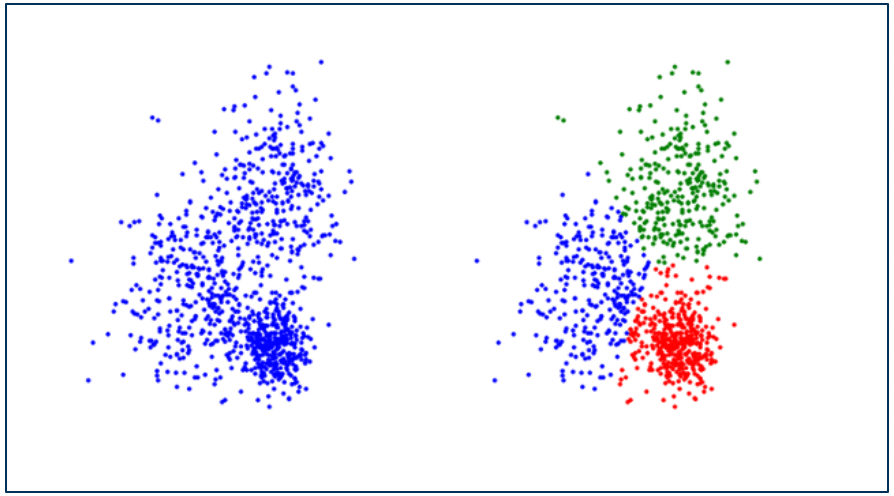
ECE 4252/8803: Fundamentals of Machine Learning (FunML)
Fall 2024

Lecture 11: Clustering



OLIVES @GeorgiaTech

Georgia Tech

Midterms

Oct 2, Wednesday

- Lectures 1-12
- Will be on Canvas
- **100 points**
- 75 mins (includes time to upload)
- Majority of the questions are conceptual
  - True/False questions
  - MCQs
  - Analyzing visualizations
- Last question is to upload your scratch notes
  - Scratch notes only used for regrade requests and is not always necessary
- Tips for studying
  - Go through slides. Exam is open notes – it is essential you are aware of which concepts are in which lectures/slides
  - On every slide, ask yourselves how it can be turned into a T/F or MCQ or visualization question
  - Go through scribe notes
  - **Don't overthink**

Midterms

Oct 2, Wednesday

- You have **75 minutes** (students with accommodations will have the appropriate time extension automatically added to their time) to complete this exam. The countdown starts once you begin and may not reset whether you re-login or disconnect from the Internet. It is your responsibility to observe the time.
- **The exam will open 10/2/24 at 8 AM ET and will be available until 9.30 AM ET. To ensure that you get the entire 75 minutes, please start the exam before 8.15 AM ET.**
- Open notes, calculator/MATLAB/Python are permitted. Internet search engines are not recommended. No communications with any other human or a machine with any level of intelligence allowed except questions to the instructors for clarification on Piazza.
- You may post questions to Piazza during the exam availability time. You MAY NOT post the question itself, solutions, or information related to how problems should be solved.
- **When you post a question on Piazza, make sure you do not reveal any details about the questions themselves. Please remember that other students may start the same quiz, but at a later time. If you need to share parts of the question, send a private message to the instructors via piazza.**
- **All numerical solutions are integers. For non-integer solutions, please round-off to two decimal points.**
- By clicking on the Submit button you affirm that you have neither given nor received assistance during the exam window.
- The last question asks you to upload your worksheets. This is optional and will only be used during regrade requests. If you are not uploading anything on canvas, your browser may prompt you saying one of the questions has not been answered. Please note that this is fine and there are no explicit points for uploading files.

Overview

In the Previous Lecture..

Introduction

- What is clustering?
- Clustering application areas
- Simple example
- Definitions
- What we need for clustering
- Formalization of the clustering problem
- Proximity measures

Common Clustering Methods

Evaluating Clustering Performance

Image Segmentation as Clustering Problem

Proximity Measures

Mahalanobis Distance

- A **distance** measure between two random variables $\boldsymbol{x}_j, \boldsymbol{x}_k$ of the **same distribution**, defined as:

$$d(\boldsymbol{x}_j, \boldsymbol{x}_k) = (\boldsymbol{x}_j - \boldsymbol{x}_k)^T \sum^{-1} (\boldsymbol{x}_j - \boldsymbol{x}_k)$$

  where $\sum$ is the covariance matrix of the **entire dataset** $\boldsymbol{X}$.

- Useful for **detecting outliers**
- When $\sum$ is an identity matrix, the Mahalanobis distance is the same as the Euclidian distance.

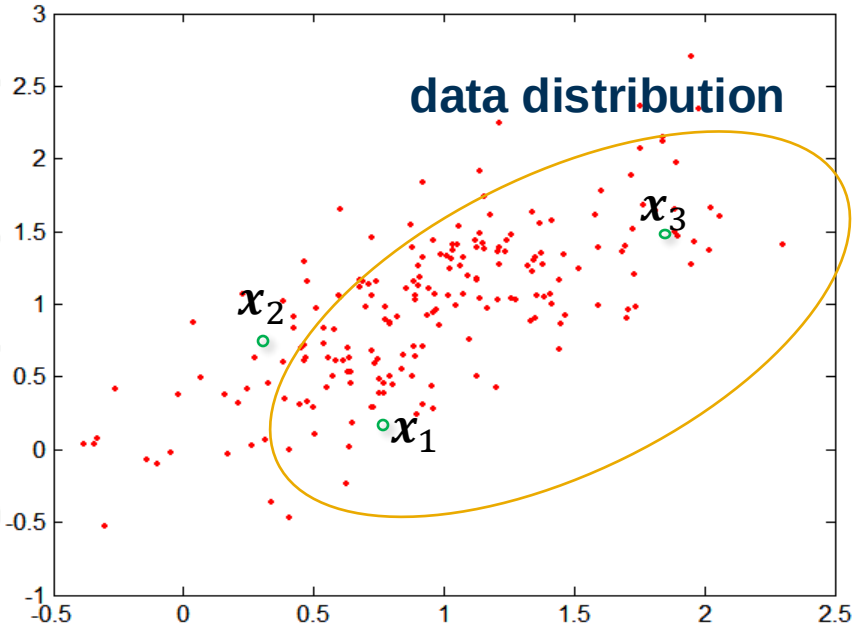
Proximity Measures

Mahalanobis Distance

- **Variance** measures the variation of a single random variable (like the grade of a student in a university)
  - $\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_{i1} - \overline{x_1})^2$, where *N* is the number of samples and $\bar{x}$ is the mean of the random variable
- **Covariance** is a measure of how much two random variables vary together (like the grade of a student and the major of a student in a university)
  - $\sigma(x_1, x_2) = \frac{1}{N-1}\sum_{i=1}^{N}(x_{i1} - \overline{x_1})(x_{i2} - \overline{x_2})$, where *N* is the number of samples, $\bar{x}$ is the mean of the random variable *x_1*, and $\overline{x_2}$ is the mean of the random variable *x_2*
- $\sigma(x_1, x_1) = \sigma_{x_1}^2$
- For a 2X2 random variables, we have a 2X2 covariance matrix:
  - $C = \begin{bmatrix} \sigma(x_1, x_1) & \sigma(x_1, x_2) \\ \sigma(x_2, x_1) & \sigma(x_2, x_2) \end{bmatrix}$
- In general, we can calculate the covariance matrix as follows: $C = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})(X_i - \bar{X})^T$
- If the mean is zero, we can use the semi-definite matrix to calculate $C = \frac{XX^T}{N-1}$

Proximity Measures

Mahalanobis Distance



$$d(x_1, x_2) = (x_1 - x_2)^T \sum^{-1} (x_1 - x_2)$$

$$= [0.5 \quad -0.5] \begin{bmatrix} 6 & -4 \\ -4 & 6 \end{bmatrix} \begin{bmatrix} +0.5 \\ -0.5 \end{bmatrix} = 5$$

$$d(x_1, x_3) = (x_1 - x_3)^T \sum^{-1} (x_1 - x_3)$$

$$= [-1.0 \quad -1.0] \begin{bmatrix} 6 & -4 \\ -4 & 6 \end{bmatrix} \begin{bmatrix} -1.0 \\ -1.0 \end{bmatrix} = 4$$

- Note that although x_2 has a closer direct Euclidean distance to x_1 than x_3, $d(x_1, x_2)$ is a larger Mahalanobis distance since x_2 is not inline with the overall data distribution.

$$x_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, x_3 = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix}$$

Suppose that $\sum$ is given $\sum = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}, \sum^{-1} = \begin{bmatrix} 6 & -4 \\ -4 & 6 \end{bmatrix}$

Note that $\sum$ is computed over the entire dataset, not just the three points

Overview

In this Lecture..

Introduction

Common Clustering Methods

- k-Means Clustering
- Variants of k-Means
- Gaussian Mixture Models
- Mean-shift Clustering

Evaluating Clustering Performance

Image Segmentation as Clustering Problem

Overview

K-Means Clustering

- The k-Means clustering problem
- Convergence criterion
- Standard k-Means Algorithm
- k-Means clustering example
- Choosing the number of clusters k
- Sensitivity to seeds (initial centroids)
- Strength of k-Means
- Weakness of k-Means
- Dealing with outliers

k-Means Clustering

Problem Statement

- Given a set of data samples $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \dots, x_{iP}]^T \in \mathbb{R}^P$, and P is the number of features
- The k-means algorithm partitions the given data into k clusters $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k\}$:
  - The number of clusters, k, is predefined by the user
  - Each sample is assigned to a cluster $\bigcup_{j=1}^{k} \mathcal{C}_j = \boldsymbol{X}$, the samples in $\mathcal{C}_j$ has smaller **Euclidean distance** than samples not in $\mathcal{C}_j$
  - Each cluster $\mathcal{C}_j$ has a cluster center (called centroid), represented by the mean of all the samples in $\mathcal{C}_j$: $m_j = \frac{1}{|\mathcal{C}_j|} \sum x_i \, , x_i \in \mathcal{C}_j$
- k-Means clustering is a **centroid-based** clustering

k-Means Clustering

Standard Algorithm

Predefine k, the k-Means algorithm works as follows:

1. Randomly choose k samples from the data to be k initialized centroids $m_1, \dots, m_k$
2. Assignment step: Assign each data point $\boldsymbol{x_i}$ to the cluster $\mathcal{C}_j$ with the nearest centroids m_j, measured by the **Euclidean distance** such that:

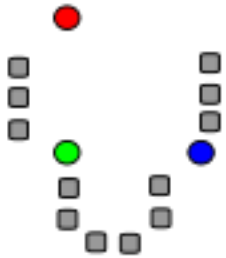
$$\mathcal{C}_j^{(t)} = \left\{ \boldsymbol{x_i} : \left\| \boldsymbol{x_i} - m_j^{(t)} \right\|_2 \leq \left\| \boldsymbol{x_i} - m_l^{(t)} \right\|_2 \; \forall l, 1 \leq l \leq k \right\}$$

3. Update step: Re-compute the centroids (mean) $m_1, \dots, m_k$ of all clusters $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k\}$ with the current cluster memberships:

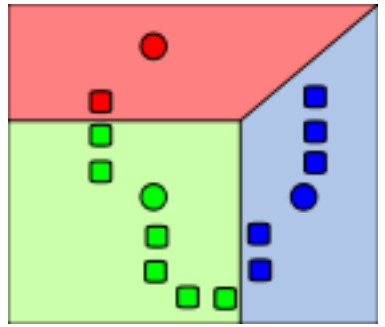
$$m_j = \frac{1}{|\mathcal{C}_j|} \sum x_i \, , x_i \in \mathcal{C}_j \; \forall j, 1 \leq j \leq k$$

4. If the convergence criterion is not met, repeat steps 2 and 3
5. During the inference, the cluster membership is assigned as:

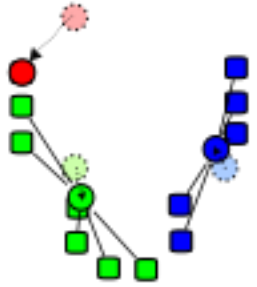
$$\underset{j}{\operatorname{argmin}} \left\| \boldsymbol{x_i} - m_j \right\|_2 \; 1 \leq j \leq k$$



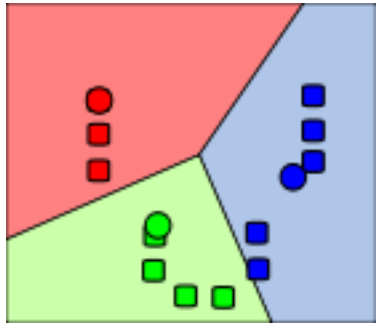
Step1: initialize 3 centroids



Step2: associating samples with the nearest centroids.



Step 3: The mean of the 3 clusters becomes the new centroids.



Step 4: Steps 2 and 3 are repeated until convergence reached.

k-Means Clustering

Convergence Criteria

- Cluster memberships become static:
  No (or minimum) re-assignments of data samples to different clusters, or
- Centroids become static:
  No (or minimum) change of centroids , or
- Change in the cost, sum of squared Euclidean distances (SSD), $(SSD^{(t)} - SSD^{(t+1)})$, measured between iterations, drops to a minimum, where

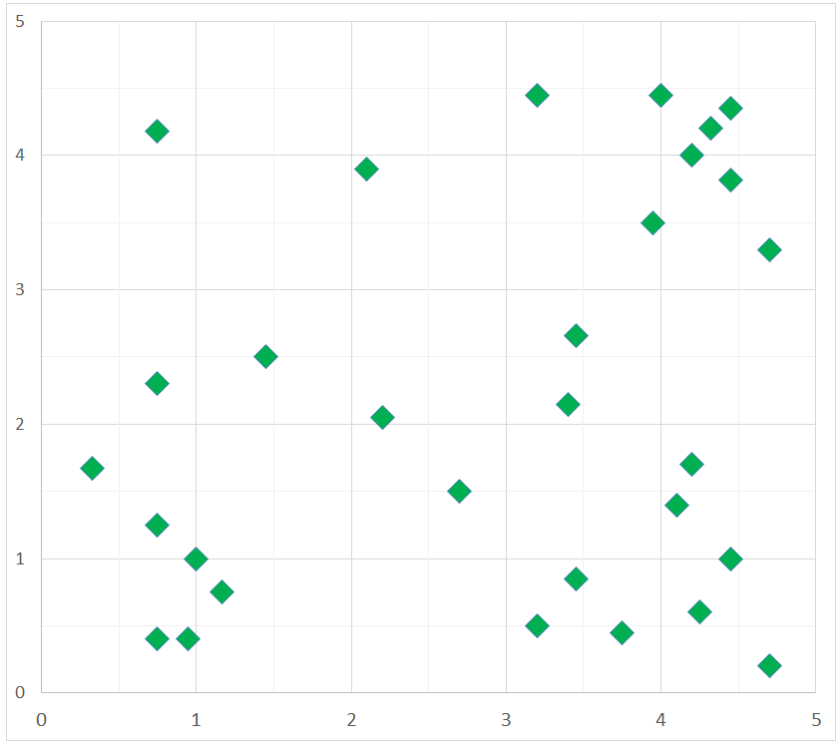
$$\text{Cost: } SSD = \sum_{j=1}^{k} \sum_{\boldsymbol{x_i} \in C_j} d\left(\boldsymbol{x}_i, \boldsymbol{m}_j\right)^2$$

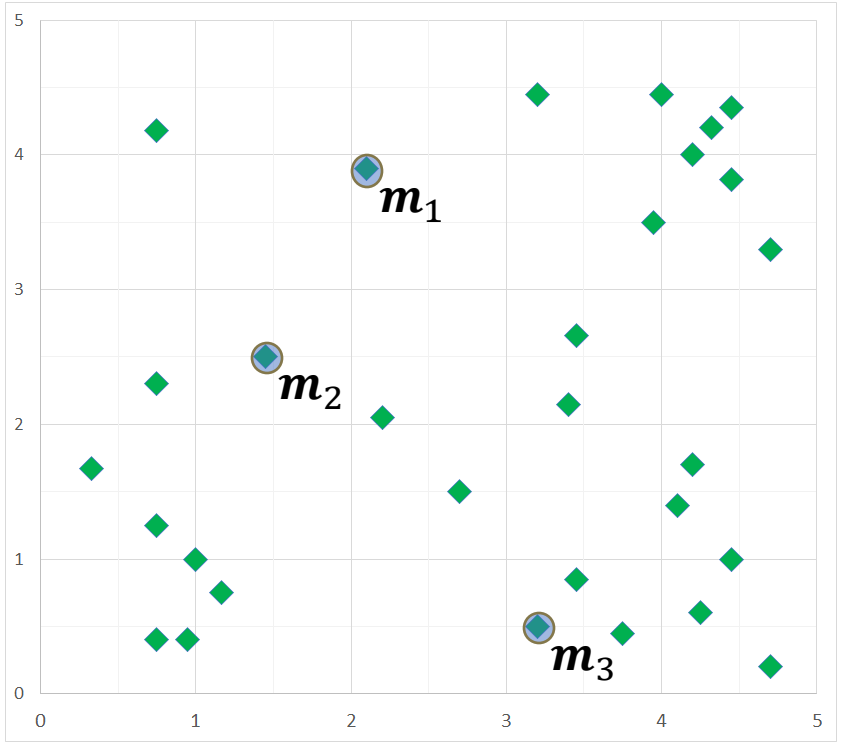
  - C_j is the j^{th}cluster
  - $\boldsymbol{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j),
  - $d\left(\boldsymbol{x}_i, \boldsymbol{m}_j\right)$ is the distance between data sample $\boldsymbol{x_i}$ and centroid $\boldsymbol{m}_j$

k-Means Clustering

Example

Step 1: Predefine k=3, randomly choose m_1, m_2, m_3 from all data points to be initialized centroids

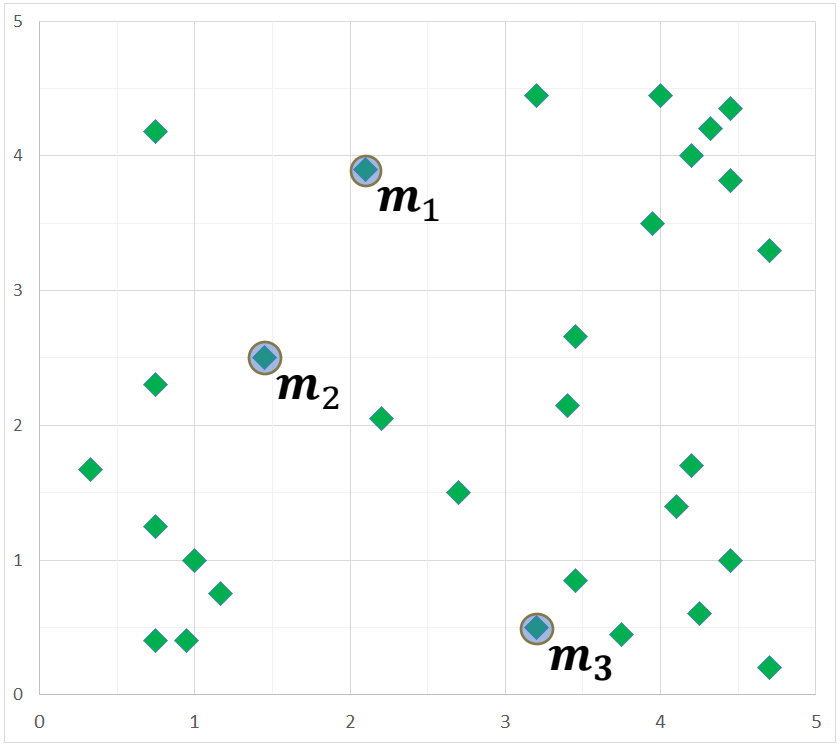


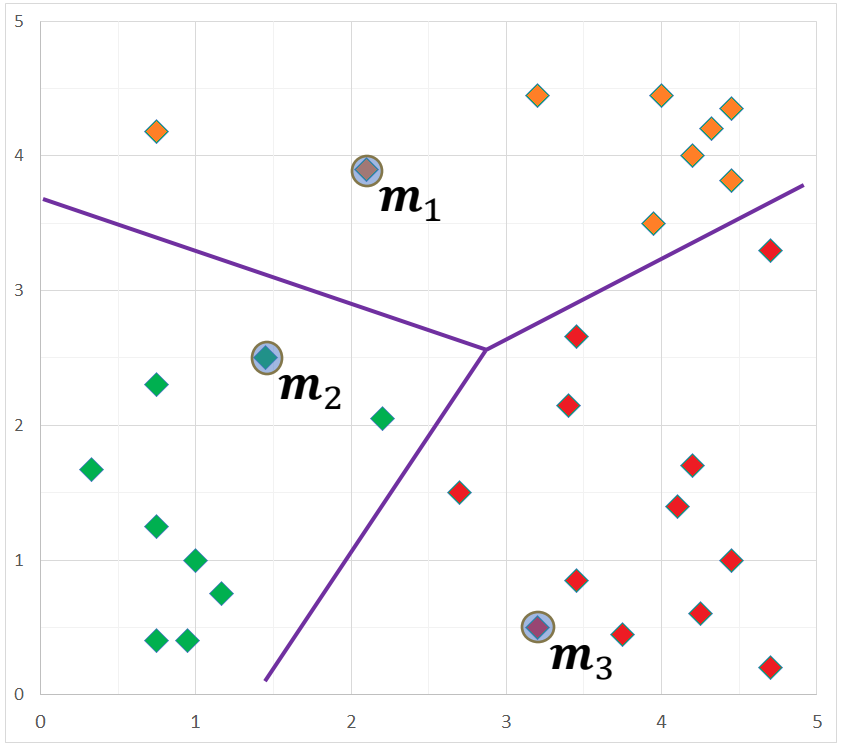


k-Means Clustering

Example

Step 2: Assignment step: determine the cluster membership for each sample by the **Euclidean distance** $C_j^{(t)} = \left\{ \boldsymbol{x_i}: \left\| \boldsymbol{x_i} - m_j^{(t)} \right\|_2 \leq \left\| \boldsymbol{x_i} - m_l^{(t)} \right\|_2 \forall l, 1 \leq l \leq 3 \right\}$



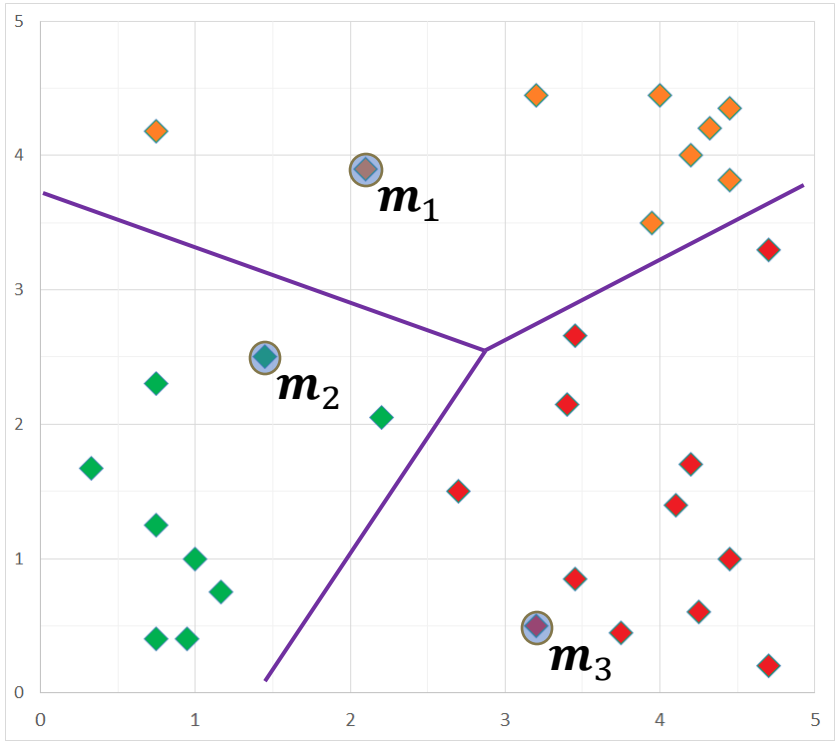


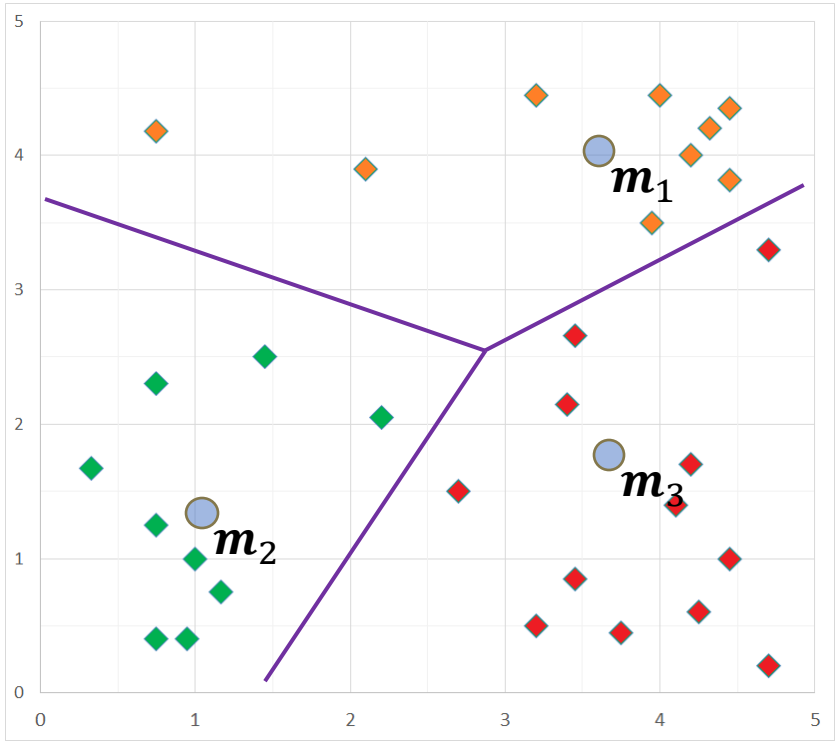
k-Means Clustering

Example

Step 3: Update step: Re-estimate centroids (mean) of the current clusters $m_j = \frac{1}{|\mathcal{C}_j|}\sum x_i \, , x_i \in \mathcal{C}_j \; \forall j, 1 \leq j \leq 3$

centroids move to the mean of the clusters



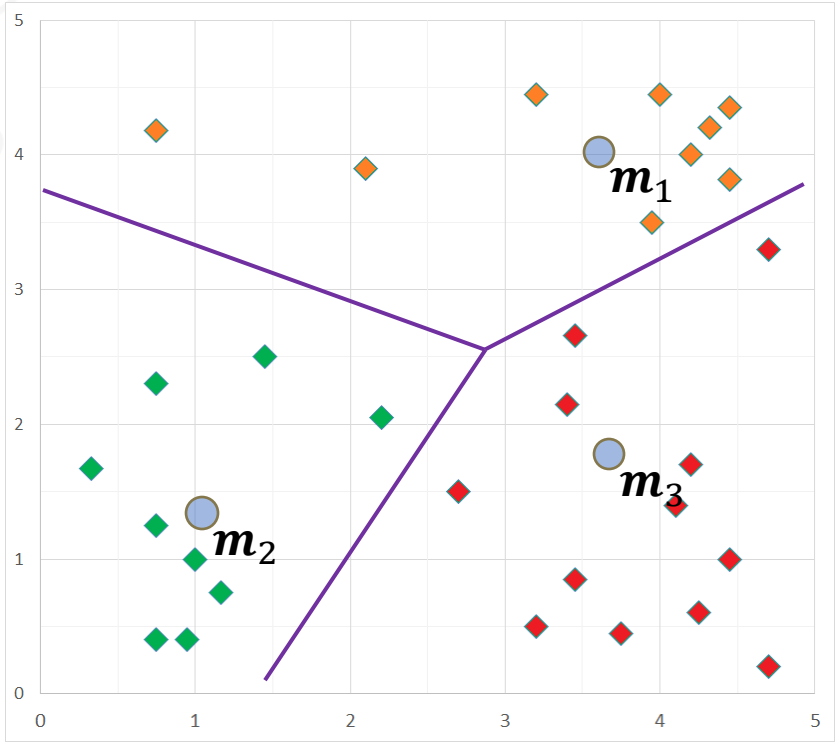


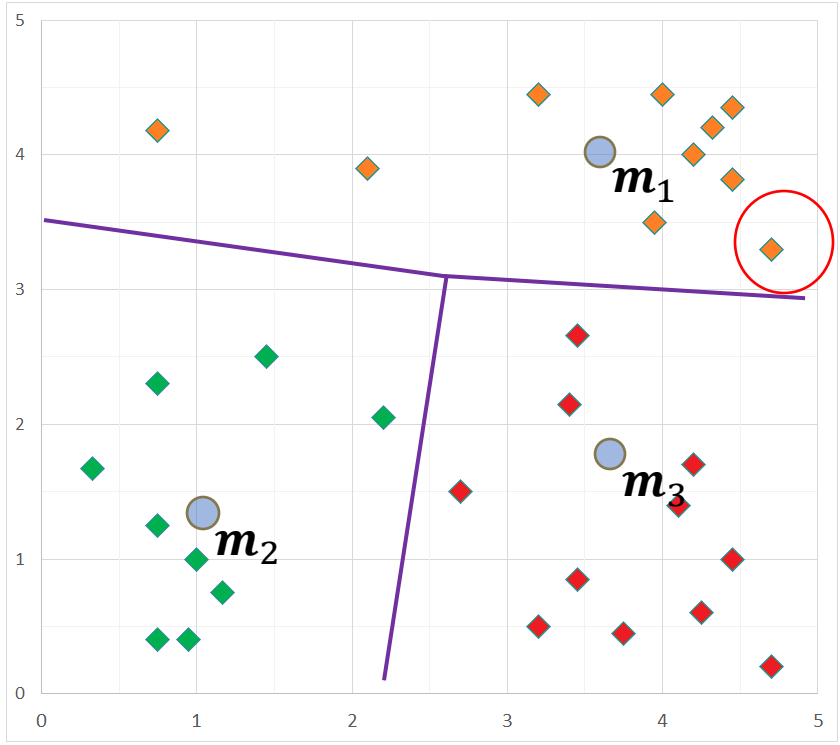
k-Means Clustering

Example

Step 2 (2nd iteration): Assignment step: re-assign cluster membership for each point

convergence not met because of cluster membership update

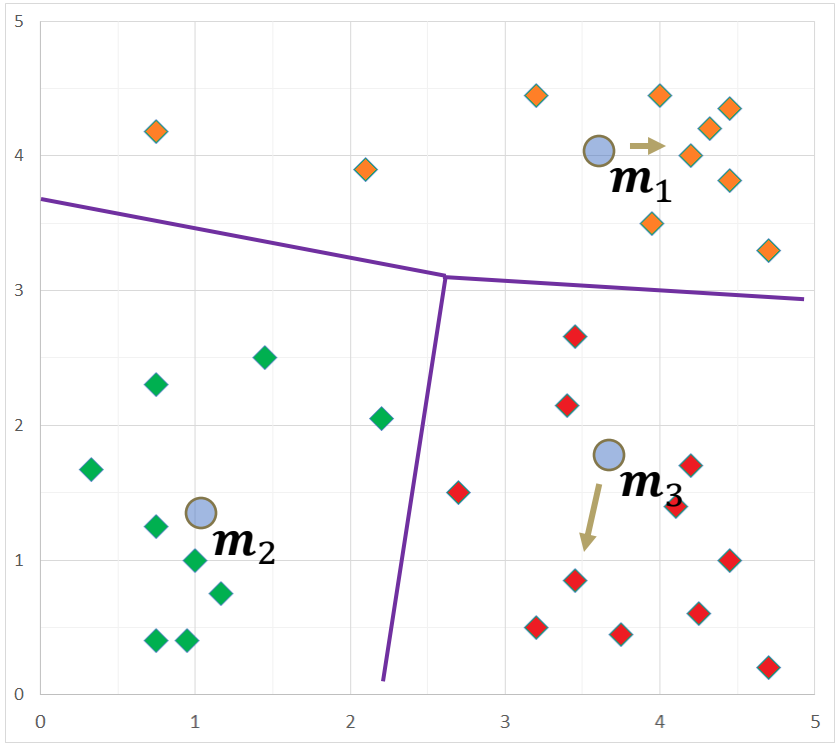


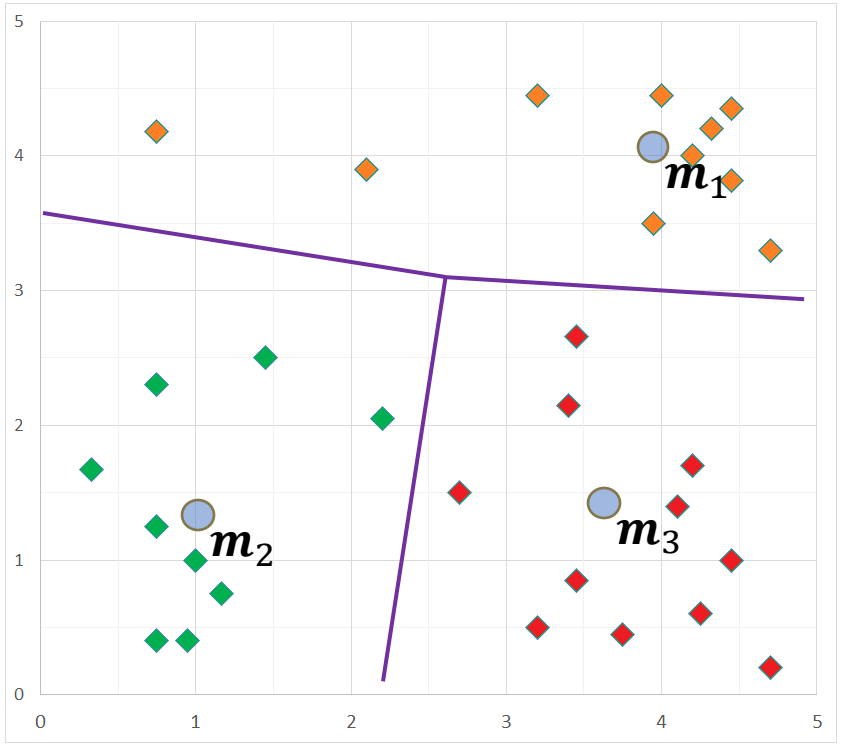


k-Means Clustering

Example

Step 3 (2nd iteration): update step: Re-estimate centroids

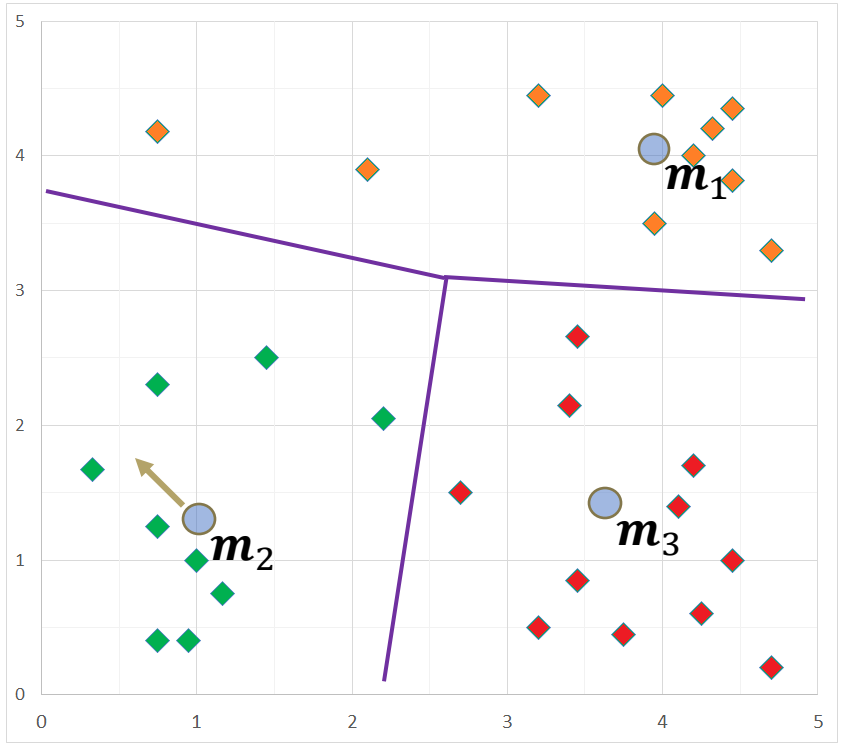


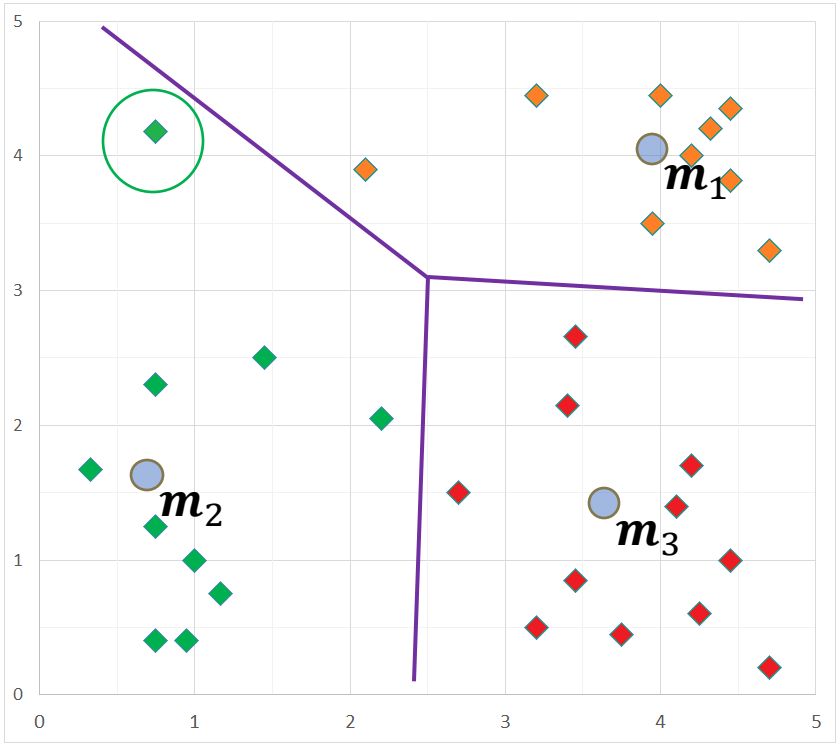


k-Means Clustering

Example

Step 2 (3rd iteration): Assignment step: Re-assign cluster membership for each point

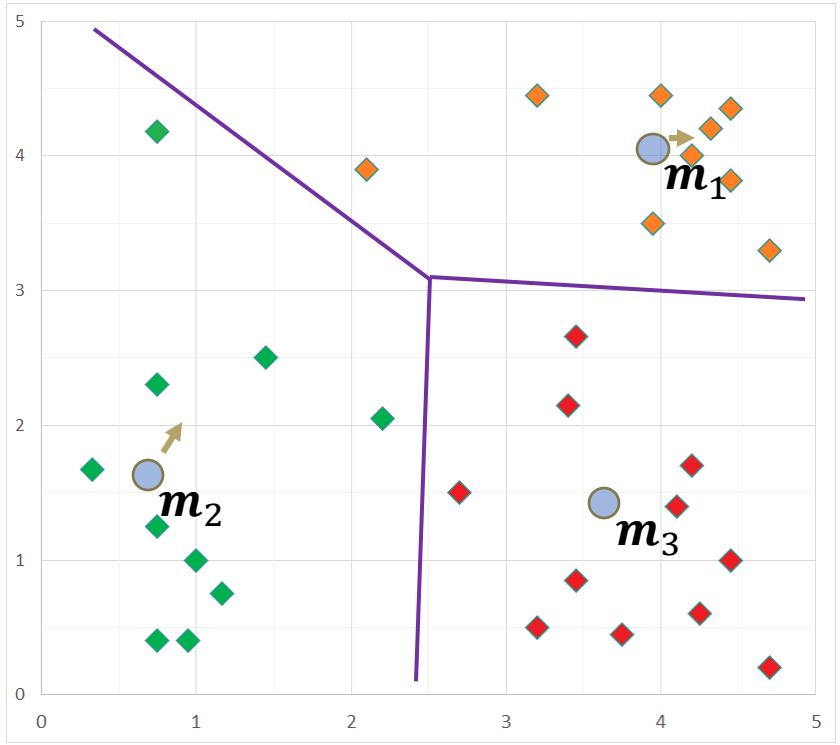


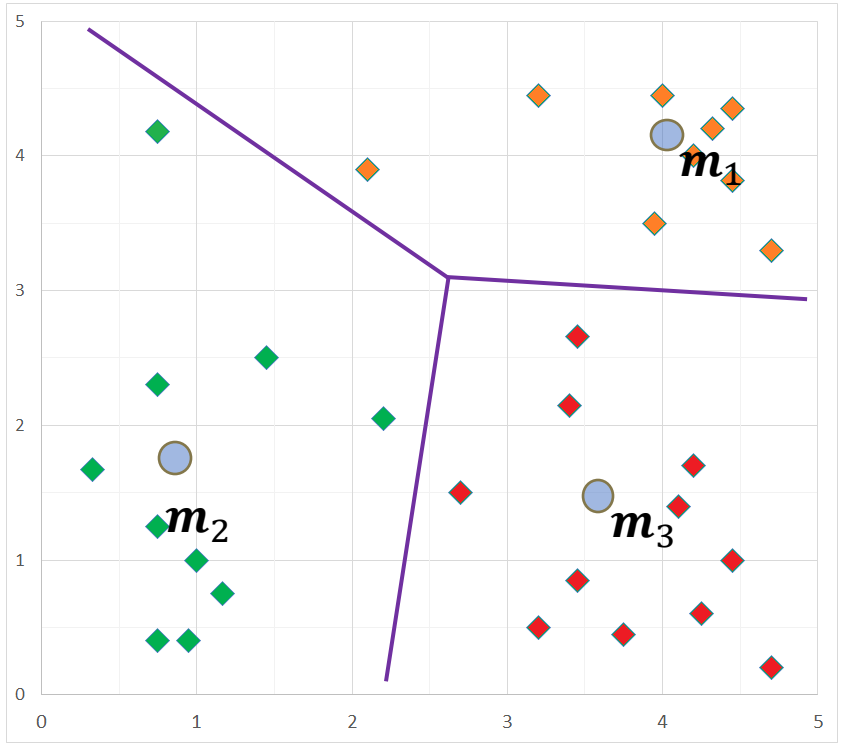


k-Means Clustering

Example

Step 3 (3rd iteration): update step Re-estimate centroids



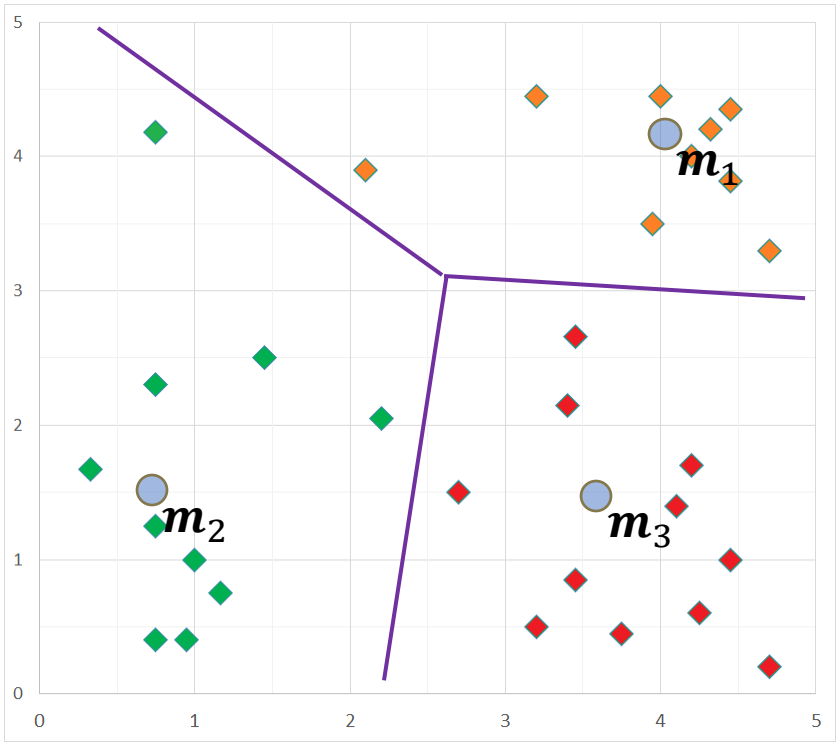


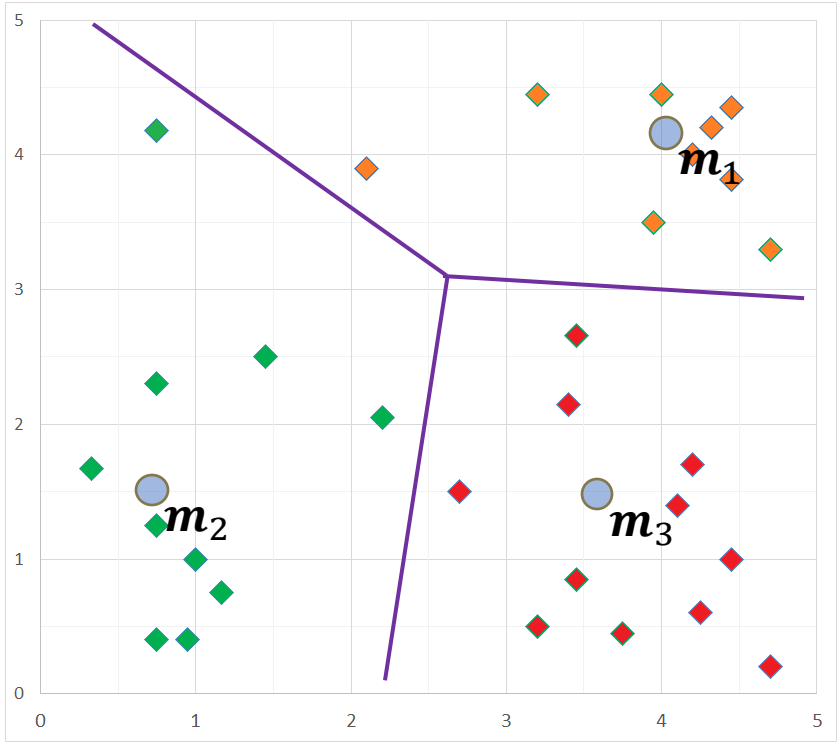
k-Means Clustering

Example

Step 2 (4th iteration): Re-assign cluster membership for each point

Since there is no change in cluster membership, algorithm convergences.



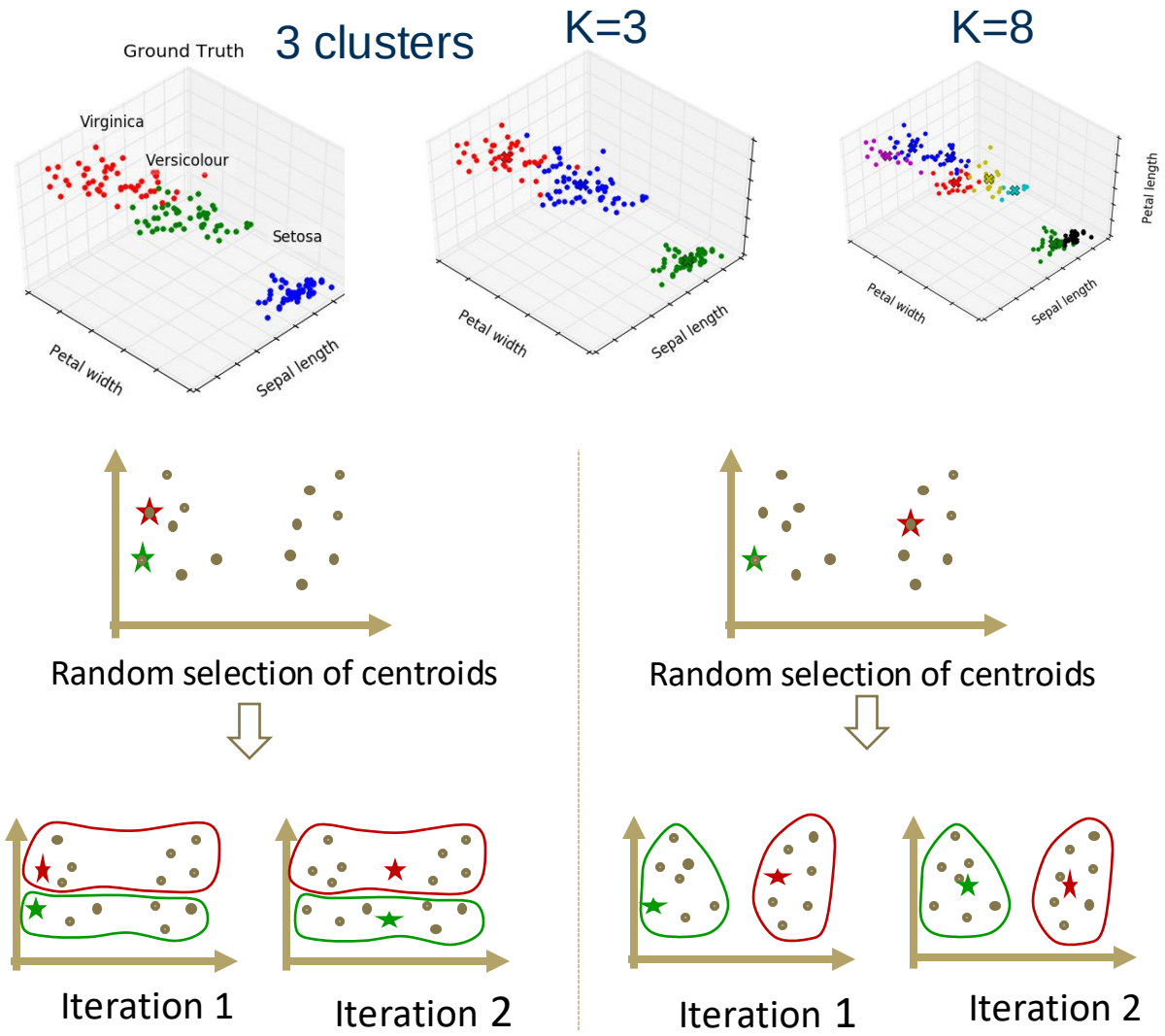


k-Means Clustering

Initialization

Initialization of k-means clustering involves two important aspects:

- Choosing the number of clusters, k
  - Setting k to be too large or too small results in poor clustering
- Methods to initialize k centroids
  - Affects the speed of convergence



k-Means Clustering

Choice of k

Three most common ways of selecting the number of clusters k

1. Utilize **domain knowledge** or any other insight about the data.
   For instance, when we want to cluster cars sold last year, k will best be the number of car manufacturers in the market
2. Run the algorithms several times for **different values of k** and select such k that results in the smallest value of total sum of squared Euclidean distances $\sum_{j=1}^{k} \sum_{\boldsymbol{x_i} \in C_j} d\left(\boldsymbol{x}_i, \boldsymbol{m}_j\right)^2$
3. Perform **cross-validation** and select such k that performs the best on a hold-out validation dataset.

k-Means Clustering

Choice of initial Centroids

- KMPP: *random farthest points, or k-means++.*
  As above with selection made based on distance and probability
- RP: *randomly selected points.*
  Selects the first k random samples selected as initial centers
- RGC: *centroids of random subsamples.*
  Data partitioned randomly by k mutually exclusive groups, the centroids of these groups are appointed as initial centroids
- RUNFP: *farthest points (running selection).*
  Starting with k samples, runs through the rest replacing a centroid with the new sample if it makes the centers farther apart
- SIMFP: *farthest points (simple selection).*
  Starting with the first sample as the first centroid, for next centroids selects farthest sample from the first centroid. Next centroid is the sample farthest for these two, and so on.
- GREP: *group representative points.*
  Starts with global mean as first centroid, selects next centroid as a sample that represents some group of points better than existing centroids.

Others are FYI only

k-Means Initialization

k-Means++

- A method for better **choosing initial centroids** to prevent arbitrarily bad local minima
- Initial cluster centers are chosen at **random** from the set of data points **one by one in sequential order**.
- The probability of choosing data point $\boldsymbol{x}$ is high if $\boldsymbol{x}$ is **NOT near** any **previously chosen centroids**.
    1. Randomly choose first center $\boldsymbol{m}_1$
    2. For each data point $\boldsymbol{x}_i$, find $d(\boldsymbol{x}_i) = \|\boldsymbol{m}_{closest} - \boldsymbol{x}_i\|$, the distance between $\boldsymbol{x}_i$ and the closest center (at the beginning this is only $\boldsymbol{m}_1$)
    3. For each point, calculate $p(\boldsymbol{x}_i) = \frac{d(\boldsymbol{x}_i)^2}{\sum_i d(\boldsymbol{x}_i)^2}$
    4. Choose next centroid from data set at random with **probability proportional** to p
    5. Repeat steps 2 and 3 until k centers have been chosen

Higher probability: NOT near any **previously chosen centroids**

- Proceed using standard k-Means clustering

k-Means Clustering

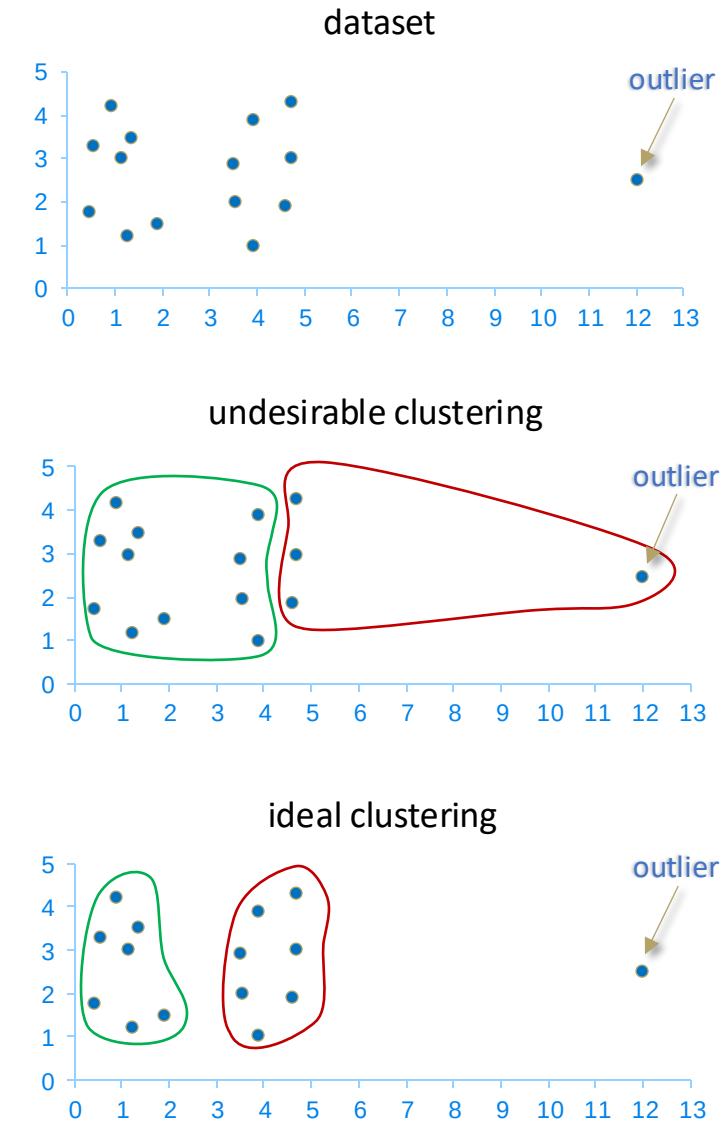
Strengths

- Simple: easy to understand and to implement
- Efficient: time complexity is $O(tkN)$, where
  - N is the number of data samples,
  - k is the number of clusters,
  - t is the number of iterations.
  - Since both k and t are small, k-Means is considered a linear algorithm.
- k-Means is the most popular clustering algorithm using Euclidean distance measure
- k-means algorithm does not guarantee convergence to a global optimum if SSD is used. The global optimum is hard to find due to complexity

k-Means Clustering

Weakness

- The user needs to specify k
- The algorithm is only applicable if the mean of the cluster is defined
  - For categorical data, k-Mode is used – the centroid is represented by most frequent values
- The algorithm is sensitive to outliers
  - Outliers are data samples that are very far away from (dissimilar to) other data points
  - Outliers could be errors in the data recording (noise) or some special data points with very different values



Mini-batch k-Means

k-Medians

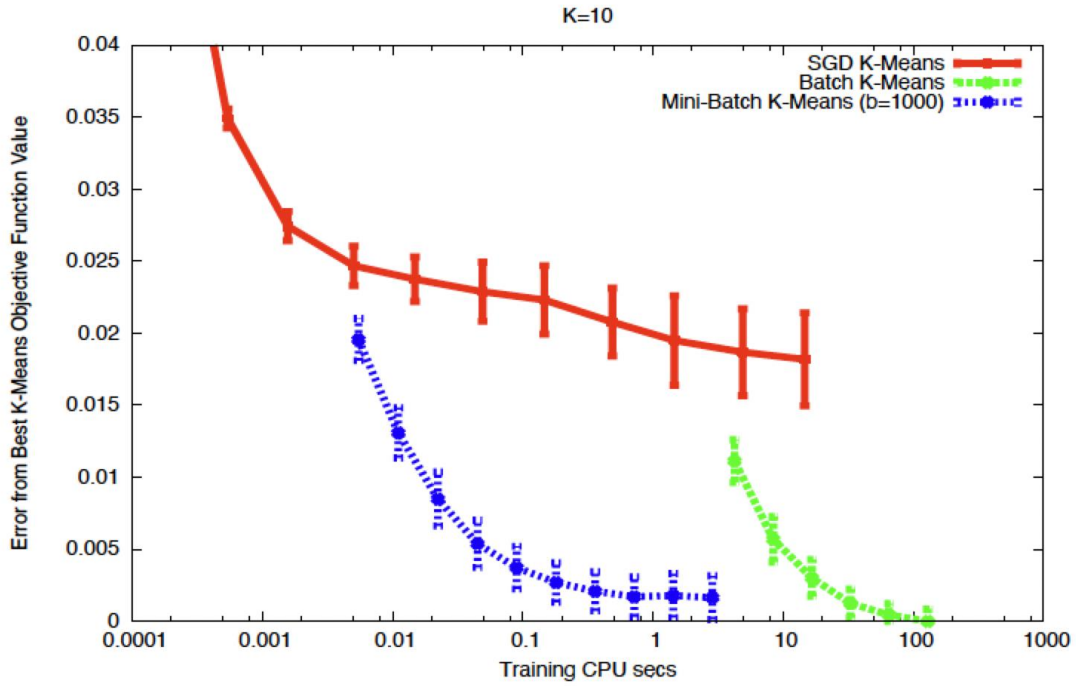
k-Medoid Clustering

Variation of k-Means Clustering

Mini-batch k-Means

- Mini-batches are subsets of the input data, randomly sampled in each training iteration.
- **Reduce the computation time**, while still attempting to optimize the same objective function.
- Proposed as an alternative to k-Means for **clustering massive datasets**.
- Produces results that are generally only s

Convergence Speed. The mini-batch k-means (blue) is orders of magnitude faster than the k-means method (green), while converges to slightly worse solution



https://web.archive.org/web/20150225224202id_/http://www.eecs.tufts.edu:80/~dsculley/papers/fastkmeans.pdf

Variation of k-Means Clustering

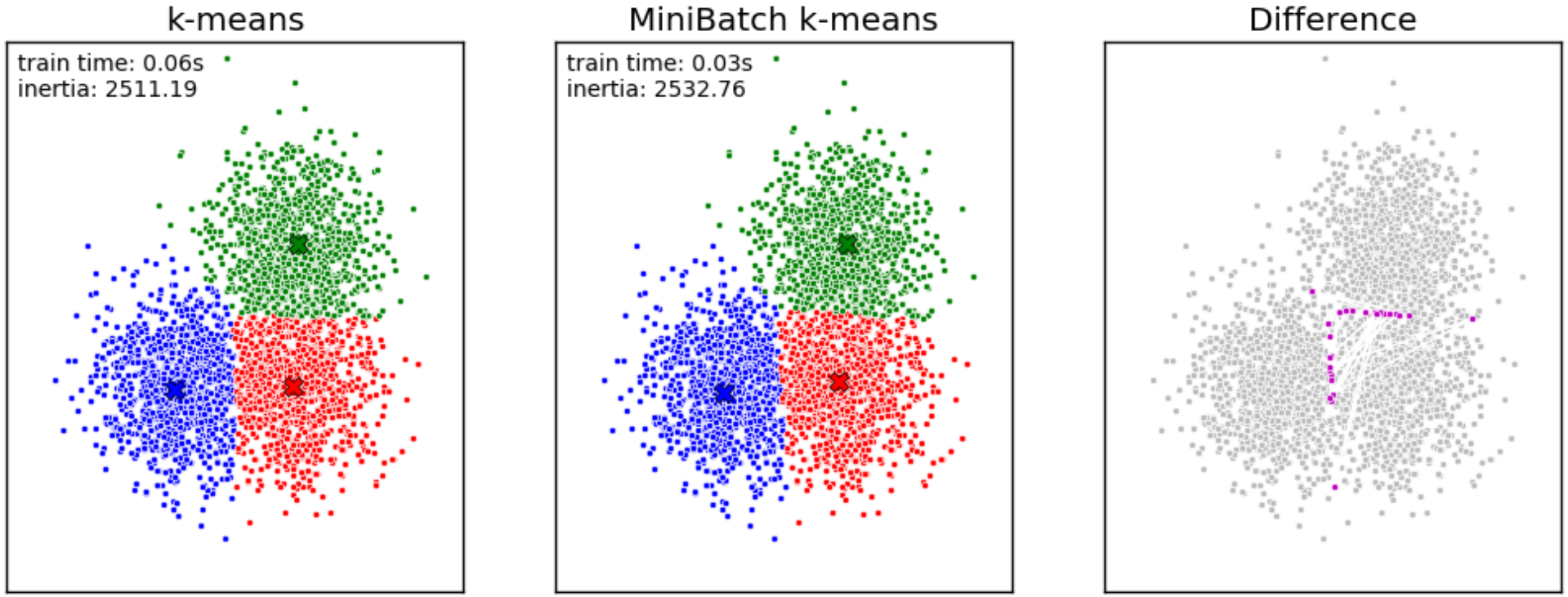
Mini-batch k-Means

- Given a set of samples, the number of clusters k, and a mini-batch size b:
  1. Initialize the k centroids with samples picked from the data set
  2. Randomly draw b samples from the dataset to form a mini-batch.
  3. Assign each sample in the mini-batch to the nearest centroid.
  4. Update centroids on a per-mini batch basis
     - For each sample in the mini-batch, the assigned centroid is updated by taking the streaming average of the sample and all previous samples assigned to that centroid.
  5. Repeat steps 3 and 4 until convergence or for predetermined number of iterations.

Variation of k-Means Clustering

Mini-batch k-Means

- Mini-Batch k-Means is faster but gives slightly different results
- Plot on the right shows a few data points that were clustered differently



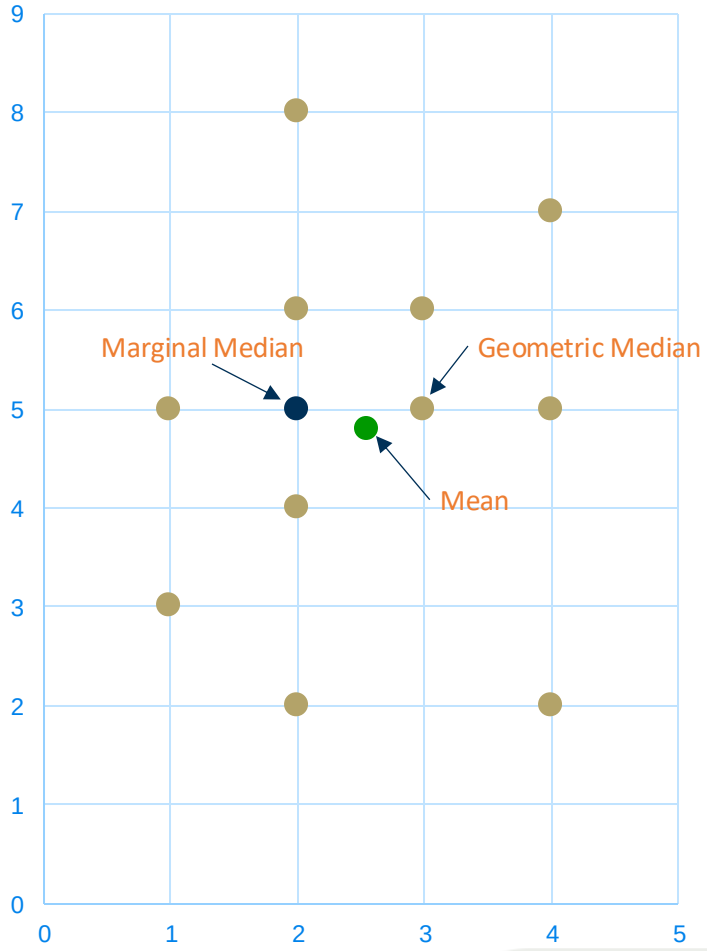
Variation of k-Means Clustering

k-Medians Clustering

- The **Geometric (Spatial) Median** generalizes this notion into higher dimensions. For a given set $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ of N points with each $\boldsymbol{x}_i \in \mathbb{R}^p$, the geometric median is defined as the **sample** $\boldsymbol{m} \in \mathcal{X}$ that satisfies:

$$\arg\min_{\boldsymbol{m} \in \mathcal{X}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{m}\|_2$$

- k-Medians uses geometric median to compute cluster centroids. Median is less vulnerable to outliers compared to mean; thus, k-medians is more robust to outliers than k-means.
- The **Marginal Median** (vector of medians) is a combination of per-dimension scalar medians.



Variation of k-Means Clustering

k-Medians Clustering

- Follows the same steps as k-Means
  - **Initialization**: select k samples from the N data points as initial medians
  - **Assignment Step**: Associate each data point to the closest median using Manhattan distance

$$\mathcal{C}_j^{(t)} = \left\{ \boldsymbol{x_i} : \left\| \boldsymbol{x_i} - m_j^{(t)} \right\|_{\mathbf{1}} \leq \left\| \boldsymbol{x_i} - m_l^{(t)} \right\|_{\mathbf{1}} \forall l, 1 \leq l \leq k \right\}$$

  - **Update Step**: Recompute a new median $\boldsymbol{m}$ for each cluster such that

$$\arg\min_{\boldsymbol{m} \in \mathcal{X}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{m}\|_2$$

  - Repeat while the cost of the configuration decreases
- As a special case, the k-Medoids algorithm is described in further detail next

Variation of k-Means Clustering

k-Mediods Clustering

- A medoid is defined as the **data point** of a cluster whose average dissimilarity to all the data points in the cluster is minimal.
- For a given set $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ of N points with each $\boldsymbol{x}_i \in \mathbb{R}^p$, the medoid is defined as the **sample** $\boldsymbol{m} \in \mathcal{X}$ that satisfies:

$$\arg\min_{\boldsymbol{m} \in \mathcal{X}} \sum_{i=1}^{N} d(\boldsymbol{x}_i, \boldsymbol{m})$$

where d is a distance measure such as L1 or L2 norms.

k-medoids can be used with arbitrary distance measures, which is a generalization of k-medians

Variation of k-Means Clustering

k-Mediods Clustering

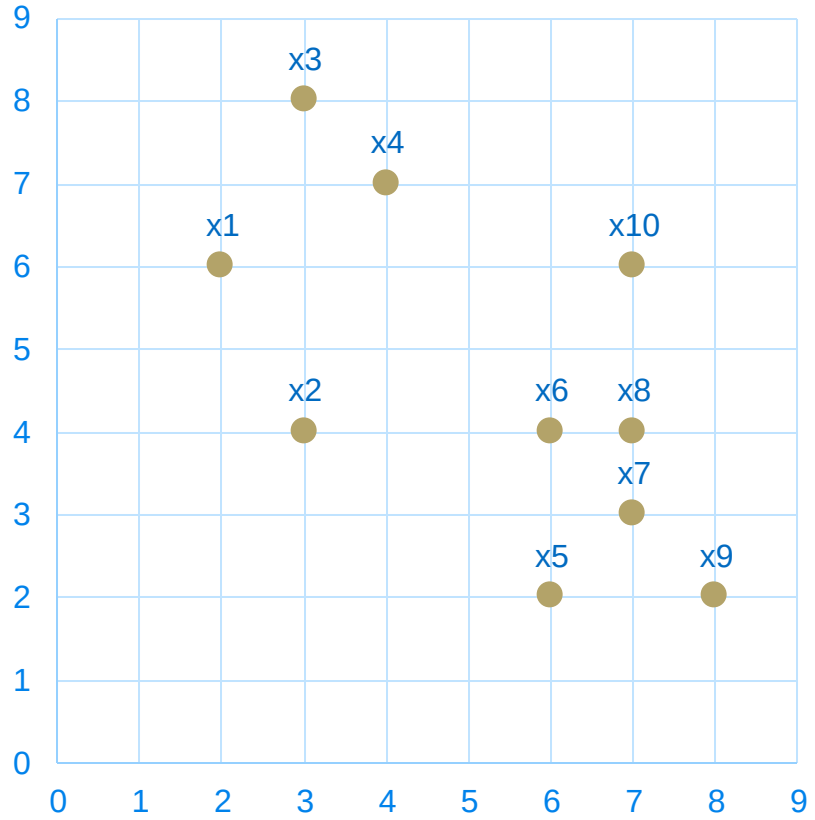
- Initialization: select k of the N data points as the medoids
- Assignment Step: Associate each data point to the closest medoid
- Update Step: while the cost (sum of distances of points to their medoid) decreases
  - For each medoid $\boldsymbol{m}$, examine associated data points for a better medoid
    For each point $\boldsymbol{x}$ in $\boldsymbol{m}$'s cluster:
    1. **Swap $\boldsymbol{m}$ and $\boldsymbol{x}$,**
    2. Re-associate points in the dataset to the closest medoid
    3. Recompute the cost If the total cost of the configuration increases in the previous step, undo swap
  - Repeat for all points in $\boldsymbol{m}$'s cluster
- Repeat while the cost of the configuration decreases

Variation of k-Means Clustering

k-Mediods Example

- Using k-Medoids and the L1-norm distance (Manhattan distance) metric, cluster the following data set of 10 samples into two clusters i.e. $k = 2$

Data object	
i	$\boldsymbol{x}_i$
1	(2, 6)
2	(3, 4)
3	(3, 8)
4	(4, 7)
5	(6, 2)
6	(6, 4)
7	(7, 3)
8	(7, 4)
9	(8, 5)
10	(7, 6)



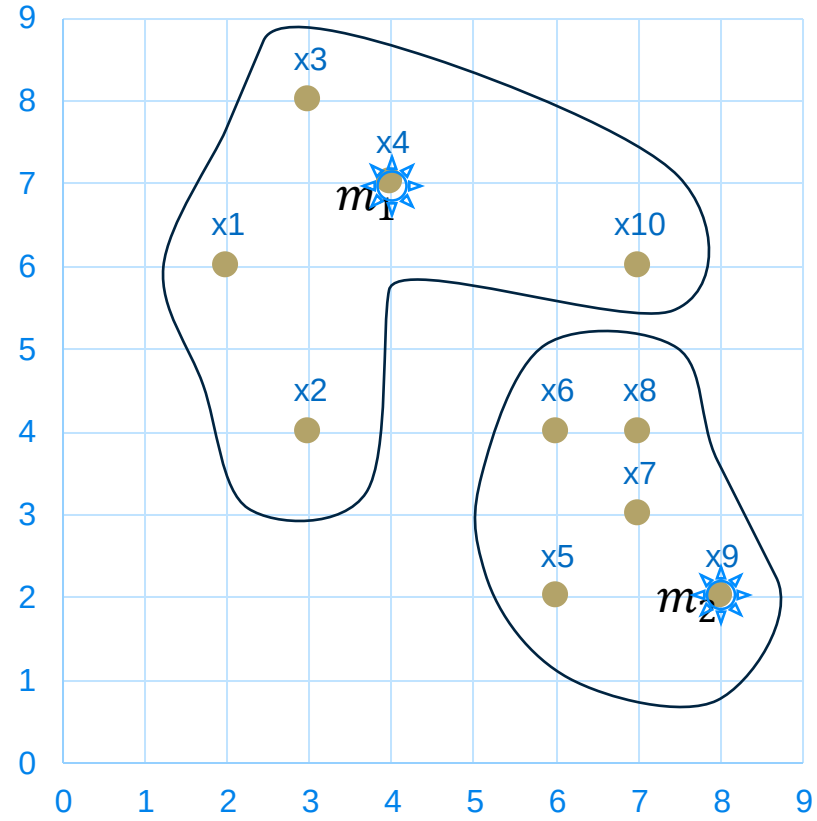
Variation of k-Means Clustering

k-Mediods Example

- **Initialization and Assignment**

  Randomly selected medoids: $m_1 = x_4 = (4,7)$ and $m_2 = x_9 = (8,2)$

Data object		Distance to	
i	x_i	$m_1 = (4,7)$	$m_2 = (8,2)$
1	(2, 6)	**3**	10
2	(3, 4)	**4**	7
3	(3, 8)	**2**	11
4	(4, 7)	0	0
5	(6, 2)	7	**2**
6	(6, 4)	5	**4**
7	(7, 3)	7	**2**
8	(7, 4)	6	**3**
9	(8, 5)	0	0
10	(7, 6)	**4**	5
Cost		**13**	**11**



Total cost of the clustering is $13 + 11 = 24$

Variation of k-Means Clustering

k-Mediods Example

- **Update Step: Iteration 1(a)** - updating medoid m_1

  Check samples associated with m_1 to see if any can provide a better total cost

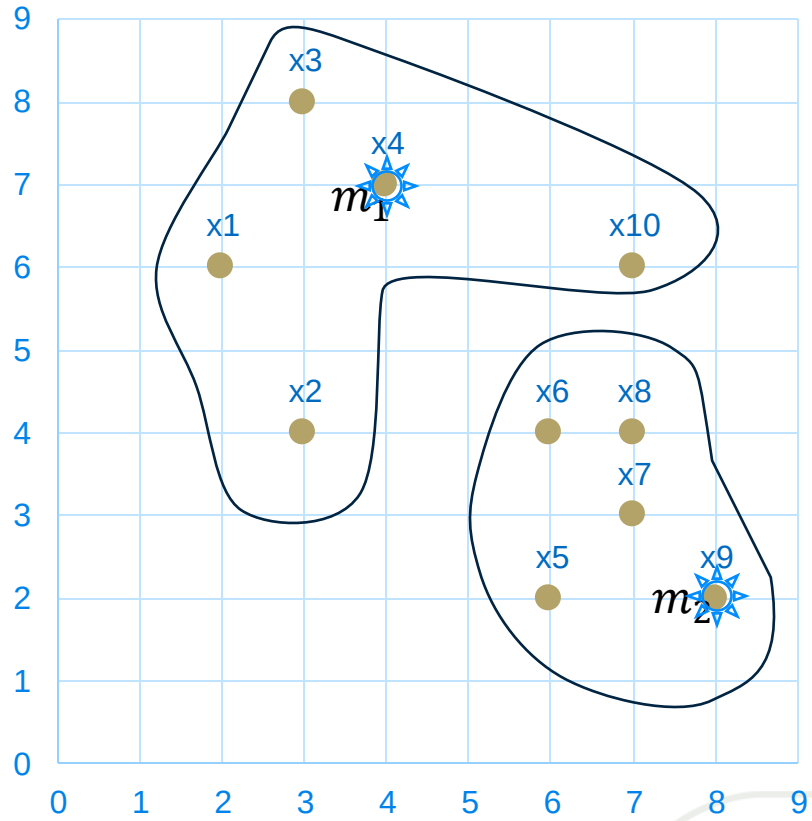
  The table below summarizes iterations of the loop

Data sample in cluster		Cost of configuration	
i	x_i	Distances from each candidate in cluster to other samples	Total for cost of candidate
1	(2, 6)	=0+3+3+3+5	14
2	(3, 4)	=3+0+4+4+8	19
3	(3, 8)	=3+4+0+2+6	15
4	(4,7)	=3+4+2+0+4	13
10	(7, 6)	=5+6+6+4+0	21

current medoid →

Current medoid x_4 has a total cost of 13, and none of the other samples provides a lower cost.

So, x_4 remains the value of medoid m_1



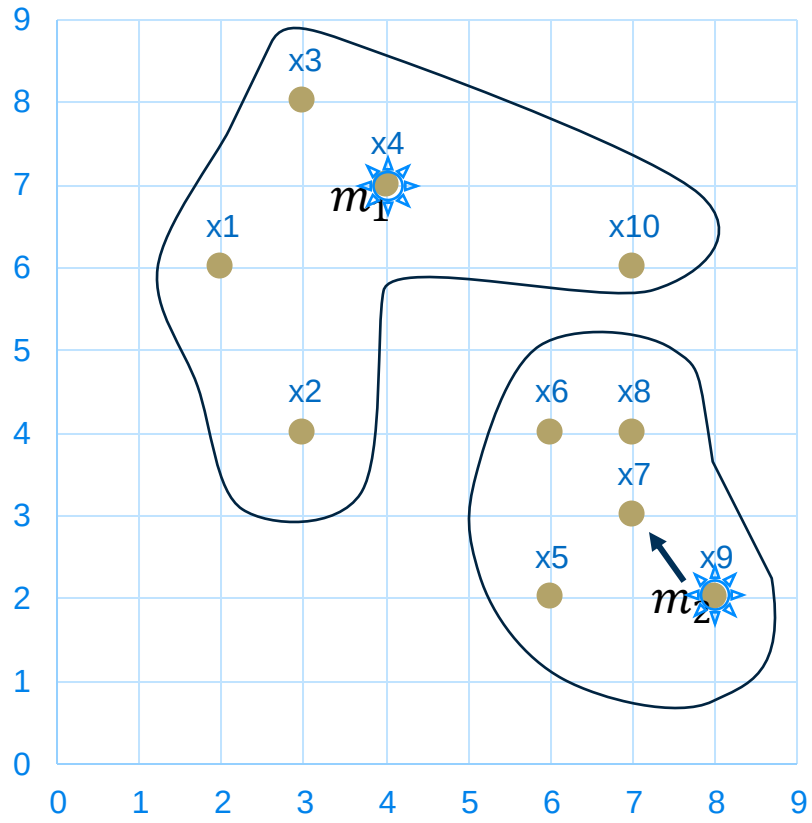
Variation of k-Means Clustering

k-Mediods Example

- **Update Step: Iteration 1(b)** - updating medoid m_2
  Check samples associated with m_2 to see if any can provide a better total cost
  The table below summarizes iterations of loop

Data sample in cluster		Cost of configuration	
i	x_i	Distances from each candidate in cluster to other samples	Total for cost of candidate
5	(6, 2)	=0+2+2+3+2	9
6	(6, 4)	=2+0+1+2+4	9
7	(7, 3)	=2+2+0+1+2	7
8	(7, 4)	=3+1+1+0+3	8
9	(8, 2)	=2+4+2+3+0	11
Minimum		x_7(7,3)	**7**

current medoid →



Sample x_7 provides a total cost of 7 which is lower than the total cost of the current medoid x_9 which is 11.
Therefore, medoid m_2 moves to x_7

Variation of k-Means Clustering

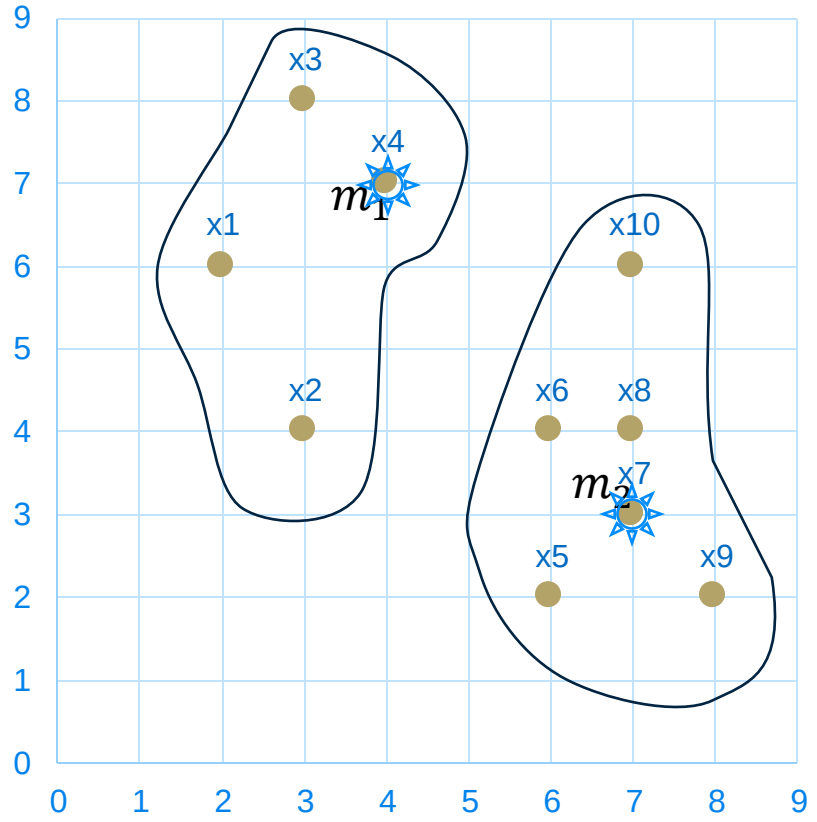
k-Mediods Example

- **Update Step: Reassign entire dataset based on new medoids**

  Recompute distances relative to new medoids: $m_1 = \boldsymbol{x}_4 = (4,7)$ and $m_2 = \boldsymbol{x}_7 = (7,3)$

Data object		Distance to	
i	$\boldsymbol{x}_i$	$\mathrm{m}_1 = (4,7)$	$\mathrm{m}_2 = (7,3)$
1	(2, 6)	**3**	8
2	(3, 4)	**4**	5
3	(3, 8)	**2**	9
4	(4, 7)	0	0
5	(6, 2)	7	**2**
6	(6, 4)	5	**2**
7	(7, 3)	7	0
8	(7, 4)	6	**1**
9	(8, 5)	0	2
10	(7, 6)	4	**3**
Cost		**9**	**8**

Total cost of the clustering is $9 + 8 = 17$



Variation of k-Means Clustering

k-Mediods Example

- **Update Step: Iteration 2(a)** - updating medoid $\boldsymbol{m}_1$

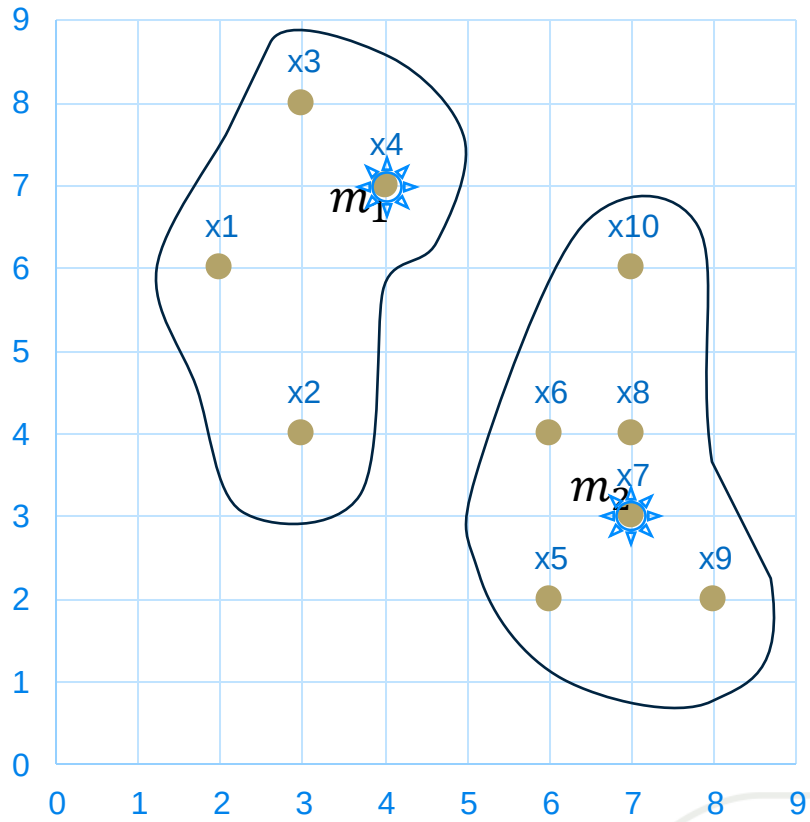
  Check samples associated with $\boldsymbol{m}_1$ to see if any can provide a better total cost

  The table below summarizes iterations of the loop

Data sample in cluster		Cost of configuration	
i	$\boldsymbol{x}_i$	Distances from each candidate in cluster to other samples	Total for cost of candidate
1	(2, 6)	=0+3+3+3	9
2	(3, 4)	=3+0+4+4	11
3 (current medoid →)	(3, 8)	=3+4+0+2	9
4	(4,7)	=3+4+2+0	9

Current medoid $\boldsymbol{x}_4$ has a total cost of 9, and none of the other samples provides a lower cost.

So, $\boldsymbol{x}_4$ remains the value of medoid $\boldsymbol{m}_1$



Variation of k-Means Clustering

k-Mediods Example

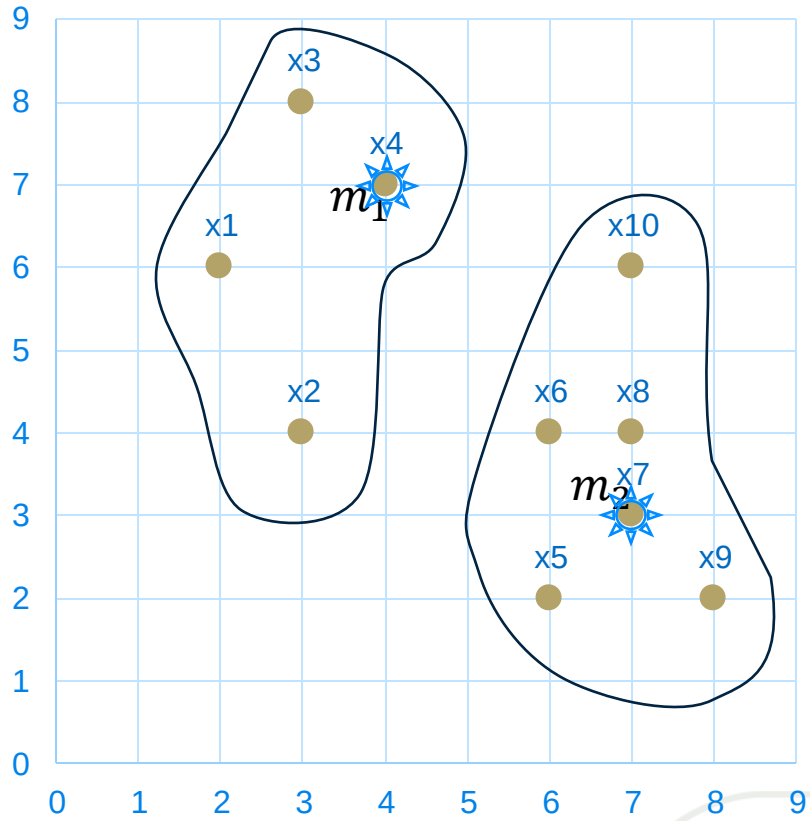
- **Update Step: Iteration 2(b)** - updating medoid m_2

  Check samples associated with m_2 to see if any can provide a better total cost

  The table below summarizes iterations of loop

Data sample in cluster		Cost of configuration	
i	x_i	Distances from candidate medoid to other samples	Total cost of candidate
5	(6, 2)	=0+2+2+3+2+5	14
6	(6, 4)	=2+0+1+2+4+3	12
7	(7, 3)	=2+2+0+1+2+3	10
8	(7, 4)	=3+1+1+0+3+2	10
9	(8, 2)	=2+4+2+3+0+5	16
10	(7,6)	=5+3+3+2+5+0	18

current medoid →



Current medoid x_7 has a total cost of 10, and none of the other samples provides a lower cost.

So, x_7 remains the value of medoid m_2

Variation of k-Means Clustering

Fuzzy C-Means Introduction

- We have seen the k-Means and many variations that belong to hard clustering, i.e., each data point can only be assigned to one cluster.
- Fuzzy C-Means is a method of **fuzzy clustering**. A data point can belong to more than one cluster, i.e. a cluster is a fuzzy set
- A fuzzy set is defined as the pair $(\boldsymbol{X}, \mathcal{M})$ where $\boldsymbol{X}$ is the set of samples and
  $\mathcal{M}: \boldsymbol{X} \rightarrow [0,1]$ is a membership function.
- For each $\boldsymbol{x}_i \in \boldsymbol{X}$ the value $\mathcal{M}(\boldsymbol{x}_i)$ is the degree of membership of $\boldsymbol{x}_i$ in $(\boldsymbol{X}, \mathcal{M})$.

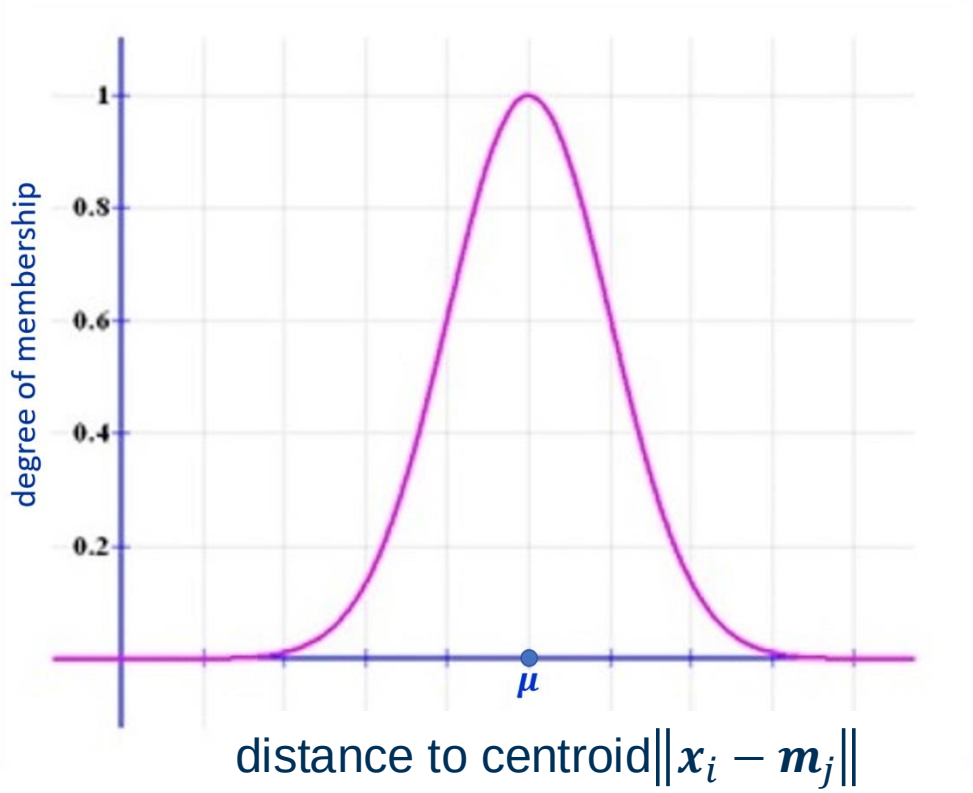
Variation of k-Means Clustering

Fuzzy C-Means Introduction

- In a clustering problem, each data point $\boldsymbol{x}_i$ has a set of coefficients, which gives the coefficient matrix $\boldsymbol{W} = w_{ij} \in [0,1], i = 1, \dots, N, j = 1, \dots, k$, where each element, w_{ij}, tells the degree of membership of element $\boldsymbol{x}_i$ in cluster $\mathcal{C}_j$.
- A fuzzy membership function determines these coefficients for data points based on their distance from the centroids.
- Example fuzzy membership function

$$w_{ij} = e^{\frac{\left(\|\boldsymbol{x}_i - \boldsymbol{m}_j\| - \mu\right)^2}{2\sigma^2}}$$

where μ is a central value, and σ is a standard deviation.



Appendix A: Notations

- x_i: a single feature
- $\boldsymbol{x}_i$: feature vector (a data sample)
- $\boldsymbol{x}_{:,i}$: feature vector of all data samples
- $\boldsymbol{X}$: matrix of feature vectors (dataset)
- N: number of data samples
- m: degree of polynomial
- P: number of features in a feature vector
- θ_i: a single model coefficient (parameter)
- $\boldsymbol{\theta}$: coefficient vector
- ε: error margin
- α: learning rate
- γ: bias factor
- Bold letter/symbol: vector
- Bold capital letters/symbol: matrix