50.007 Machine Learning

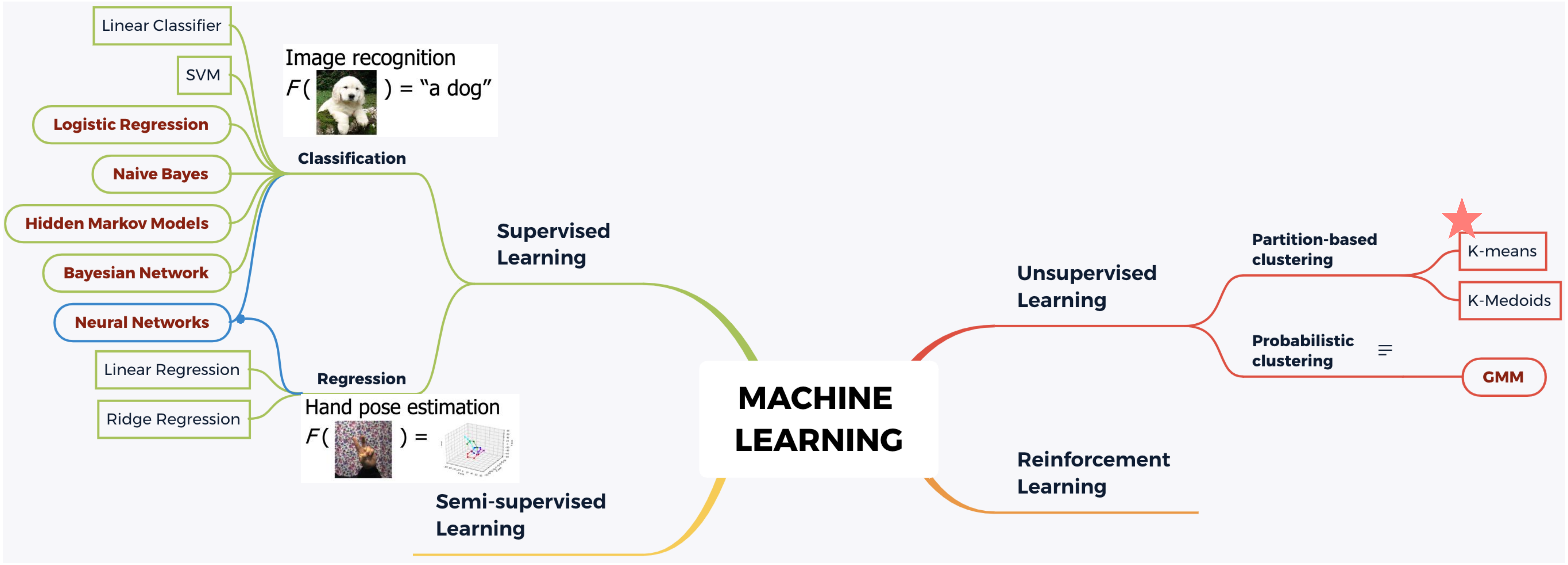
2026 Spring

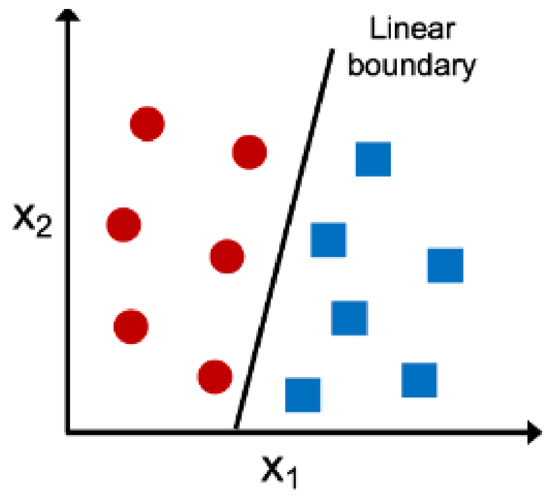
5. Clustering: k-Means

Na Zhao
Assistant Professor, ISTD

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Roadmap


Linear Classifier
SVM
Logistic Regression
Naive Bayes
Hidden Markov Models
Bayesian Network
Neural Networks
Linear Regression
Ridge Regression
Image recognition
F( ) = "a dog"
Classification
Regression
Hand pose estimation
F( ) =
Supervised Learning
Semi-supervised Learning
MACHINE LEARNING
Unsupervised Learning
Reinforcement Learning
Partition-based clustering
K-means
K-Medoids
Probabilistic clustering
GMM



1. Training Set (Linearly Separable)

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})$$

2. Model (Linear Classifier)

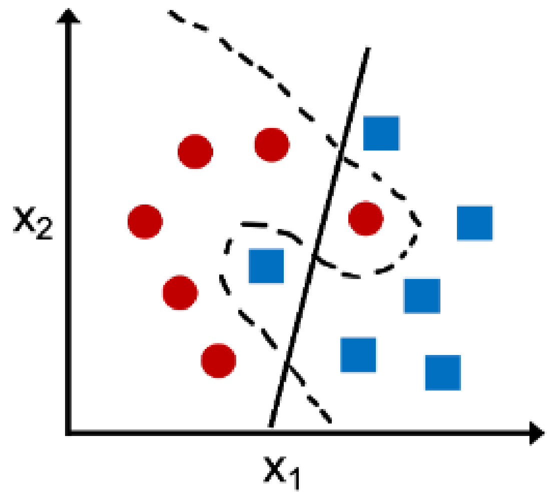
$$h(x;\theta) = \text{sign}(\theta_1 x_1 + \cdots + \theta_d x_d)$$

3. Training Error (Zero-one Loss)

$$\boldsymbol{\varepsilon}_n(\theta) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} [\![\, y(\theta^\top x) \leq 0 \,]\!]$$

4. Algorithm (The Perceptron Algorithm)

$$\text{if } y^{(t)} \neq h(x^{(t)};\theta^{(k)}) \text{ then}$$
$$\theta^{(k+1)} = \theta^{(k)} + y^{(t)}x^{(t)}$$



1. Training Set (Not Necessarily Linearly Separable)

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})$$

2. Model (Linear Classifier)

$$h(x;\theta) = \text{sign}(\theta_1 x_1 + \cdots + \theta_d x_d)$$

3. Training Error (Hinge Loss)

$$R_n(\theta) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \max\{1 - y(\theta^\top x), 0\}$$

4. Algorithm (Sub-Stochastic Gradient Descent)

$$\text{select } t \in \{1, \dots, n\} \text{ at random,}$$
$$\text{if } y^{(t)}(\theta^{(k)} \cdot x^{(t)}) \leq 1, \text{ then}$$
$$\theta^{(k+1)} = \theta^{(k)} + \eta_k y^{(t)} x^{(t)}$$

Learning a function

$$y = f(x)$$
$$x \in \mathbb{R}^d$$
$$y \in \{1, 2, \dots, k\}$$

Classification

Linear Regression

$$\boxed{R_n(\theta) = \frac{1}{n}\sum_{t=1}^{n}(y^{(t)} - \theta \cdot x^{(t)})^2/2}$$

$\theta^{(0)} = 0$ (vector) *# Random initialization*

select $t \in \{1, \ldots, n\}$ at random,

$$\theta^{(k+1)} = \theta^{(k)} + \eta_k(y^{(t)} - \theta \cdot x^{(t)})x^{(t)}$$ *# Update*

Repeat the update step until stopping criterion is met.

$$f(x; \theta, \theta_0) = \theta \cdot x + \theta_0 = \sum_{i=1}^{d} \theta_i x_i + \theta_0$$

Learning a function
$y = f(x)$
$x \in \mathbb{R}^d$
$y \in \mathbb{R}^m$

Regression

Ridge Regression

$$\boxed{J_{n,\lambda}(\theta) = \overset{\lambda > 0}{\frac{\lambda}{2}} \|\theta\|^2 + \frac{1}{n}\sum_{t=1}^{n}(y^{(t)} - \theta \cdot x^{(t)})^2/2}$$

$\theta^{(0)} = 0$ (vector) *# Random initialization*

select $t \in \{1, \ldots, n\}$ at random, closed form solution

$$\theta^{(k+1)} = \boxed{(1 - \lambda\eta_k)}\theta^{(k)} + \eta_k(y^{(t)} - \theta \cdot x^{(t)})x^{(t)}$$ *# Update*

Repeat the update step until stopping criterion is met.

Learning Objectives

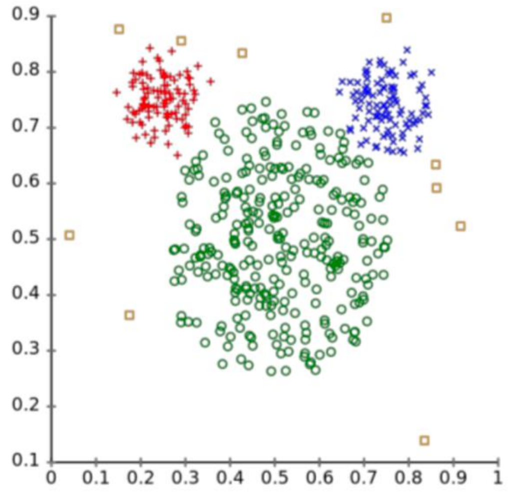
You should be able to know:

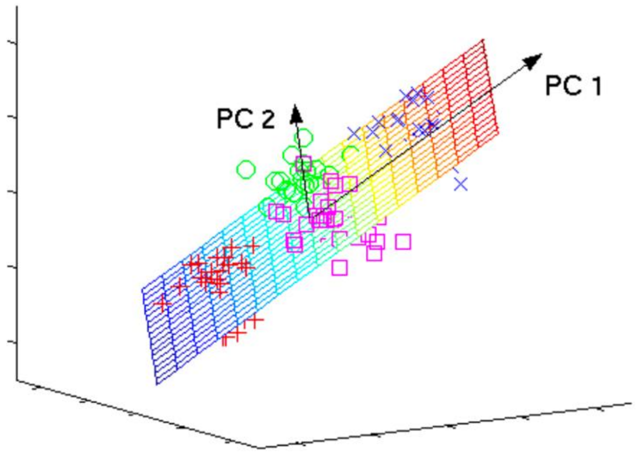
1. What is the k-means algorithm and what type of problem does it solve?
2. How to implement the k-means algorithm and the rationales behind the two update steps?
3. What convergence properties does the k-means algorithm have?

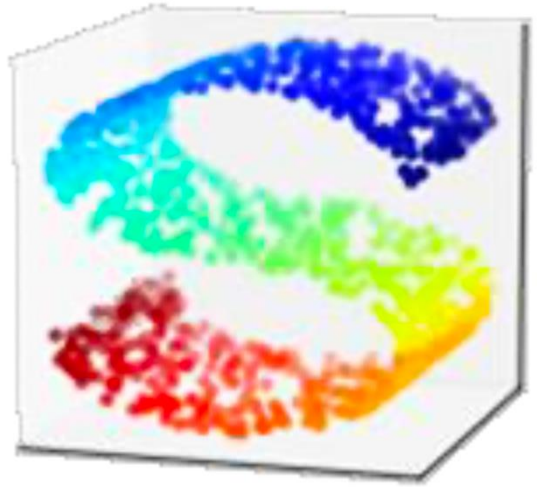
Unsupervised Learning

- No labels/responses.
- Finding structure in data.

non-linear data







Clustering

$T: \mathbb{R}^d \to \{1, 2, \dots, k\}$

Subspace Learning **Manifold Learning**

$T: \mathbb{R}^d \to \mathbb{R}^m$

Clustering

Clustering

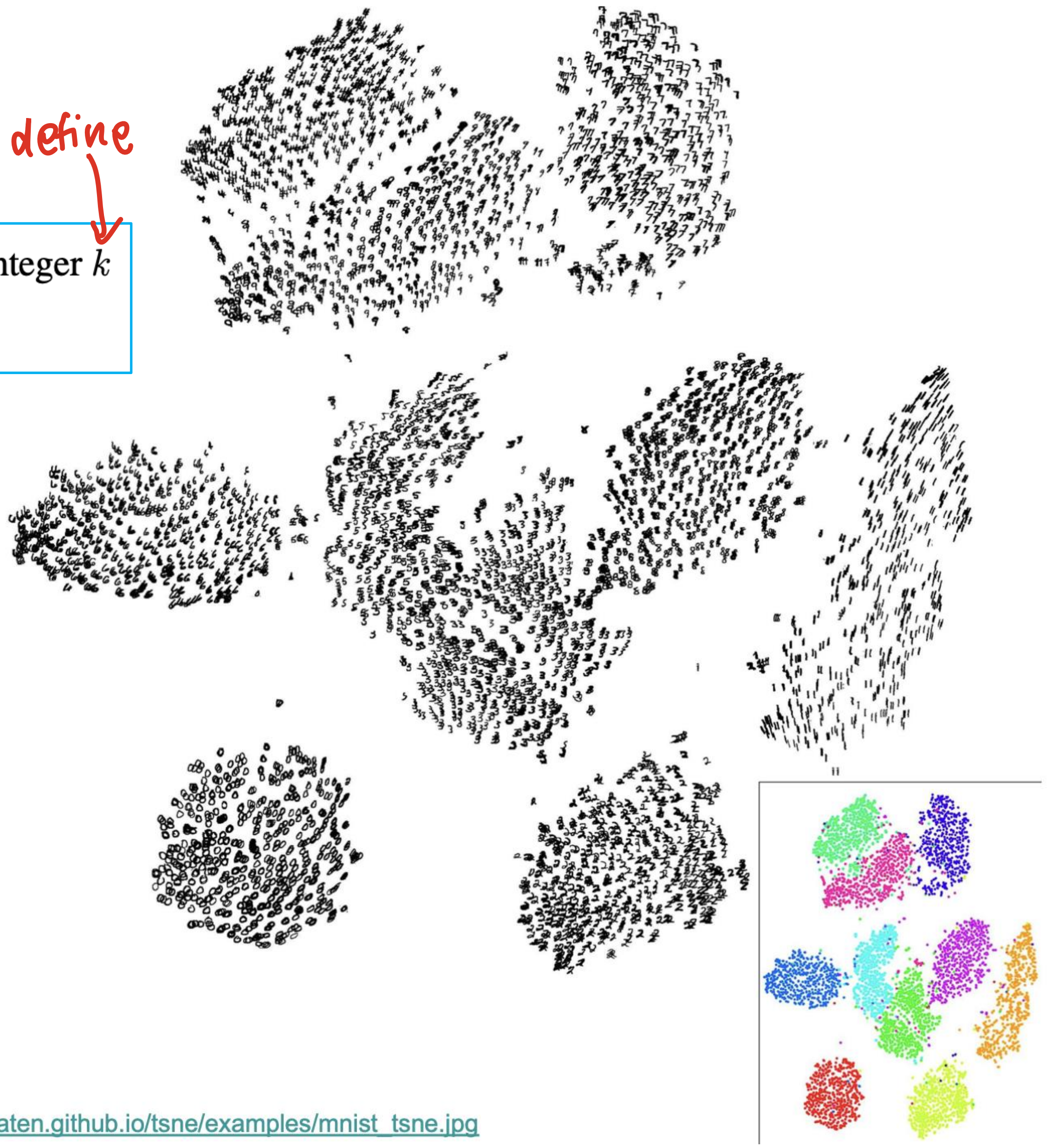
pre-define

Input: Training set $S_n = \{x^{(i)}; i = 1, \ldots; n\}$, where $x^{(i)} \in \mathcal{R}^d$, integer k

Output: A set of clusters $C_1, \ldots, C_k$.

- **Clusters**:
  - – subgroups or subpopulations in the data
    - *Similar* to one another within the same cluster
    - *Dissimilar* to the objects in other clusters

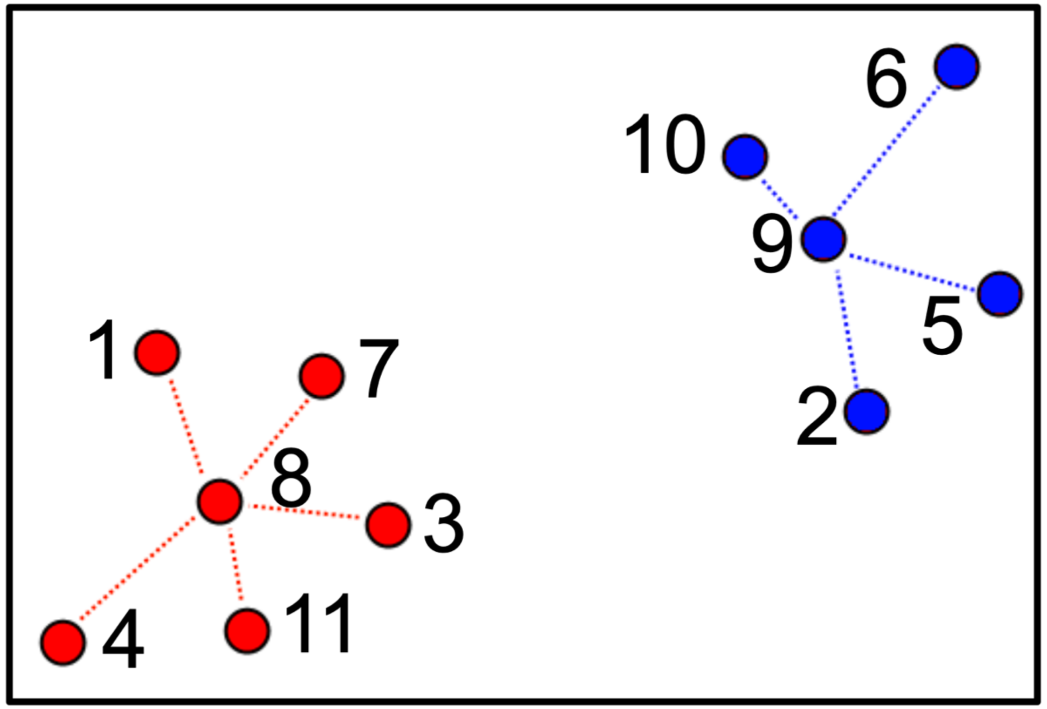
- **Goals**:
  - – Discovering the subgroups
  - – Estimating which subgroup a data point belongs to



http://lvdmaaten.github.io/tsne/examples/mnist_tsne.jpg

How to specify a Cluster

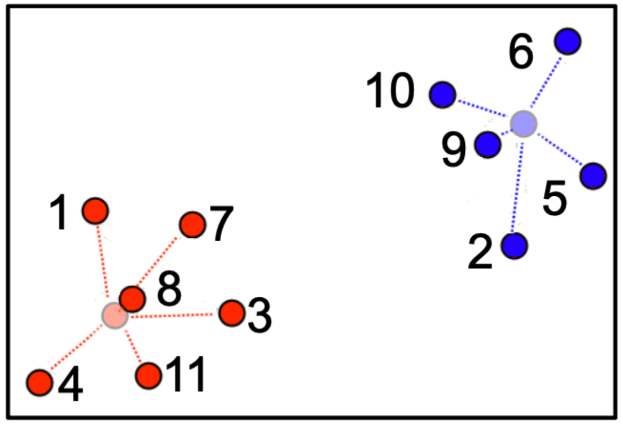
- By listing all its elements
  - $\mathcal{C}1$ = {1,3,4,7,8,11 1}
  - $\mathcal{C}2$ = {2,5,6,9,10}



How to specify a Cluster

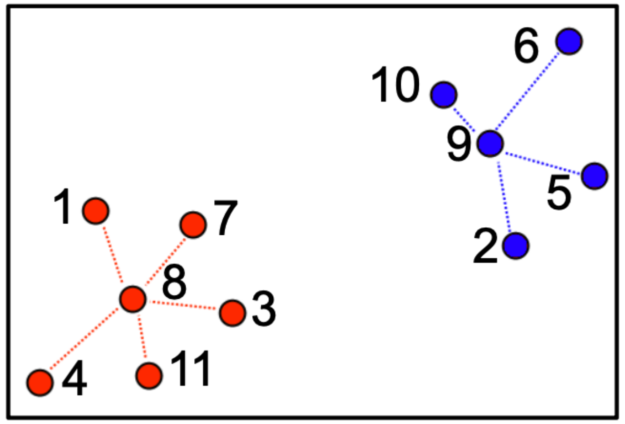
- Using a representative
  a) A point in center of cluster (e.g., centroid, median, mode)
  b) A point in the training data (exemplar/medoid)

$z^{(1)} = \binom{1}{1}, z^{(2)} = \binom{5}{4}$



centroid

$z^{(1)} = 8, z^{(2)} = 9$



exemplar

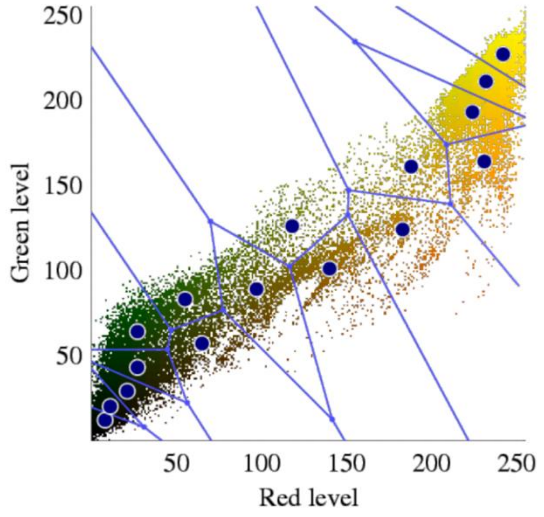
Examples of Clustering Applications

- Data compression

8 bits

RGB
(12,165,73)
(11,167,79)
:

save as label



$\longrightarrow$

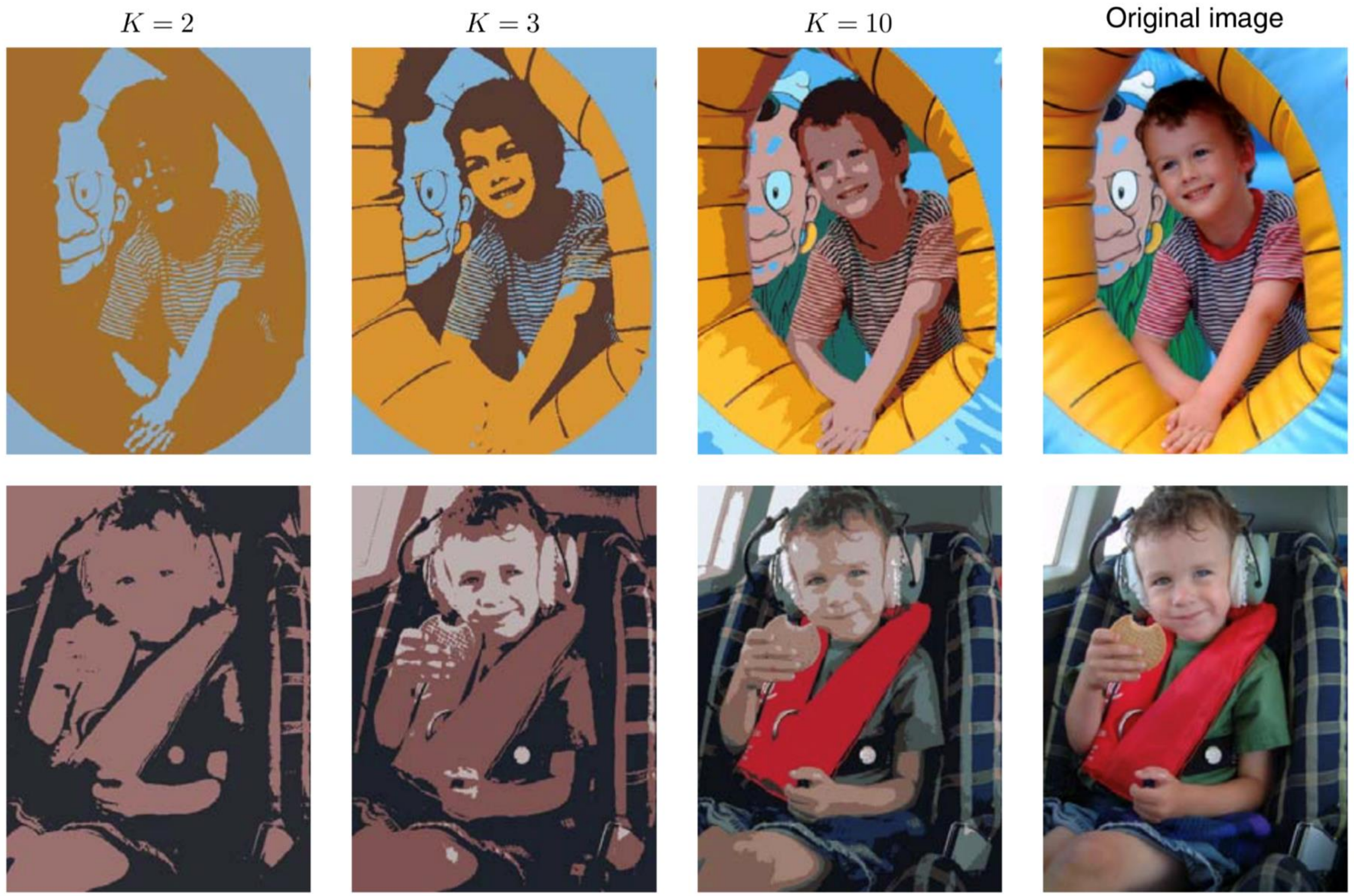
Labels
5
32 $\rightarrow$ 5 bits (2^5)
:

Dictionary
1 ~ (10, 160, 70)
2 ~ (40, 240, 20)
:

Examples of Clustering Applications

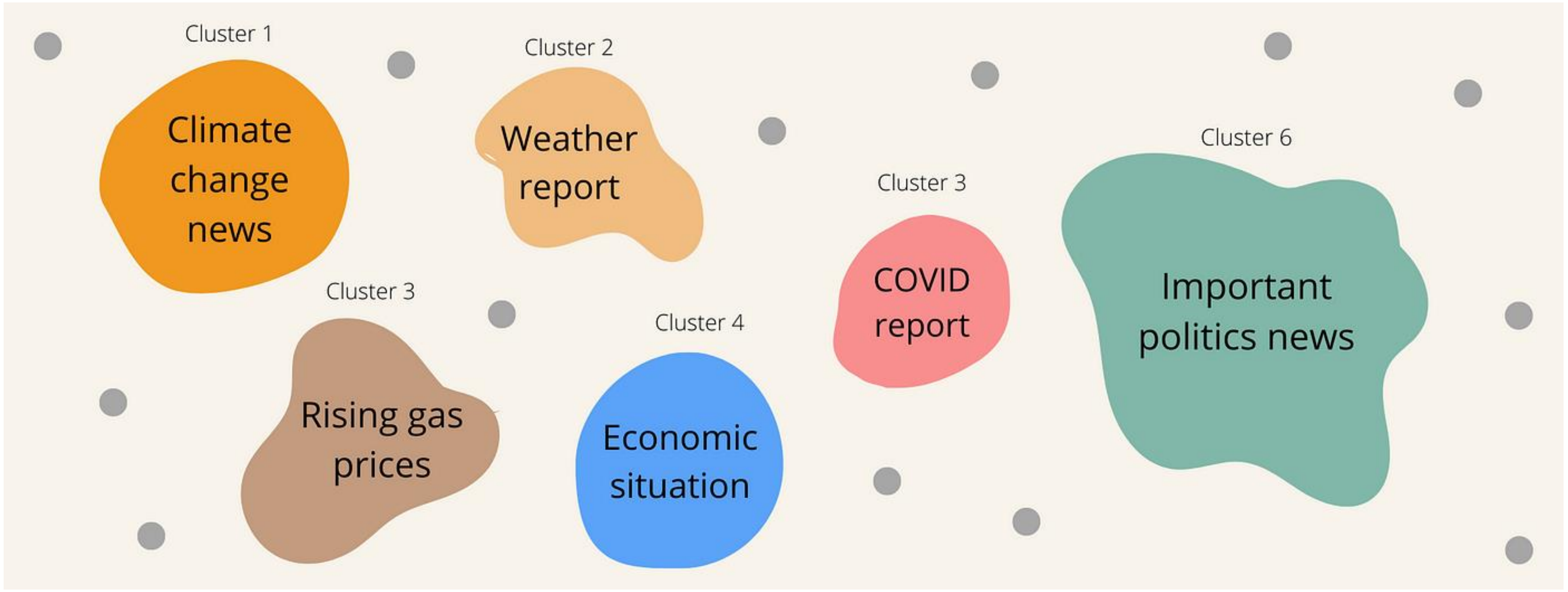
- Image segmentation

higher k, higher segmentation



Examples of Clustering Applications

- News aggregation



img source

Cluster Selection Criterion

for comparison of points

- We need a **criterion** to compare pairs of points, determining whether
  - they are *similar* (in the same cluster)
  - Or *dissimilar* (in different clusters)

- There are a variety of options for similarity/distance metric, and the choice of metric depends on the *nature of the data* and the *characteristics* you aim to capture.

- The two most commonly used ones are:
  - Cosine Similarity
  - Euclidean Distance

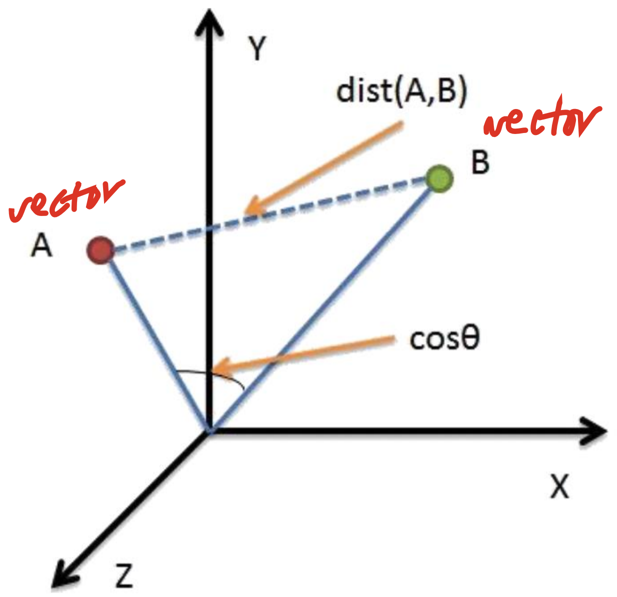
Similarity Metric

$\cos \theta = \frac{a \cdot b}{|a||b|}$, θ is ∠ btw a & b

- **Cosine similarity** is simply the angle between two vectors (data points):

$$\cos(x^{(i)}, x^{(j)}) = \frac{x^{(i)} \cdot x^{(j)}}{\|x^{(i)}\| \, \|x^{(j)}\|} = \frac{\sum_{l=1}^{d} x_l^{(i)} x_l^{(j)}}{\sqrt{\sum_{l=1}^{d} (x_l^{(i)})^2} \sqrt{\sum_{l=1}^{d} (x_l^{(j)})^2}}$$

↳ $\in [-1, 1]$



  - It measures the similarity between two vectors of an inner product space.
  - It is measured by the *cosine of the angle* between two vectors and determines *whether two vectors are pointing in roughly the same direction*.
  - The smaller the angle, higher the cosine similarity.

↳ $\cos(0) = 1$, same dir
$\cos(180) = -1$; opp dir
$\cos(90) = 0$

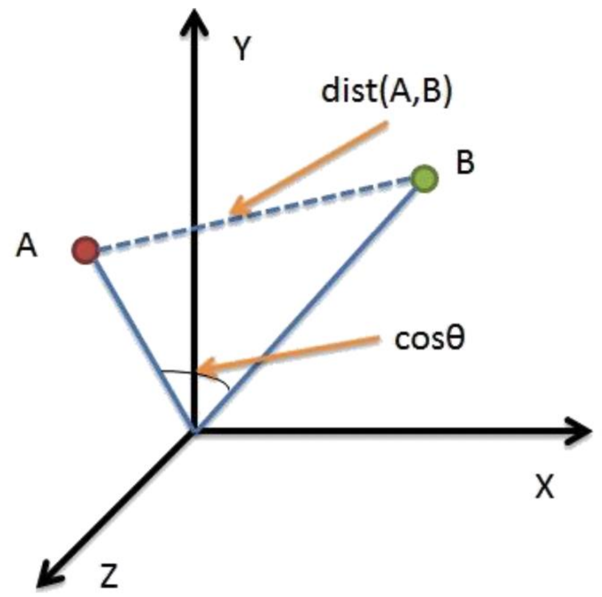
$1 - \cos(x_i, x_j)$, $\in [0, 2]$
to convert similarity matrix to distance matrix

Distance Metric

- In this lecture, we will primarily use **squared Euclidian distance**:

$$\mathrm{dist}(x^{(i)}, x^{(j)}) = \left\|x^{(i)} - x^{(j)}\right\|^2 = \sum_{l=1}^{d}(x_l^{(i)} - x_l^{(j)})^2$$

  - It measures the *straight-line distance* between two points in space.
  - Comparing to Euclidean distance, **squared Euclidean distance** produces the *same result* but avoids an unnecessary square-root calculation and sidesteps issues of numerical precision.
  - The smaller the Euclidian distance, higher the similarity.



Clustering – Cost Function

- Once we have the distance metric, we can specify an objective function for clustering:

$$Cost(C^1, \dots, C^k, z^{(1)}, \dots, z^{(k)}) = \sum_{j=1}^{k} \sum_{i \in C^j} d(x^{(i)}, z^{(j)})$$

  - k: the number of clusters (*pre-defined*)
  - C^j: the j-th cluster
  - z^j: the representative of a cluster
  - $d(x^i, z^j)$: the distance metric that measures the pairwise distance between i-th sample in C^j and z^j

- The cost here depends on both the clusters and how the representatives are chosen for each cluster, but we can derive one from another.

k-Means Clustering

k-Means Clustering

- ***k*-means** is a *partition*-based clustering method:
  - Hard assignments of data points to clusters: each point must belong to one and only one cluster.
- Cost Function:

$$\text{cost}(C_1, C_2, \ldots, C_k, z^{(1)}, \ldots, z^{(k)}) = \sum_{j=1\ldots k} \sum_{i \in C_j} \left\| x^{(i)} - z^{(j)} \right\|^2$$

  - Use squared Euclidian distance as the metric
  - Each cluster is specified by a “centroid”:

$$z^{(j)} = \frac{1}{|C_j|} \sum\nolimits_{i \in C_j} x^{(i)}$$

  - Clusters can be formed once their representatives are known:

$$C_j = \{i \in \{1, \ldots, n\} \text{ s.t. the closest representative of } x^{(i)} \text{ is } z^{(j)}\}$$

k-Means Clustering

- Clusters can be formed once their representatives are known:

$$C_j = \{i \in \{1,\ldots,n\} \text{ s.t. the closest representative of } x^{(i)} \text{ is } z^{(j)}\}$$

- *In other words*, these clusters define an optimal clustering *w.r.t.* our cost function for a fixed setting of the representatives:

$$\begin{aligned}\text{cost}(z^{(1)},\ldots,z^{(k)}) &= \min_{C_1,\ldots,C_k} \text{cost}(C_1,\ldots,C_k,z^{(1)},\ldots,z^{(k)}) \\ &= \min_{C_1,\ldots,C_k} \sum_{j=1\ldots k} \sum_{i\in C_j} \left\|x^{(i)} - z^{(j)}\right\|^2 \\ &= \sum_{i=1,\ldots,n} \min_{j=1,\ldots,k} \left\|x^{(i)} - z^{(j)}\right\|^2\end{aligned}$$

k-Means Algorithm

- ***k*-means** algorithm, *a.k.a.* **Lloyd's algorithm**, is an ***iterative*** algorithm that alternatingly finds:
  - best clusters for centroids (*assigning data points to clusters*)
  - and best centroids for clusters (*computing the cluster means*)

```
1. Initialize centroids $z^{(1)}, \ldots, z^{(k)}$
2. Repeat until there is no further change in cost
    (a) For each $j = 1, \ldots, k : C_j = \{i \text{ s.t. } x^{(i)} \text{ is closest to } z^{(j)}\}$
    (b) For each $j = 1, \ldots, k : z^{(j)} = \frac{1}{|C_j|} \sum_{i \in C_j} x^{(i)}$ (cluster mean)
Each iteration requires $\mathcal{O}(kn)$ operations.
```

Or until some maximum number of iterations is exceeded

$n = k$

c_j

$\uparrow$

$$\left.\begin{matrix} x_1 \; .. \; x_2 \cdots x_n \\ \\ z_1 \cdots z_2 \cdots z_K \end{matrix}\right\} \; n \times k$$

k-Means Convergence

- The **convergence** of k-means algorithm is assured, as each phase reduces the value of the objective function. *However*,
  - *Not necessarily* yield a solution that is optimal (*i.e.,* may converge to a local rather than global minimum)
  - *Different initializations* lead to *different solutions*

Img generated by Bard

Why is the convergence of the *k*-means algorithm guaranteed?

k-Means Convergence

Repeat until there is no further change in cost

(a) For each $j = 1, \dots, k : C_j = \{i \text{ s.t. } x^{(i)} \text{ is closest to } z^{(j)}\}$

(b) For each $j = 1, \dots, k : z^{(j)} = \frac{1}{|C_j|} \sum_{i \in C_j} x^{(i)}$ (cluster mean)

- Step (a) finds new clusters $C'_1, \dots, C'_k$ for fixed centroids:

$$\begin{aligned} \text{cost}(C_1, \dots, C_k, z^{(1)}, \dots, z^{(k)}) &\overset{\text{(a)}}{\geq} \min_{C_1, \dots, C_k} \text{cost}(C_1, \dots, C_k, z^{(1)}, \dots, z^{(k)}) \\ &= \text{cost}(C'_1, \dots, C'_k, z^{(1)}, \dots, z^{(k)}) \end{aligned}$$

  - The **inequality (a)** is equality *only when the algorithm converges*.

- Step (b) finds new centroids $z'^{(1)}, \dots, z'^{(k)}$ for the new clusters:

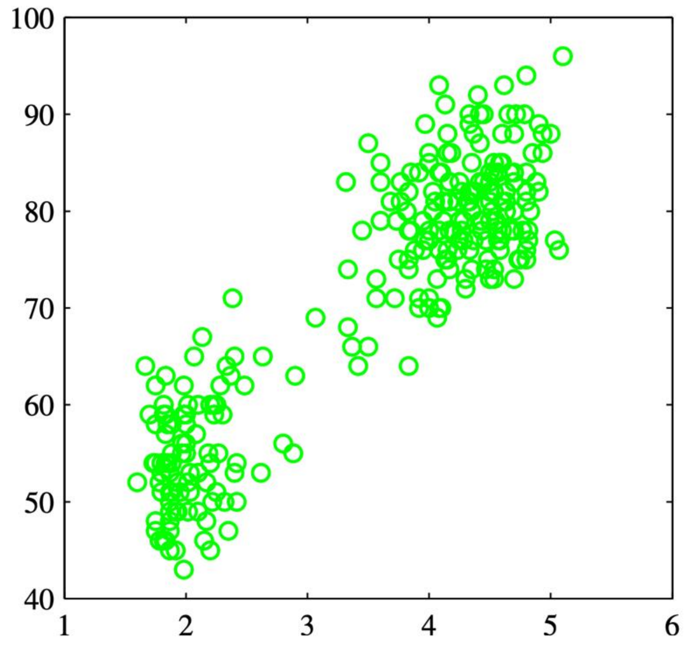
$$\begin{aligned} \text{cost}(C'_1, \dots, C'_k, z^{(1)}, \dots, z^{(k)}) &\overset{\text{(b)}}{\geq} \min_{z^{(1)}, \dots, z^{(k)}} \text{cost}(C'_1, \dots, C'_k, z^{(1)}, \dots, z^{(k)}) \\ &= \text{cost}(C'_1, \dots, C'_k, z'^{(1)}, \dots, z'^{(k)}) \end{aligned}$$

  - The **inequality (b)** is equality *only when the centroids are optimal for the given clusters (converges)*.

- Inequality (a) and (b) guarantee that the k-means algorithm monotonically decreases the cost. As the cost has a ***lower bound*** (non-negative), the algorithm must ***converge***.

Example - Old Faithful Dataset

- Geyser in Yellowstone National Park
- 272 observations on 2 variables:
  - waiting time between eruptions (*vertical* axis)
  - duration of the eruption (*horizontal* axis)



Credit to: Bruce T. Gourley www.brucegourley.com.

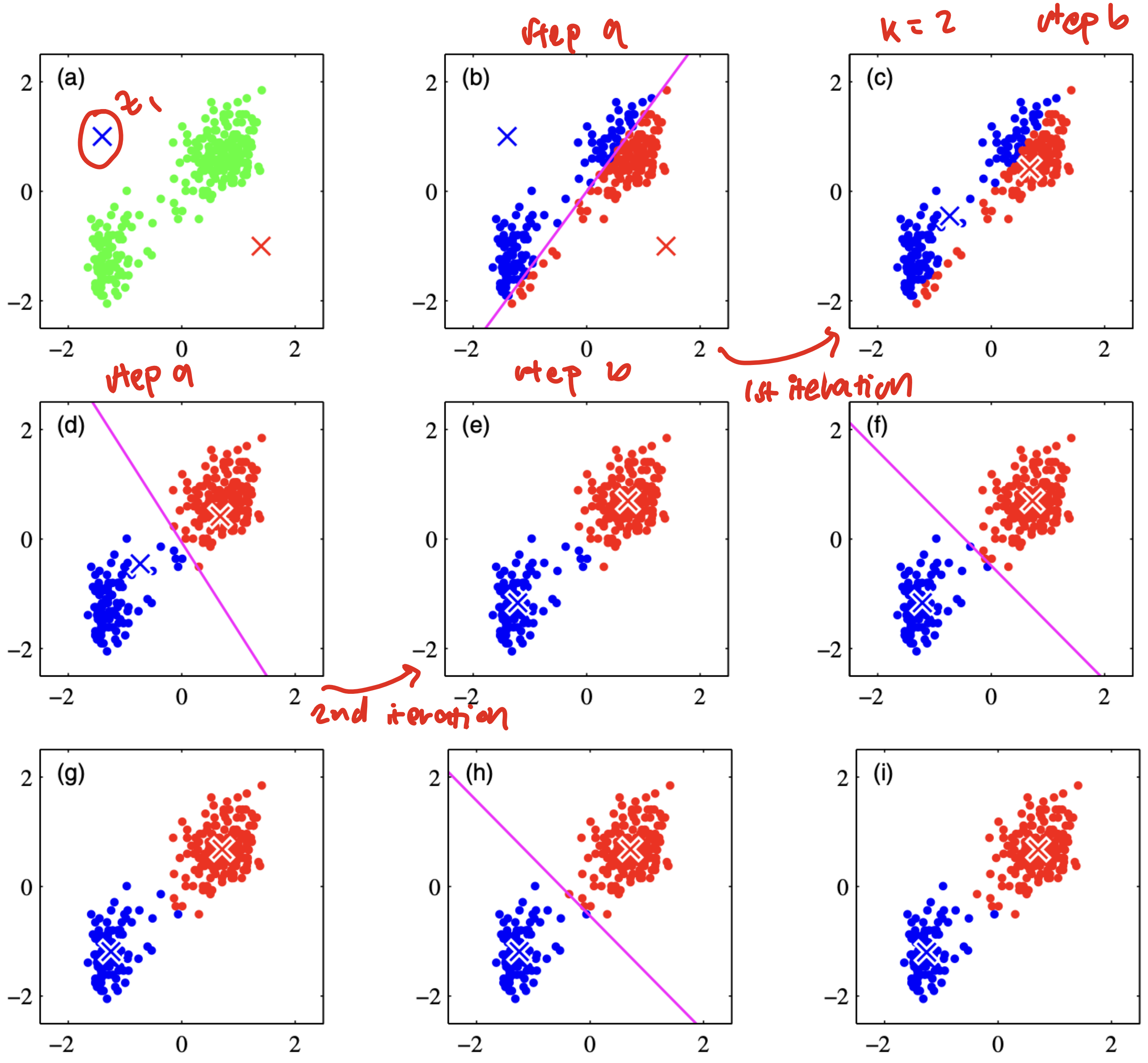


Illustration of the k-means algorithm using the re-scaled Old Faithful data set.

(a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres $z^{(1)}$ and $z^{(2)}$ are shown by the red and blue crosses, respectively.

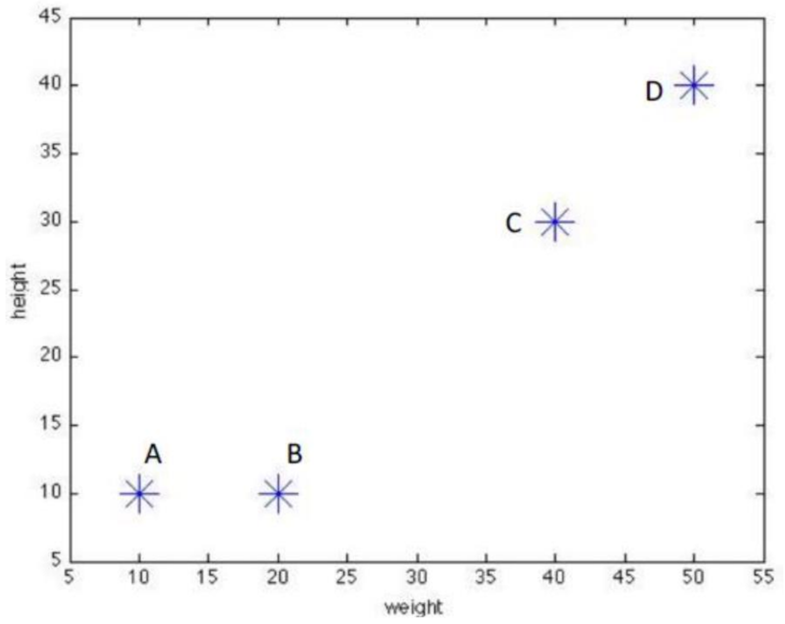
(b) In this cluster assignment step, each data point is assigned either to the red cluster or to the blue cluster, according to *which cluster centre is nearer*. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on.

(c) In the subsequent centroid update step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster.

(d)–(i) show successive cluster assignment and centroid update steps through to final convergence of the algorithm.

Exercise - k-Means Clustering

- Suppose we have 4 boxes of different sizes and we want to divide them into 2 clusters via k-means. Each box represents one point with two attributes (X,Y):



$$A = (10,10),$$
$$B = (20,10),$$
$$C = (40,30),$$
$$D = (50,40)$$

Exercise: k-Means Clustering

- **Initial centroids**: suppose we choose points A and B as the initial centroids/centres, so $z^1 = (10, 10)$ and $z^2 = (20, 10)$.
- **Calculate object-centroid distance**:

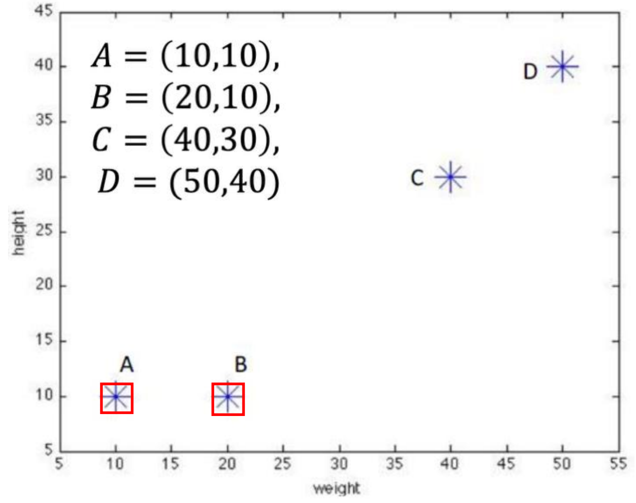
	A	B	C	D
Centre 1	0	10	36.06	50
Centre 2	10	0	28.28	43.43

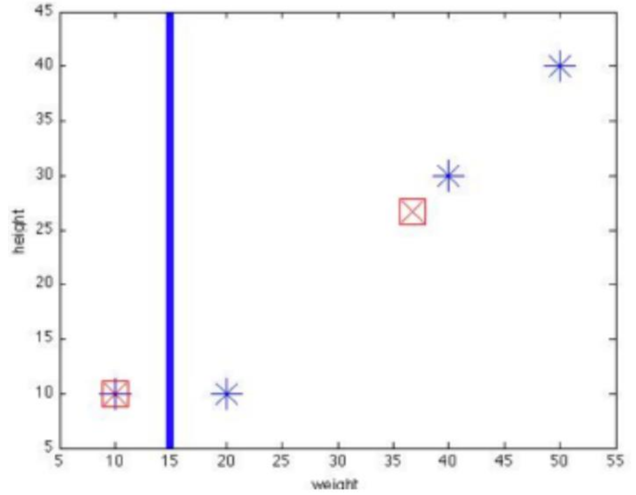
- **Object clustering**:

	A	B	C	D
Centre 1	1	0	0	0
Centre 2	0	1	1	1

- **Determine new centroids**:

$$z^1 = (10, 10), \;\; z^2 = \left(\frac{20+40+50}{3}, \frac{10+30+40}{3}\right) = (36.7, 26.7)$$





Exercise: k-Means Clustering

- **Re-calculate** object-centroid distance:

	A	B	C	D
Centre 1	0	10	36.06	50
Centre 2	31.4	23.6	4.7	18.9

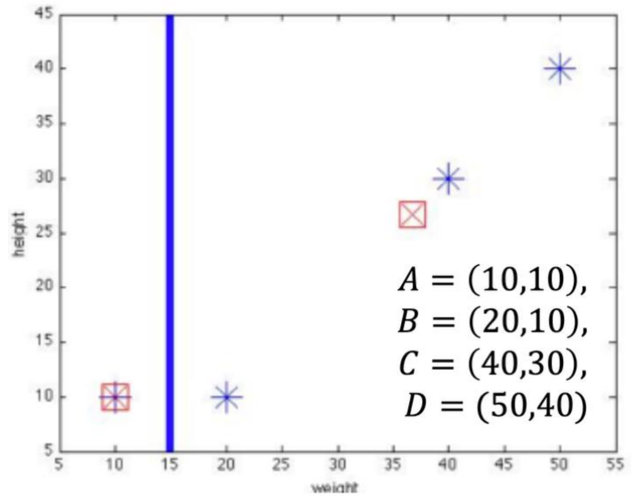
- **Object clustering**:

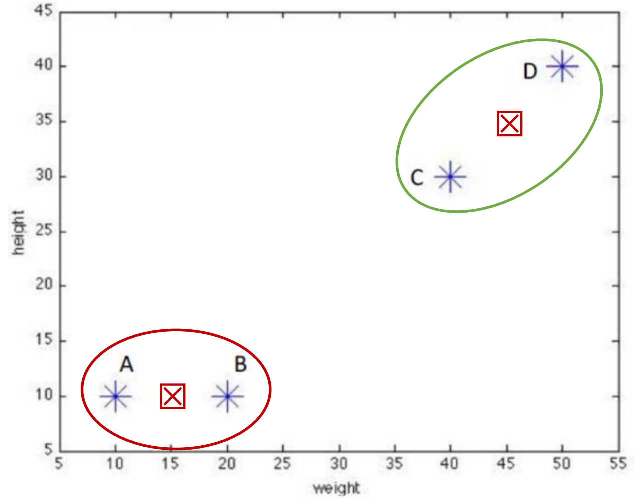
	A	B	C	D
Centre 1	1	1	0	0
Centre 2	0	0	1	1

- **Determine new centroids**:

$$z^1 = \left(\frac{10+20}{2}, \frac{10+10}{2}\right) = (15,10), \;\; z^2 = \left(\frac{40+50}{2}, \frac{30+40}{2}\right) = (45, 35)$$

➢ If repeat the above steps, you will find that *the cluster membership did not change*, so the k-means *terminates*.



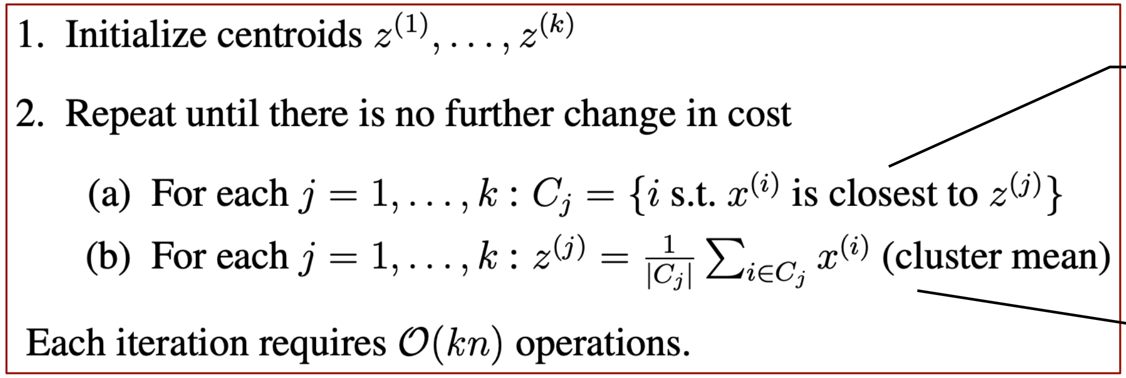


Summary

1. What is the k-means algorithm and what type of problem does it solve?

   k-means is a *partition*-based clustering method in unsupervised learning, it assigns each sample to one & only one cluster.

2. How to implement the k-means algorithm and the rationales behind the two update steps?



```
1. Initialize centroids $z^{(1)}, \dots, z^{(k)}$
2. Repeat until there is no further change in cost
    (a) For each $j = 1, \dots, k : C_j = \{i \text{ s.t. } x^{(i)} \text{ is closest to } z^{(j)}\}$
    (b) For each $j = 1, \dots, k : z^{(j)} = \frac{1}{|C_j|} \sum_{i \in C_j} x^{(i)}$ (cluster mean)
Each iteration requires $\mathcal{O}(kn)$ operations.
```

Fix centroids, find best clusters

Fix clusters, find best centroids

3. What convergence properties does the k-means algorithm have?

   The convergence of k-means is guaranteed, but the optimality of the solution is not guaranteed. Different solutions may arise based on the initialization of centroids.

Acknowledgements

- Some slides and content of this lecture are adopted from:
  - MIT 6.036 Introduction to Machine Learning
  - SUTD 50.007 Machine Learning, Spring 2023 (Asst Prof. Malika Meghjani)
  - Bishop, C. M. (2006). Pattern recognition and machine learning. Springer. (Chapter 9)

Tutorial 1 – **Linear Algebra**

- **4:30 – 5:30 pm**
- **10 Feb 2026 (Tuesday)**
- **Lecture Theatre 2**
  (Building 1 Level 2,3)