

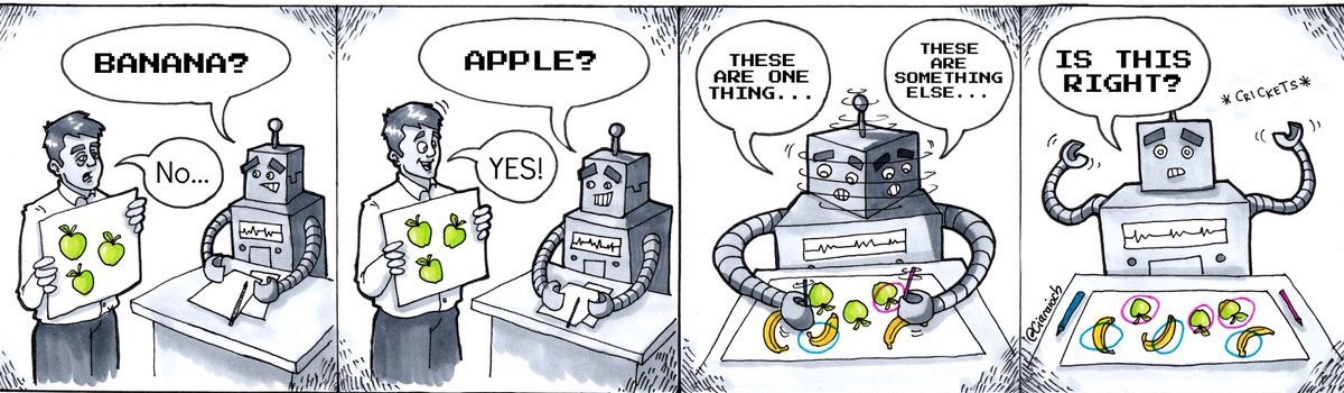
LMP 1210 - Basic principles of machine learning in biomedical research

Rahul G. Krishnan

Canada CIFAR AI Chair

Tier II Canada Research Chair
in Computational Medicine

Unsupervised Learning vs Supervised Learning

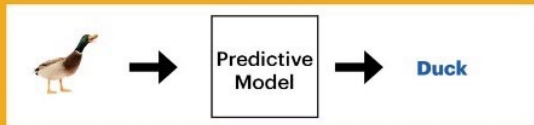
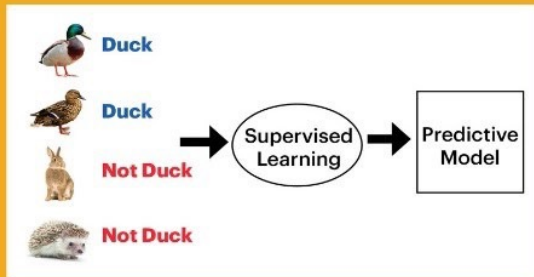


Supervised Learning

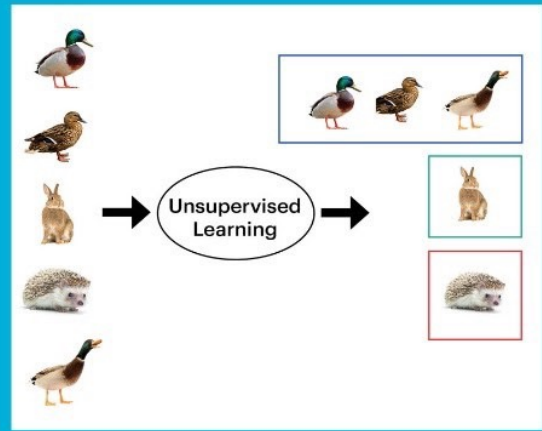
Unsupervised Learning

Unsupervised Learning vs Supervised Learning

Supervised Learning (Classification Algorithm)



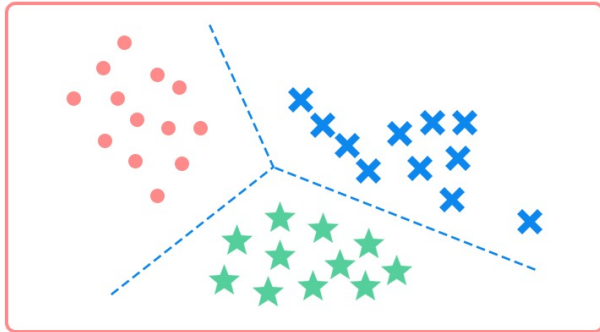
Unsupervised Learning (Clustering Algorithm)



Western Digital.

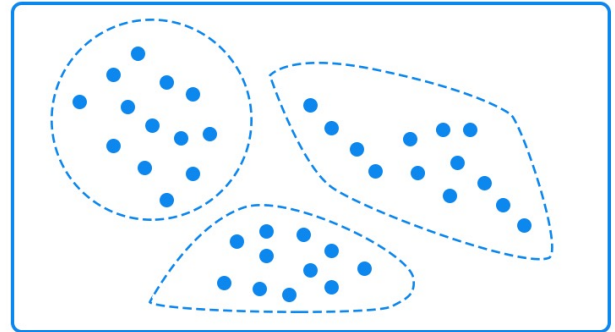
Unsupervised Learning vs Supervised Learning

Classification



Supervised learning

Clustering



Unsupervised learning

Unsupervised Learning vs Supervised Learning

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

Motivating Examples of Unsupervised Learning

- Some examples of situations where you would use **unsupervised learning**:
 - ▶ You want to understand how a scientific field has changed over time. You want to take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?
 - ▶ You are a biologist studying animal behaviour, so you want to infer a high-level description of their behaviour from video. You don't know the set of behaviour ahead of time.
 - ▶ You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (when refrigerator, washing machine, etc. were operating or not).
- Common theme: you have some data, and you want to infer the structure underlying the data.
- This structure is **latent**, which means it is not observed.

Clustering: Informal Goals

Goal: Automatically partition **unlabeled** data into groups of similar datapoints.

Question: When and why would we want to do this?

Useful for:

- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
 - Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).

Clustering: Applications

- Cluster news articles or web pages or search results by topic.



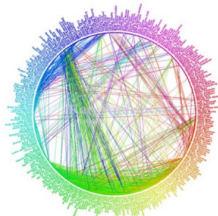
- Cluster protein sequences by function or genes according to expression profile.



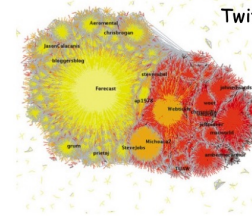
```

    * * * * * * * * * * * * * *
-NTREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-NTREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-NTREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-NTREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-NTREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-MARGGPDFETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-NYREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-NYREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-NTREGGFDIETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-MARGGPDFETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-MARGGPDFETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
-MARGGPDFETICSHRTPKRLINLLRQSRANTTEGKRELPDP--GGGG--ISITVILMAMVAVLVLELLELPPLEL---GPELPGK---SSPHS---QVFPAPPVGD--99
    
```

- Cluster users of social networks by interest (community detection).



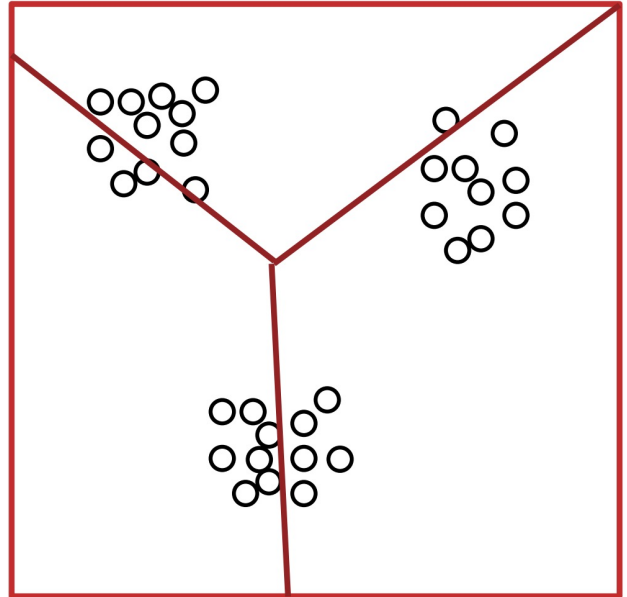
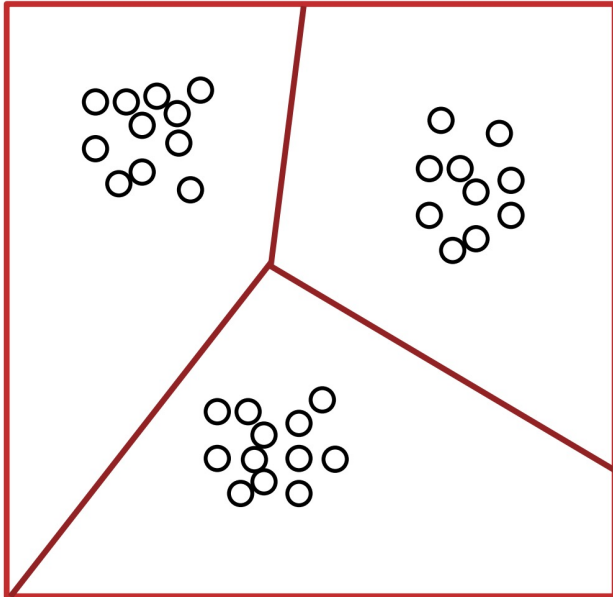
Facebook network



Twitter Network

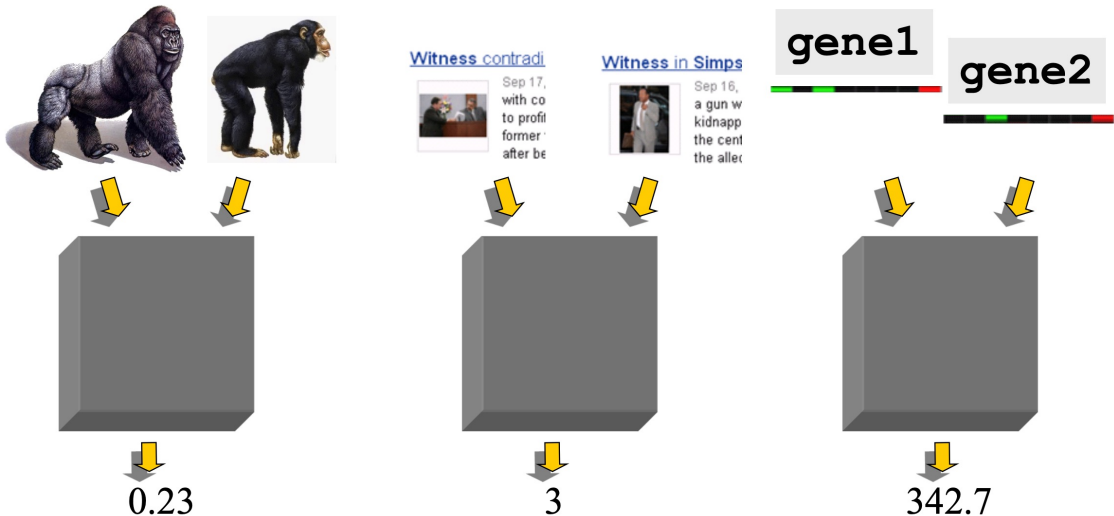
What is a good clustering?

Question: Which of these partitions is “better”?

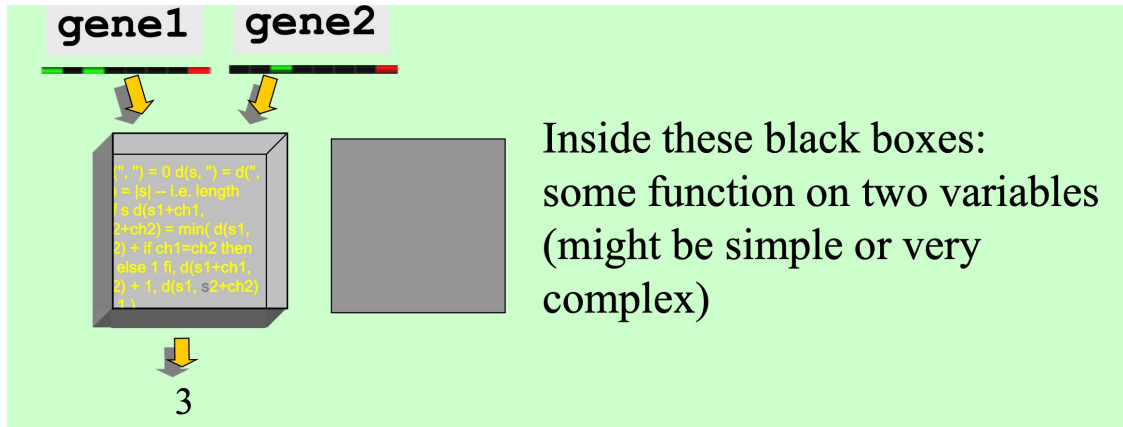


Clustering: Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1, O_2)$



Clustering: Distance Measures



A few examples:

- Euclidian distance

$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

- Correlation coefficient

$$s(x, y) = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y}$$

- Similarity rather than distance
- Can determine similar trends

Desirable Properties of Distance Measures

- Scalability (in terms of both time and space)
- Ability to deal with different data types
- Minimal requirements for domain knowledge to determine input parameters
- Interpretability and usability

Optional

- Incorporation of user-specified constraints

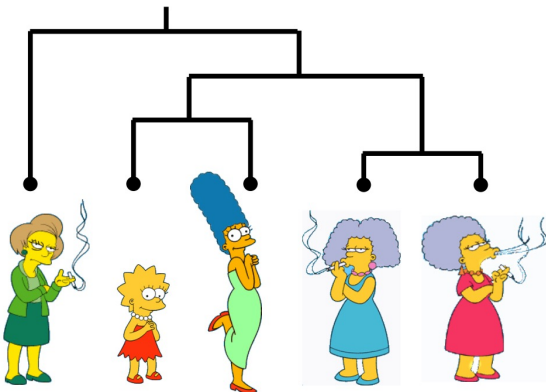
Two Types of Clustering

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion (focus of this class)

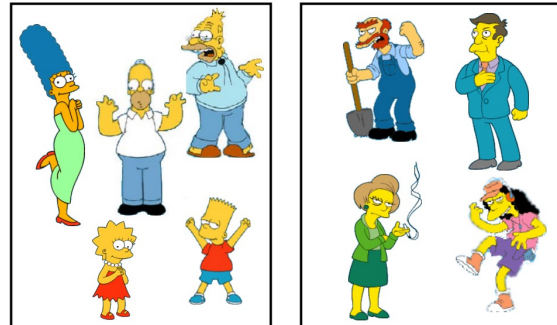
Bottom up or top down

Top down

Hierarchical



Partitional

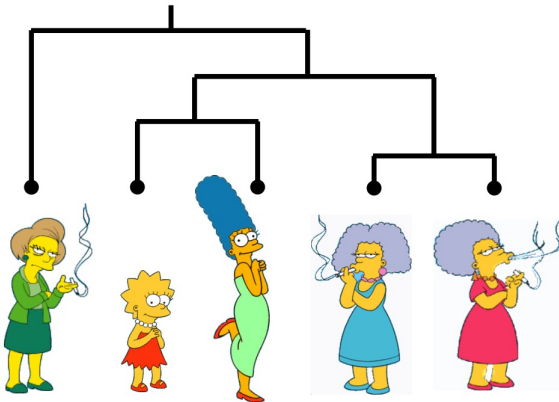


(How to) Hierarchical Clustering

The number of dendrograms with n leaves = $(2n - 3)! / [(2^{n-2}) (n - 2)!]$

Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	...
10	34,459,425

Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

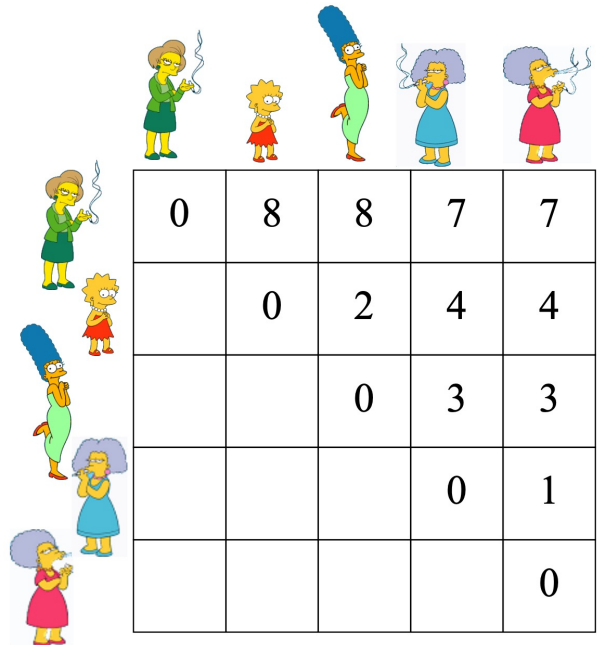


(How to) Hierarchical Clustering











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


$$D(\text{Marge}, \text{Lisa}) = 8$$


$$D(\text{Maggie}, \text{Edna}) = 1$$



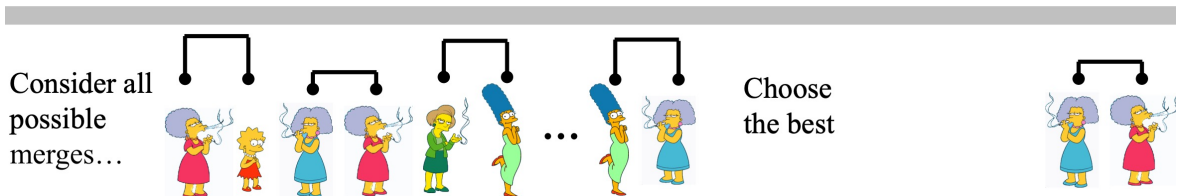
A distance matrix for five Simpsons characters: Marge Simpson, Lisa Simpson, Marge Simpson (with tall beehive), Edna Krabappel, and Maggie Simpson. The matrix is lower triangular, with the diagonal elements all being 0. The values in the matrix are: (Marge, Lisa) = 8, (Marge, Marge with beehive) = 8, (Marge, Edna) = 7, (Marge, Maggie) = 7, (Lisa, Marge with beehive) = 2, (Lisa, Edna) = 4, (Lisa, Maggie) = 4, (Marge with beehive, Edna) = 3, (Marge with beehive, Maggie) = 3, (Edna, Maggie) = 1, and (Maggie, Maggie) = 0.

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

(How to) Hierarchical Clustering

Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

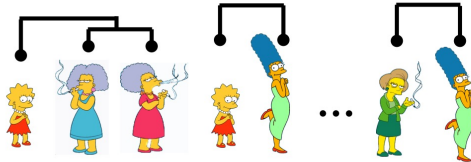


(How to) Hierarchical Clustering

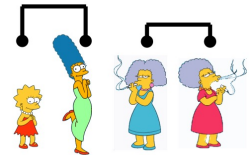
Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

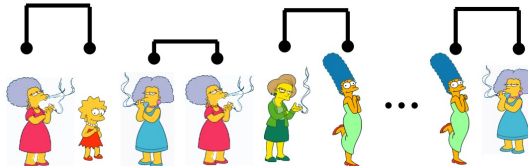
Consider all possible merges...



Choose the best



Consider all possible merges...



Choose the best

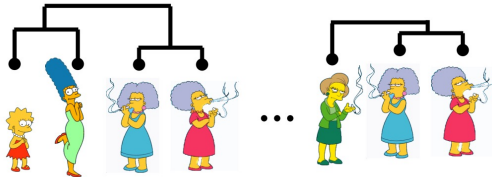


(How to) Hierarchical Clustering

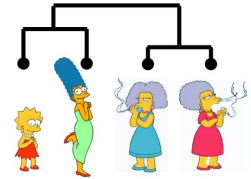
Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

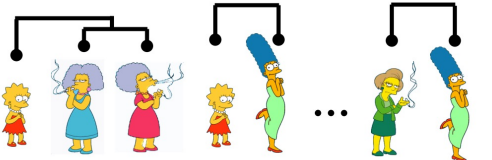
Consider all possible merges...



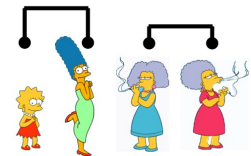
Choose the best



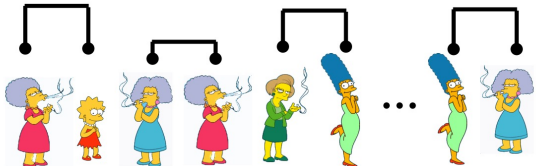
Consider all possible merges...



Choose the best



Consider all possible merges...



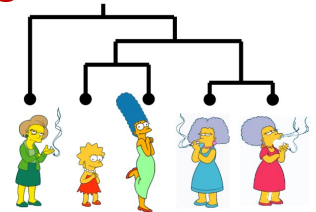
Choose the best



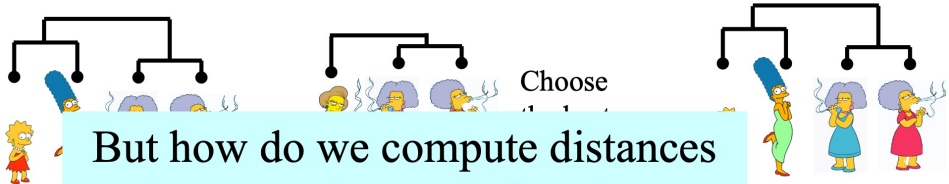
(How to) Hierarchical Clustering

Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

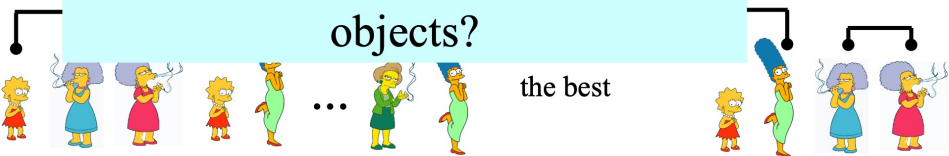


Consider all possible merges...



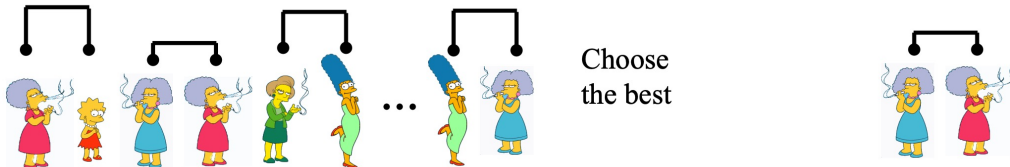
Choose the best

Consider all possible merges...



the best

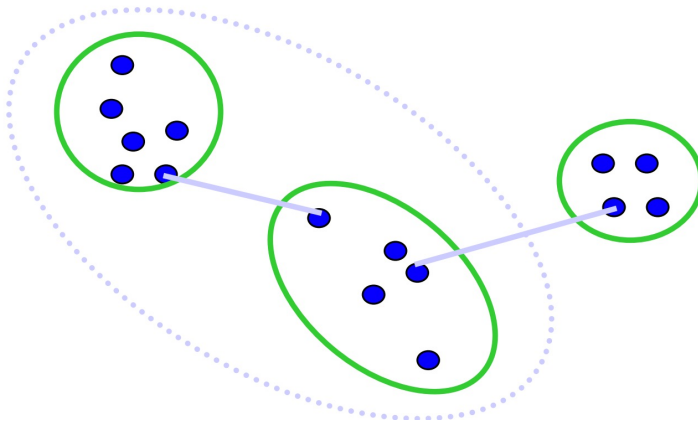
Consider all possible merges...



Choose the best

Computing distance between clusters: Single Link

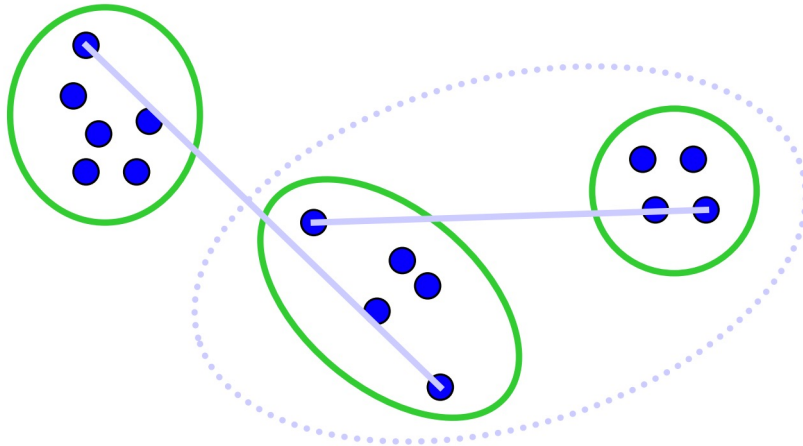
- cluster distance = distance of two **closest** members in each class



- Potentially
long and skinny
clusters

Computing distance between clusters: Complete Link

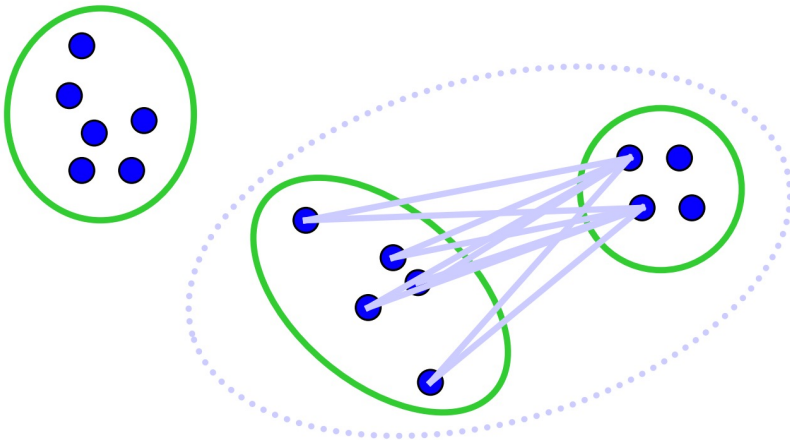
- cluster distance = distance of two farthest members



+ tight clusters

Computing distance between clusters: Average Link

- cluster distance = average distance of all pairs

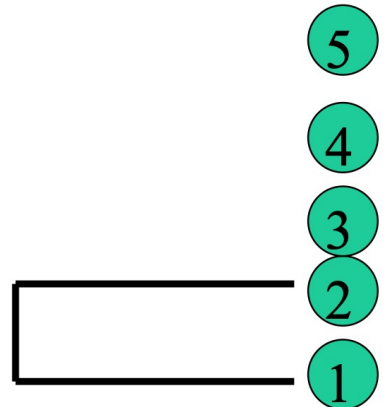


**the most widely
used measure**

**Robust against
noise**

Example: Single Link

	1	2	3	4	5
1	0				
2	2	0			
3	6	3	0		
4	10	9	7	0	
5	9	8	5	4	0



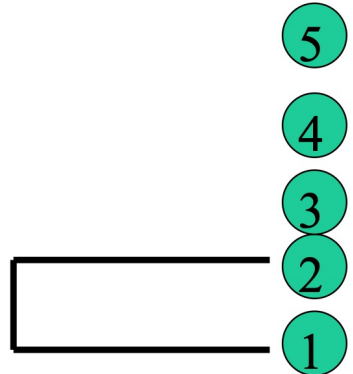
Example: Single Link

$$\begin{array}{c} 1 \ 2 \ 3 \ 4 \ 5 \\ \left[\begin{array}{ccccc} 0 & & & & \\ 2 & 0 & & & \\ 6 & 3 & 0 & & \\ 10 & 9 & 7 & 0 & \\ 9 & 8 & 5 & 4 & 0 \end{array} \right] \end{array} \quad \rightarrow \quad \begin{array}{c} (1,2) \ 3 \ 4 \ 5 \\ \left[\begin{array}{cccc} 0 & & & \\ 3 & 0 & & \\ 4 & 9 & 7 & 0 \\ 5 & 8 & 5 & 4 & 0 \end{array} \right] \end{array}$$

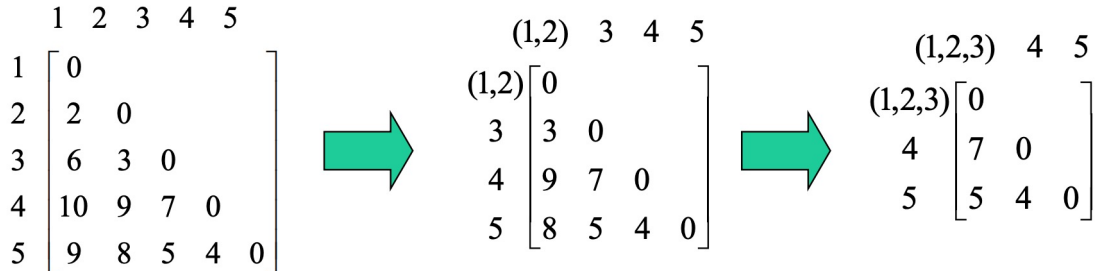
$$d_{(1,2),3} = \min\{d_{1,3}, d_{2,3}\} = \min\{6,3\} = 3$$

$$d_{(1,2),4} = \min\{d_{1,4}, d_{2,4}\} = \min\{10,9\} = 9$$

$$d_{(1,2),5} = \min\{d_{1,5}, d_{2,5}\} = \min\{9,8\} = 8$$

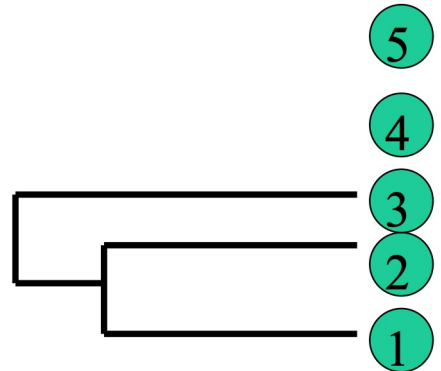


Example: Single Link

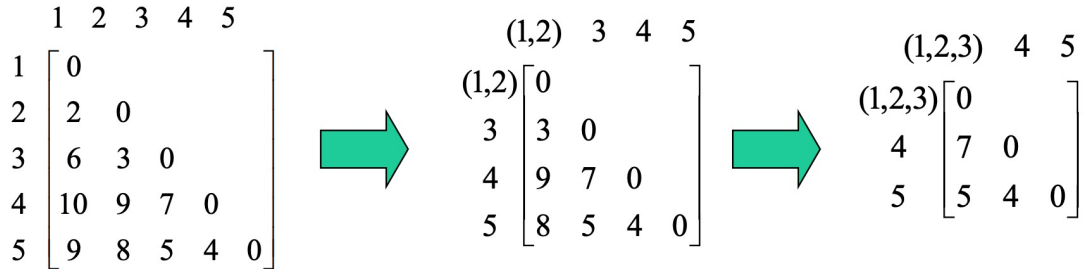


$$d_{(1,2,3),4} = \min\{d_{(1,2),4}, d_{3,4}\} = \min\{9, 7\} = 7$$

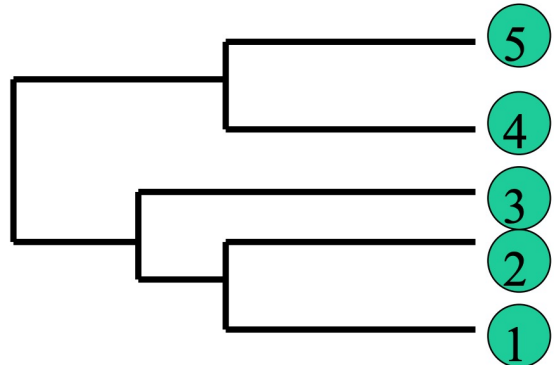
$$d_{(1,2,3),5} = \min\{d_{(1,2),5}, d_{3,5}\} = \min\{8, 5\} = 5$$



Example: Single Link



$$d_{(1,2,3),(4,5)} = \min\{d_{(1,2,3),4}, d_{(1,2,3),5}\} = 5$$

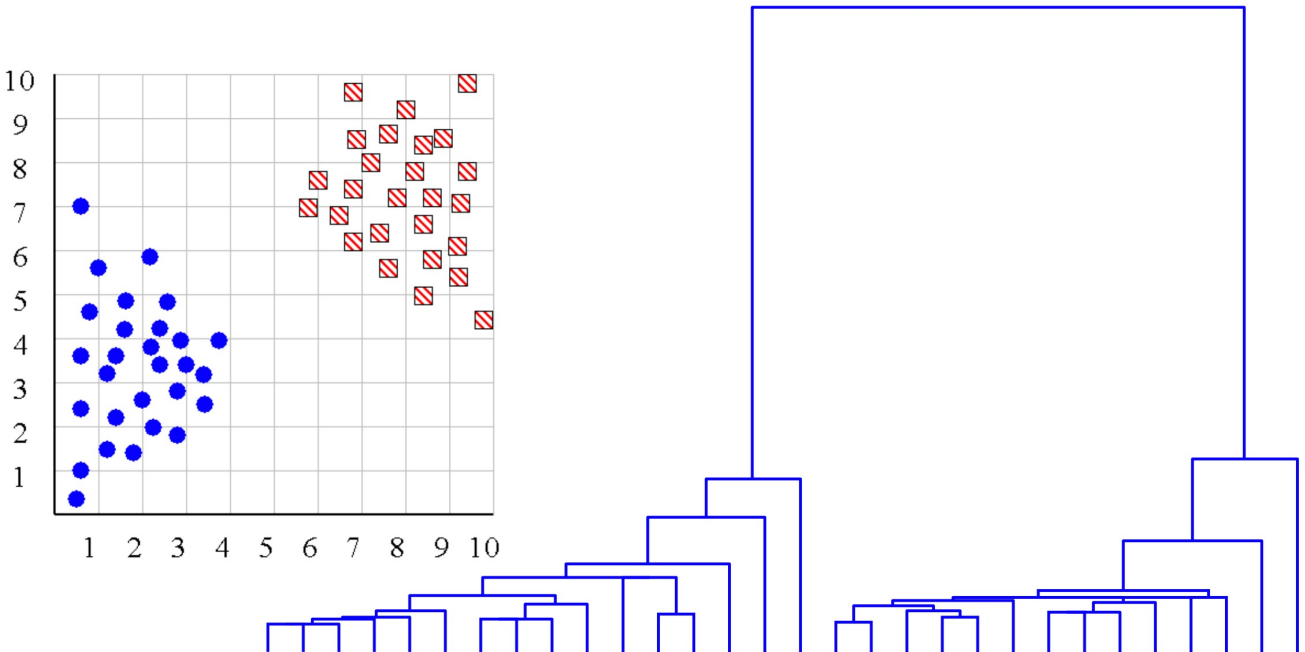


Summary: Hierarchical Clustering

- No need to specify the number of clusters in advance.
- Hierarchical structure maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects.
- Like any heuristic search algorithms, local optima are a problem.
- Interpretation of results is (very) subjective.

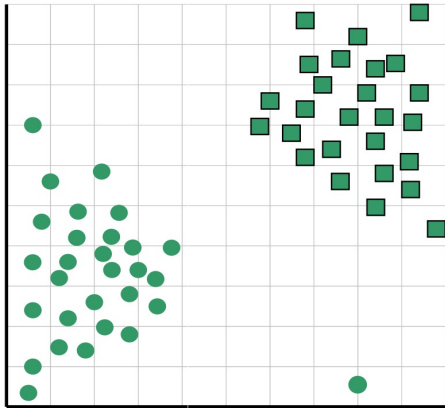
How many clusters?

In some cases we can determine the “correct” number of clusters. However, things are rarely this clear cut, unfortunately.

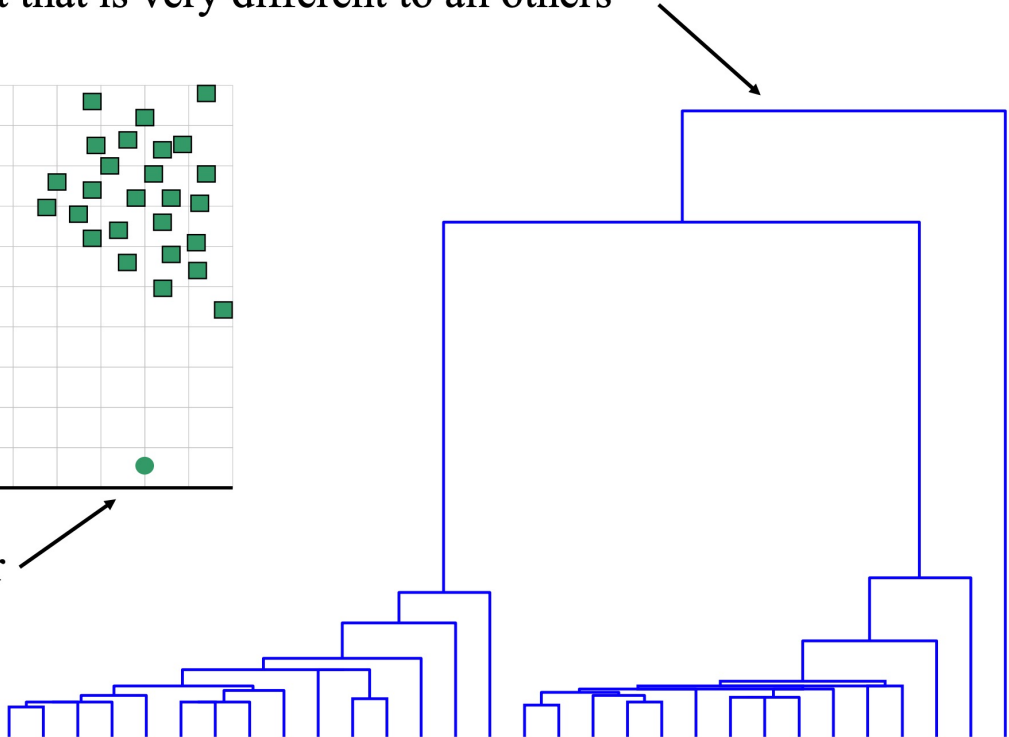


How many clusters?

The single isolated branch is suggestive of a data point that is very different to all others

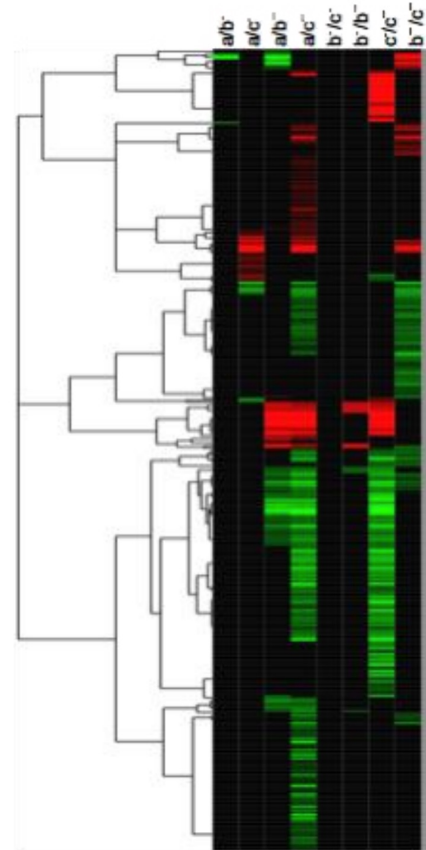


Outlier



Hierarchical Clustering for RNA-seq data

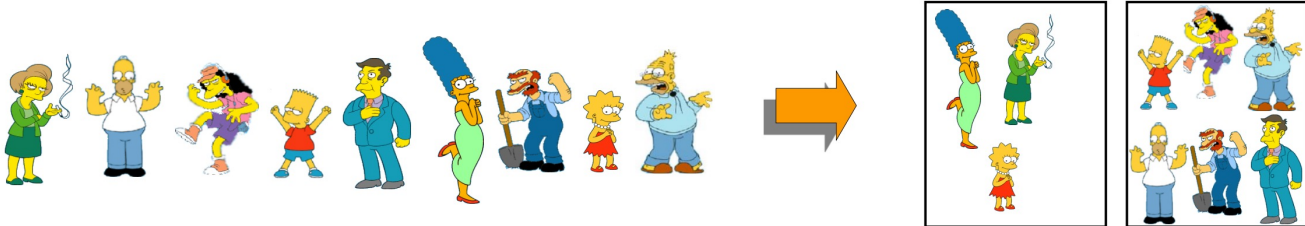
- Microarrays measures the activities of all genes in different conditions
- Clustering genes can help determine new functions for unknown genes



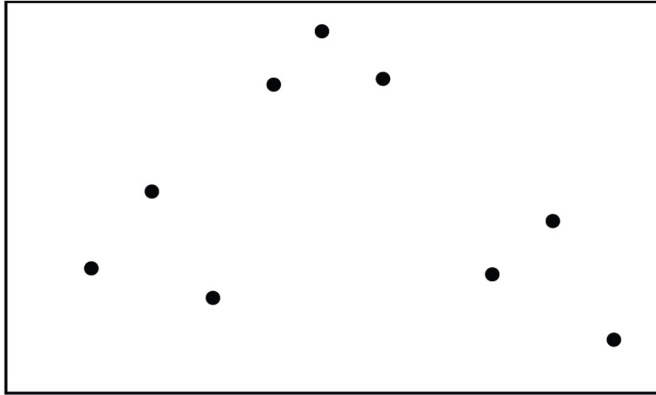
Break

K-means

- Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.
- Since the output is only one set of clusters the user has to specify the desired number of clusters K .



K-means : Setup



- Assume that the data points $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ live in an Euclidean space, i.e., $\mathbf{x}^{(n)} \in \mathbb{R}^D$.
- Assume that each data point belongs to one of K clusters
- Assume that the data points from same cluster are similar, i.e., close in Euclidean distance.
- How can we identify those clusters and the data points that belong to each cluster?

K-means : Objectives

Let's formulate this as an optimization problem

- **K-means Objective:**

Find cluster centres $\{\mathbf{m}_k\}_{k=1}^K$ and assignments $\{\mathbf{r}^{(n)}\}_{n=1}^N$ to minimize the sum of squared distances of data points $\{\mathbf{x}^{(n)}\}$ to their assigned cluster centres

- ▶ Data sample $n = 1, \dots, N$: $\mathbf{x}^{(n)} \in \mathbb{R}^D$ (observed),
- ▶ Cluster centre $k = 1, \dots, K$: $\mathbf{m}_k \in \mathbb{R}^D$ (not observed),
- ▶ Responsibilities: Cluster assignment for sample n :
 $\mathbf{r}^{(n)} \in \mathbb{R}^K$ 1-of-K encoding (not observed)

- Mathematically:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} J\left(\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}\right) = \min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \left\| \mathbf{m}_k - \mathbf{x}^{(n)} \right\|^2,$$

where $r_k^{(n)} = \mathbb{I}\{\mathbf{x}^{(n)} \text{ is assigned to cluster } k\}$, e.g.,
 $\mathbf{r}^{(n)} = [0, \dots, 1, \dots, 0]^\top$.

- Finding an optimal solution is an NP-hard problem!

K-means : Optimization

- Optimization problem:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \underbrace{\sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \left\| \mathbf{m}_k - \mathbf{x}^{(n)} \right\|^2}_{\text{distance between } \mathbf{x}^{(n)} \text{ and its assigned cluster centre}}$$

- Since $r_k^{(n)} = \mathbb{I}\{\mathbf{x}^{(n)} \text{ is assigned to cluster } k\}$ (e.g., $\mathbf{r}^{(n)} = [0, \dots, 1, \dots, 0]^\top$), the inner sum is over K terms but only one of them is non-zero.
- For example, if data point $\mathbf{x}^{(n)}$ is assigned to cluster $k = 3$, then $\mathbf{r}^{(n)} = [0, 0, 1, 0, \dots]$ and

$$\sum_{k=1}^K r_k^{(n)} \left\| \mathbf{m}_k - \mathbf{x}^{(n)} \right\|^2 = \left\| \mathbf{m}_3 - \mathbf{x}^{(n)} \right\|^2.$$

K-means : Optimization

Optimization problem:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \left\| \mathbf{m}_k - \mathbf{x}^{(n)} \right\|^2$$

- Problem is hard when minimizing jointly over the parameters $\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}$.
- But if we fix one and minimize over the other, then it becomes easy.
- Doesn't guarantee the same solution!

K-means : Optimization

Optimization problem:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2$$

• Note:

- ▶ If we fix the centres $\{\mathbf{m}_k\}$, we can easily find the optimal assignments $\{\mathbf{r}^{(n)}\}$ for each sample n

$$\min_{\mathbf{r}^{(n)}} \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2.$$

- ▶ Assign each point to the cluster with the nearest centre

$$r_k^{(n)} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}^{(n)} - \mathbf{m}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

- ▶ E.g. if $\mathbf{x}^{(n)}$ is assigned to cluster \hat{k} ,

$$\mathbf{r}^{(n)} = \underbrace{[0, 0, \dots, 1, \dots, 0]^T}_{\text{Only } \hat{k}\text{-th entry is 1}}$$

K-means : Optimization

- If we fix the assignments $\{\mathbf{r}^{(n)}\}$, then we can easily find optimal centres $\{\mathbf{m}_k\}$
 - ▶ Set each cluster's centre to the average of its assigned data points:
For $l = 1, 2, \dots, K$

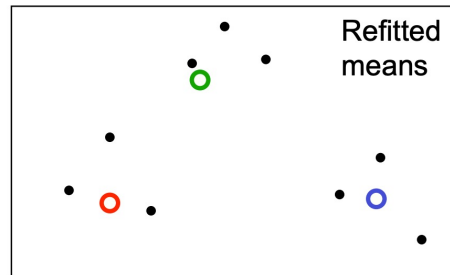
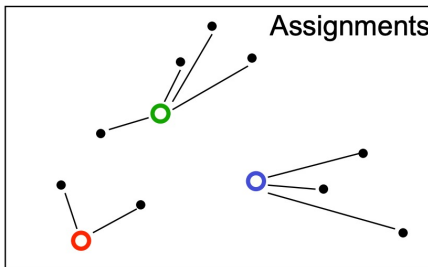
$$\begin{aligned} 0 &= \frac{\partial}{\partial \mathbf{m}_l} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2 \\ &= 2 \sum_{n=1}^N r_l^{(n)} (\mathbf{m}_l - \mathbf{x}^{(n)}) \quad \implies \quad \mathbf{m}_l = \frac{\sum_n r_l^{(n)} \mathbf{x}^{(n)}}{\sum_n r_l^{(n)}} \end{aligned}$$

- Let's alternate between minimizing $J(\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\})$ with respect to $\{\mathbf{m}_k\}$ and $\{\mathbf{r}^{(n)}\}$
- This is called [alternating minimization](#).

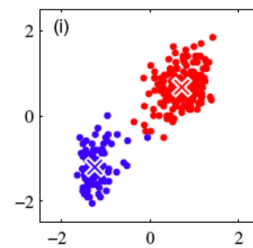
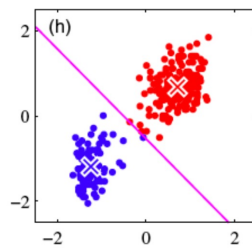
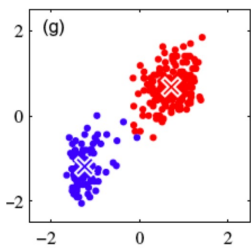
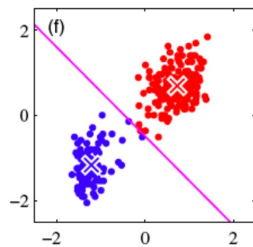
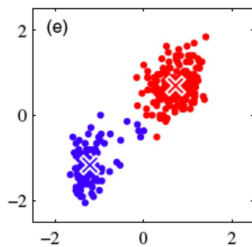
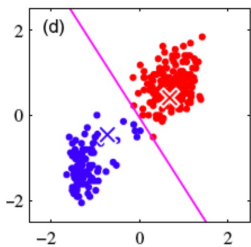
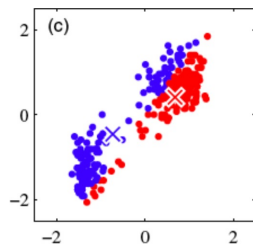
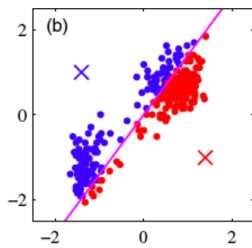
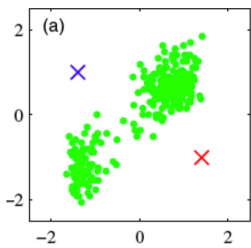
K-means : Optimization

High level overview of algorithm:

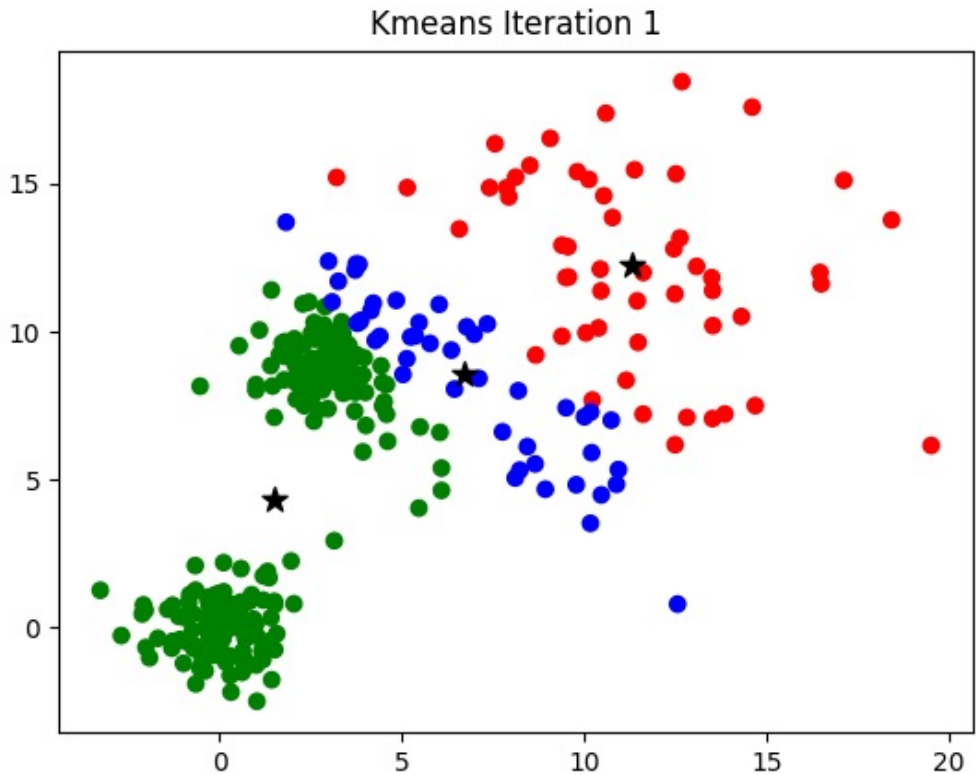
- **Initialization:** randomly initialize cluster centres
- The algorithm iteratively alternates between two steps:
 - ▶ **Assignment step:** Assign each data point to the closest cluster
 - ▶ **Refitting step:** Move each cluster centre to the mean of the data assigned to it



K-means : Example



K-means : Example



K-means : Algorithms

- **Initialization**: Set K cluster means $\mathbf{m}_1, \dots, \mathbf{m}_K$ to random values
- Repeat until convergence (until assignments do not change):
 - ▶ **Assignment**: Optimize J w.r.t. $\{\mathbf{r}\}$: Each data point $\mathbf{x}^{(n)}$ assigned to nearest centre

$$\hat{k}^{(n)} = \arg \min_k \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2$$

and **Responsibilities** (1-hot or 1-of- K encoding)

$$r_k^{(n)} = \mathbb{I}\{\hat{k}^{(n)} = k\} \quad \text{for } k = 1, \dots, K$$

- ▶ **Refitting**: Optimize J w.r.t. $\{\mathbf{m}\}$: Each centre is set to mean of data assigned to it

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}.$$

K-means : Applications

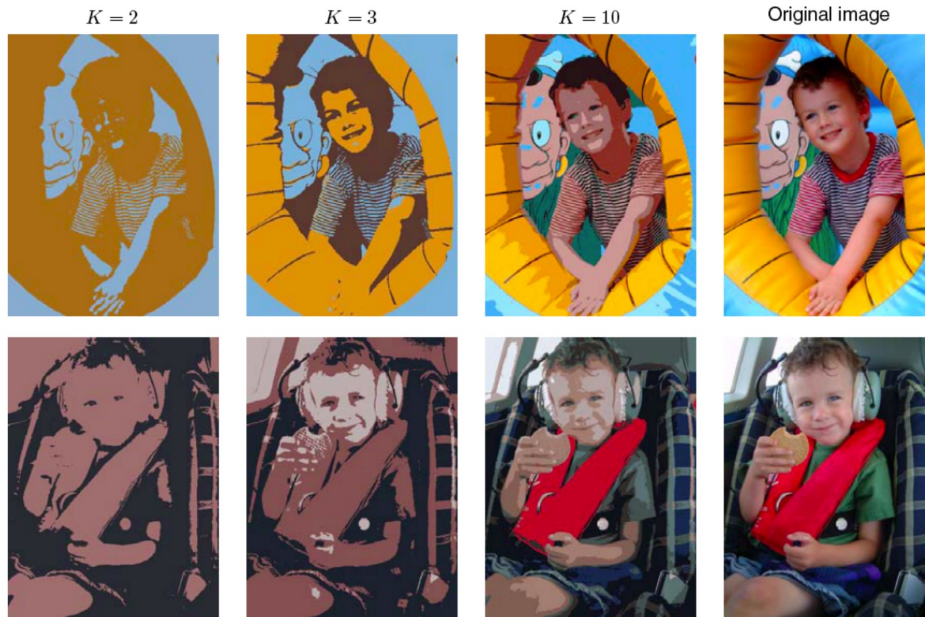
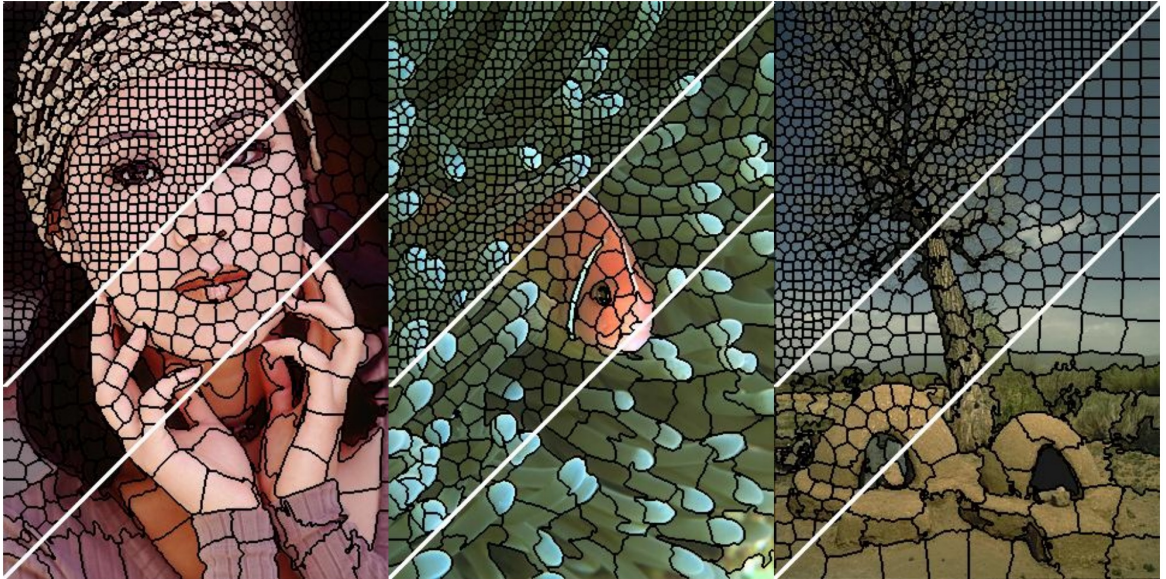


Figure from Bishop

- Given image, construct “dataset” of pixels represented by their RGB pixel intensities
- Run k-means, replace each pixel by its cluster centre

K-means : Applications



- Given image, construct “dataset” of pixels, represented by their RGB pixel intensities and grid locations
- Run k-means (with some modifications) to get superpixels

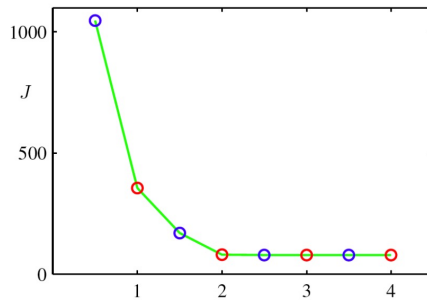
K-means : Potential Issues

- Why does update set \mathbf{m}_k to mean of assigned points?
- What if we used a different distance measure?
- How can we choose the best distance?
- How to choose K ?
- Will it converge?

Hard cases – unequal spreads, non-circular spreads, in-between points

K-means : Potential Issues

- K-means algorithm reduces the cost at each iteration.
 - ▶ Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centres is reduced.
 - ▶ Whenever a cluster centre is moved, J is reduced.
- **Test for convergence:** If the assignments do not change in the assignment step, we have converged (to at least a local minimum).
- This will always happen after a finite number of iterations, since the number of possible cluster assignments is finite

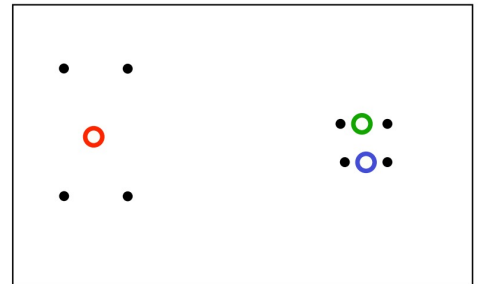


- K-means cost function after each assignment step (blue) and refitting step (red). The algorithm has converged after the third refitting step.

K-means : Potential Issues

- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points

A bad local optimum



Clustering: A generative view

- Next: probabilistic formulation of clustering
- We need a sensible measure of what it means to cluster the data well
 - ▶ This makes it possible to judge different methods
 - ▶ It may help us decide on the number of clusters
- An obvious approach is to imagine that the data was produced by a generative model
 - ▶ Then we adjust the model parameters using maximum likelihood i.e. to maximize the probability that it would produce exactly the data we observed

Clustering: A generative view

- We'll be working with the following generative model for data \mathcal{D}
- Assume a datapoint \mathbf{x} is generated as follows:
 - ▶ Choose a cluster z from $\{1, \dots, K\}$ such that $p(z = k) = \pi_k$
 - ▶ Given z , sample \mathbf{x} from a Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_z, \mathbf{I})$
- Can also be written:

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

$$p(\mathbf{x}|z = k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{I})$$

Clustering: A generative view

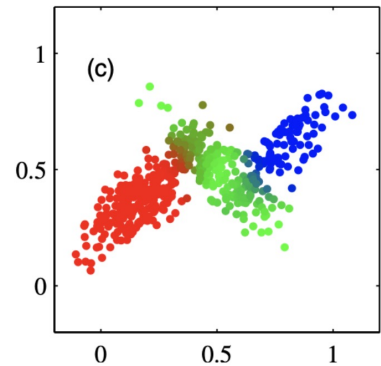
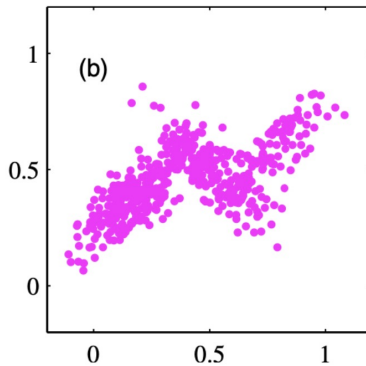
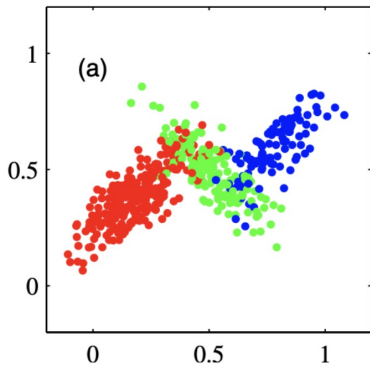
- This defines joint distribution $p(z, \mathbf{x}) = p(z)p(\mathbf{x}|z)$ with parameters $\{\pi_k, \boldsymbol{\mu}_k\}_{k=1}^K$
- The marginal of \mathbf{x} is given by $p(\mathbf{x}) = \sum_z p(z, \mathbf{x})$
- $p(z = k|\mathbf{x})$ can be computed using Bayes rule

$$p(z = k|\mathbf{x}) = \frac{p(\mathbf{x} | z = k)p(z = k)}{p(\mathbf{x})}.$$

This tells us the probability that \mathbf{x} comes from the k^{th} cluster.

Clustering: A generative view

- 500 points drawn from a mixture of 3 Gaussians.



a) Samples from $p(\mathbf{x} | z)$

b) Samples from the marginal $p(\mathbf{x})$

c) Responsibilities $p(z | \mathbf{x})$

Clustering: A generative view

- How should we choose the parameters $\{\pi_k, \boldsymbol{\mu}_k\}_{k=1}^K$?
- Maximum likelihood principle: choose parameters to maximize likelihood of **observed data**
- We don't observe the cluster assignments z , we only see the data \mathbf{x}
- Given data $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^N$, choose parameters to maximize:

$$\log p(\mathcal{D}) = \sum_{n=1}^N \log p(\mathbf{x}^{(n)})$$

- We can find $p(\mathbf{x})$ by marginalizing out z :

$$p(\mathbf{x}) = \sum_{k=1}^K p(z = k, \mathbf{x}) = \sum_{k=1}^K p(z = k)p(\mathbf{x}|z = k)$$

Gaussian Mixture Model (GMM)

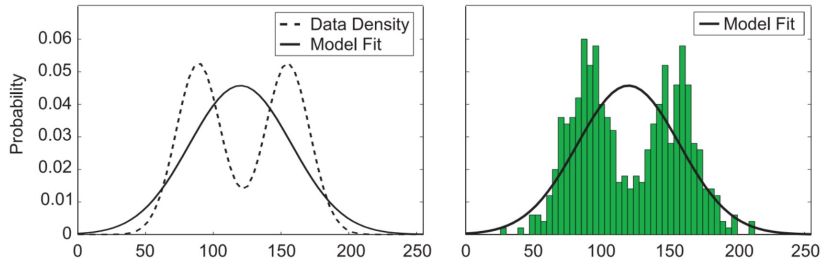
What is $p(\mathbf{x})$?

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

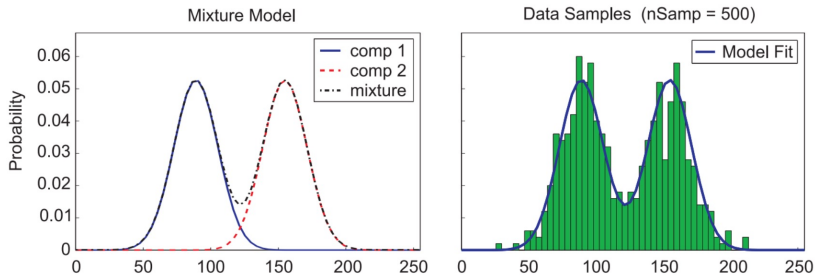
- This distribution is an example of a **Gaussian Mixture Model (GMM)**, and π_k are known as the **mixing coefficients**
- In general, we would have different covariance for each cluster, i.e., $p(\mathbf{x}|z = k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. For this lecture, we assume $\boldsymbol{\Sigma}_k = \mathbf{I}$ for simplicity.
- If we allow arbitrary covariance matrices, GMMs are **universal approximators of densities** (if you have enough Gaussians). Even diagonal GMMs are universal approximators.

Gaussian Mixture Model (GMM)

- If you fit one Gaussian distribution to data:

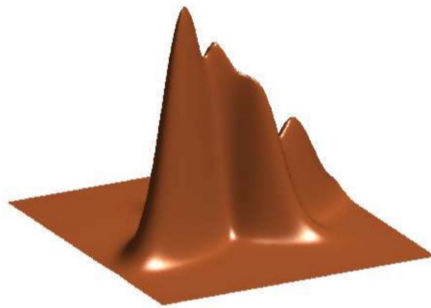
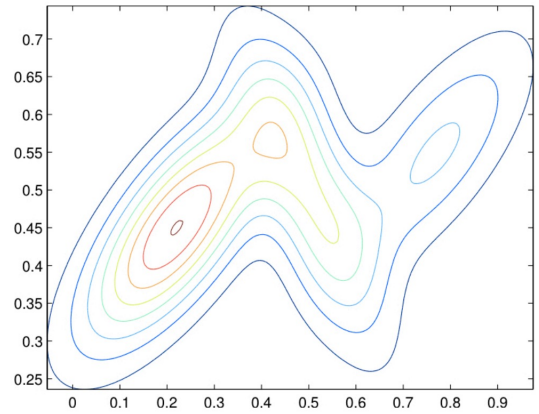
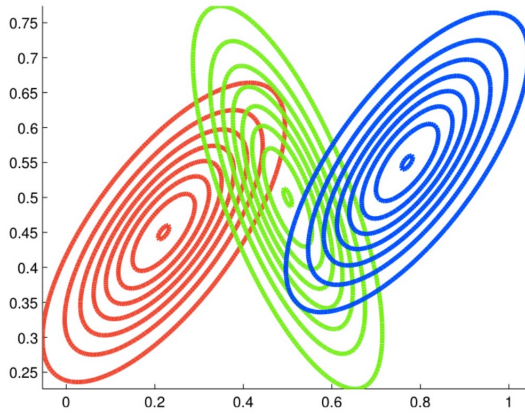


- Now, we are trying to fit a GMM with $K = 2$:



[Slide credit: K. Kutulakos]

Gaussian Mixture Model (GMM)



(How to) Gaussian Mixture Model (GMM)

Optional!



Maximum likelihood objective:

$$\log p(\mathcal{D}) = \sum_{n=1}^N \log p(\mathbf{x}^{(n)}) = \sum_{n=1}^N \log \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) \right)$$

- How would you optimize this w.r.t. parameters $\{\pi_k, \boldsymbol{\mu}_k\}$?
 - ▶ No closed-form solution when we set derivatives to 0
 - ▶ Difficult because sum inside the log
- One option: gradient ascent. Can we do better?
- Can we have a closed-form update?

(How to) Gaussian Mixture Model (GMM)

Optional!



- **Observation:** if we knew $z^{(n)}$ for every $\mathbf{x}^{(n)}$, (i.e. our dataset was $\mathcal{D}_{\text{complete}} = \{(z^{(n)}, \mathbf{x}^{(n)})\}_{n=1}^N$) the maximum likelihood problem is easy:

$$\begin{aligned}\log p(\mathcal{D}_{\text{complete}}) &= \sum_{n=1}^N \log p(z^{(n)}, \mathbf{x}^{(n)}) \\ &= \sum_{n=1}^N \log p(\mathbf{x}^{(n)} | z^{(n)}) + \log p(z^{(n)}) \\ &= \sum_{n=1}^N \sum_{k=1}^K \mathbb{I}\{z^{(n)} = k\} \left(\log \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k \right)\end{aligned}$$

(How to) Gaussian Mixture Model (GMM)

Optional!



$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^N \sum_{k=1}^K \mathbb{I}\{z^{(n)} = k\} \left(\log \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k \right)$$

- We have been optimizing something similar for Naive bayes classifiers
- By maximizing $\log p(\mathcal{D}_{\text{complete}})$, we would get this:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^N \mathbb{I}\{z^{(n)} = k\} \mathbf{x}^{(n)}}{\sum_{n=1}^N \mathbb{I}\{z^{(n)} = k\}} = \text{class means}$$
$$\hat{\pi}_k = \frac{1}{N} \sum_{n=1}^N \mathbb{I}\{z^{(n)} = k\} = \text{class proportions}$$

(How to) Gaussian Mixture Model (GMM)

Optional!



- We haven't observed the cluster assignments $z^{(n)}$, but we can compute $p(z^{(n)}|\mathbf{x}^{(n)})$ using Bayes rule
- Conditional probability (using Bayes rule) of z given \mathbf{x}

$$\begin{aligned} p(z = k|\mathbf{x}) &= \frac{p(z = k)p(\mathbf{x}|z = k)}{p(\mathbf{x})} \\ &= \frac{p(z = k)p(\mathbf{x}|z = k)}{\sum_{j=1}^K p(z = j)p(\mathbf{x}|z = j)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{I})}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \mathbf{I})} \end{aligned}$$

(How to) Gaussian Mixture Model (GMM)

Optional!



$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^N \sum_{k=1}^K \mathbb{I}\{z^{(n)} = k\} (\log \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k)$$

- We don't know the cluster assignments $\mathbb{I}\{z^{(n)} = k\}$ (they are our latent variables), but we know their expectation $\mathbb{E}[\mathbb{I}\{z^{(n)} = k\} | \mathbf{x}^{(n)}] = p(z^{(n)} = k | \mathbf{x}^{(n)})$.
- If we plug in $r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)})$ for $\mathbb{I}\{z^{(n)} = k\}$, we get:

$$\sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} (\log \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k)$$

- This is still easy to optimize! Solution is similar to what we have seen:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^N r_k^{(n)} \mathbf{x}^{(n)}}{\sum_{n=1}^N r_k^{(n)}} \quad \hat{\pi}_k = \frac{\sum_{n=1}^N r_k^{(n)}}{N}$$

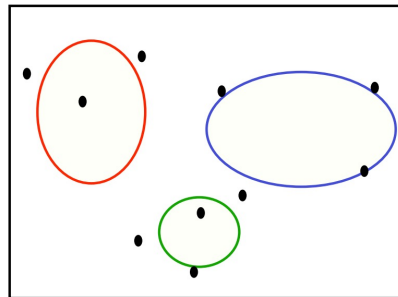
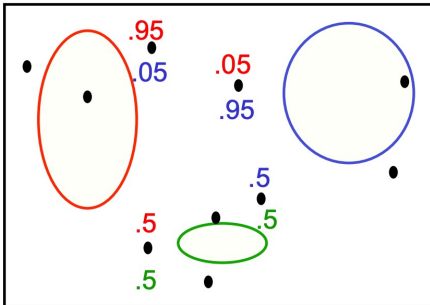
- Note: this only works if we treat $r_k^{(n)} = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I})}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_j, \mathbf{I})}$ as fixed.

(How to) Gaussian Mixture Model (GMM)

Optional!



- This motivates the **Expectation-Maximization algorithm**, which alternates between two steps:
 1. **E-step**: Compute the posterior probabilities $r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)})$ given our current model, i.e., how much do we think a cluster is responsible for generating a datapoint.
 2. **M-step**: Use the equations on the last slide to update the parameters, assuming $r_k^{(n)}$ are held fixed – change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.



(How to) Gaussian Mixture Model (GMM)

Optional!



- **Initialize** the means $\hat{\boldsymbol{\mu}}_k$ and mixing coefficients $\hat{\pi}_k$
- Iterate until convergence:
 - ▶ **E-step**: Evaluate the responsibilities $r_k^{(n)}$ given current parameters

$$r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)}) = \frac{\hat{\pi}_k \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_k, \mathbf{I})}{\sum_{j=1}^K \hat{\pi}_j \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_j, \mathbf{I})} = \frac{\hat{\pi}_k \exp\{-\frac{1}{2} \|\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}_k\|^2\}}{\sum_{j=1}^K \hat{\pi}_j \exp\{-\frac{1}{2} \|\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}_j\|^2\}}$$

- ▶ **M-step**: Re-estimate the parameters given current responsibilities

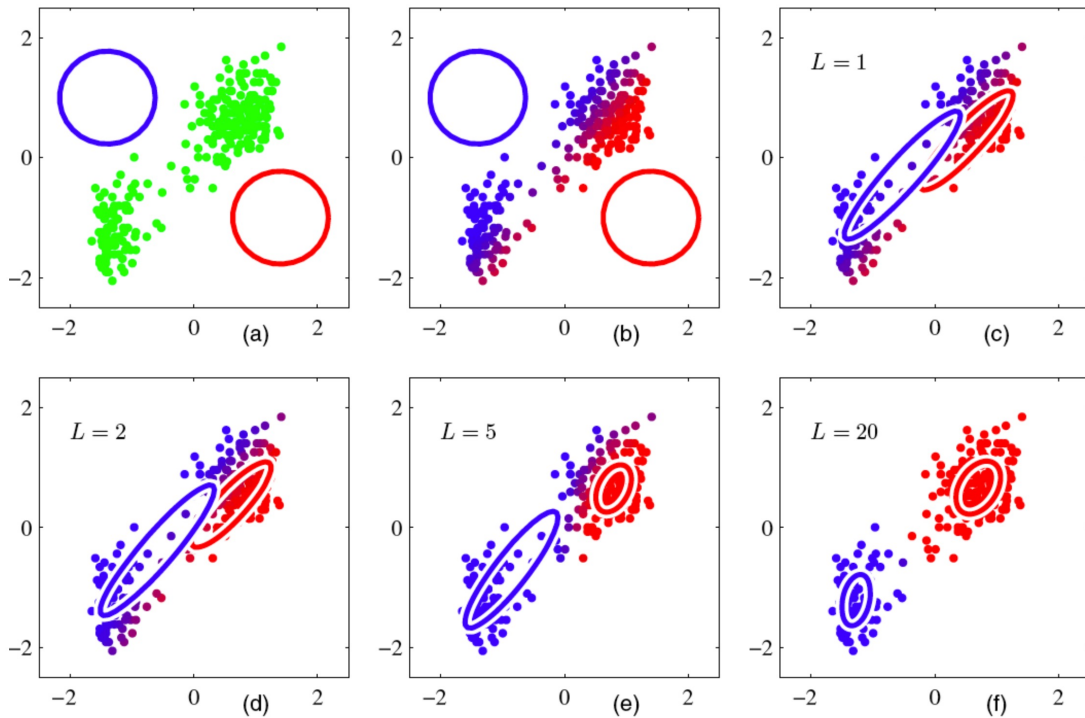
$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{n=1}^N r_k^{(n)} \mathbf{x}^{(n)}$$

$$\hat{\pi}_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^N r_k^{(n)}$$

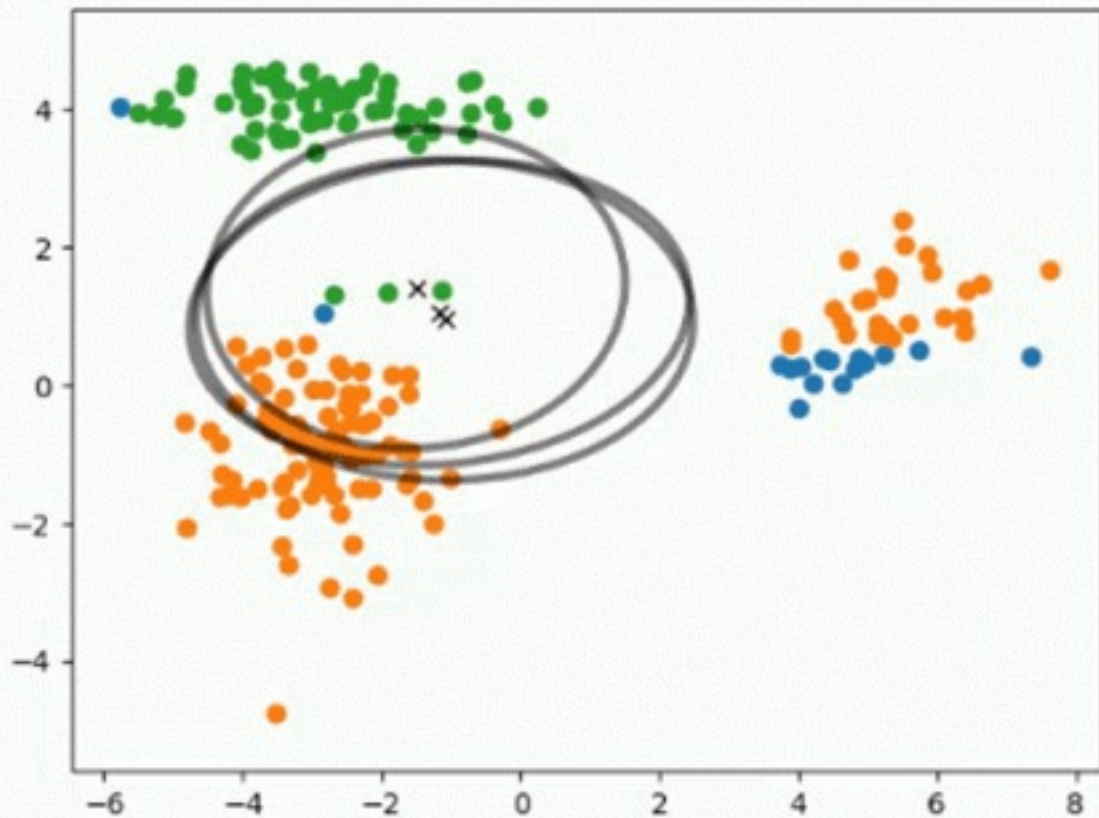
- ▶ Evaluate log likelihood and check for convergence

$$\log p(\mathcal{D}) = \sum_{n=1}^N \log \left(\sum_{k=1}^K \hat{\pi}_k \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_k, \mathbf{I}) \right)$$

Gaussian Mixture Model (GMM) : Example



Gaussian Mixture Model (GMM) : Example



GMM vs K-means

- The K-Means Algorithm:
 1. **Assignment step**: Assign each data point to the closest cluster
 2. **Refitting step**: Move each cluster centre to the average of the data assigned to it
- The EM Algorithm:
 1. **E-step**: Compute the posterior probability over z given our current model
 2. **M-step**: Maximize the probability that it would generate the data it is currently responsible for.

GMM vs K-means

Convergence:

K-Means tends to **converge** much faster than a **GMM**

Speed:

Each iteration of **K-Means** is **computationally less intensive** than each iteration of a **GMM**

Initialization:

To **initialize** a **GMM**, we typically first run **K-Means** and use the resulting cluster centers as the means of the Gaussian components

Output:

A **GMM** yields a **probability distribution** over the cluster assignment for each point; whereas **K-Means** gives a single **hard assignment**