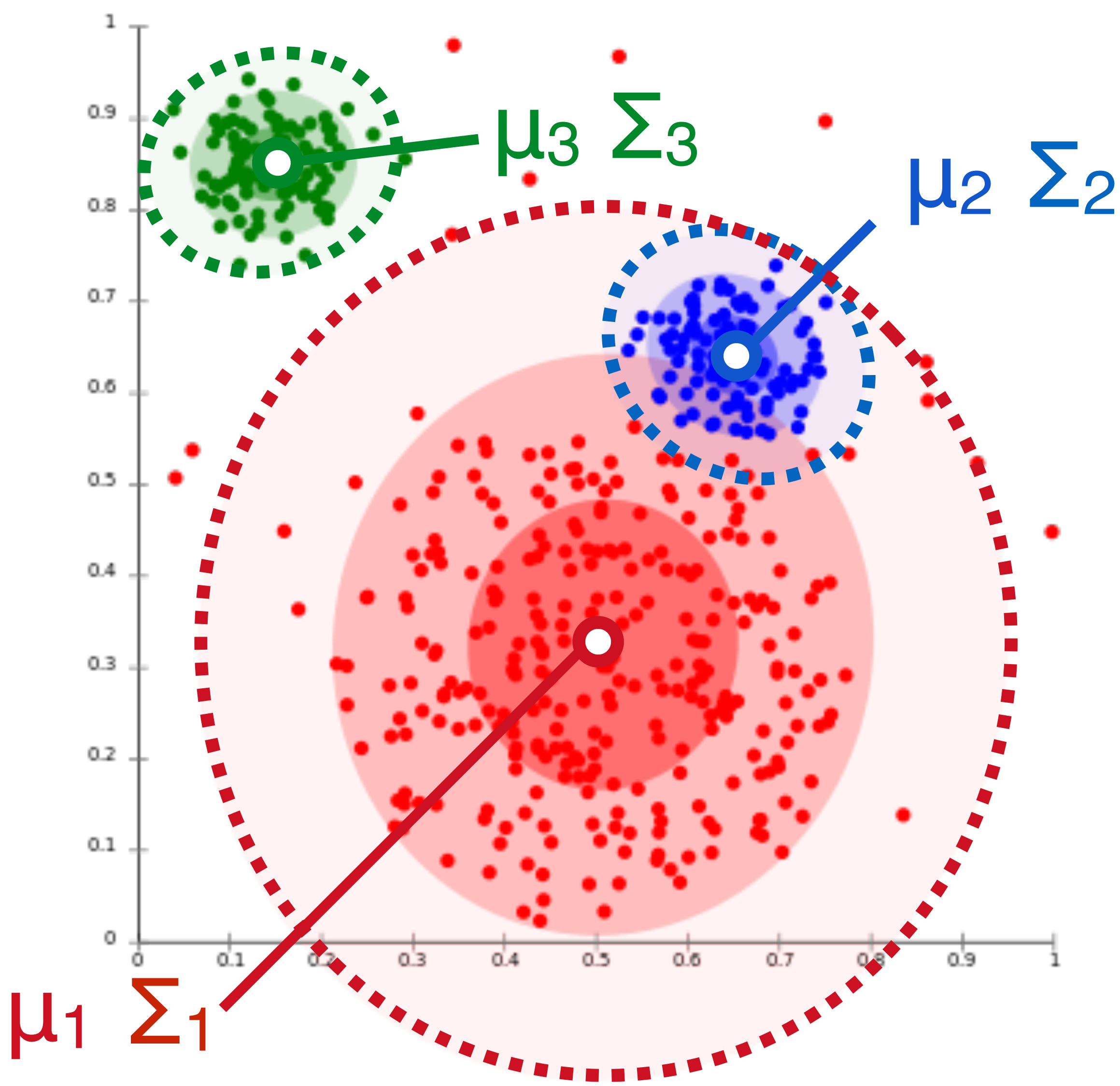
Soft assignments to clusters

$$\gamma_{nk} = p(z_n = k \mid \boldsymbol{x}_n)$$

(posterior probability)

Mixture models

Gaussian Mixture Models



Idea 1: Points in each cluster are sampled for a Gaussian

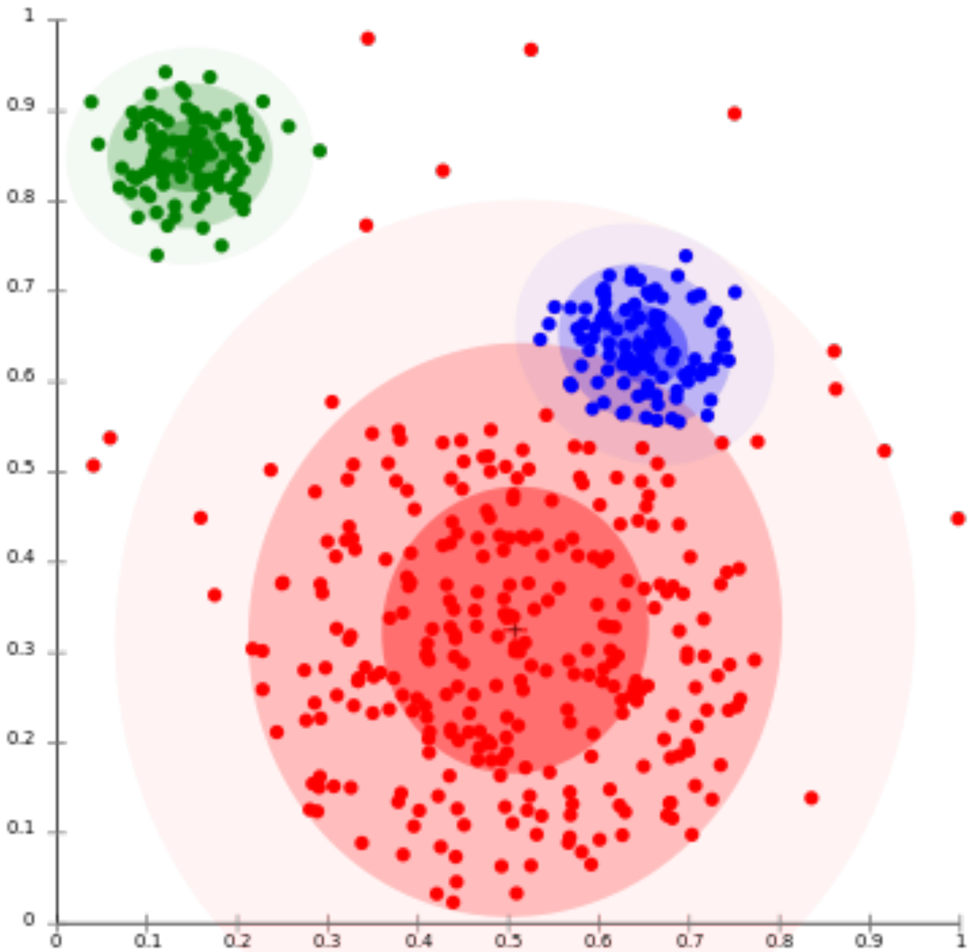
$$x_n \mid z_n = k \sim \mathrm{Norm}(\mu_k, \Sigma_k)$$

Idea 2: Compute probability that point belongs to each cluster

$$\gamma_{nk} = p(z_n = k \mid \boldsymbol{x}_n)$$

Weights sum to 1: $\sum_{k=1}^{K} \gamma_{nk} = 1$

“Hard EM” with Gaussians



$$\boldsymbol{\theta} := \{\mu_{1:K}, \Sigma_{1:K}, \pi\}$$

Algorithm

Initialize parameters to $\boldsymbol{\theta}^0$

Repeat until convergence

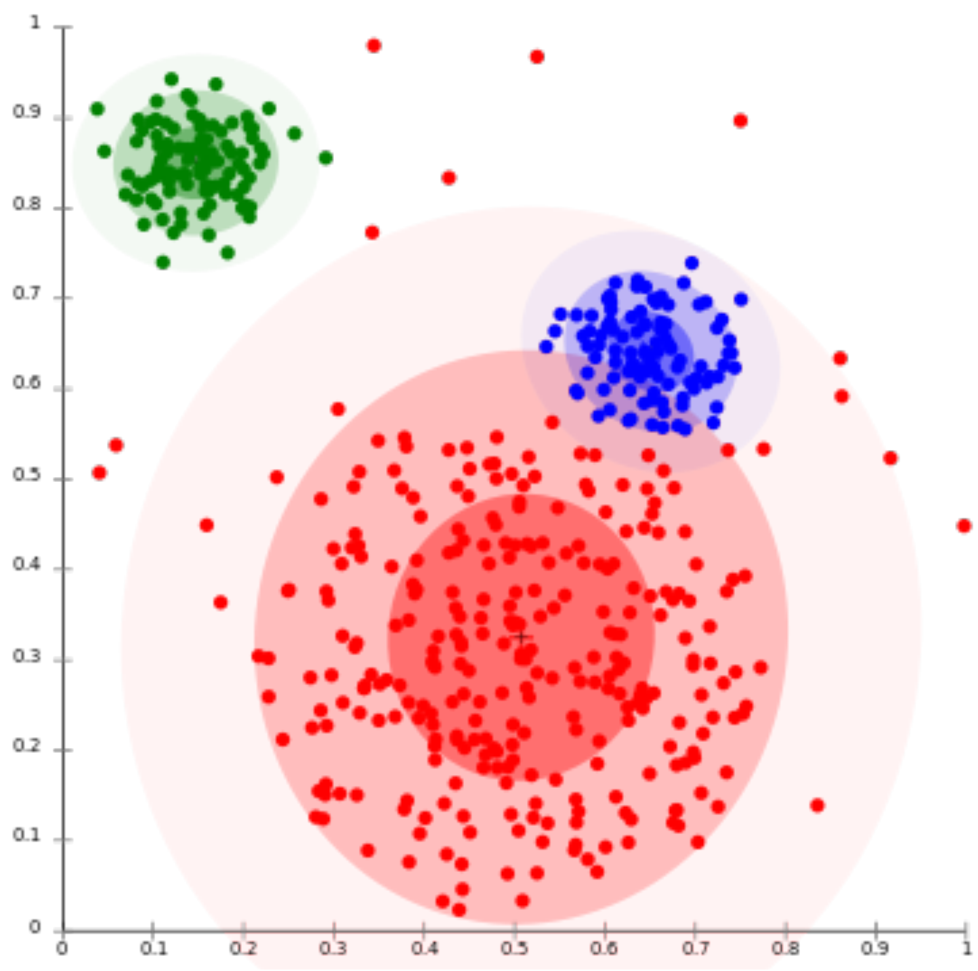
1. Update cluster assignments

$$\boldsymbol{z}^i = \underset{\boldsymbol{z}}{\operatorname{argmax}}\; p(\boldsymbol{X}, \boldsymbol{z} \mid \boldsymbol{\theta}^{i-1})$$

2. Update parameters

$$\boldsymbol{\theta}^i = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\; p(\boldsymbol{X}, \boldsymbol{z}^i \mid \boldsymbol{\theta})$$

“Hard EM” with Gaussians



Assignment Update

$$z_n = \underset{k}{\operatorname{argmax}}\, p(z_n = k \mid \boldsymbol{x}_n, \boldsymbol{\theta})$$

Parameter Updates

$$N_k := \sum_{n=1}^{N} z_{nk} \qquad z_{nk} := I[z_n = k]$$

$$\boldsymbol{\pi} = (N_1/N, \ldots, N_K/N)$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk}\, \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk}\, (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

“Hard” EM: General

Initialize parameters **randomly**

while not converged

1. **E-Step:**
   Set the latent variables to the the values that maximizes likelihood, treating parameters as observed
2. **M-Step:**
   Set the parameters to the values that maximizes likelihood, treating latent variables as observed

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

“Hard” EM: General

Initialize **parameters randomly**

while not converged

1. **E-Step:**
   Set the **latent variables** to the the values that maximizes likelihood, treating parameters as observed

   Hallucinate labels

2. **M-Step:**
   Set the **parameters** to the values that maximizes likelihood, treating latent variables as observed

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

"Hard" EM: General

Initialize **parameters randomly**

while not converged

1. **E-Step:**
   Set the **latent variables** to the the values that maximizes likelihood, treating parameters as observed

   Hallucinate labels

2. **M-Step:**
   Set the **parameters** to the values that maximizes likelihood, treating latent variables as observed

   Train (as if supervised)

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

Algorithm 1 Hard EM for MMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$)

Randomly initialize parameters, $\boldsymbol{\theta}, \boldsymbol{\phi}$

while not converged **do**

E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)}|z;\boldsymbol{\theta}) + \log p(z;\boldsymbol{\phi})$$

M-Step:

$$\phi \leftarrow \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(z^{(i)};\boldsymbol{\phi})$$

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|z;\boldsymbol{\theta})$$

return $(\boldsymbol{\phi}, \boldsymbol{\theta})$

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

Algorithm 1 Hard EM for MMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$)

Randomly initialize parameters, $\boldsymbol{\theta}, \phi$

while not converged **do**

E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)}|z;\boldsymbol{\theta}) + \log p(z;\boldsymbol{\phi})$$

Just loop over potential assignments

M-Step:

$$\phi \leftarrow \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(z^{(i)};\boldsymbol{\phi})$$

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|z;\boldsymbol{\theta})$$

return $(\phi, \boldsymbol{\theta})$

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

Algorithm 1 Hard EM for MMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$)

Randomly initialize parameters, $\boldsymbol{\theta}, \phi$

while not converged **do**

E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)} | z; \boldsymbol{\theta}) + \log p(z; \boldsymbol{\phi})$$

Just loop over potential assignments

M-Step:

$$\phi \leftarrow \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(z^{(i)}; \boldsymbol{\phi})$$

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} | z; \boldsymbol{\theta})$$

Supervised learning

return $(\phi, \boldsymbol{\theta})$

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

Algorithm 1 Hard EM for GMMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$)

    Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \mathbf{\Sigma}$

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

Algorithm 1 Hard EM for GMMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$)

  Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$

  while not converged **do**

    E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)} | z; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z; \boldsymbol{\phi})$$

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

Algorithm 1 Hard EM for GMMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$)

Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$

while not converged **do**

E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)} | z; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z; \boldsymbol{\phi})$$

M-Step:

$$\phi_k \leftarrow \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k), \forall k$$

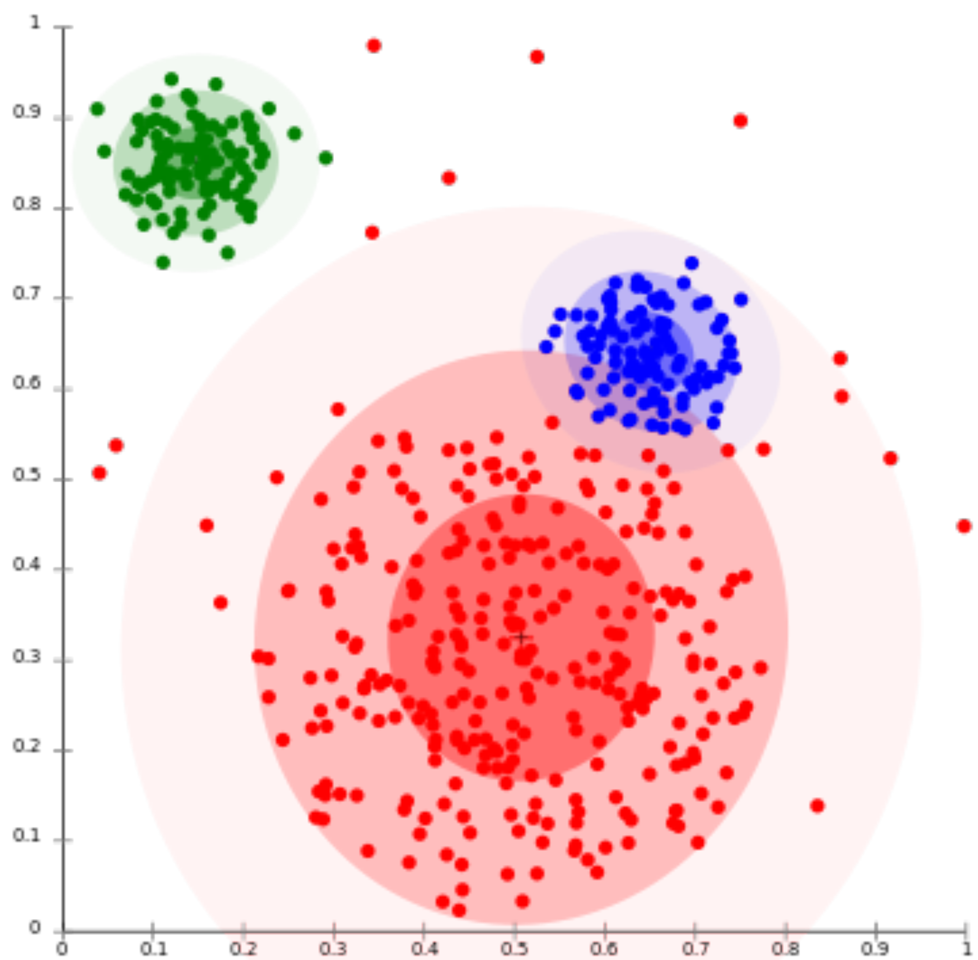
$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k) \mathbf{x}^{(i)}}{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k)}, \forall k$$

$$\boldsymbol{\Sigma}_k \leftarrow \frac{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k) (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k)}, \forall k$$

return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

“Hard EM” with Gaussians



Assignment Update

$$z_n = \operatorname*{argmax}_k p(z_n = k \mid \boldsymbol{x}_n, \boldsymbol{\theta})$$

Parameter Updates

$$N_k := \textstyle\sum_{n=1}^{N} z_{nk} \qquad z_{nk} := I[z_n = k]$$

$$\boldsymbol{\pi} = (N_1/N, \ldots, N_K/N)$$

$$\boldsymbol{\mu}_k = \tfrac{1}{N_k} \textstyle\sum_{n=1}^{N} z_{nk}\, \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \tfrac{1}{N_k} \textstyle\sum_{n=1}^{N} z_{nk}\, (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

How can we deal with overlapping clusters in a better way?

Learn ***Soft*** Assignments to Clusters

Posterior on Cluster Assignments (from Bayes' Rule)

Likelihood *Prior*

$$\gamma_{nk} = p(z_n = k \mid \boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n \mid z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)}$$

Posterior *Marginal Likelihood*

Learn *Soft* Assignments to Clusters

Posterior on Cluster Assignments (from Bayes' Rule)

Likelihood *Prior*

$$\gamma_{nk} = p(z_n = k \mid \boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n \mid z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)}$$

Posterior *Marginal Likelihood*

Prior

$$p(z_n = k) = \pi_k$$

Likelihood

$$p(\boldsymbol{x}_n \mid z_n = k) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)}$$

Marginal Likelihood

$$p(x_n) = \sum_{k=1}^{K} p(\boldsymbol{x}_n \mid z_n = k)p(z_n = k)$$

Learn *Gaussian* for Each Cluster

Maximum Likelihood Estimation

$$\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*, \boldsymbol{\pi}^* = \underset{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}}{\operatorname{argmax}} \quad \log p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})$$

Idea: Use weights $\gamma_{nk} = p(z_n = k \mid x_n)$ to compute estimates

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_n \gamma_{nk} \boldsymbol{x}_n \qquad N_k = \sum_n \gamma_{nk}$$

Cluster Mean

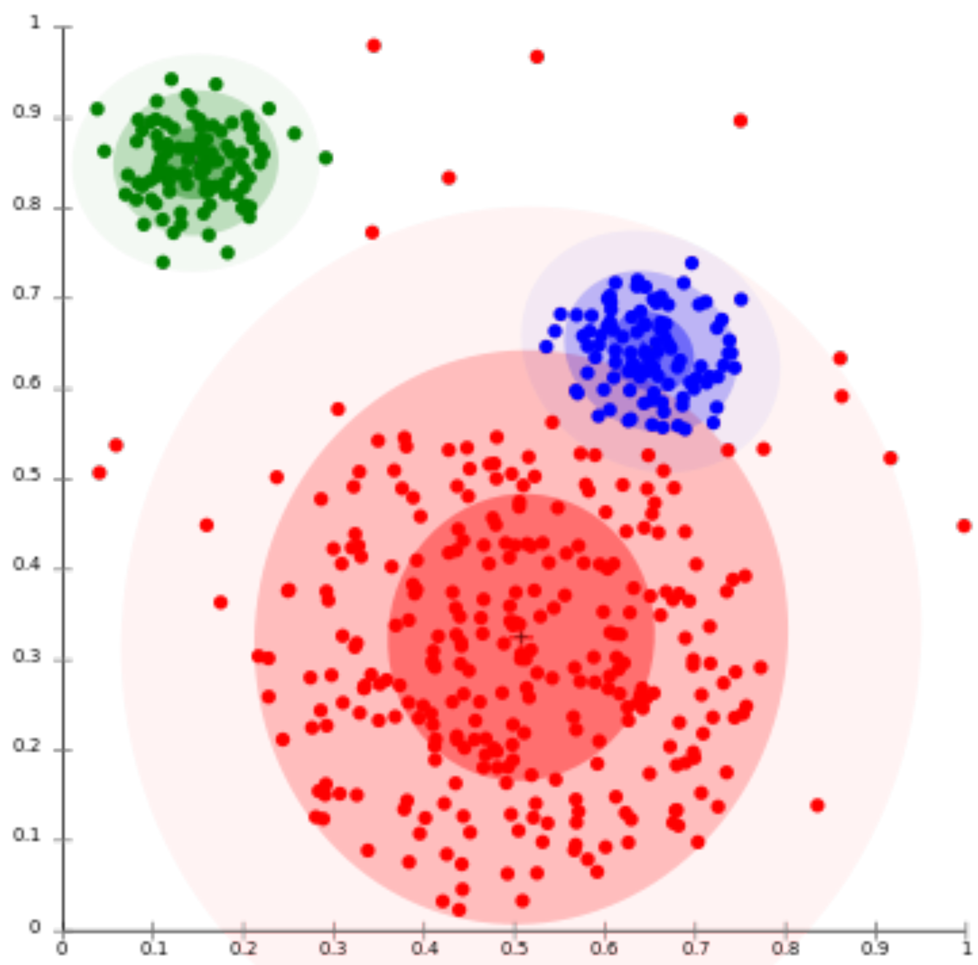
$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \mu_k)(\boldsymbol{x}_n - \mu_k)^\top$$

Cluster Covariance

$$\pi_k = \frac{N_k}{N}$$

Fraction of points in each cluster

"Hard EM" with Gaussians



Idea: Replace **hard** assignments with **soft** assignments

Assignment Update

$$z_n = \operatorname*{argmax}_k p(z_n = k \mid \boldsymbol{x}_n, \boldsymbol{\theta})$$

Parameter Updates

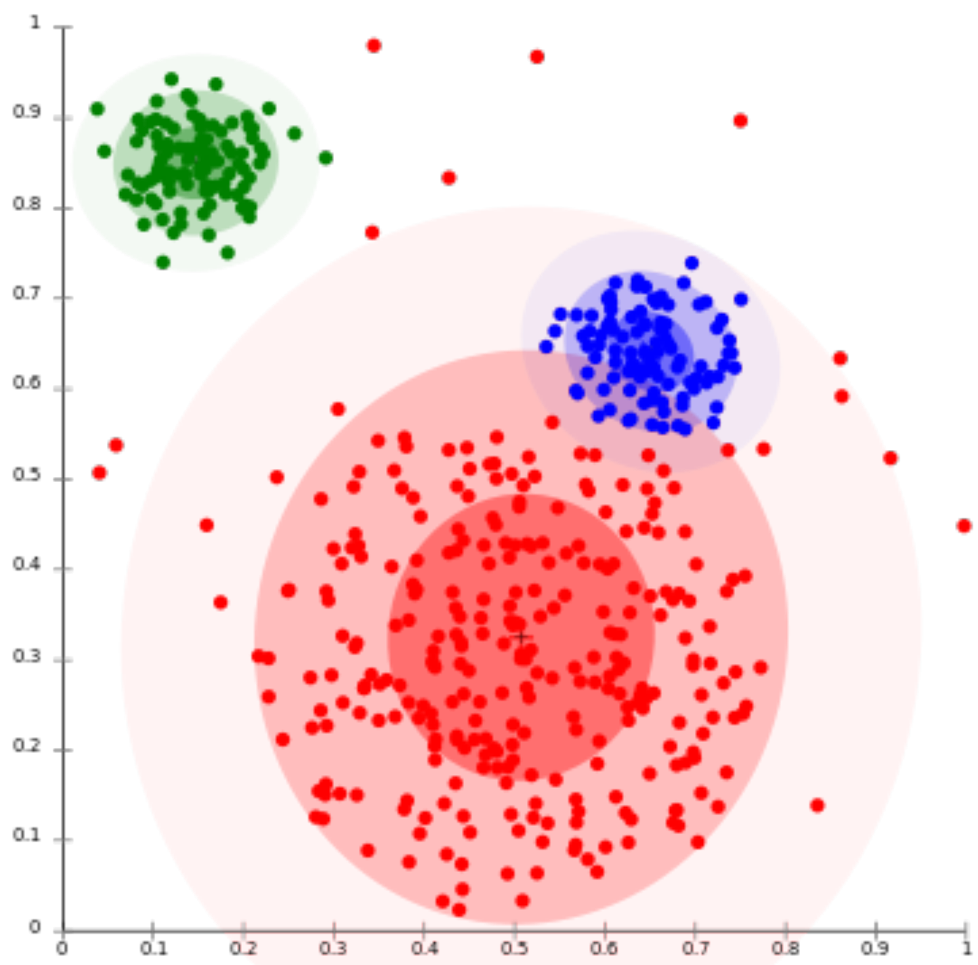
$$N_k := \sum_{n=1}^{N} z_{nk} \qquad z_{nk} := I[z_n = k]$$

$$\boldsymbol{\pi} = (N_1/N, \ldots, N_K/N)$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

Gaussian Mixture Models



Idea: Replace **hard** assignments with **soft** assignments

Soft Assignment Update

$$\gamma_{nk} := p(z_n = k \mid \boldsymbol{x}_n, \boldsymbol{\theta})$$

Parameter Updates

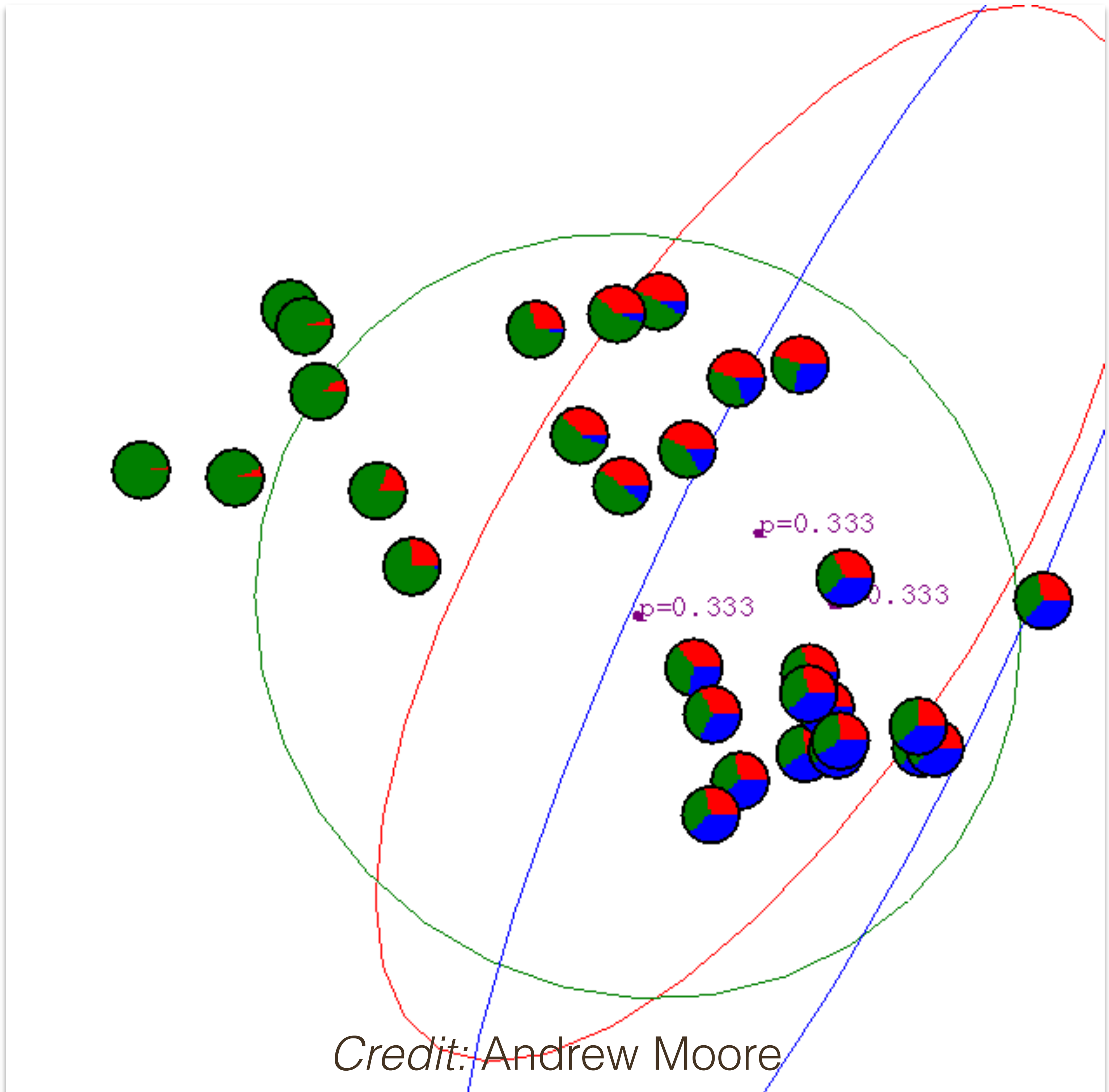
$$N_k := \sum_{n=1}^{N} \gamma_{nk}$$

$$\boldsymbol{\pi} = (N_1/N, \ldots, N_K/N)$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \boldsymbol{x}_n$$

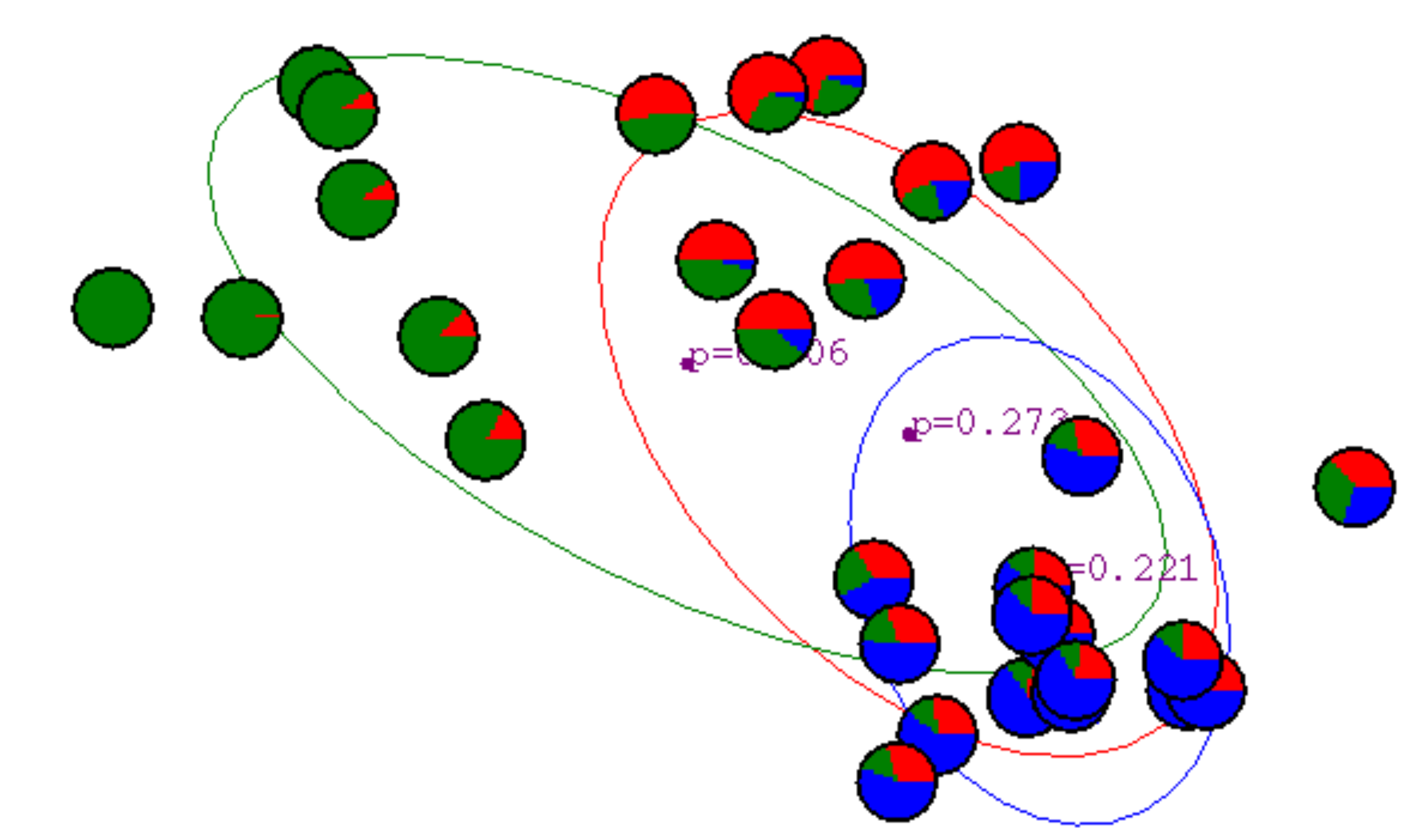
$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

EM for Gaussian Mixtures



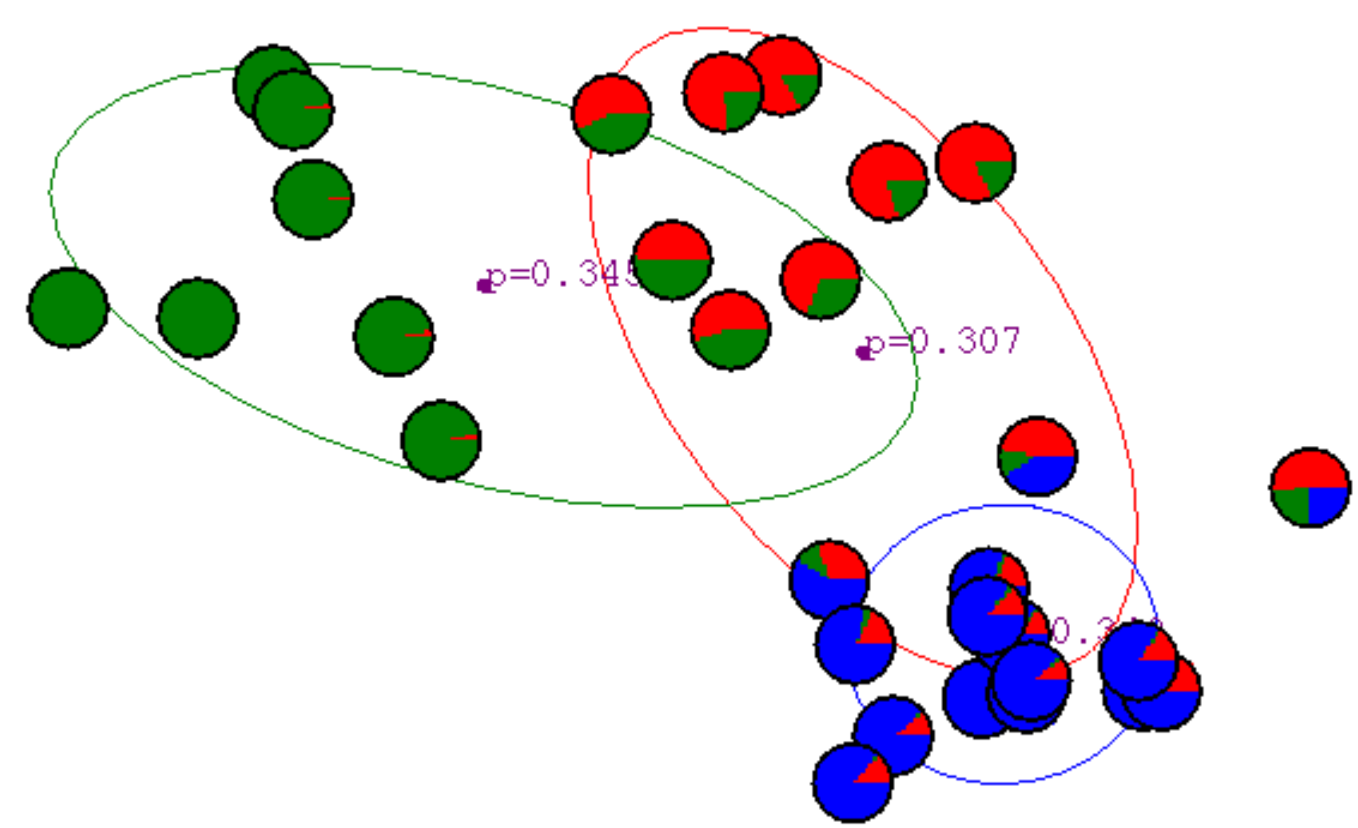
Credit: Andrew Moore

EM for Gaussian Mixtures



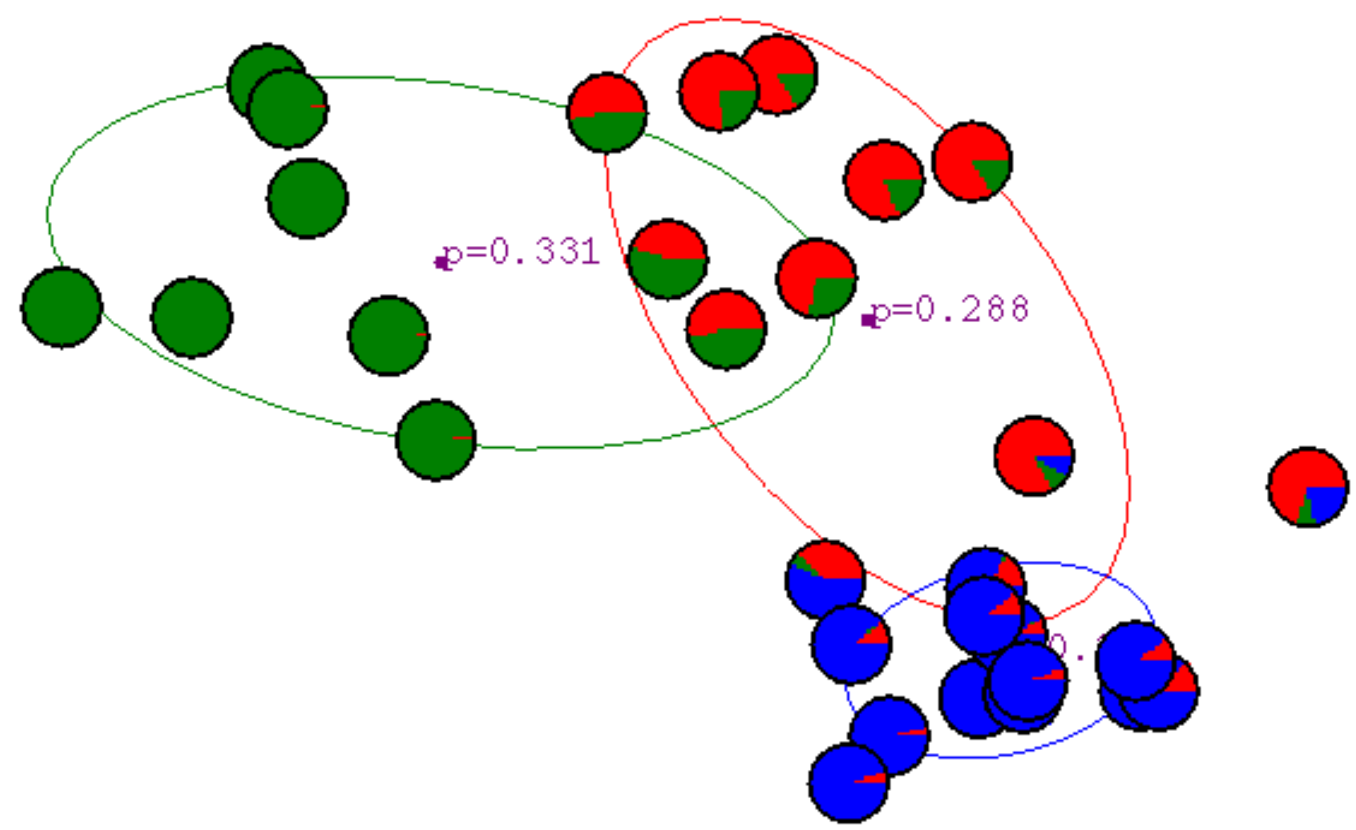
Credit: Andrew Moore

EM for Gaussian Mixtures



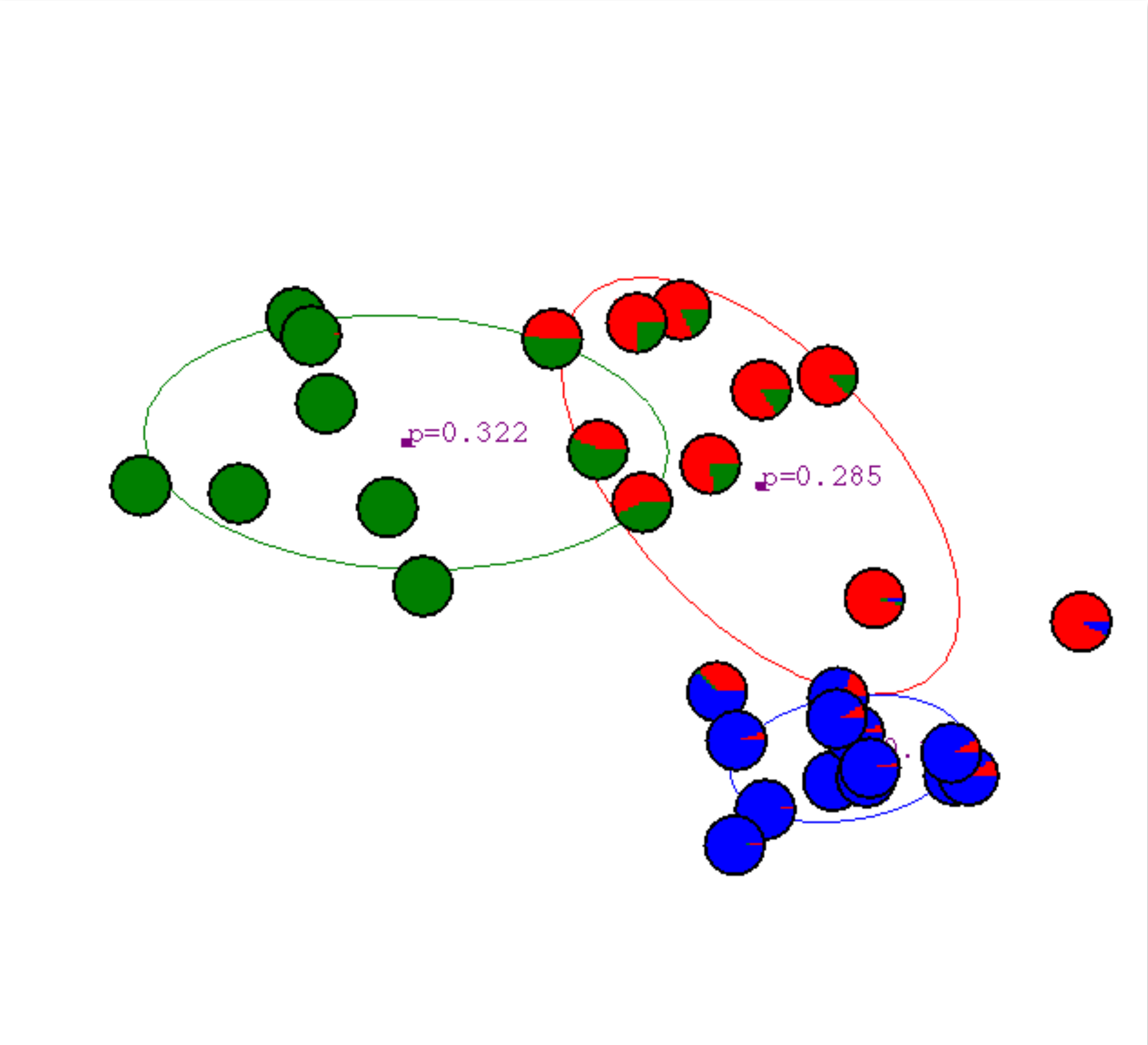
Credit: Andrew Moore

EM for Gaussian Mixtures



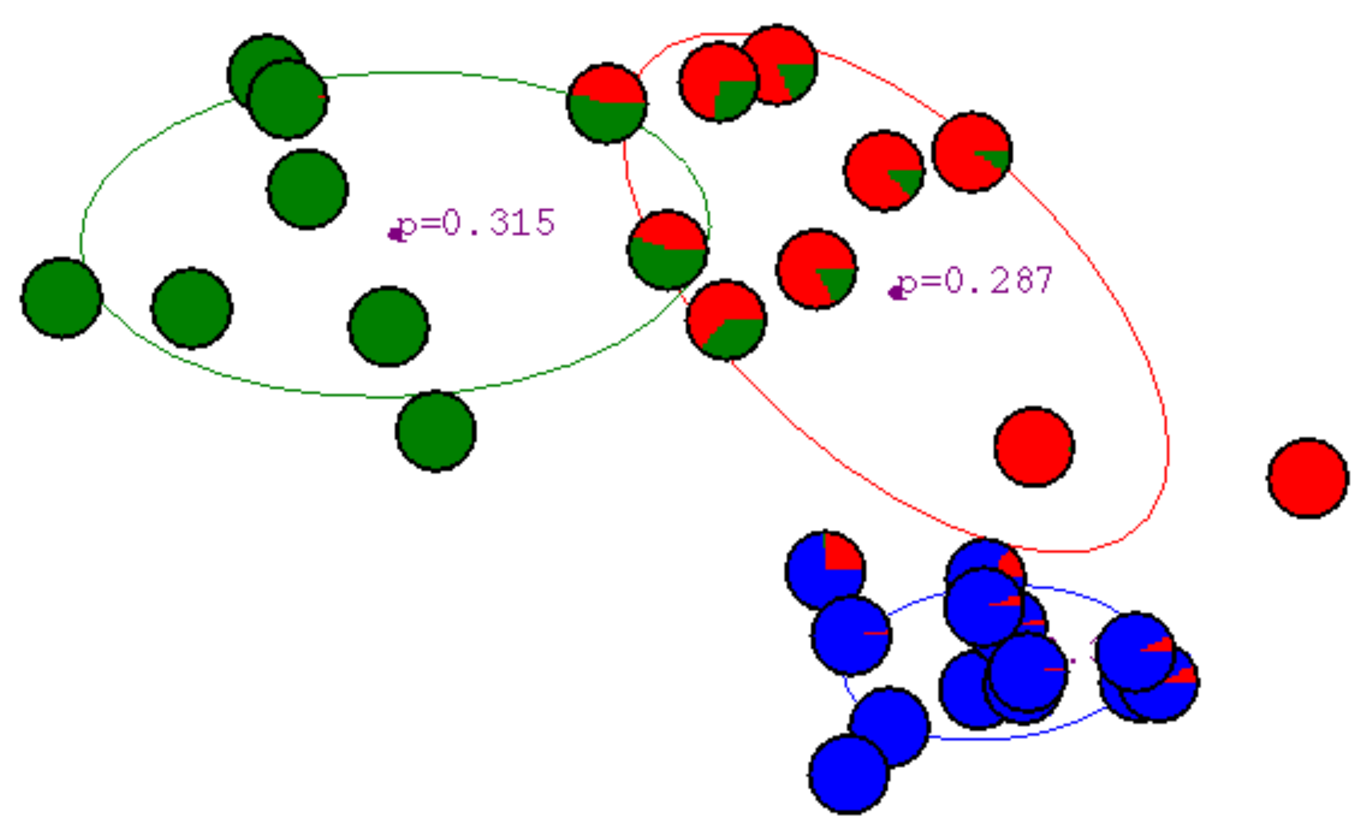
Credit: Andrew Moore

EM for Gaussian Mixtures



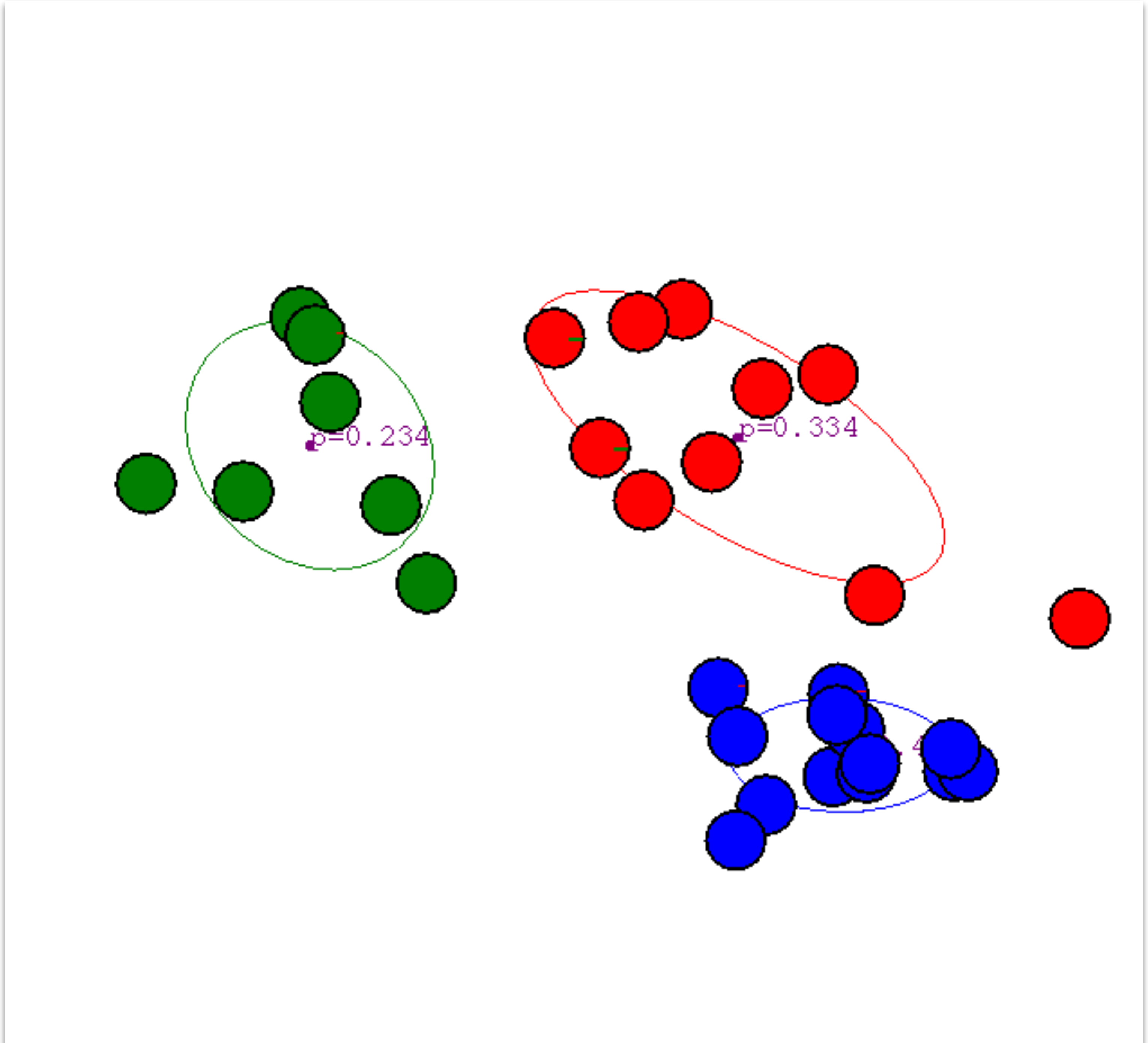
Credit: Andrew Moore

EM for Gaussian Mixtures



Credit: Andrew Moore

EM for Gaussian Mixtures



Credit: Andrew Moore

Consider Naive Bayes

The model

$$p(c|w_{1:N}, \pi, \theta) \propto p(c|\pi) \prod_{n=1}^{N} p(w_n|\theta_c)$$

$$p(\mathcal{D}|\theta_{1:C}, \pi) \quad = \quad \prod_{d=1}^{D} \left(p(c_d|\pi) \prod_{n=1}^{N} p(w_n|\theta_{c_d}) \right)$$

In-class exercise: How would we use EM here?

In-class exercise

Initialize **parameters randomly**

while not converged

1. **E-Step:**
   Set the **latent variables** to the the values that maximizes likelihood, treating parameters as observed
2. **M-Step:**
   Set the **parameters** to the values that maximizes likelihood, treating latent variables as observed

Slide credit: ***Matt Gormley and Eric Xing (CMU)***

Let's review
(on board)

Summing up

- *Mixture models* can be used to perform probabilistic clustering

Summing up

- *Mixture models* can be used to perform probabilistic clustering
- General idea: Assume instances are generated from distinct *components.* Each component has its own model parameters.

Summing up

- *Mixture models* can be used to perform probabilistic clustering

- General idea: Assume instances are generated from distinct *components.* Each component has its own model parameters.

- Fitting: More difficult here than in standard supervised learning because we do not observe z.
(One) **Solution**: Expectation-Maximization.