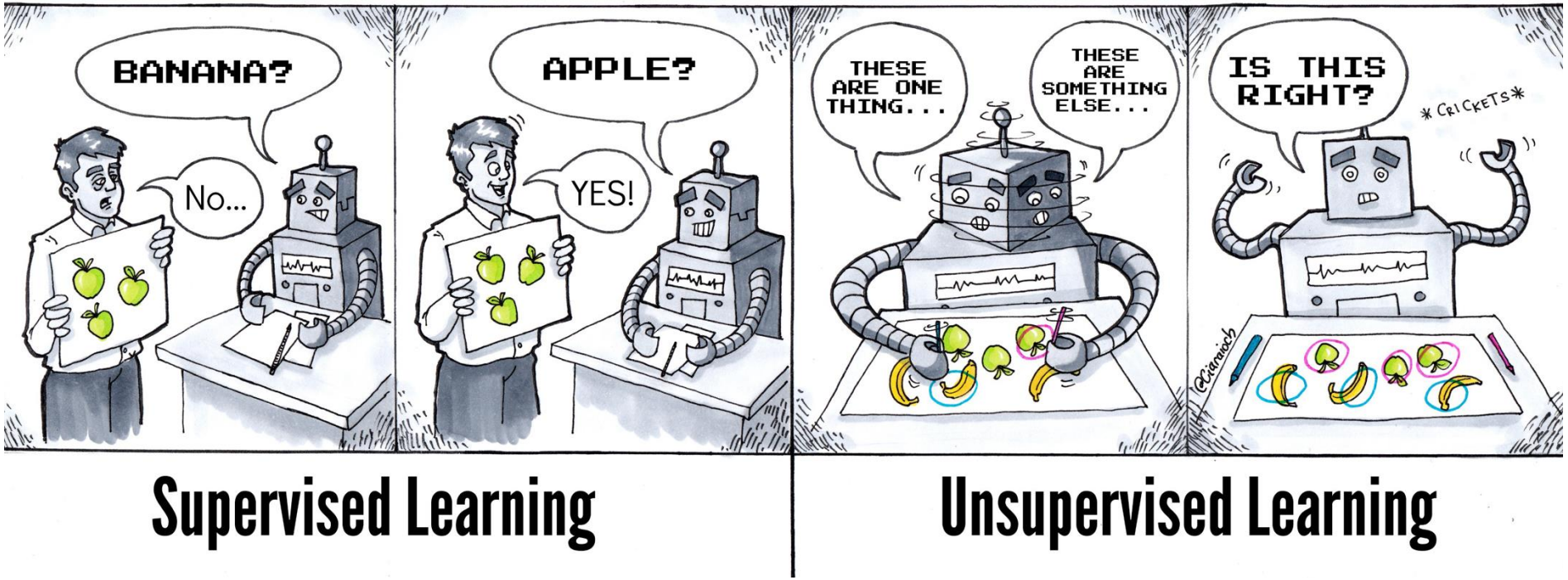
IN3050/IN4050, Lecture 10
Unsupervised Learning

1: Introduction
2: Learning how to generate similar data
3: Learning how data are grouped
4: Learning how to compress data
5: Learning how to represent data in low dimension

IN3050/IN4050, Lecture 10
Unsupervised Learning

1: Introduction

Ole Christian Lingjærde



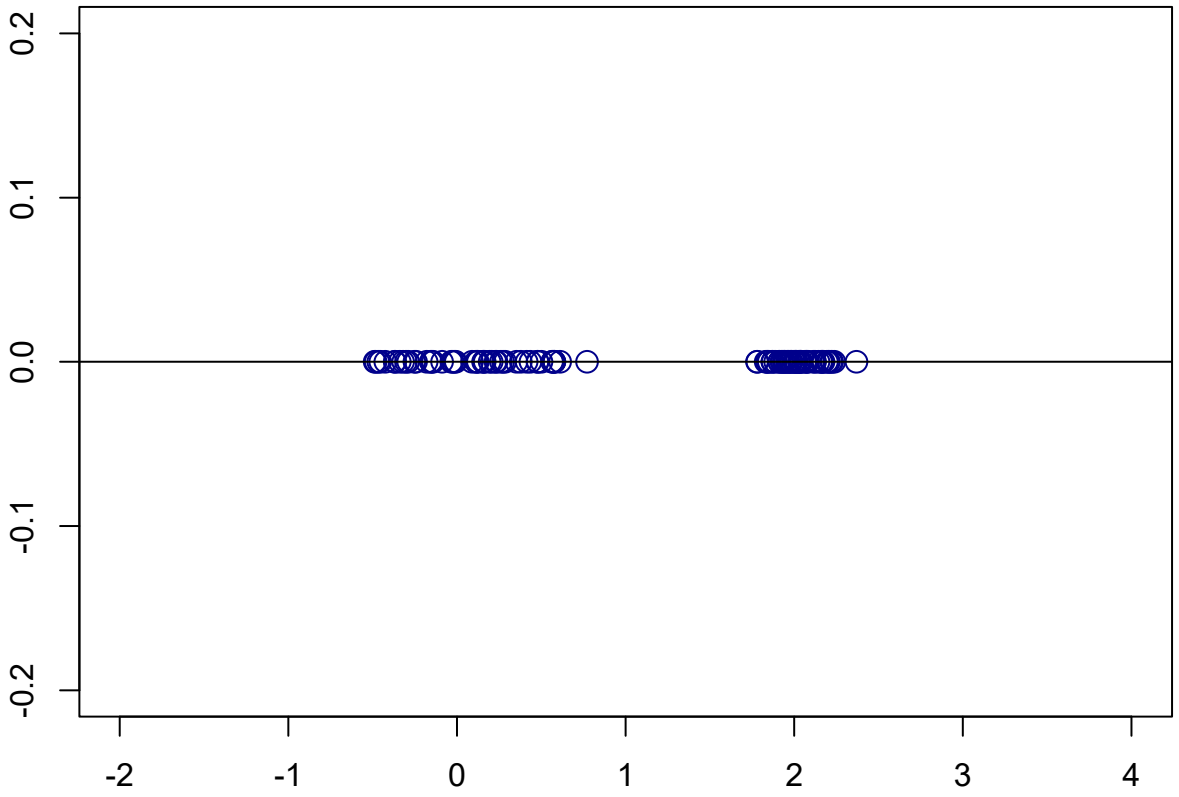
https://twitter.com/athena_schools/status/1063013435779223553/photo/1

Supervised vs. unsupervised learning

Supervised learning: we have a training set consisting of a feature vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ and a response y for each sample. Goal is to learn the mapping f: $\boldsymbol{x} \mapsto y$.

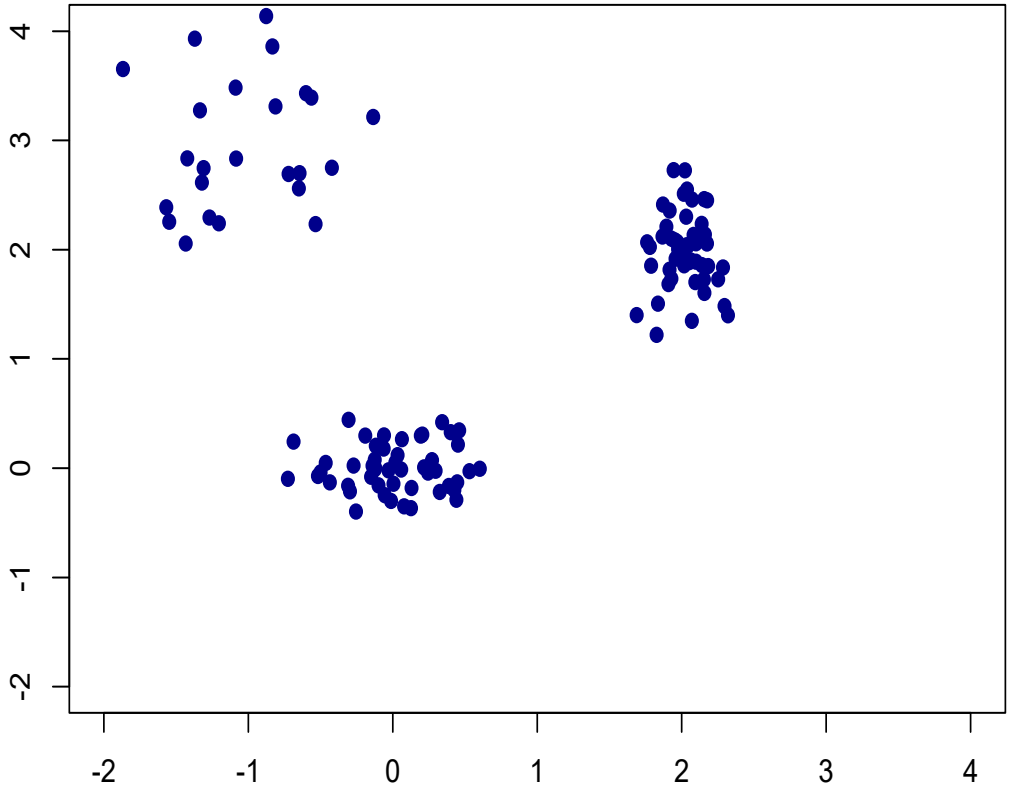
Unsupervised learning: we have a training set consisting of only a feature vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ for each sample. Goal is to learn something interesting about the distribution of the $\boldsymbol{x}$'s.

Example (1D)



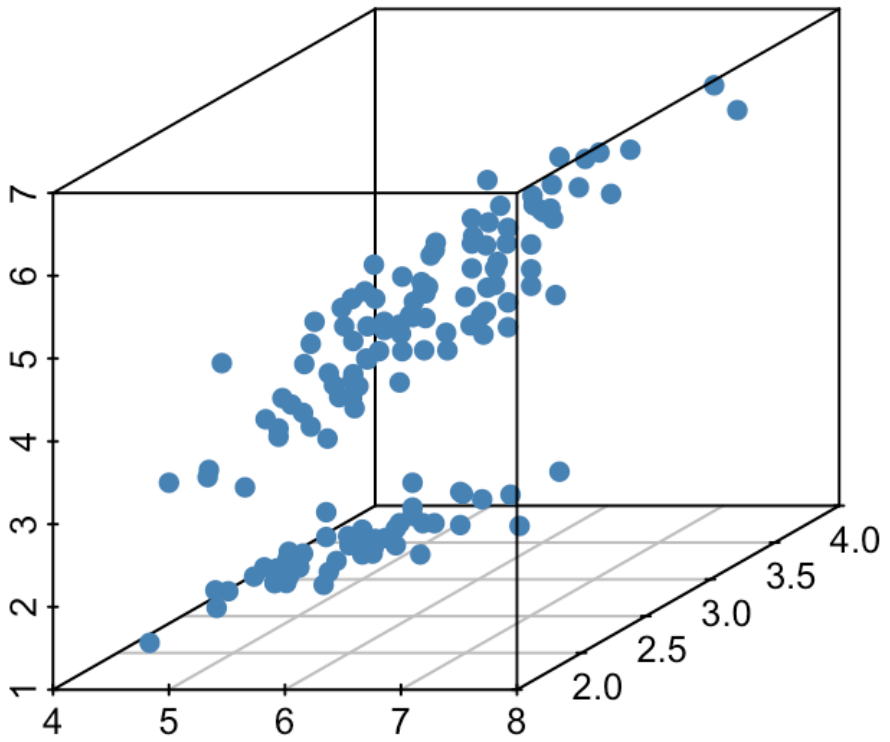
Each sample has a single feature x (a real number)

Example (2D)



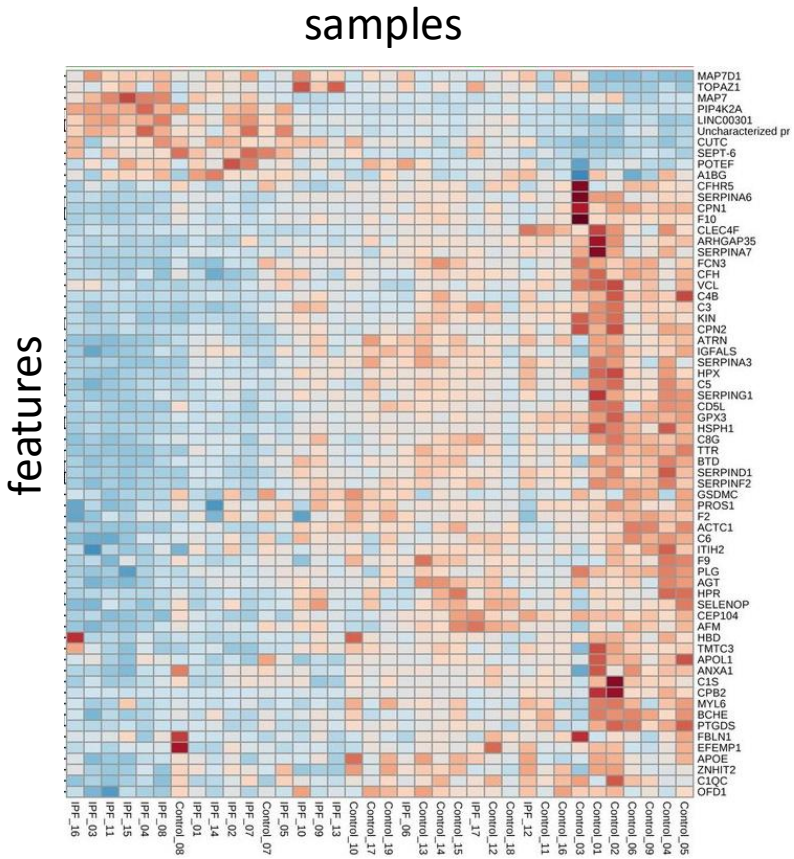
Each sample has two features $\boldsymbol{x} = (x_1, x_2)$

Example (3D)



Each sample has three features $\boldsymbol{x} = (x_1, x_2, x_3)$

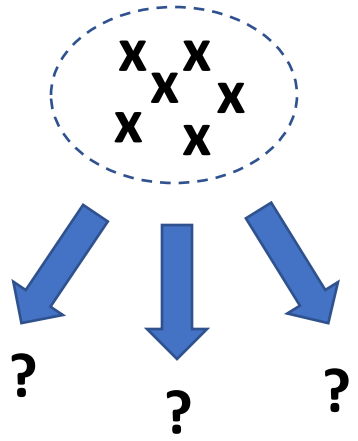
Example (p dimensions)



Each sample has 66 features $\boldsymbol{x} = (x_1, x_2, x_{3,\ldots,} x_{66})$

What can we learn in unsupervised learning?

- How to generate similar data
  → Density estimation, generative models
- How many (and which) groups are present
  → hierarchical clustering, k-means clustering
- How to compress the data
  → Autoencoders
- How to visualize the data in low dimension
  → PCA, learning vector quantization, self organizing maps



Evaluating the performance

Supervised learning:

- Goodness of fit to training data
- Goodness of fit to new data
- Prediction performance

Unsupervised learning:

- No "truth" to compare with
- Performance more open to interpretation
- Still: often possible to define a useful loss function and to determine which of two solutions is best

IN3050/IN4050, Lecture 10
Unsupervised Learning

3: Learning how data are grouped

Ole Christian Lingjærde

Clustering

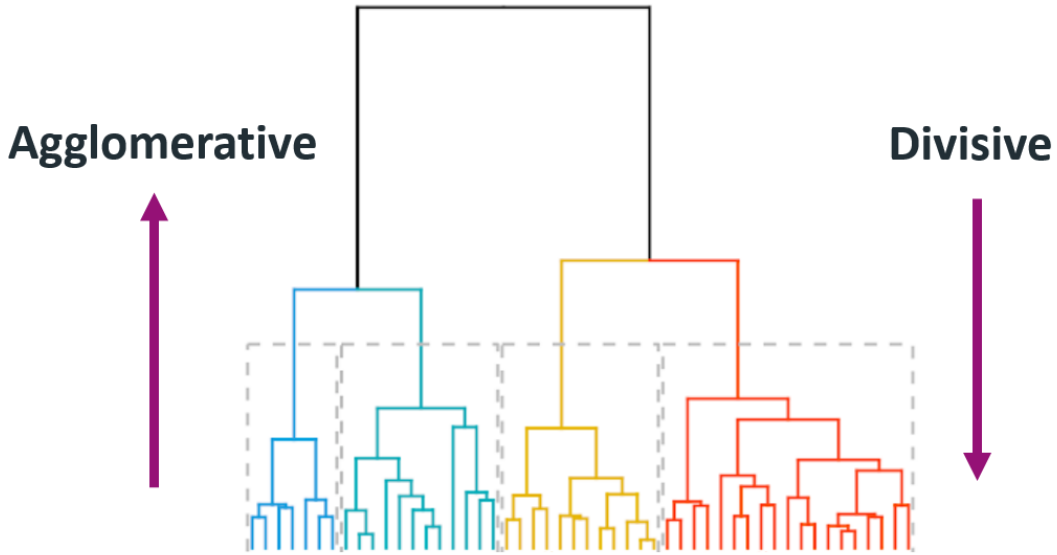
Learning how the data are grouped is called clustering.

Some clustering methods:

- Hierarchical clustering
- k-means clustering
- Competitive learning

Hierarchical clustering

1) Initially, each sample is a separate cluster
2) Find the two clusters that are closest together and merge them into one cluster
3) Repeat point 2 until there is only one cluster left



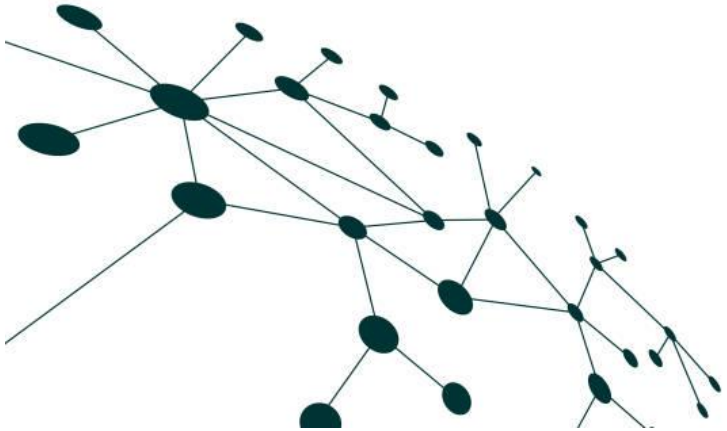
https://harshsharma1091996.medium.com/hierarchical-clustering-996745fe656b

Hierarchical clustering: important aspects

- Must keep track of the whole history of cluster mergers in order to plot the tree (= dendrogram)
- In each iteration we have to compute the distance between every pair of the remaining clusters
- We have to define what we mean by the distance between two clusters

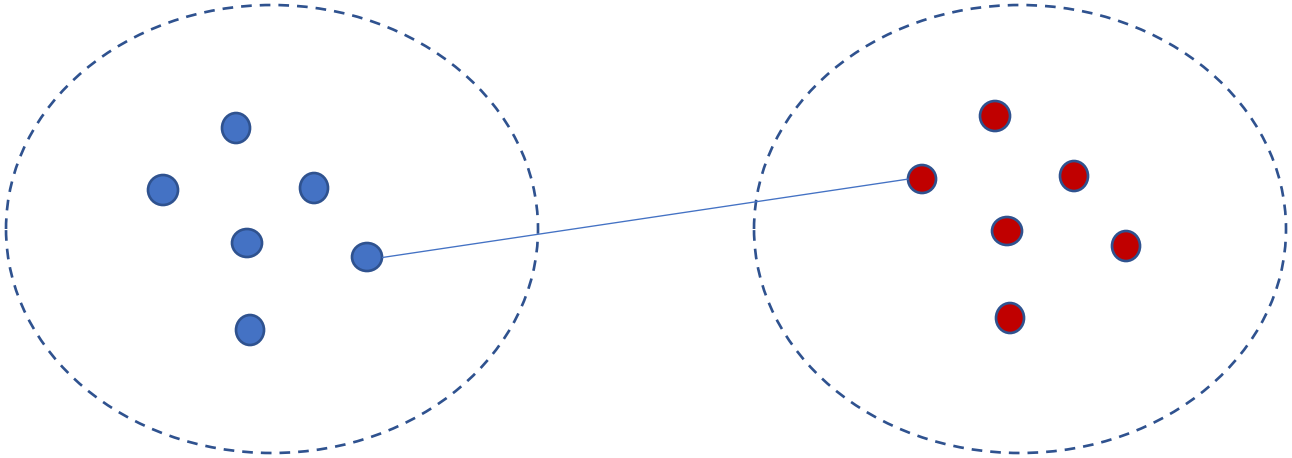
Hierarchical clustering: how to define the distance between two clusters C_1 and C_2

- Referred to as linkage method
- Has great influence on result
- Most common linkage methods:
  - Single linkage
  - Average linkage
  - Complete linkage
  - Ward linkage



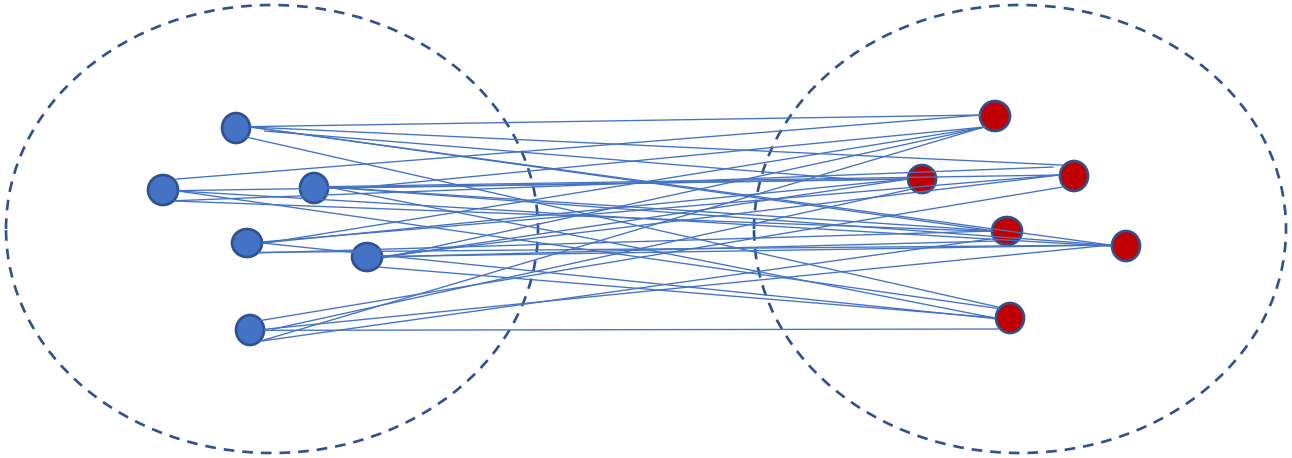
Single linkage

$d(C_1,C_2)$ = smallest distance between a sample in C_1 and a sample in C_2



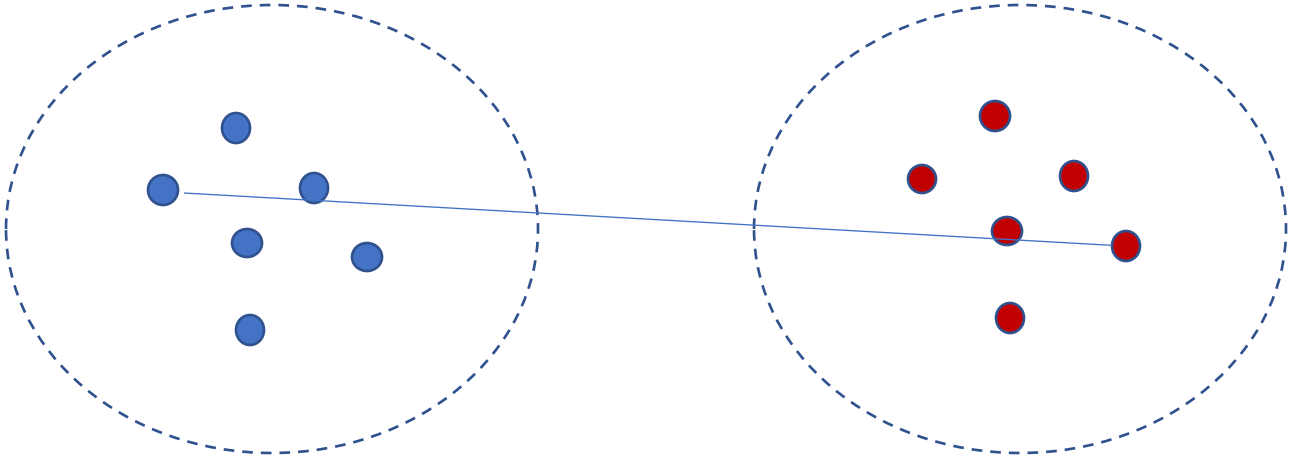
Average linkage

$d(C_1,C_2)$ = average distance between samples in C_1 and samples in C_2



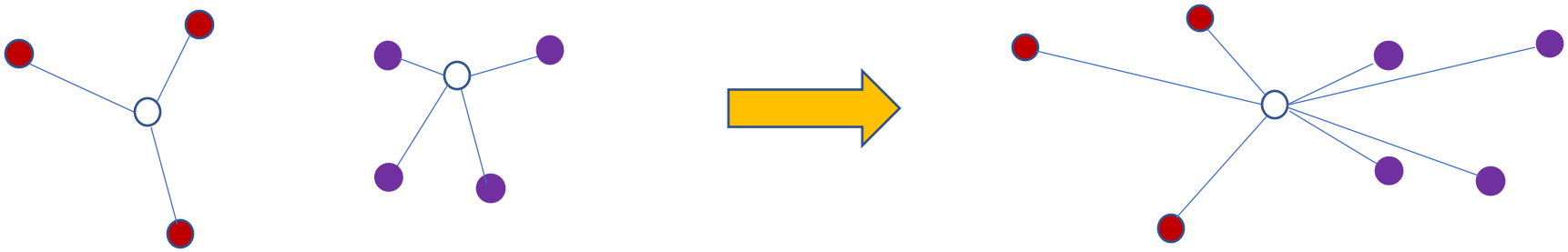
Complete linkage

$d(C_1,C_2)$ = maximal distance between a sample in C_1 and a sample in C_2



Ward linkage

$d(C_1,C_2)$ = increase in within-cluster variance if clusters are merged



Hierarchical clustering: how to define the distance between two samples $\mathbf{x}_1$ and $\mathbf{x}_2$

Manhattan distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{p} | \, x_{1i} - x_{2i} \, |$$

Euclidean distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{p} (x_{1i} - x_{2i})^2$$

Cosine distance:

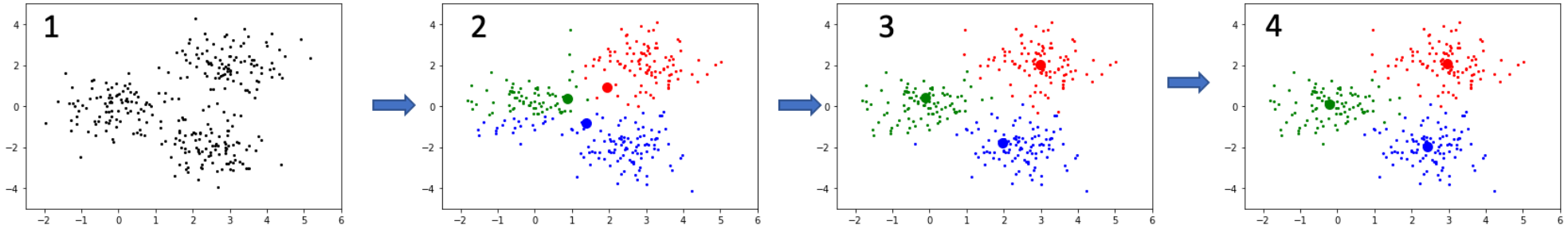
$$d(\mathbf{x}_1, \mathbf{x}_2) = 1 - \cos\theta = 1 - \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{||\mathbf{x}_1|| \cdot ||\mathbf{x}_1||}$$

k-means clustering

As before we wish to cluster samples $x_1, \ldots, x_n \in \mathbf{R}^p$

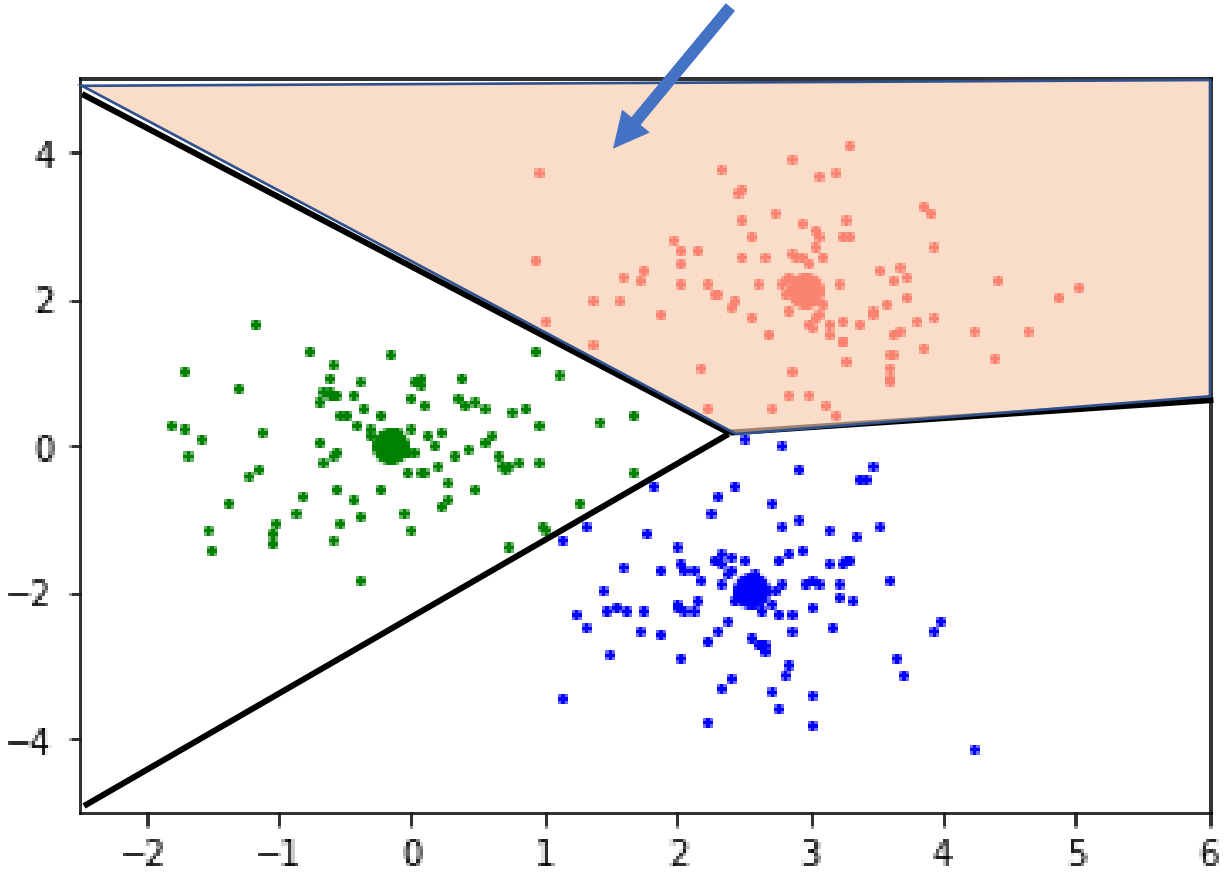
1) Randomly select k cluster centers $c_1, \ldots, c_k \in \mathbf{R}^p$
2) Assign each sample to the nearest cluster center
3) Update cluster centers to mean of samples in cluster
4) Repeat steps 2-3 until convergence

Example: k-means with k=3



Voronoi diagram

Voronoi cell: part of the plane that
is closer to one particular cluster center
than to any other cluster center



k-means clustering: algorithmic details

- Initial cluster center selection:
  - Random (uniform, normal) coordinates within data range
  - Random selection of the data points
  - Manual selection

- Distance between a sample x_i and a cluster center c_i:

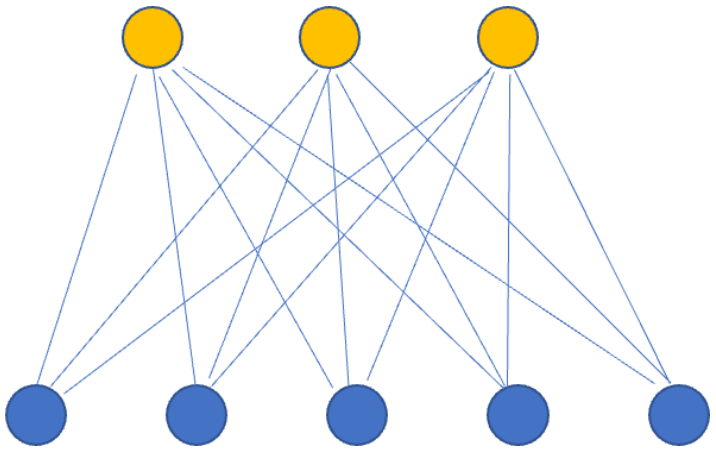
$$d(\boldsymbol{x}_i, \boldsymbol{c}_i) = \Sigma\left(x_{ij} - c_{ij}\right)^2 \qquad \text{(or some other metric)}$$

- Cluster centers are updated by calculating the average of all the sample vectors assigned to it:

$$\boldsymbol{c}_i^{new} = \frac{1}{m} \sum \boldsymbol{x}_i \qquad \text{(or median)}$$

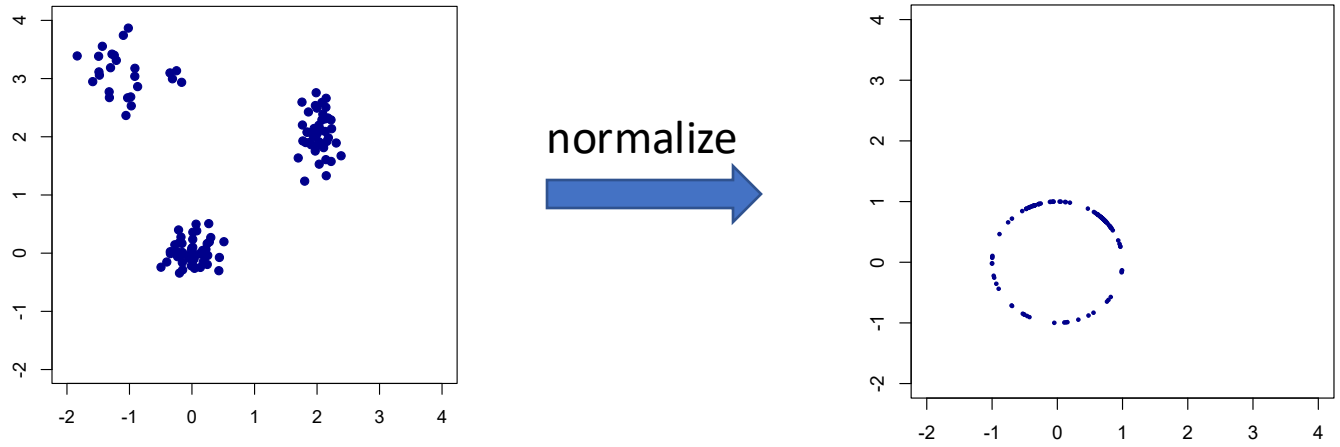
The k-means neural network

- Normalize all samples ($\|x\|$=1 for all input vectors)
- Computing output values for input x:
  - Calculate $h_k = x \cdot w_k$ for all output nodes
  - Set activation value to 1 for output node with maximal h_k and set activation to 0 for all other output nodes
- Learning rule: $\Delta w_{ij} = \eta(x_j - w_{ij})$



k-means ≠ k-means neural network

A: The normalization in k-means neural network means we lose information (only angular distance is preserved):



B: The learning rule in k-means neural network is on-line (i.e. weights are updated after each sample)