



# Supervised Learning and Clustering

Melih Kandemir

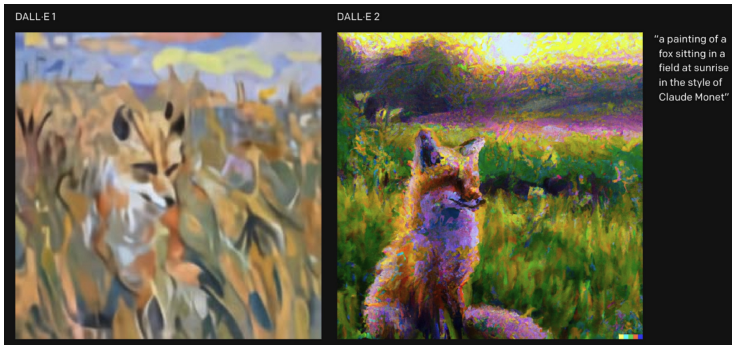
University of Southern Denmark  
Department of Mathematics and Computer Science (IMADA)  
kandemir@imada.sdu.dk

December 4, 2024

# Section 1: What is machine learning?

A computer program is said to learn  
from experience **E**  
with respect to a task **T**  
and performance measure **P**,  
if its performance at task **T**, as measured in **P**, improves with  
experience **E**.

# What can machine learning do?



# What can machine learning do?

## Chess



AlphaZero vs. Stockfish

W:29.0% D:70.6% L:0.4%



W:2.0% D:97.2% L:0.8%

## Shogi



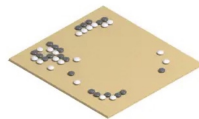
AlphaZero vs. Elmo

W:84.2% D:2.2% L:13.6%



W:98.2% D:0.0% L:1.8%

## Go



AlphaZero vs. AGO

W:68.9% L:31.1%



W:53.7% L:46.3%

AZ wins ■ AZ draws ■ AZ loses ■ AZ white ○ AZ black ●

# What can machine learning do?

M

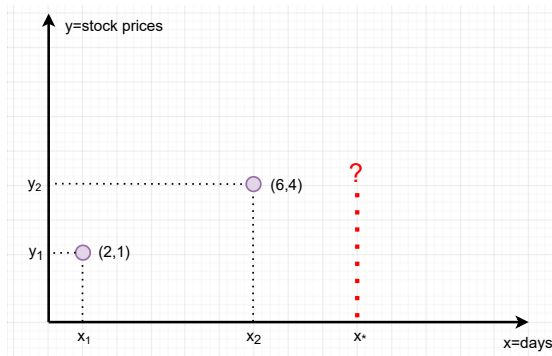
Tell me how ChatGPT works.



ChatGPT is a large language model that uses deep learning techniques to generate human-like text. It is based on the GPT (Generative Pre-trained Transformer) architecture, which uses a transformer neural network to process and generate text. The model is pre-trained on a massive dataset of text, such as books, articles, and websites, so it can understand the patterns and structure of natural language. When given a prompt or a starting point, the model uses this pre-trained knowledge to generate text that continues the given input in a coherent and natural way.



# Stock price prediction

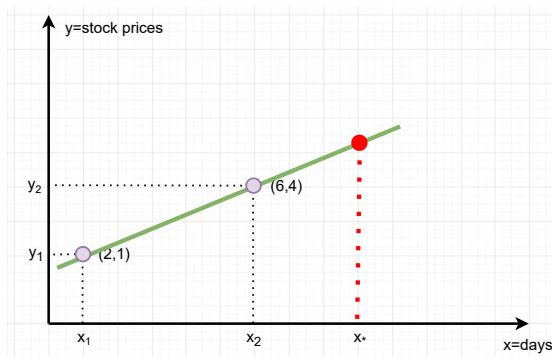


We are given the following two observations:

- The stock price at day number 2 is 1 DKK.
- The stock price at day number 6 is 4 DKK.

How can we predict the price at an arbitrary day  $x_*$  in the future?

# How far do we get with high school math?

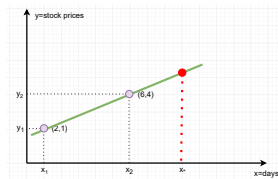


- Find the equation of the line passing through the two points

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \rightarrow y - 1 = \frac{4 - 1}{6 - 2}(x - 2) \rightarrow y = 0.75x - 0.5$$

- Treat the line as a predictor function:  $f(x) = 0.75x - 0.5$

# Which recipe did we follow?



- Collect a **data set**, i.e. a set of observations

$$S = \{(2, 1), (6, 4)\}$$

- Assume a **model** about how the system works: **a line**
- **Fit** the model to data, i.e. **learn**: Find the equation of the line passing through the two points

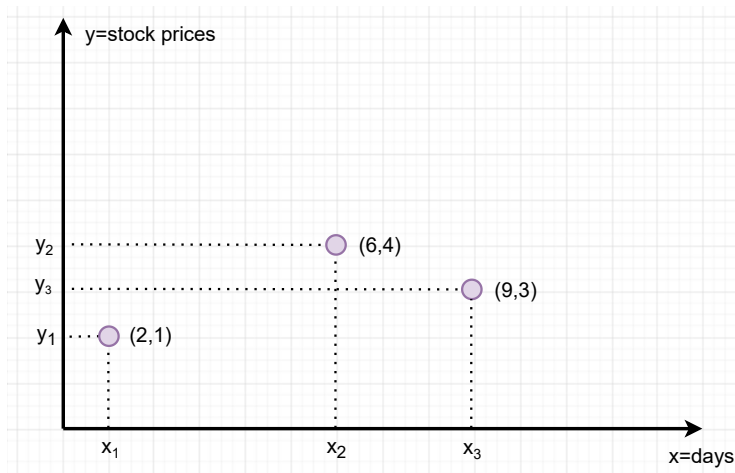
$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \rightarrow y - 1 = \frac{4 - 1}{6 - 2}(x - 2)$$
$$\rightarrow y = 0.75x - 0.5$$

- Use  $f(x) = 0.75x - 0.5$  to make new predictions, i.e. **generalize**.



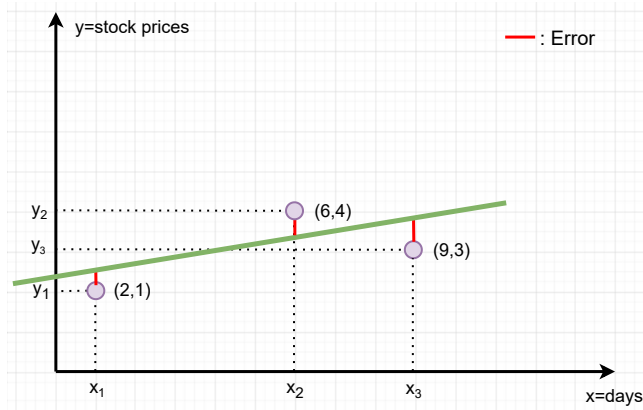
# What if we have a third observation?

Now machine learning starts.



There is no line that can go through all the three data points!

# How about finding the **best** line $y = ax + b$ ?



- For each choice of  $a, b$ , we will have a different line. Which one is the best? Best in terms of what?
- **Hint:** Whichever line we choose, the model will make an **error**. Quantify the error first, minimize the error next.

# Error

Since it does not matter if our prediction  $y_*$  was above or below the target value  $y$ , it is reasonable to define the error as:

$$e(y, y_*) = \left( \underbrace{y_*}_{\text{predicted}} - \underbrace{y}_{\text{observed}} \right)^2$$

What is the geometric shape of the function that has  $y_*$  on the x-axis and  $e(y, y_*)$  on the y-axis?

# Loss

Each time our model makes a prediction, it will incur some error. The sum of these errors gives us the **loss** of our model (a line defined by a slope  $a$  and an intercept  $b$ ) on the data set  $S^1$ :

$$\begin{aligned}L(a, b, S) &= \frac{1}{3} \sum_{i=1}^3 e(f(x_i, a, b), y_i) \\&= \frac{1}{3} \sum_{i=1}^3 (f(x_i, a, b) - y_i)^2 \\&= \frac{1}{3}(2a + b - 1) + \frac{1}{3}(6a + b - 4) + \frac{1}{3}(9a + b - 3) \\&= \sum_{i=1}^3 \frac{1}{3} (ax_i + b - y_i)^2\end{aligned}$$

---

<sup>1</sup>Remember:  $S = \{(x_1, y_1) = (2, 1), (x_2, y_2) = (6, 4), (x_3, y_3) = (9, 3)\}$

# Learning: Choosing the best of three lines

Let

$$\mathcal{H} := \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$$

be all the options we have. How would you choose the best one?

# Learning: Choosing the best of three lines

Evaluate the loss with each option

$$L(a_1, b_1, S) := \frac{1}{3}(2a_1 + b_1 - 1) + \frac{1}{3}(6a_1 + b_1 - 4) + \frac{1}{3}(9a_1 + b_1 - 3)$$

$$L(a_2, b_2, S) := \frac{1}{3}(2a_2 + b_2 - 1) + \frac{1}{3}(6a_2 + b_2 - 4) + \frac{1}{3}(9a_3 + b_3 - 3)$$

$$L(a_3, b_3, S) := \frac{1}{3}(2a_3 + b_3 - 1) + \frac{1}{3}(6a_2 + b_2 - 4) + \frac{1}{3}(9a_3 + b_3 - 3)$$

and choose the line that gives the smallest loss. In formal terms, choose  $a_{best}, b_{best}$  that satisfies

$$L(a_{best}, b_{best}, S) = \min\{L(a_1, b_1, S), L(a_2, b_2, S), L(a_3, b_3, S)\}.$$

Choosing the **argument** that **minimizes** a function can be expressed shortly via the **arg min** operation:

$$a_{best}, b_{best} = \arg \min_{(a,b) \in \mathcal{H}} L(a, b, S)$$

# Learning: Choosing the best of infinitely many lines

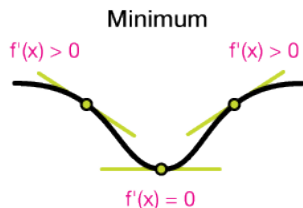
- In fact  $a, b$  can be any real number. So we have infinite number of options. How shall we find out the best one?
- Best:  $a, b$  that minimizes the loss:

$$a_{best}, b_{best} = \arg \min_{a, b \in \mathbb{R}} L(a, b, S)$$

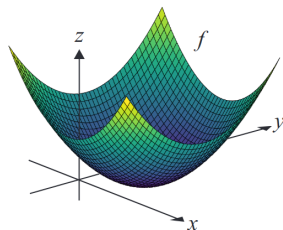
- What is special with the point  $L(a_{best}, b_{best}, S)$ ? **Hint:** How does the loss function **change** in the neighborhood of this point?
  - ▶ If there is upward slope, turn back and go down.
  - ▶ If there is downward slope, go further ahead.
  - ▶ If there is no slope, here it is!

How can we express the change of a function value with respect to a change in its input values in mathematical terms?

# The derivative!



(a) Minima and slopes



(b) 2-dimensional loss

We are looking for values  $a_{best}, b_{best}$  that satisfy:

$$L'_a(a = a_{best}, b = b_{best}, S) = 0,$$

$$L'_b(a = a_{best}, b = b_{best}, S) = 0.$$



# Recipe for finding $a_{best}$ and $b_{best}$

- 1 Set  $L'_b(a, b, S) = 0$  and solve for  $b$ , which will be an expression of  $b_{best}$  in terms of  $a$ .
- 2 Set  $L'_a(a, b = b_{best}, S) = 0$  and solve for  $a$ , which will be a function of  $b_{best}$ . Replace wherever you see  $b_{best}$  by the expression found in Step (1).
- 3 Replace the solution in Step (2) into the expression found in Step (1) to find  $b_{best}$ .

## Step 1: Find $b_{best}$ in terms of $a$

$$\begin{aligned} 0 &= \frac{1}{3} \sum_{i=1}^3 ((ax_i + b - y_i)^2)'_b && \text{as } (f(x) + g(x))' = f'(x) + g'(x) \\ &= \sum_{i=1}^3 (ax_i + b - y_i) && \text{as } f(g(x))' = f'(g(x))g'(x) \\ &= 3b - \sum_{i=1}^3 (y_i - ax_i) && \text{Rearrange terms} \end{aligned}$$

Then leaving  $b$  alone on one side, we get

$$b_{best} = \underbrace{\frac{1}{3} \sum_{i=1}^3 y_i}_{\bar{y}} - a \underbrace{\left( \frac{1}{3} \sum_{i=1}^3 x_i \right)}_{\bar{x}} = \bar{y} - a\bar{x}$$

where  $\bar{x}$  and  $\bar{y}$  are the *center of mass* of the data.

## Step 2: Solve for $a_{best}$

$$\begin{aligned}0 &= \frac{1}{3} \sum_{i=1}^3 ((ax_i + b_{best} - y_i)^2)'_a \\ &= \frac{1}{3} \sum_{i=1}^3 ((ax_i + (\bar{y} - a\bar{x}) - y_i)^2)'_a \\ &= \sum_{i=1}^3 (a(x_i - \bar{x}) - (y_i - \bar{y}))(x_i - \bar{x})\end{aligned}$$

Leaving  $a$  alone on one side, we get

$$a_{best} = \frac{\sum_{i=1}^3 (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^3 (x_i - \bar{x})^2}$$

## Step 3: Evaluate $b_{best}$

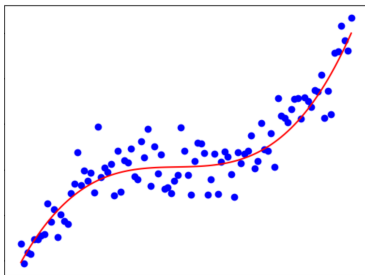
$$\begin{aligned} b_{best} &= \bar{y} - a_{best}\bar{x} \\ &= \bar{y} - \left( \frac{\sum_{i=1}^3 (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^3 (x_i - \bar{x})^2} \right) \bar{x} \end{aligned}$$

## The best fitting line to data set $S$ is then

$$f(x) = \frac{\sum_{i=1}^3 (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^3 (x_i - \bar{x})^2} x + \bar{y} - \left( \frac{\sum_{i=1}^3 (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^3 (x_i - \bar{x})^2} \right) \bar{x}$$

We can use this function to predict the output of any  $x$  we want.

# What if?



- The input-output relationship is nonlinear, i.e. no line-shaped predictor evaluated on the input can even approximate the output.
- We can evaluate the derivatives for arbitrary  $a, b$ , but, cannot find  $a, b$  values that make the derivatives zero?

# Re-express the derivative in a more generic way

$$L'_a(a, b, S) = \frac{2}{3} \sum_{i=1}^3 \underbrace{(ax_i + b - y_i)}_{f(x_i; a, b) - y_i} \underbrace{x_i}_{(ax_i)'_a}$$

The solution template would be the same if:

- we had not 3 but  $m$  data points, i.e.  $S = (x_1, y_1), \dots, (x_m, y_m)$
- our model were not a line but an arbitrary function  $f(x; \theta)$
- we computed the derivative of an arbitrary parameter  $\theta$  of this function

$$L'_\theta(\theta, S) = \frac{2}{m} \sum_{i=1}^m (f(x_i; \theta) - y_i) f'_\theta(x_i; \theta)$$

# The Gradient Descent Algorithm

- Choose a **learning rate**  $\alpha > 0$ .
- Initialize  $\theta$  to an arbitrary value
- **repeat**
  - ▶  $\theta \leftarrow \theta - \alpha L'_\theta(\theta, S)$
- **until**  $\theta$  no longer changes



# Making sense of gradient descent

Placing the generic loss derivative calculated in the previous slide into the update rule:

$$\theta \leftarrow \theta - \frac{2\alpha}{m} \sum_{i=1}^m (f(x_i; \theta) - y_i) f'_\theta(x_i)$$

In natural language, change parameter  $\theta$  proportionally to

- The signed prediction error of the current model  $f(x_i; \theta) - y_i$
- The rate the predictions of the model changes  $f'_\theta(x_i)$

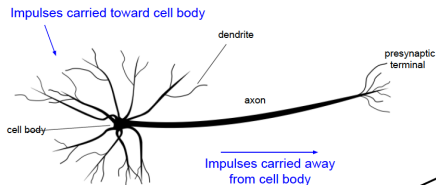
This means,

- Make big changes when predictions are bad
- Stop changing when predictions are good

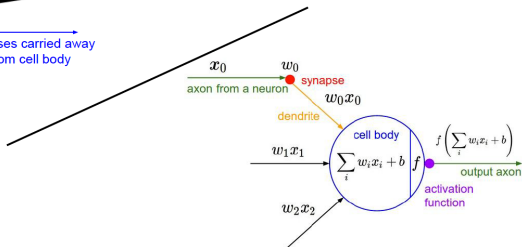
# Perceptron: The atom of intelligence

**North-West:** Biological neuron

**South-East:** Artificial neuron (perceptron)

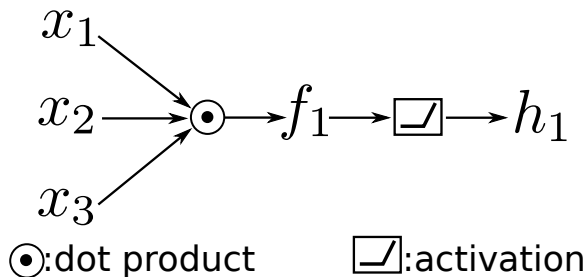


This image by Felipe Peruchio is licensed under [CC BY 3.0](#)



# The computational graph of a perceptron

**Computational graph:** block diagram of mathematical operations.



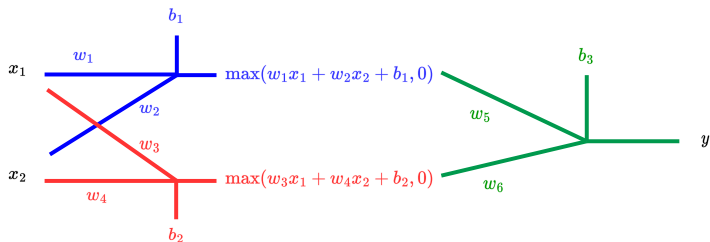
The neuron has weights assigned to each input channel:

- $w_1$  for  $x_1$ ,  $w_2$  for  $x_2$ ,  $w_3$  for  $x_3$ , and a bias term  $b$  to model the intercept just as in the line fitting example above.

It processes the given input in two steps:

- Linear mapping:  $f = \sum_{i=1}^3 w_i x_i + b$
- Nonlinear activation, for example:  $h = \max(0, f)$

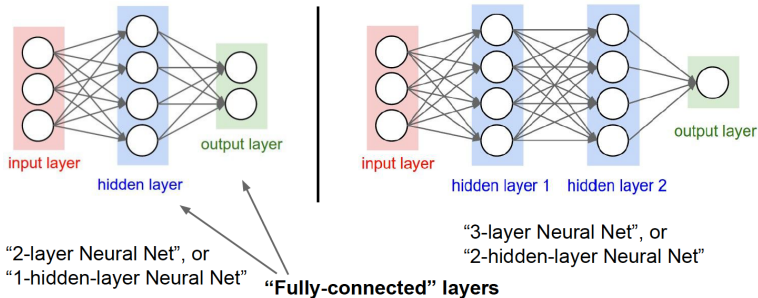
# Neural net: A group of connected perceptrons



This neural net will make the following prediction for the input observations  $(x_1, x_2)$ :

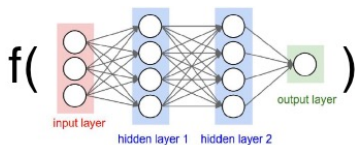
$$f(x_1, x_2) = w_5 \left( \max(w_1 x_1 + w_2 x_2 + b_1, 0) \right) + w_6 \left( \max(w_3 x_1 + w_4 x_2 + b_2, 0) \right) + b_3$$

# The Multi-Layer Perceptron (MLP)



# Large Language Models

- The predictor is not a line but a neural network



- The input is not a single value but a long sequence of words:

$$f(\text{her mother would like}) = \text{to}$$

# Large Language Models

- The neural net learns to pay more attention to important information in the input sequence:

$$f(\underbrace{\text{her mother}}_{\text{ignore}} \underbrace{\text{would like}}_{\text{attend}}) = \text{to}$$

Attended words influence the output, unattended words are ignored.

- The model generates text progressively by taking its own predictions as input, i.e. **autoregression**:

$$f(\text{her mother would like}) = \text{to}$$

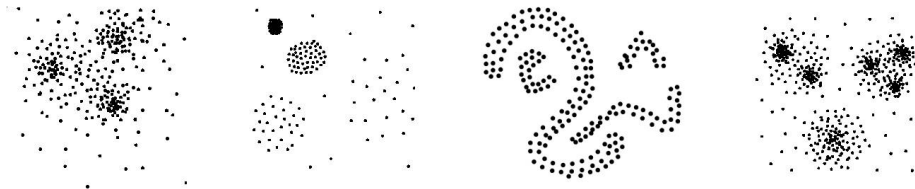
$$f(\text{her mother would like to}) = \text{buy}$$

$$f(\text{her mother would like to buy}) = \text{an}$$

$$f(\text{her mother would like to buy an}) = \text{icecream}$$

## Section 2: Purpose of Clustering

- identify a finite number of categories (classes, groups: clusters) in a given dataset
- **similar** objects shall be grouped in the same cluster, **dissimilar** objects in different clusters
- similarity is highly subjective, depending on the application scenario





# A Dataset can be Clustered in Different Meaningful Ways

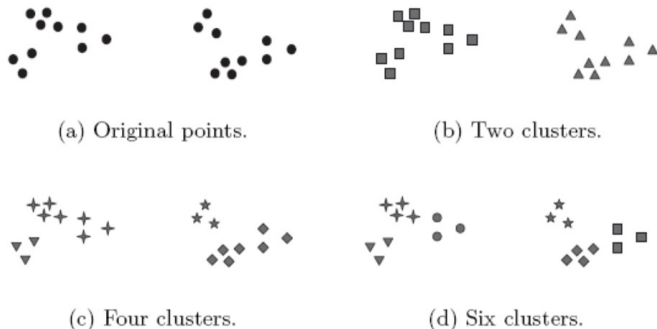
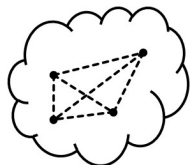


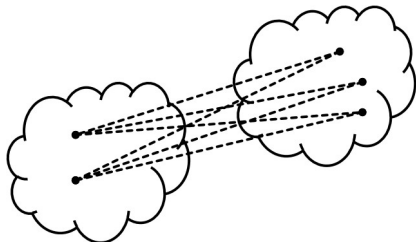
Figure 8.1. Different ways of clustering the same set of points.

# Criteria of Quality: Cohesion and Separation

- **cohesion:** how strong are the cluster objects connected (how similar, pairwise, to each other)?
- **separation:** how well is a cluster separated from other clusters?



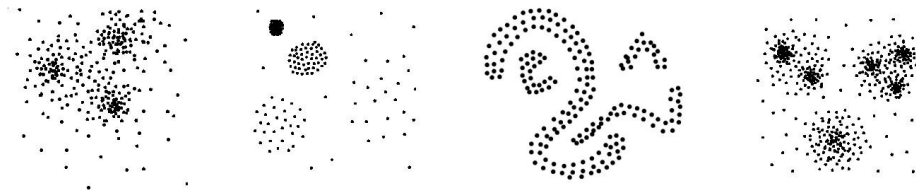
small within cluster distances



large between cluster distances

# Optimization of Cohesion

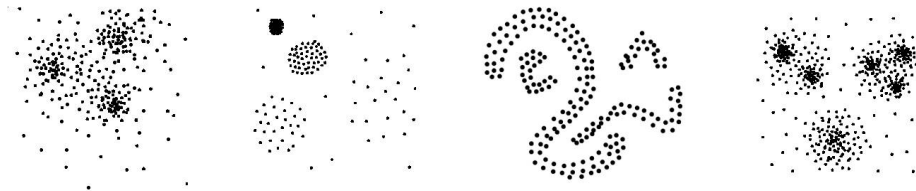
Partitional clustering algorithms partition a dataset into  $k$  clusters, typically minimizing some cost function (compactness criterion), i.e., optimizing cohesion.



# Assumptions for Partitioning Clustering

Central assumptions for approaches in this family are typically:

- number  $k$  of clusters known (i.e., given as input)
- clusters are characterized by their compactness
- compactness measured by some distance function (e.g., distance of all objects in a cluster from some cluster representative is minimal)
- criterion of compactness typically leads to convex or even spherically shaped clusters



# Construction of Central Points: Basics

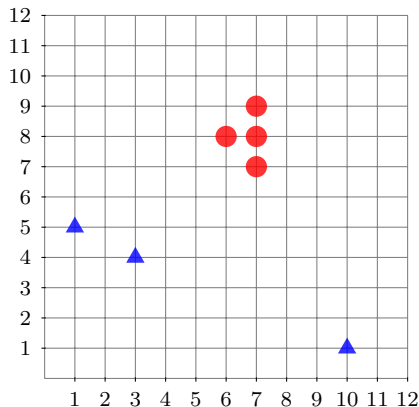
- data points (objects) are  $d$ -dimensional real-valued vectors

$$x = (x_1, \dots, x_d) \in \mathbb{R}^d$$

- and we measure distances between data point pairs by

$$\|x - y\|_2 = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$$

- **centroid**  $\mu_C$ : mean vector of all points in cluster  $C$



$$\mu_{C_i} = \frac{1}{|C_i|} \cdot \sum_{o \in C_i} o$$

# Construction of Central Points: Basics

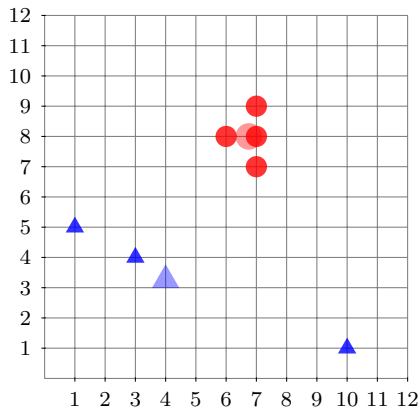
- data points (objects) are  $d$ -dimensional real-valued vectors

$$x = (x_1, \dots, x_d) \in \mathbb{R}^d$$

- and we measure distances between data point pairs by

$$\|x - y\|_2 = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$$

- **centroid**  $\mu_C$ : mean vector of all points in cluster  $C$



$$\mu_{C_i} = \frac{1}{|C_i|} \cdot \sum_{o \in C_i} o$$

# Construction of Central Points: Basics

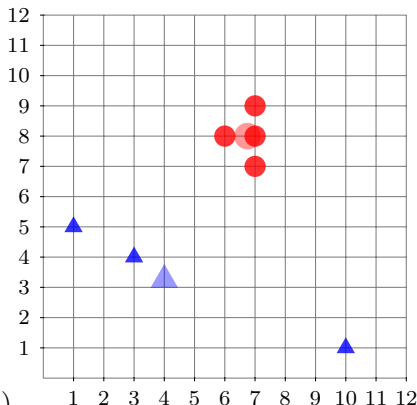
- measure of compactness for a cluster  $C$ :

$$TD^2(C) = \sum_{p \in C} \|p - \mu_C\|_2^2$$

(a.k.a. SSQ: sum of squares)

- measure of compactness for a clustering

$$TD^2(C_1, C_2, \dots, C_k) = \sum_{i=1}^k TD^2(C_i)$$



# Construction of Central Points: Basics

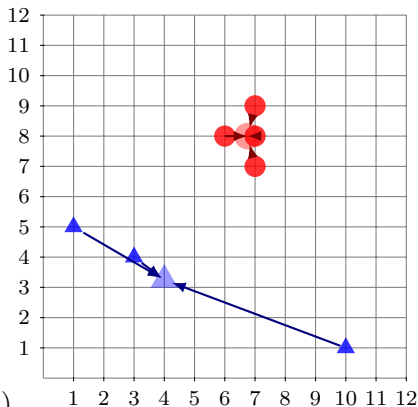
- measure of compactness for a cluster  $C$ :

$$TD^2(C) = \sum_{p \in C} \|p - \mu_C\|_2^2$$

(a.k.a. SSQ: sum of squares)

- measure of compactness for a clustering

$$TD^2(C_1, C_2, \dots, C_k) = \sum_{i=1}^k TD^2(C_i)$$



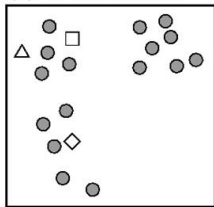


# Basic Algorithm

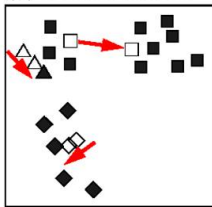
## Clustering by Minimization of Variance

- start with  $k$  (e.g., randomly selected) points as cluster representatives (or with a random partition into  $k$  “clusters”)
- repeat:
  - ▶ assign each point to the closest representative
  - ▶ compute new representatives based on the given partitions (centroid of the assigned points)
- until there is no change in assignment

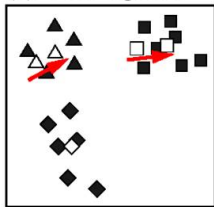
(a) Initialization



(b) First Iteration



(c) Convergence



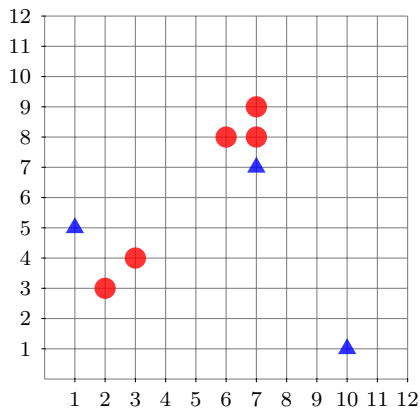
# $k$ -means

$k$ -means is a variant of the basic algorithm:

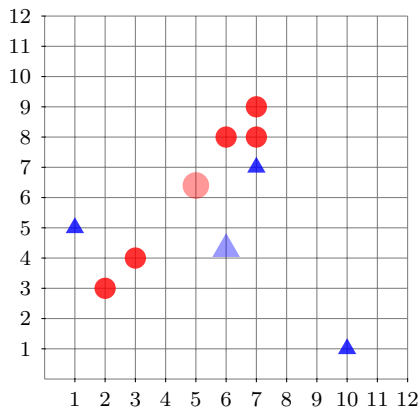
- a centroid is immediately updated when some point changes its assignment
- $k$ -means has very similar properties, but the result now depends on the order of data points in the input file

The name “ $k$ -means” is often used indifferently for any variant of the basic algorithm.

# $k$ -means Clustering – Lloyd/Forgy Algorithm



# $k$ -means Clustering – Lloyd/Forgy Algorithm

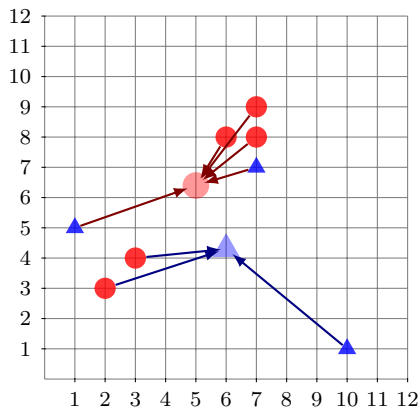


recompute centroids:

$$\mu \approx (6.0, 4.3)$$

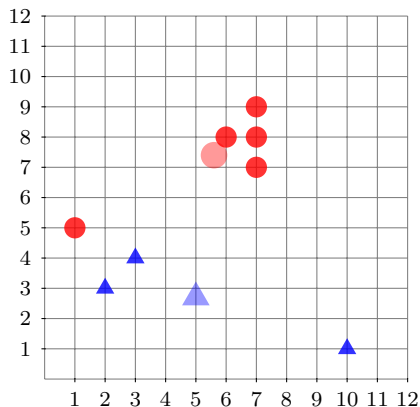
$$\mu \approx (5.0, 6.4)$$

# $k$ -means Clustering – Lloyd/Forgy Algorithm



reassign points

# $k$ -means Clustering – Lloyd/Forgey Algorithm

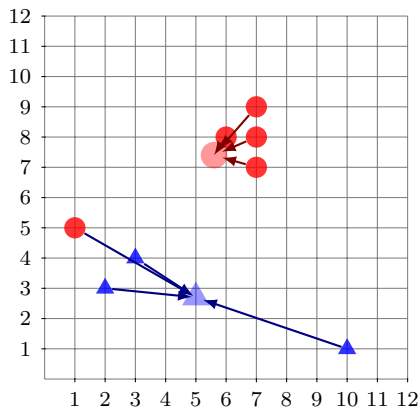


recompute centroids:

$$\mu \approx (5.0, 2.7)$$

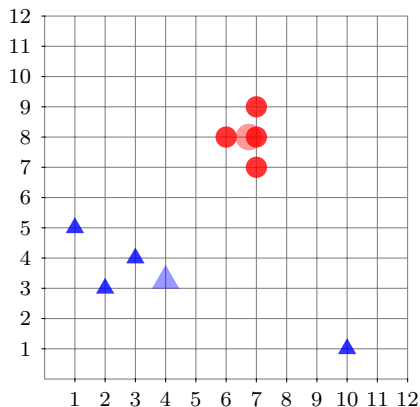
$$\mu \approx (5.6, 7.4)$$

# $k$ -means Clustering – Lloyd/Forgey Algorithm



reassign points

# $k$ -means Clustering – Lloyd/Forgy Algorithm



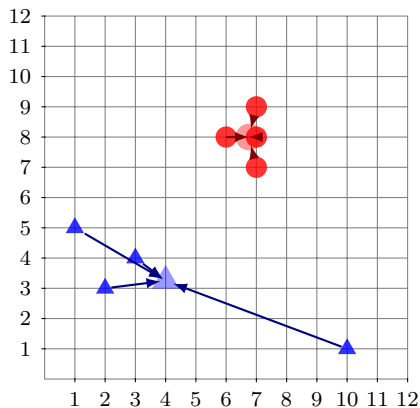
recompute centroids:

$$\mu \approx (4.0, 3.25)$$

$$\mu \approx (6.75, 8.0)$$

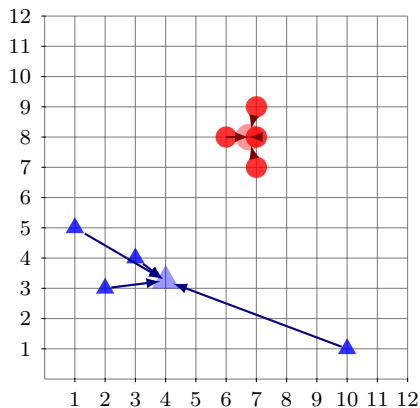


# $k$ -means Clustering – Lloyd/Forgey Algorithm



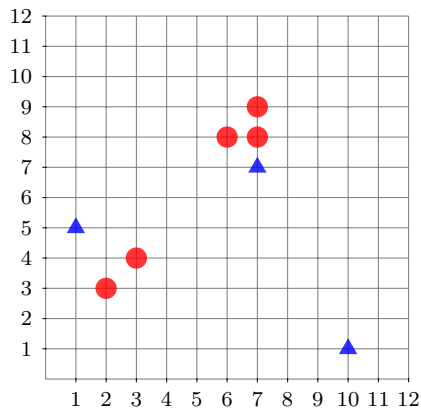
reassign points

# $k$ -means Clustering – Lloyd/Forgey Algorithm

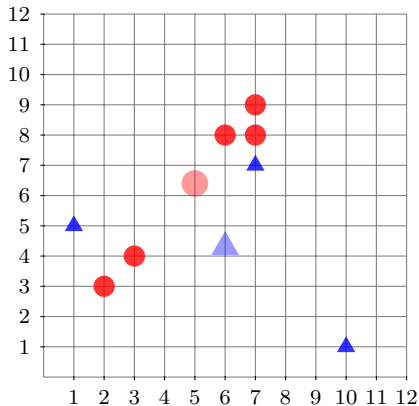


reassign points  
no change  
convergence!

# $k$ -means Clustering – MacQueen Algorithm



# $k$ -means Clustering – MacQueen Algorithm

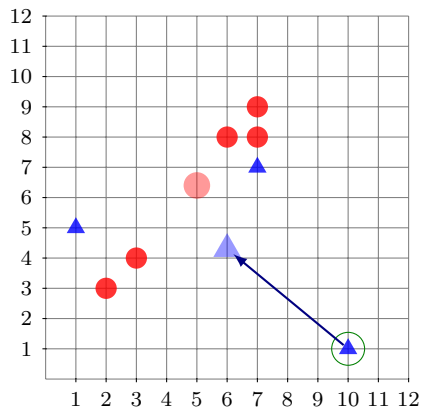


Centroids  
(e.g.: from  
previous iteration):

$$\mu \approx (6.0, 4.3)$$

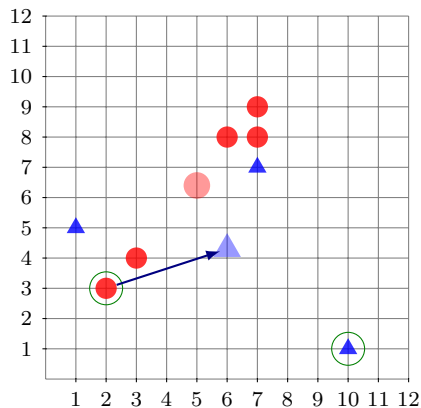
$$\mu \approx (5.0, 6.4)$$

# $k$ -means Clustering – MacQueen Algorithm



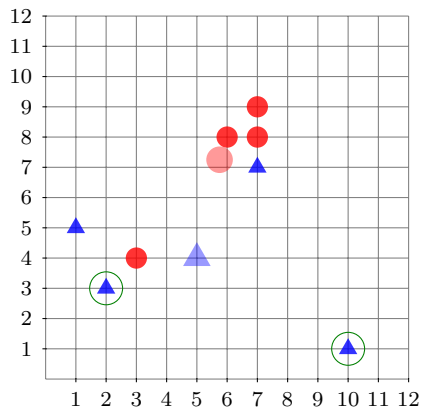
assign first point

# $k$ -means Clustering – MacQueen Algorithm



assign second point

# $k$ -means Clustering – MacQueen Algorithm

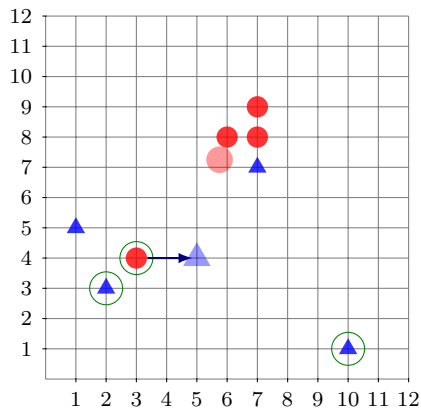


recompute centroids:

$$\mu \approx (5.0, 4.0)$$

$$\mu \approx (5.75, 7.25)$$

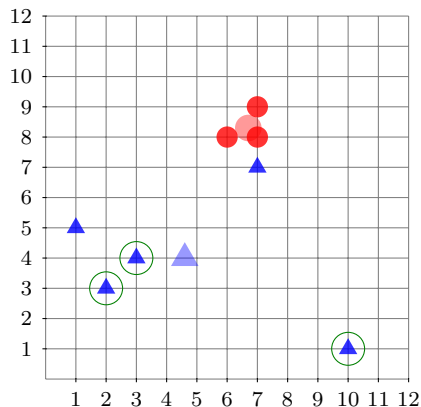
# $k$ -means Clustering – MacQueen Algorithm



assign third point



# $k$ -means Clustering – MacQueen Algorithm

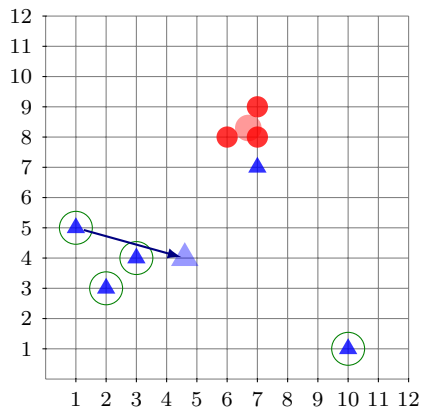


recompute centroids:

$$\mu \approx (4.6, 4.0)$$

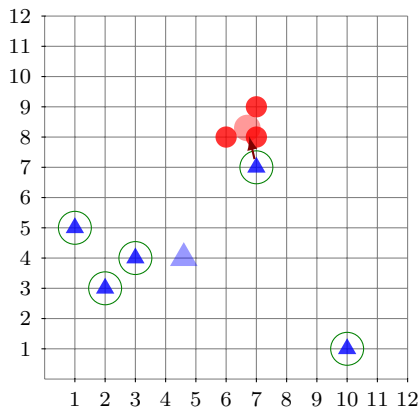
$$\mu \approx (6.7, 8.3)$$

# $k$ -means Clustering – MacQueen Algorithm



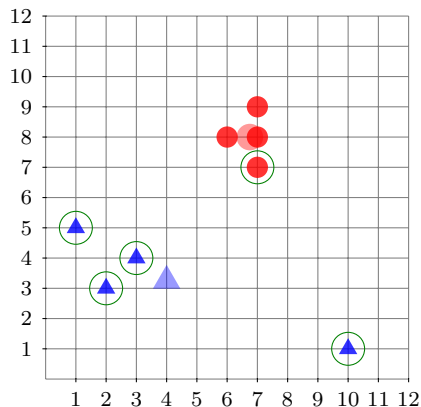
assign fourth point

# $k$ -means Clustering – MacQueen Algorithm



assigning fifth point

# $k$ -means Clustering – MacQueen Algorithm

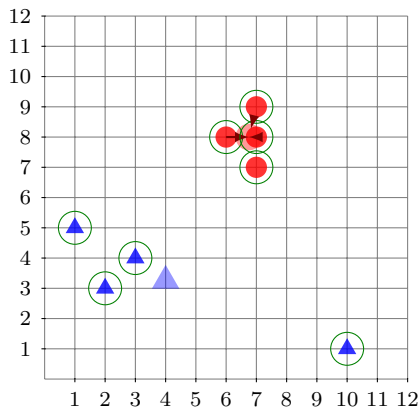


recompute centroids:

$$\mu \approx (4.0, 3.25)$$

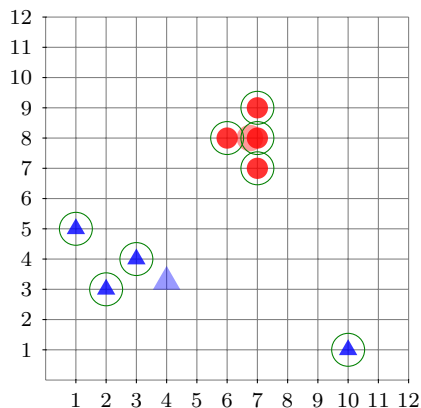
$$\mu \approx (6.75, 8.0)$$

# $k$ -means Clustering – MacQueen Algorithm



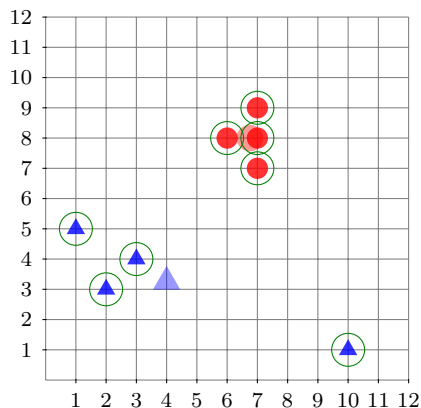
reassign more points

# $k$ -means Clustering – MacQueen Algorithm



reassign more points  
possibly more iterations

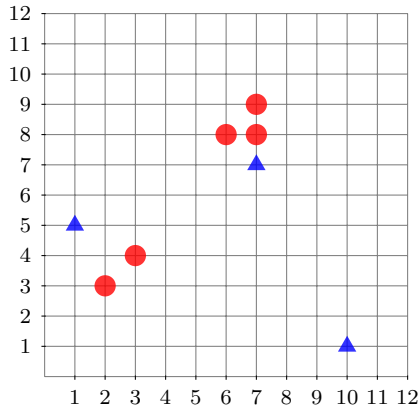
# $k$ -means Clustering – MacQueen Algorithm



reassign more points  
possibly more iterations  
convergence

# $k$ -means Clustering – MacQueen Algorithm

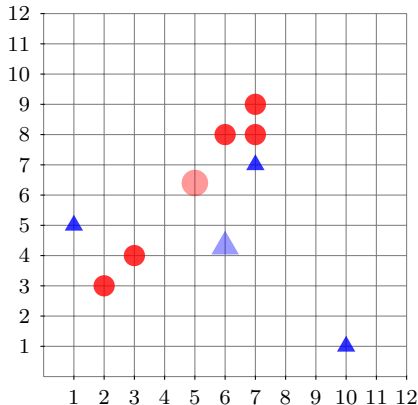
## Alternative Run – Different Order





# $k$ -means Clustering – MacQueen Algorithm

## Alternative Run – Different Order



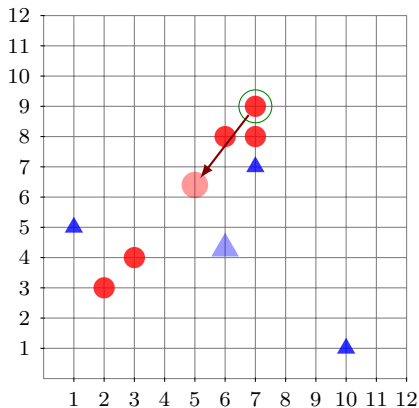
Centroids  
(e.g.: from  
previous iteration):

$$\mu \approx (6.0, 4.3)$$

$$\mu \approx (5.0, 6.4)$$

# $k$ -means Clustering – MacQueen Algorithm

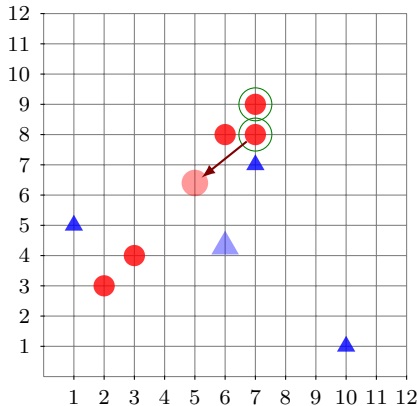
## Alternative Run – Different Order



assign first point

# $k$ -means Clustering – MacQueen Algorithm

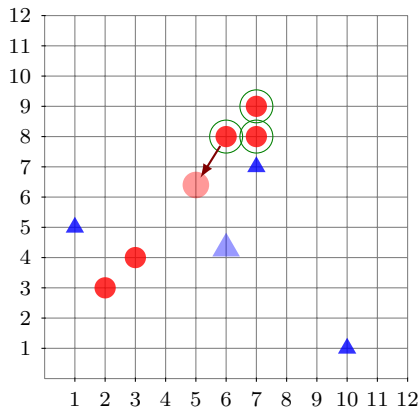
## Alternative Run – Different Order



assign second point

# $k$ -means Clustering – MacQueen Algorithm

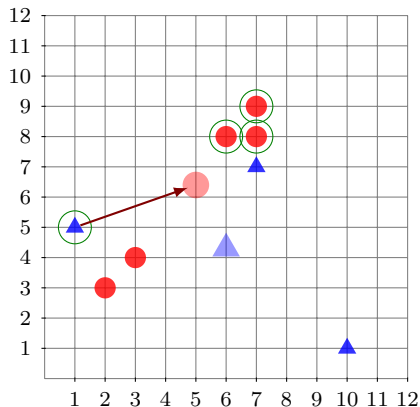
## Alternative Run – Different Order



assign third point

# $k$ -means Clustering – MacQueen Algorithm

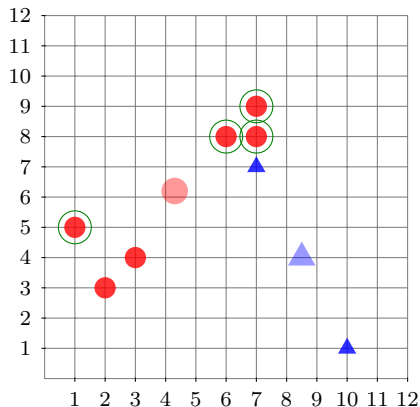
## Alternative Run – Different Order



assign fourth point

# $k$ -means Clustering – MacQueen Algorithm

## Alternative Run – Different Order



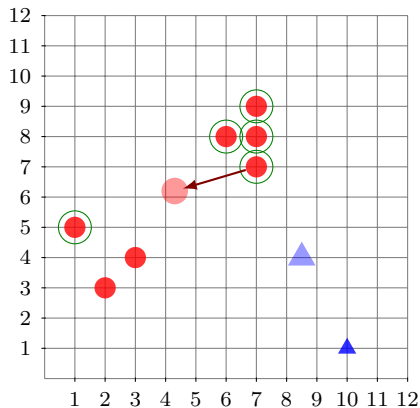
recompute centroids:

$$\mu \approx (4.0, 8.5)$$

$$\mu \approx (4.3, 6.2)$$

# $k$ -means Clustering – MacQueen Algorithm

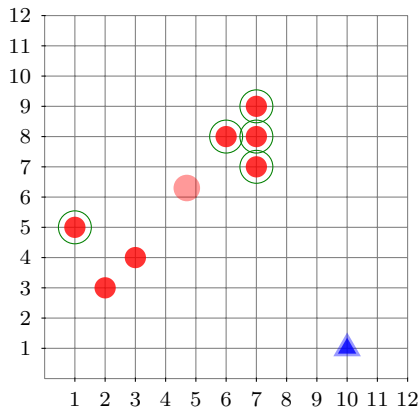
## Alternative Run – Different Order



assign fifth point

# $k$ -means Clustering – MacQueen Algorithm

## Alternative Run – Different Order



recompute centroids:

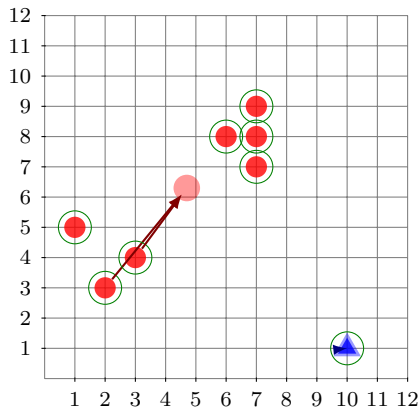
$$\mu \approx (10.0, 1.0)$$

$$\mu \approx (4.7, 6.3)$$



# $k$ -means Clustering – MacQueen Algorithm

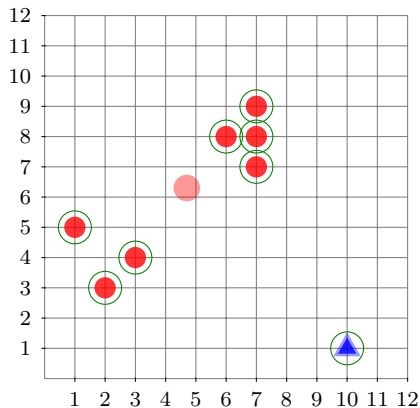
## Alternative Run – Different Order



reassign more points

# $k$ -means Clustering – MacQueen Algorithm

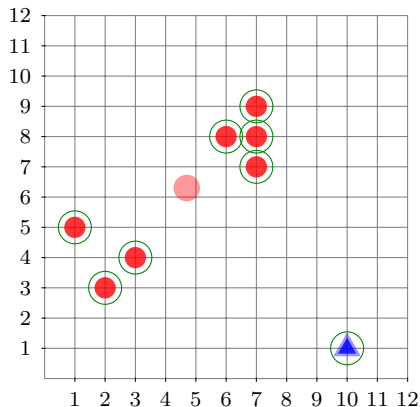
## Alternative Run – Different Order



reassign more points  
possibly more iterations

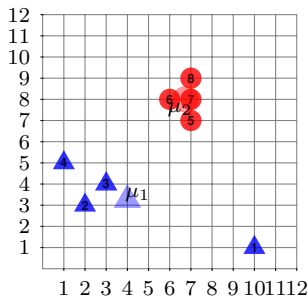
# $k$ -means Clustering – MacQueen Algorithm

## Alternative Run – Different Order



reassign more points  
possibly more iterations  
convergence

# k-means Clustering – Quality



First solution:  $TD^2 = 61\frac{1}{2}$

$$SSQ(\mu_1, p_1) = |4 - 10|^2 + |3.25 - 1|^2 = 36 + 5\frac{1}{16} = 41\frac{1}{16}$$

$$SSQ(\mu_1, p_2) = |4 - 2|^2 + |3.25 - 3|^2 = 4 + \frac{1}{16} = 4\frac{1}{16}$$

$$SSQ(\mu_1, p_3) = |4 - 3|^2 + |3.25 - 4|^2 = 1 + \frac{9}{16} = 1\frac{9}{16}$$

$$SSQ(\mu_1, p_4) = |4 - 1|^2 + |3.25 - 5|^2 = 9 + 3\frac{1}{16} = 12\frac{1}{16}$$

$$TD^2(C_1) = 58\frac{3}{4}$$

$$SSQ(\mu_2, p_5) = |6.75 - 7|^2 + |8 - 7|^2 = \frac{1}{16} + 1 = 1\frac{1}{16}$$

$$SSQ(\mu_2, p_6) = |6.75 - 6|^2 + |8 - 8|^2 = \frac{9}{16} + 0 = \frac{9}{16}$$

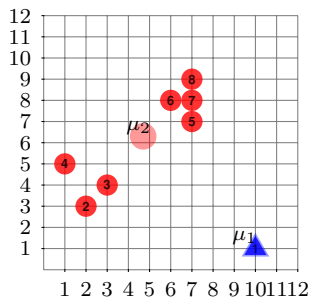
$$SSQ(\mu_2, p_7) = |6.75 - 7|^2 + |8 - 8|^2 = \frac{1}{16} + 0 = \frac{1}{16}$$

$$SSQ(\mu_2, p_8) = |6.75 - 7|^2 + |8 - 9|^2 = \frac{1}{16} + 1 = 1\frac{1}{16}$$

$$TD^2(C_2) = 2\frac{3}{4}$$

Note:  $SSQ(\mu, p) = \|\mu - p\|_2^2$ .

# k-means Clustering – Quality



$$SSQ(\mu_1, p_1) = |10 - 10|^2 + |1 - 1|^2 = 0$$
$$TD^2(C_1) = 0$$

$$SSQ(\mu_2, p_2) \approx |4.7 - 2|^2 + |6.3 - 3|^2 \approx 18.2$$

$$SSQ(\mu_2, p_3) \approx |4.7 - 3|^2 + |6.3 - 4|^2 \approx 8.2$$

$$SSQ(\mu_2, p_4) \approx |4.7 - 1|^2 + |6.3 - 5|^2 \approx 15.4$$

$$SSQ(\mu_2, p_5) \approx |4.7 - 7|^2 + |6.3 - 7|^2 \approx 5.7$$

$$SSQ(\mu_2, p_6) \approx |4.7 - 6|^2 + |6.3 - 8|^2 \approx 4.6$$

$$SSQ(\mu_2, p_7) \approx |4.7 - 7|^2 + |6.3 - 8|^2 \approx 8.2$$

$$SSQ(\mu_2, p_8) \approx |4.7 - 7|^2 + |6.3 - 9|^2 \approx 12.6$$

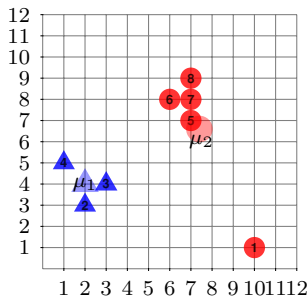
$$TD^2(C_2) \approx 72.86$$

First solution:  $TD^2 = 61\frac{1}{2}$

Second solution:  $TD^2 \approx 72.68$

Note:  $SSQ(\mu, p) = \|\mu - p\|_2^2$ .

# k-means Clustering – Quality



First solution:  $TD^2 = 61\frac{1}{2}$

Second solution:  $TD^2 \approx 72.68$

Optimal solution:  $TD^2 = 54\frac{2}{5}$

Note:  $SSQ(\mu, p) = \|\mu - p\|_2^2$ .

$$SSQ(\mu_1, p_2) = |2 - 2|^2 + |4 - 3|^2 = 0 + 1 = 1$$

$$SSQ(\mu_1, p_3) = |2 - 3|^2 + |4 - 4|^2 = 1 + 0 = 1$$

$$SSQ(\mu_1, p_4) = |2 - 1|^2 + |4 - 5|^2 = 1 + 1 = 2$$

$$TD^2(C_1) = 4$$

$$SSQ(\mu_2, p_1) = |7.4 - 10|^2 + |6.6 - 1|^2 = 6\frac{19}{25} + 31\frac{9}{25} = 38\frac{3}{25}$$

$$SSQ(\mu_2, p_5) = |7.4 - 7|^2 + |6.6 - 7|^2 = \frac{4}{25} + \frac{4}{25} = \frac{8}{25}$$

$$SSQ(\mu_2, p_6) = |7.4 - 6|^2 + |6.6 - 8|^2 = 1\frac{24}{25} + 1\frac{24}{25} = 3\frac{23}{25}$$

$$SSQ(\mu_2, p_7) = |7.4 - 7|^2 + |6.6 - 8|^2 = \frac{4}{25} + 1\frac{24}{25} = 2\frac{3}{25}$$

$$SSQ(\mu_2, p_8) = |7.4 - 7|^2 + |6.6 - 9|^2 = \frac{4}{25} + 5\frac{19}{25} = 5\frac{23}{25}$$

$$TD^2(C_2) = 50\frac{2}{5}$$

# Discussion

## pros

- efficient:  $\mathcal{O}(k \cdot n)$  per iteration, number of iterations is usually in the order of 10.
- easy to implement, thus very popular

## cons

- $k$ -means converges towards a **local** minimum
- $k$ -means (MacQueen-variant) is order-dependent
- deteriorates with noise and outliers (all points are used to compute centroids)
- clusters need to be convex and of (more or less) equal extension
- number  $k$  of clusters is hard to determine
- strong dependency on initial partition (in result quality as well as runtime)