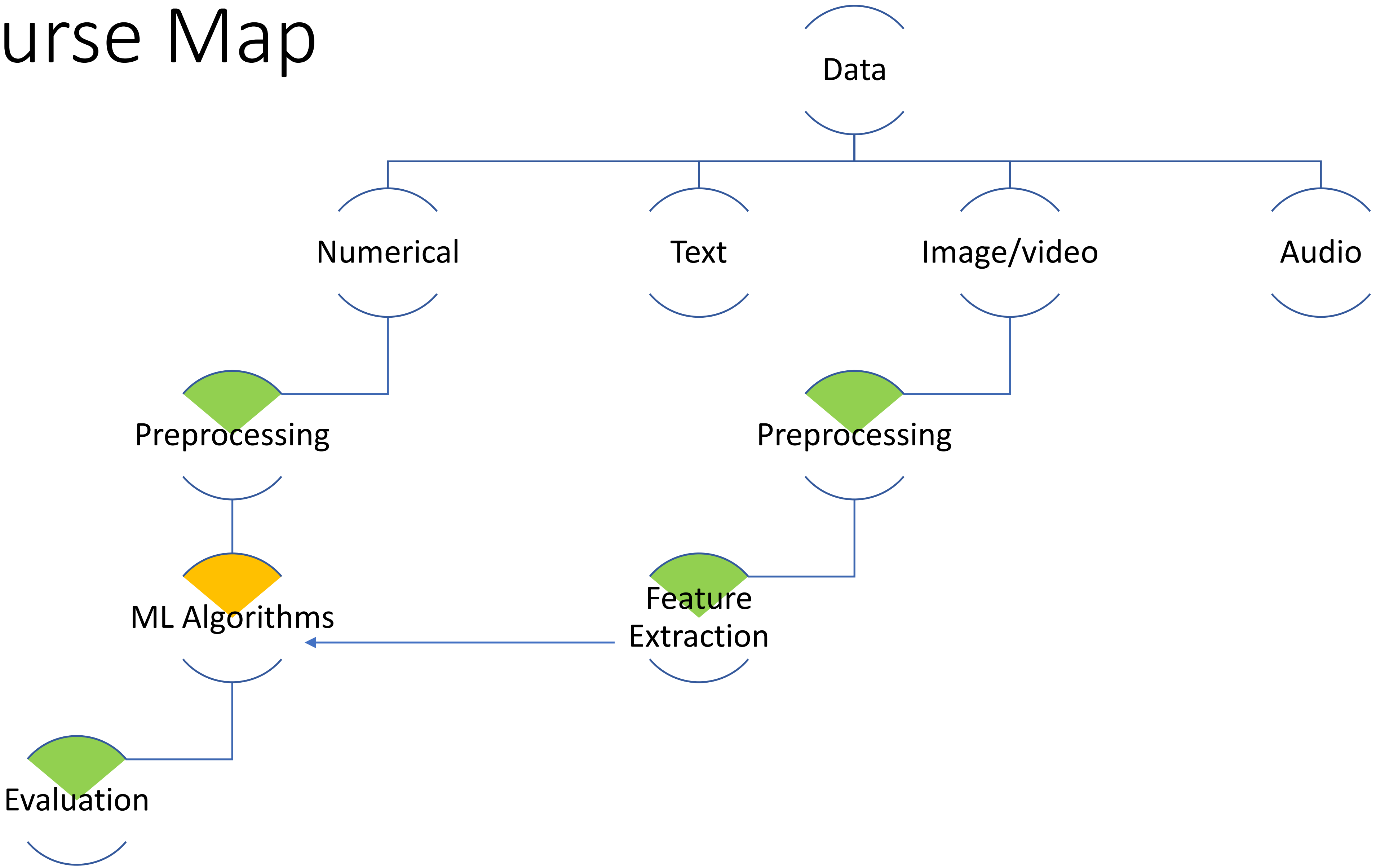
AI 330: Machine Learning

Fall 2023

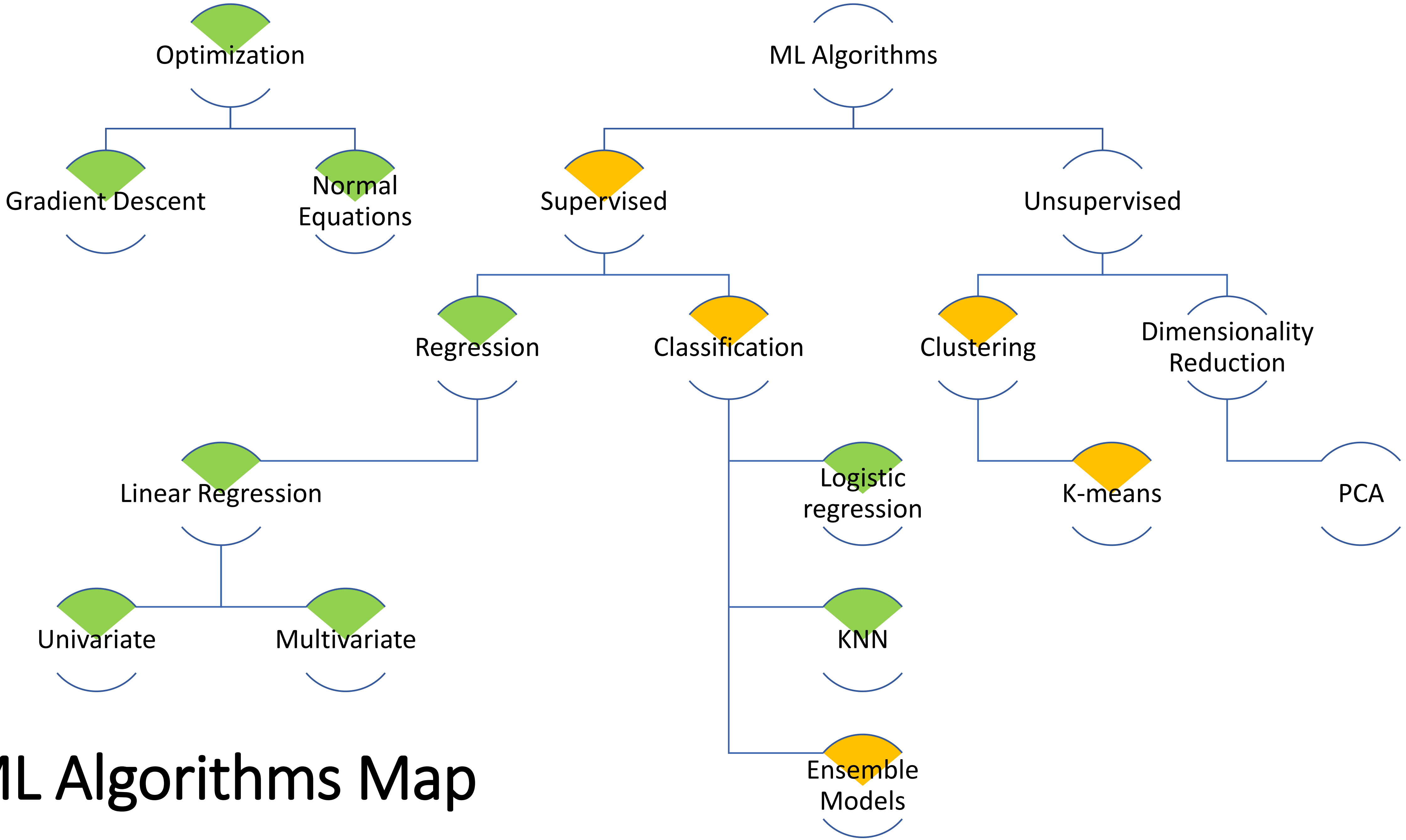
- **Dr. Wessam EL-Behaidy**
  - Associate Professor, Computer Science Department,
  - Faculty of Computers and Artificial Intelligence,
  - Helwan University.

Dr. Ensaf Hussein
Associate Professor, Computer Science Department,
Faculty of Computers and Artificial Intelligence,
Helwan University.

Course Map



Data
Numerical
Text
Image/video
Audio
Preprocessing
Preprocessing
ML Algorithms
Feature Extraction
Evaluation



ML Algorithms Map

Update DataCamp Deadline

- Due to early exam schedule, **the deadline of DataCamp (phase 2)** will be:

20 Dec. instead 24 Dec.

Project Delivery

- The project delivery will be:

Online on Sat. 16 Dec.

Lecture 9

Ensemble Models
&
K-means

Slides of:
Machine Learning Specialization https://www.coursera.org/specializations/machine-learning-introduction
at Stanford University (**Andrew Ng**)

Ensemble Models

What is Ensemble Learning?

Ensemble learning is a machine learning technique that combines the outputs of diverse models (often called “weak learners”) to create a more precise, robust and reliable prediction.

It **aims** to mitigate errors or biases that may exist in individual models.

By considering multiple perspectives and utilizing the strengths of different models, ensemble learning **improves** the overall performance of the learning system.

Ensemble Models

Its drawbacks and challenges

- being computationally expensive and
- time-consuming

due to the need for training and storing multiple models, and combining their outputs.

Simple Ensemble Techniques

Simple but powerful techniques, namely:

- Max Voting
- Averaging
- Weighted Averaging

Max Voting

- It is generally used for **classification** problems.
- It has two types: **Hard** and **soft** voting

Hard Voting (depends on class labels)

- It is also known as **majority voting**.
- Its idea is collecting predictions for each class label and predicting the class label with the most votes.

For instance, suppose the ensemble of classifiers contained 3 members: C1, C2, and C3, that assign the following classifications to a training sample: [0,0,1]

➔ The class label using max voting y=**mode** [0,0,1]= 0 .

Max Voting

Soft Voting (depends on class prediction probabilities)

- It involves collecting predicted probabilities for each class label and predicting the class label with the **largest probability**.

➔class label **y=**$\boldsymbol{argmax_i \sum_{j=1}^{n} W_j P_{ij}}$, where P_{ij}= predicted probabilities of classifier j for class i. W_j is is the weight that can be assigned to the jth classifier. (**Weight can be used or not**)

- **For instance**, C1(x)=[0.9,0.1], C2(x)=[0.2,0.8], and C3(x)=[0.6,0.4]. It has constant weights for ensemble members [0.5, 0.6, 0.3].
- The **prediction** would be as follows:
  - class 0: y0 [0.5*0.9 + 0.6*0.2 + 0.3*0.6] = 0.75
  - class 1: y1 [0.5*0.1 + 0.6*0.8 + 0.3*0.4] = 0.65

  ➔would yield a prediction y = 0.
-

Max Voting

Advantage of **max voting** :

- It is simple to understand and implement.

Drawbacks of **max voting**:

- It is useless when the baseline classifiers predictions have the same results.

Average Voting

- It is generally used for **regression** problems.
- The idea of averaging voting is an average of the predictions is used to make the final prediction.
- Average prediction is calculated using the **arithmetic mean**, which is the sum of the predictions divided by the total predictions made. ➔ class label **y=**$\boldsymbol{argmax_i}(\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{P_{ij}})$, where P_{ij}= predicted probabilities of classifier j for class i.
- **For instance**, suppose the ensemble of classifiers contained three members: C1(x)=[0.9,0.1], C2(x)=[0.2,0.8], and C3(x)=[0.6,0.4].
- The **mean prediction** would be as follows:
  - class 0: y0 [(0.9 + 0.2 + 0.6)/3] = 0.566
  - class 1: y1 [(0.1 + 0.8 + 0.4)/3] = 0.433

  ➔would yield a prediction y = 0.

Average Voting

Its advantage:

- it is more accurate in performance than majority voting and reduces overfitting.

Its drawbacks:

- Computationally more expensive than the max voting method
- It assumes that all baseline models in the ensemble are equally effective. However, it is not the case as some models may be better than others.

Weighted Average Voting

- It is a slightly modified version of averaging voting.
- Its idea is to give different weights to the baseline learners, indicating the importance of each model in prediction.
- ➔ class label **y=**$\boldsymbol{argmax_i \frac{\sum_{j=1}^{n} W_j P_{ij}}{\sum_{j=1}^{n} W_j}}$, where P_{ij}= predicted probabilities of classifier j for class i. W_j is is the weight that can be assigned to the jth classifier

- **For instance**, C1(x)=[97.2,2.8], C2(x)= [100.0,0], and C3(x)=[95.8,4.2]. It has constant weights for ensemble members [0.84, 0.87, 0.75].
  - class 0: y0 = ((97.2 * 0.84) + (100.0 * 0.87) + (95.8 * 0.75))**/** (0.84 + 0.87 + 0.75) =97.763.
  - class 1: y1 = ((2.8 *0.84) + (0 * 0.87) + (4.2 * 0.75))**/** (0.84 + 0.87 + 0.75) =2.235.

  ➔ would yield a prediction y = 0.

Weighted Average Voting

Its advantage:

- It is more accurate than the simple average-voting method.

Its drawbacks:

- Computationally more expensive than the average voting method, which makes it of little application.

Its challenge :

- is choosing each member’s relative weighting.

Advanced Ensemble Techniques

- Advanced techniques:
  - **Bagging**
    - Random Forest
  - **Boosting**
    - Adaptive Boosting (AdaBoost)
    - Gradient Boosting
    - Extreme Gradient Boosting (XGBoost)
  - **Stacking**

Three characteristics of any ensemble model

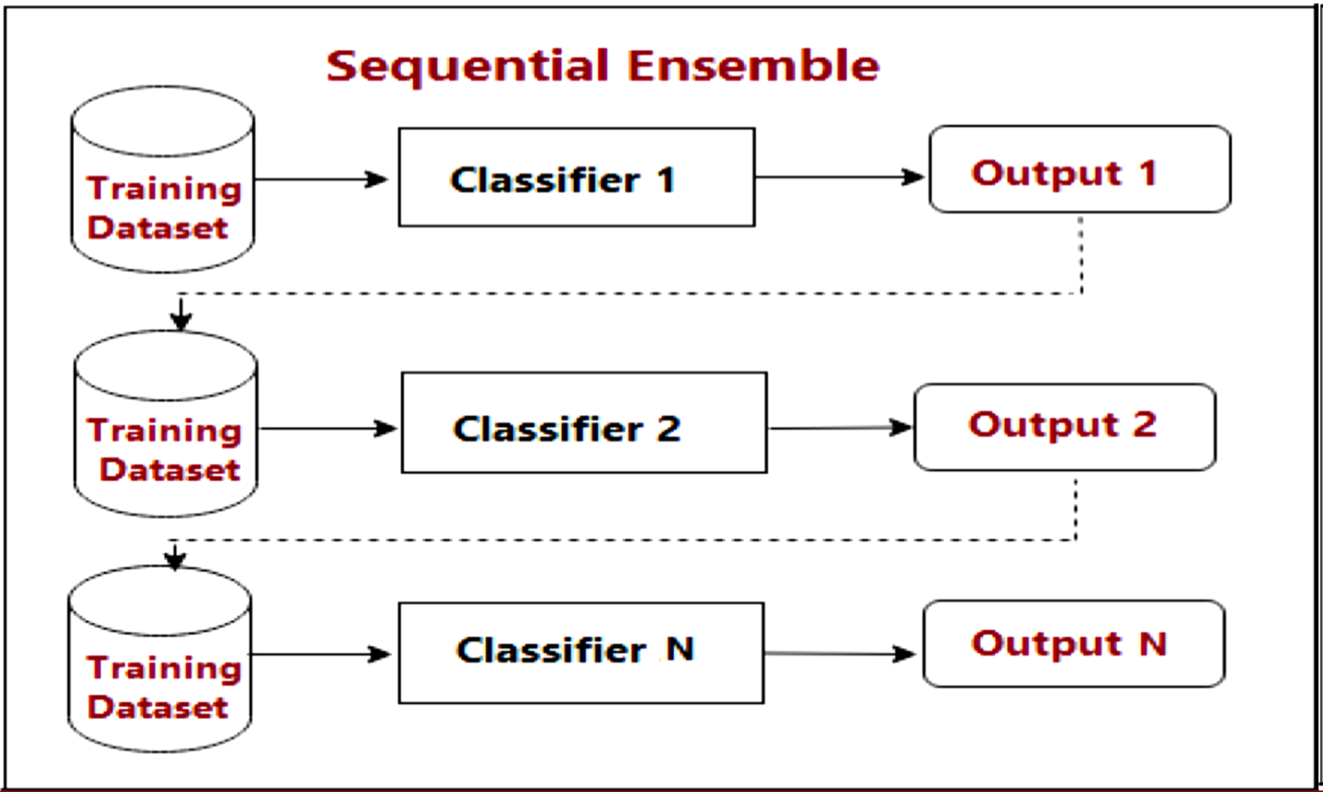
- **3 characteristics can affect the ensemble performance:**

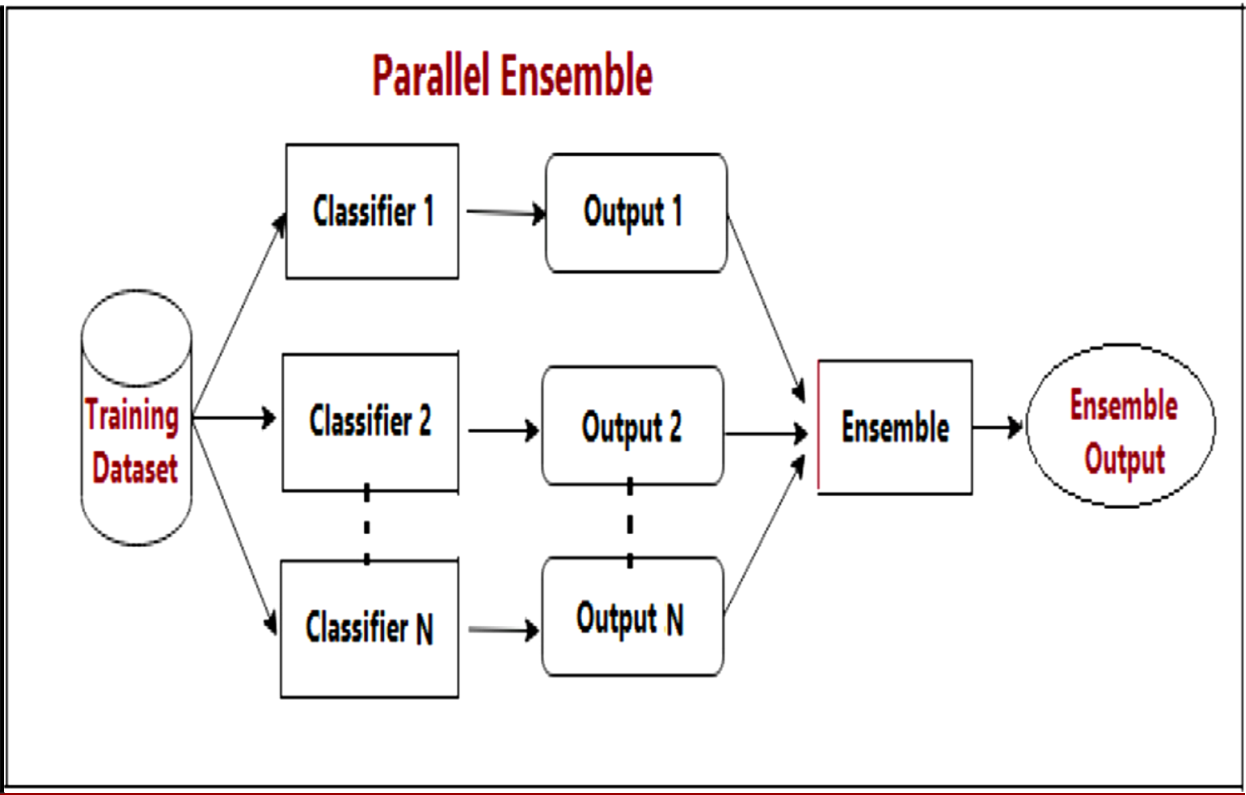
1) **Dependency**
2) **Heterogeneity**
3) **Fusion methods**

Three characteristics of any ensemble model

- **3 characteristics can affect the ensemble performance:**

1) **Dependency** on the trained baseline models, whether they are **sequential** or **parallel**.



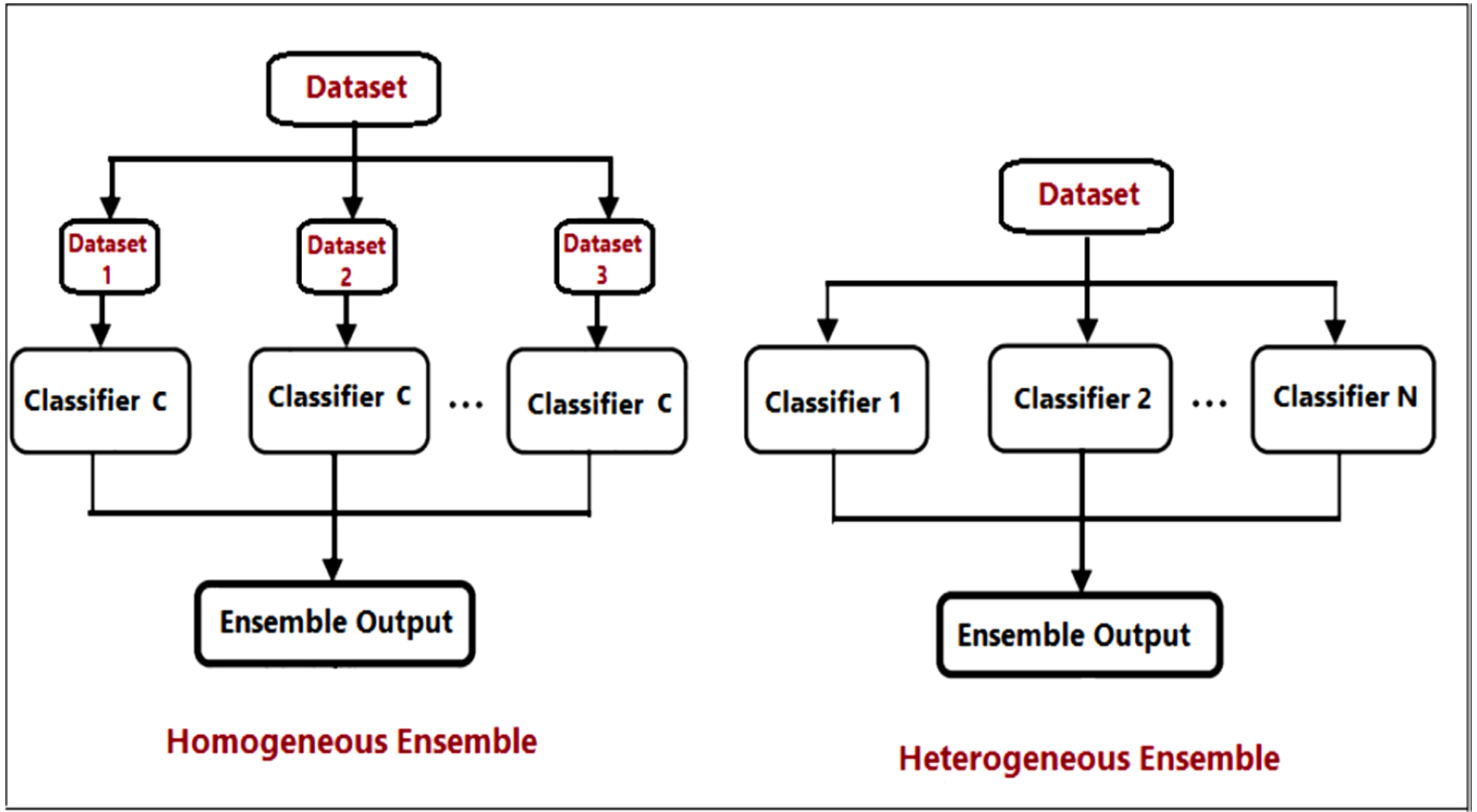


Three characteristics of any ensemble model

- **3 characteristics can affect the ensemble performance:**

2) **Heterogeneity** of the involved baseline classifiers, whether **homogeneous** or **heterogeneous**.

Homogeneous:
Same classifier
Different dataset



Heterogeneous:
Different classifiers
Same dataset

Three characteristics of any ensemble model

- **3 characteristics can affect the ensemble performance:**

3) **Fusion methods**, which involve choosing a suitable process for combining outputs of the baseline classifiers using different **weight voting** or **meta-learning** method.

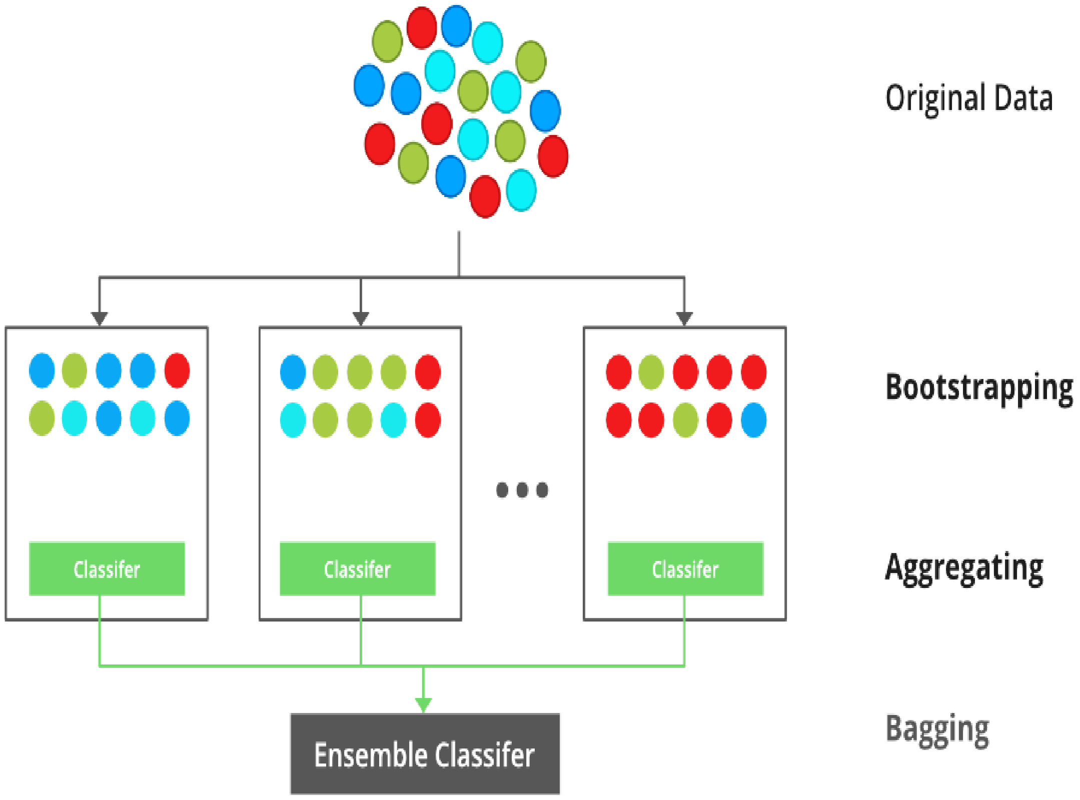
Bagging

- **Homogeneous** weak learners,
- learns them **independently** from each other in <u>parallel</u>, and
- combines them using **averaging** or **max vote** process.

➔ to obtain a model with a lower variance.

- Also known as:

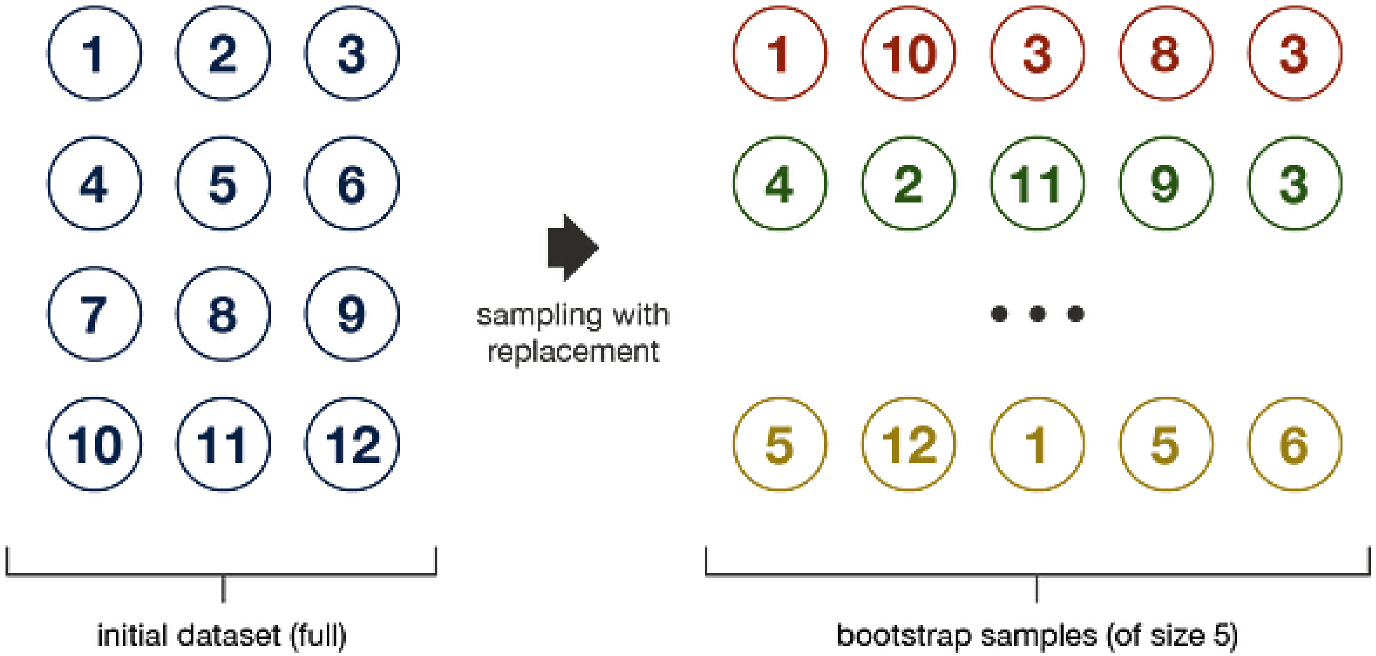
  “**bootstrap aggregating**”



Use new datasets (**bootstrap**) to teach the same algorithm several times and then predict the final answer via simple averaging answers (**aggregate**).

Bagging: Bootstrapping

- **Bootstrapping** is a statistical technique that used in generating samples of size B (called **bootstrap samples**) from an initial dataset of size N by randomly drawing **sampling with replacement** B observations.
- Data in random subsets may repeat. For example, from a set like “1-2-3-4-5” we can get subsets like “2-5-3”, “5-2-2”, “4-1-5” and so on.



Bagging Example

- **Random forest** approach is a **bagging method.** However, random forests also use **sample over features** trick to make the multiple fitted trees a bit less correlated with each others.
- **Sampling over features:** when growing each tree, instead of only sampling over the observations in the dataset to generate a bootstrap sample, we also sample over features and keep only a random subset of them to build the tree.
- Advantage:
    1) It **reduces the correlation** between the different returned outputs.
    2) **It makes the decision making process more robust to missing data**: observations (from the training dataset or not) with missing data can still be regressed or classified based on the trees that take into account only features where data are not missing.

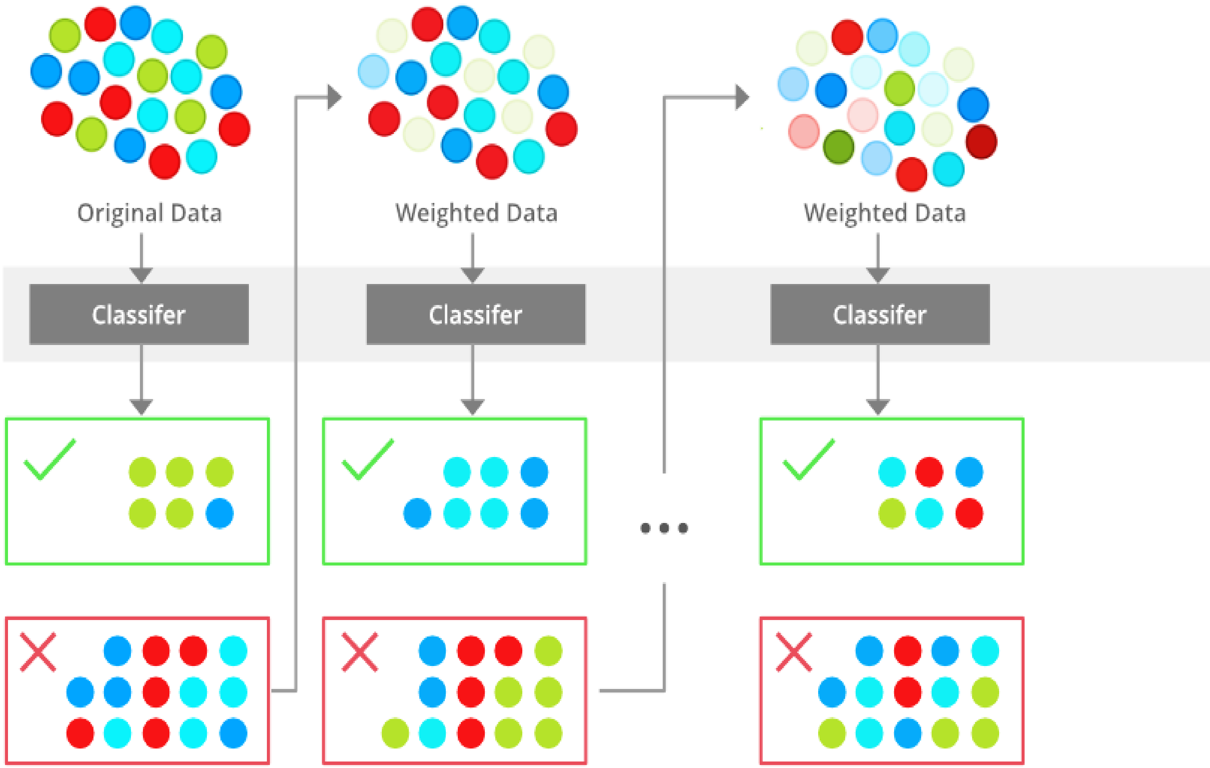
➔ random forest algorithm = bagging + random feature selection ➔ robust models

Boosting

- **Homogeneous** weak learners,
- **Dependent** on each other by fitting sequentially multiple weak learners
- combines them using **weighted averaging** of weak learners.

➔ to obtain stronger model with a lower bias.

- Boosting, like bagging, can be used for regression as well as for classification problems.



Boosting Algorithms are trained one by one **sequentially**. Each subsequent one paying most of its attention to data points that were **mis-predicted** by the previous one. Repeat until you are happy.

Boosting Steps

Boosting works in the following steps:

1. A subset is created from the **original dataset**, and all data points are given **equal weights**.
2. A **base model** is created on this subset.
3. This model is used to **make predictions** on the whole dataset.
4. **Errors** are calculated using the actual values and predicted values.
5. The observations which are **incorrectly predicted**, are given **higher weights**.
6. **Another model** is created and predictions are made on the dataset (this model tries to correct the errors from the previous model).
7. Similarly, multiple models are created, each correcting the errors of the previous model.
8. The final model (strong learner) is the **weighted average** of all the models (weak learners).

Boosting Algorithms

- Two important boosting algorithms: **adaboost** and **gradient boosting**.
- These two meta-algorithms **differ** on how they create and aggregate the weak learners during the sequential process

AdaBoost (Adaptive Boosting)

- AdaBoost is the most popular boosting algorithms.
- It assigns weights to training instances and **adjusts these weights** based on the performance of weak learners.
- It focuses on **misclassified instances**, allowing subsequent weak learners to **concentrate on these samples**.
- The final prediction is determined by aggregating the predictions of all weak learners through a **weighted majority vote**.

Gradient Boosting

- Gradient Boosting is a widely used boosting algorithm that builds an ensemble of **decision trees**.
- Gradient boosting learns from samples with **large pseudo-residuals**. Moreover, **each tree** in gradient boosting is **weighted equally** when making final classification.

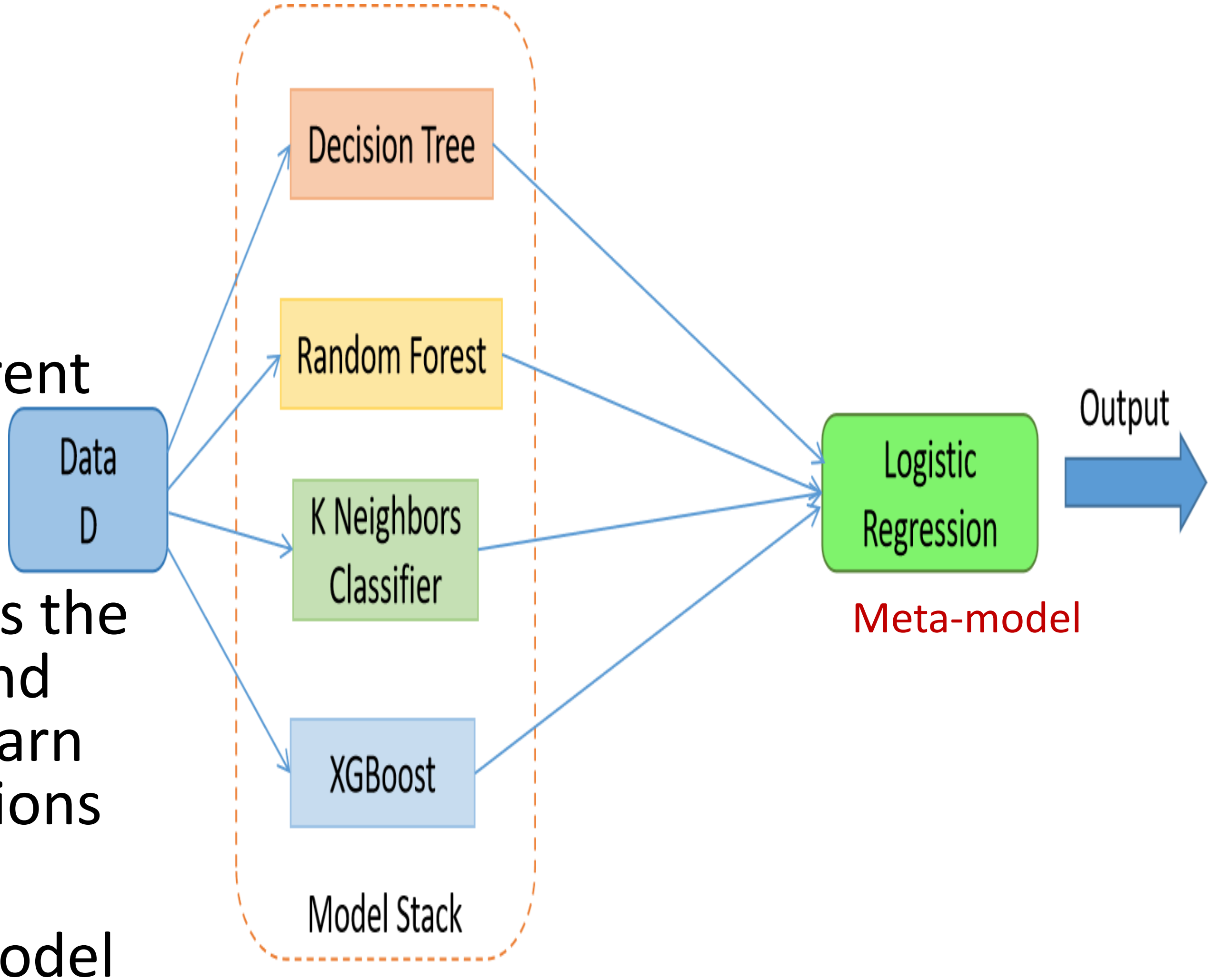
- Pseudo residuals are intermediate error terms i.e. difference between the actual value and intermediate predicted value.

XGBoost (Extreme Gradient Boosting)

- XGBoost is an advanced boosting algorithm that combines gradient boosting with **regularization** techniques to prevent overfitting.
- It incorporates both **tree-based** models and **linear** models to enhance performance and efficiency.
- It is known for its speed, scalability, and ability to handle large-scale datasets effectively.

Stacking

- **Heterogeneous** weak learners,
- learns them **independently** from each other in parallel, and
- combines them using **meta-learner** to output a prediction based on the different weak learner's predictions.



- A **meta-model** inputs the predictions as the features and the target being the ground truth values in data D, it attempts to learn how to best combine the input predictions to make a better output prediction.
- **Note:** The input predictions to meta-model are the predictions on the validation dataset, unseen during training.

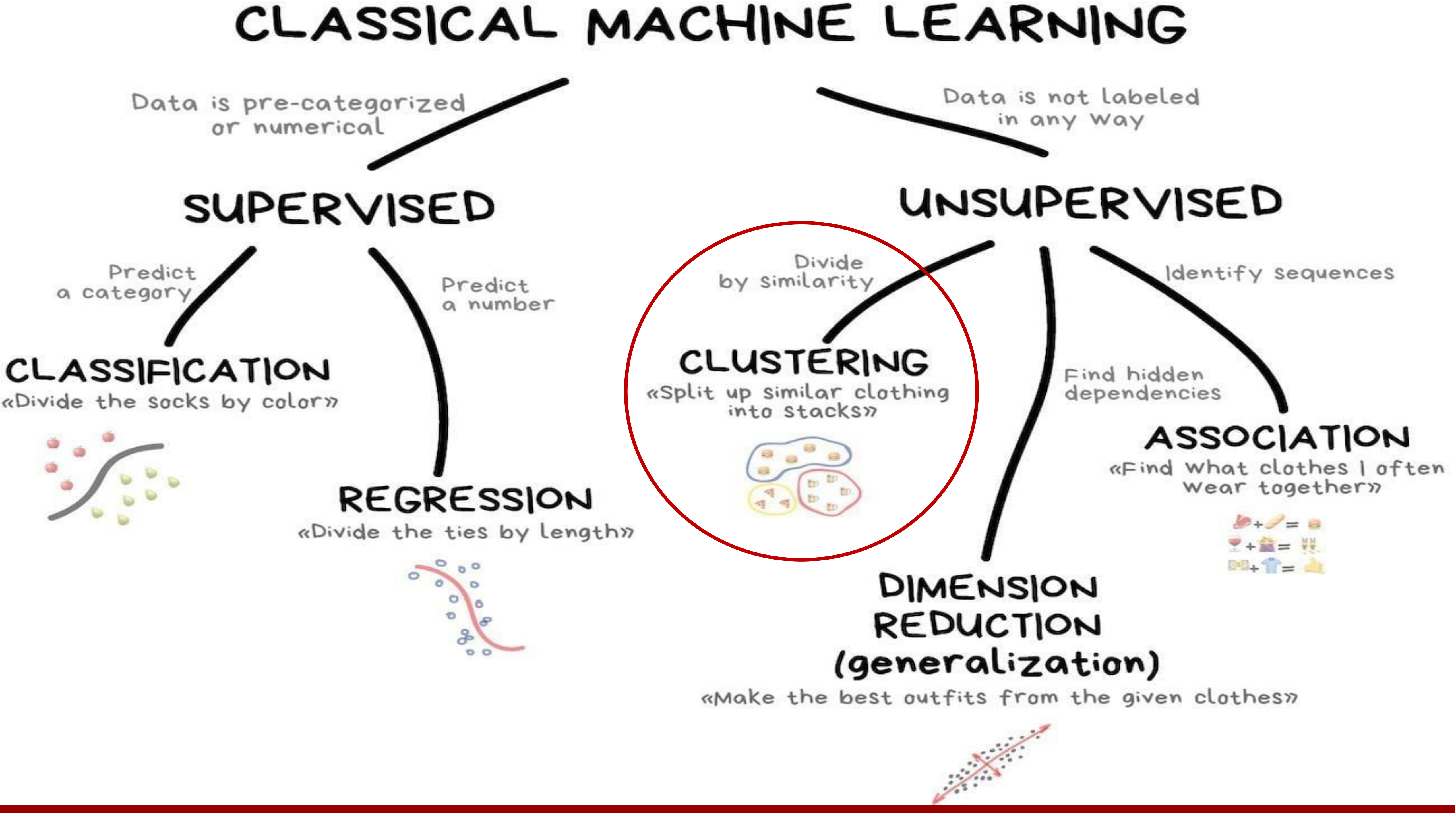
Stacking

- The ensemble members are referred to as base-models (level-0 learners), whereas the model that combines the predictions is referred to as the meta-model (level-1 model).
- Meta Model can be **Linear Regression** for regression problem or **Logistic Regression** for classification problem.
- Not always the base models are heterogeneous; different configurations of the same models can be used or the same model trained on different datasets.

Summary

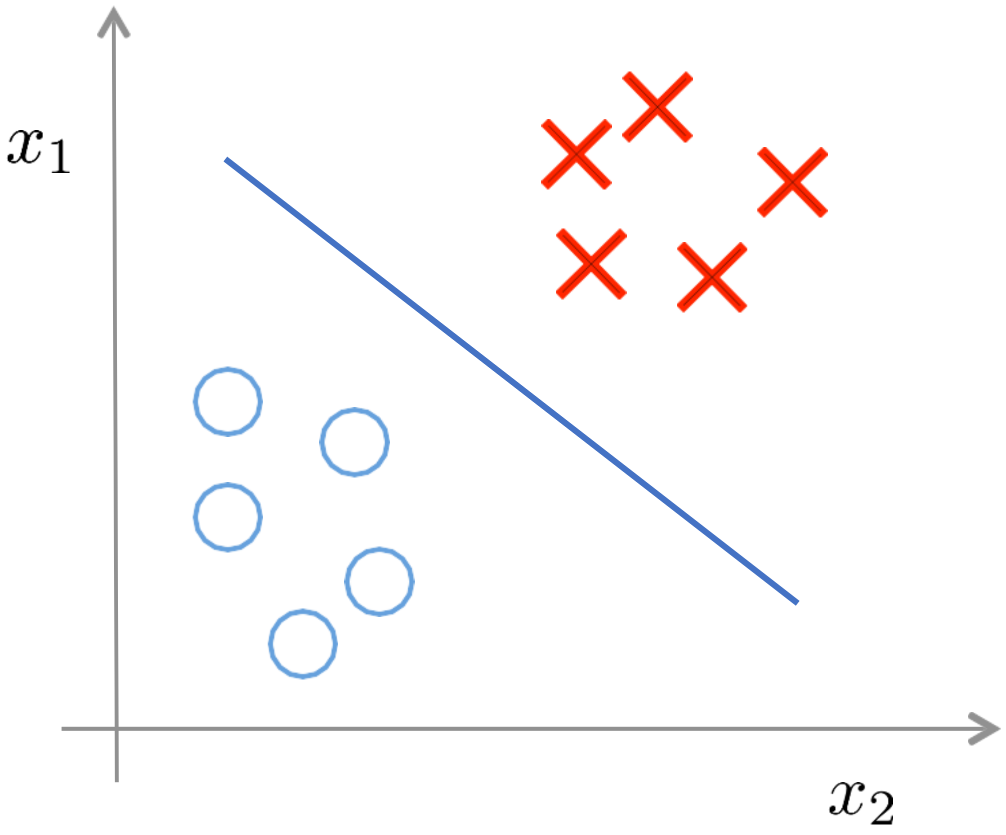
	Bagging	Boosting	Stacking
Purpose	Reduce Variance	Reduce Bias	Improve Accuracy
Base Learner Types	Homogeneous	Homogeneous	Heterogeneous
Base Learner Training	Parallel	Sequential	Parallel
Aggregation	Max Voting, Averaging	Weighted Averaging	Meta Model

Unsupervised learning introduction


CLASSICAL MACHINE LEARNING
Data is pre-categorized or numerical
Data is not labeled in any way
SUPERVISED
UNSUPERVISED
Predict a category
Predict a number
Divide by similarity
Identify sequences
Find hidden dependencies
CLASSIFICATION
«Divide the socks by color»
REGRESSION
«Divide the ties by length»
CLUSTERING
«Split up similar clothing into stacks»
ASSOCIATION
«Find what clothes I often wear together»
DIMENSION REDUCTION (generalization)
«Make the best outfits from the given clothes»

Supervised learning:

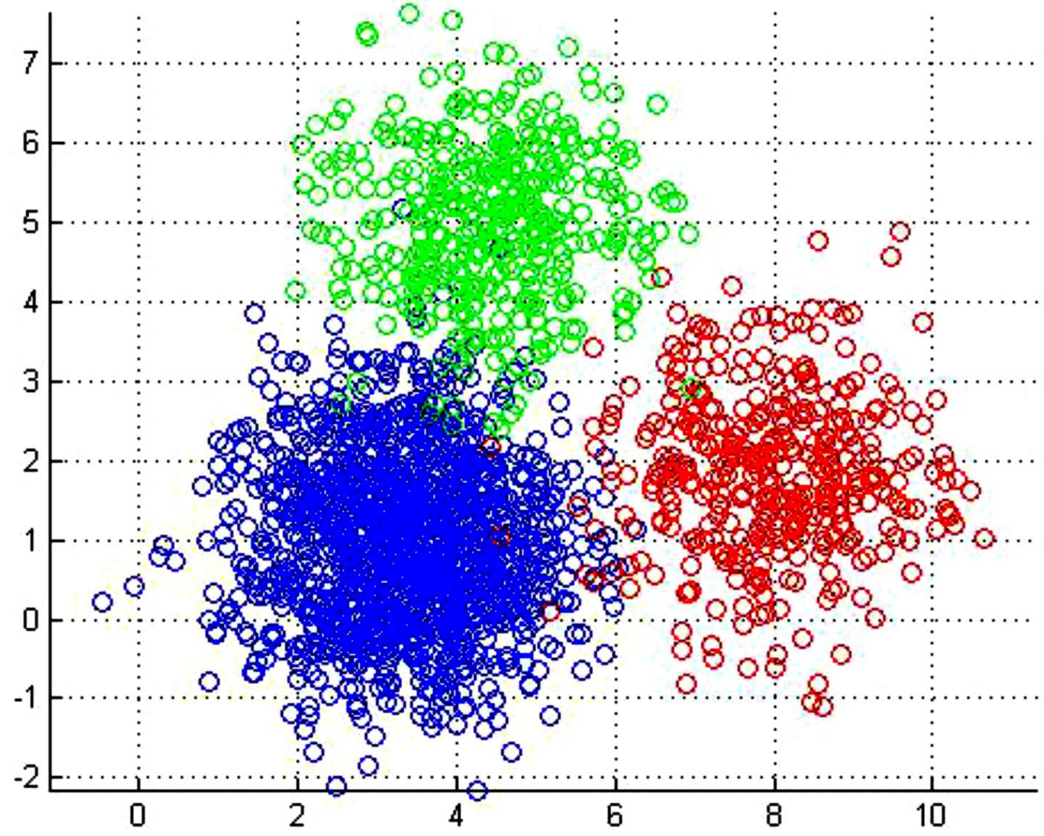
- Labels are given.



Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$

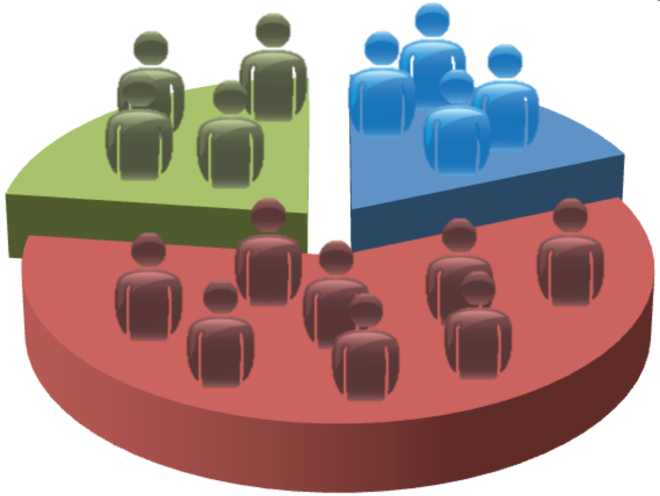
Unsupervised learning: Clustering

- No labels are given.
- Instead, we have to find the similarities between data and group the similar data into clusters.

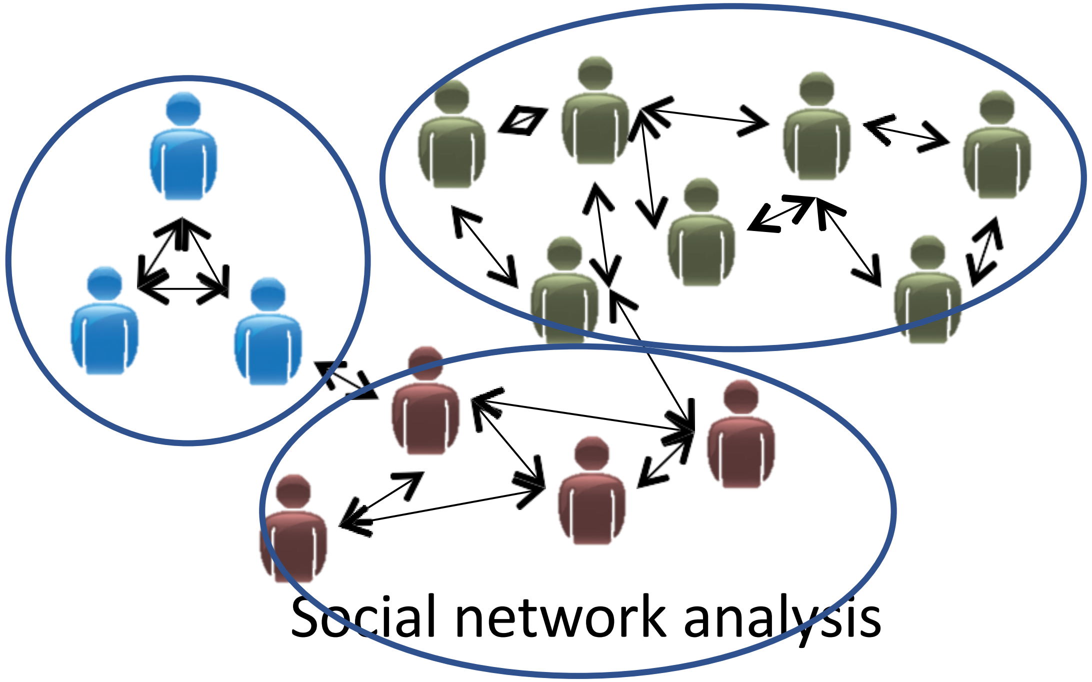


Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Applications of clustering

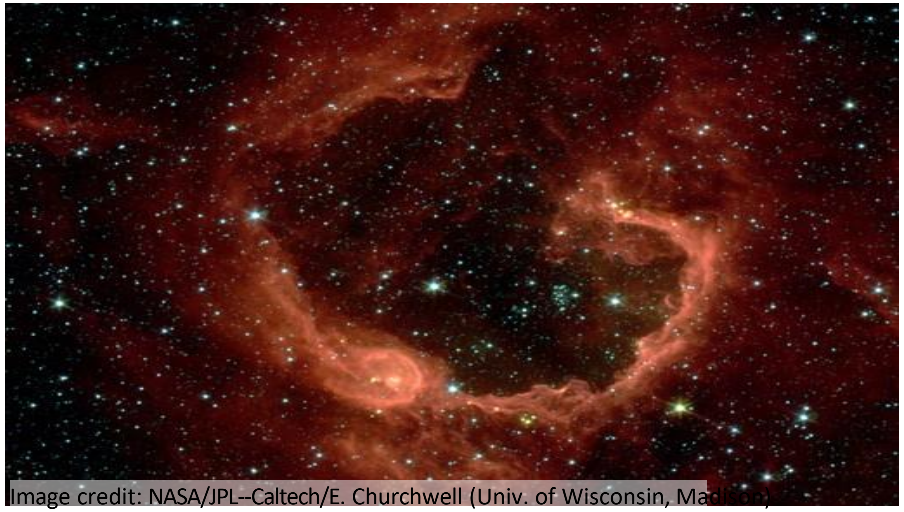


Market segmentation



Social network analysis

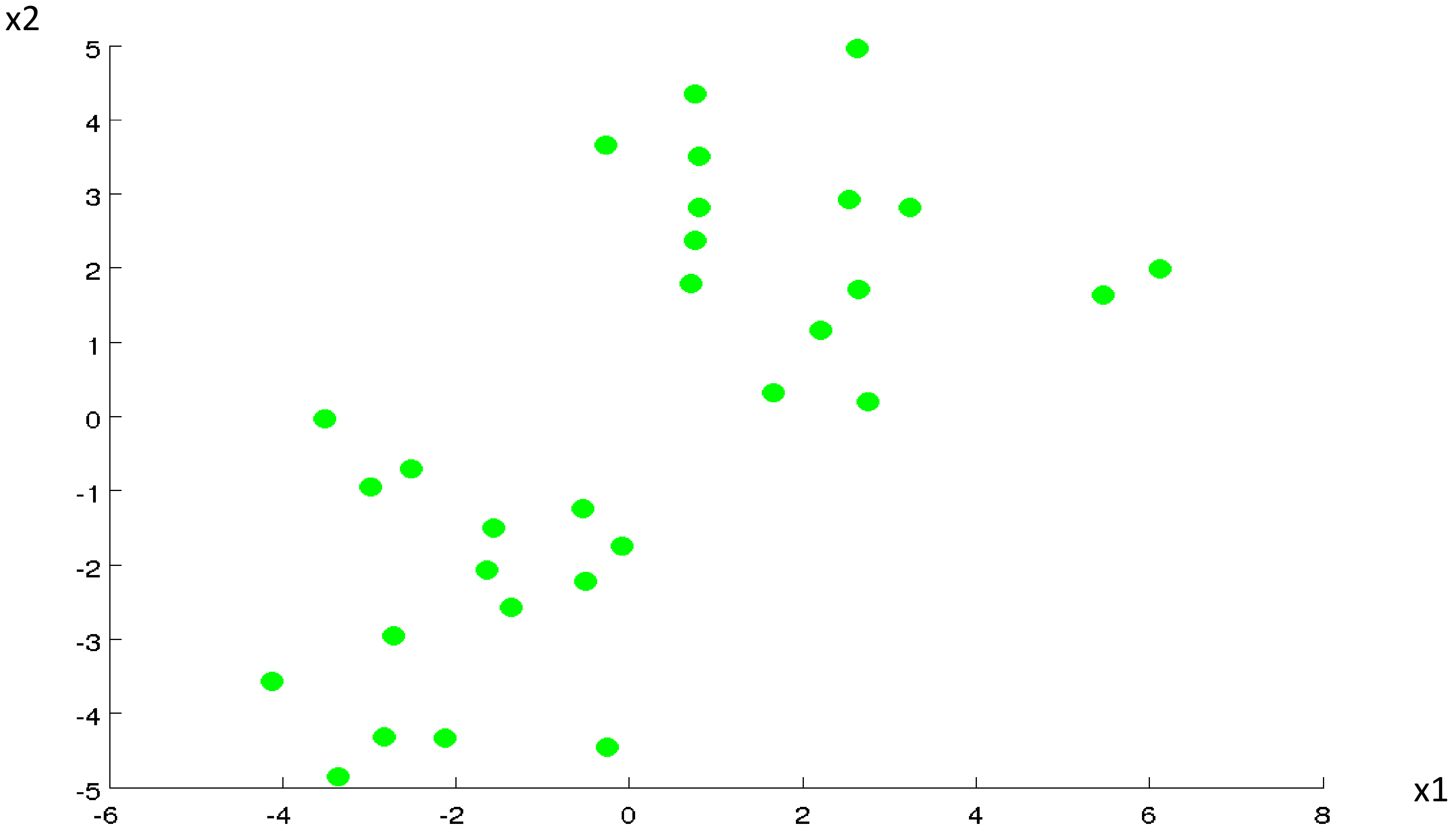
Organize computing clusters



Astronomical data analysis

K-means Algorithm

K-means Algorithm



K-means Algorithm

- Input:
  - K Number of clusters
  - Training dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

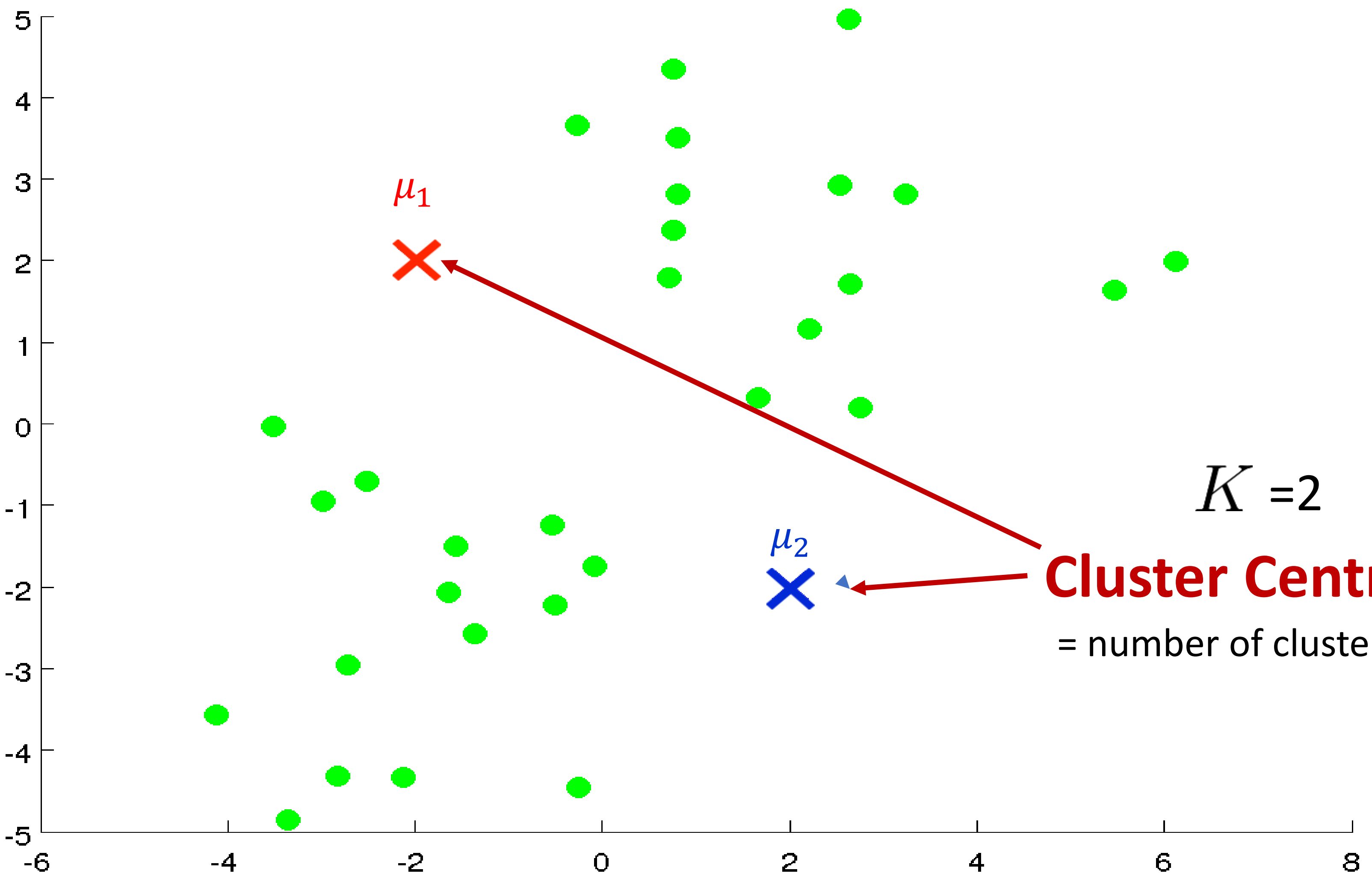
$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

m$\Rightarrow$ $number\ of\ samples$
n => number of features

For example:
If n= 2

So, $x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$

K-means Step: Randomly initialize cluster centroids



K =2

Cluster Centroids

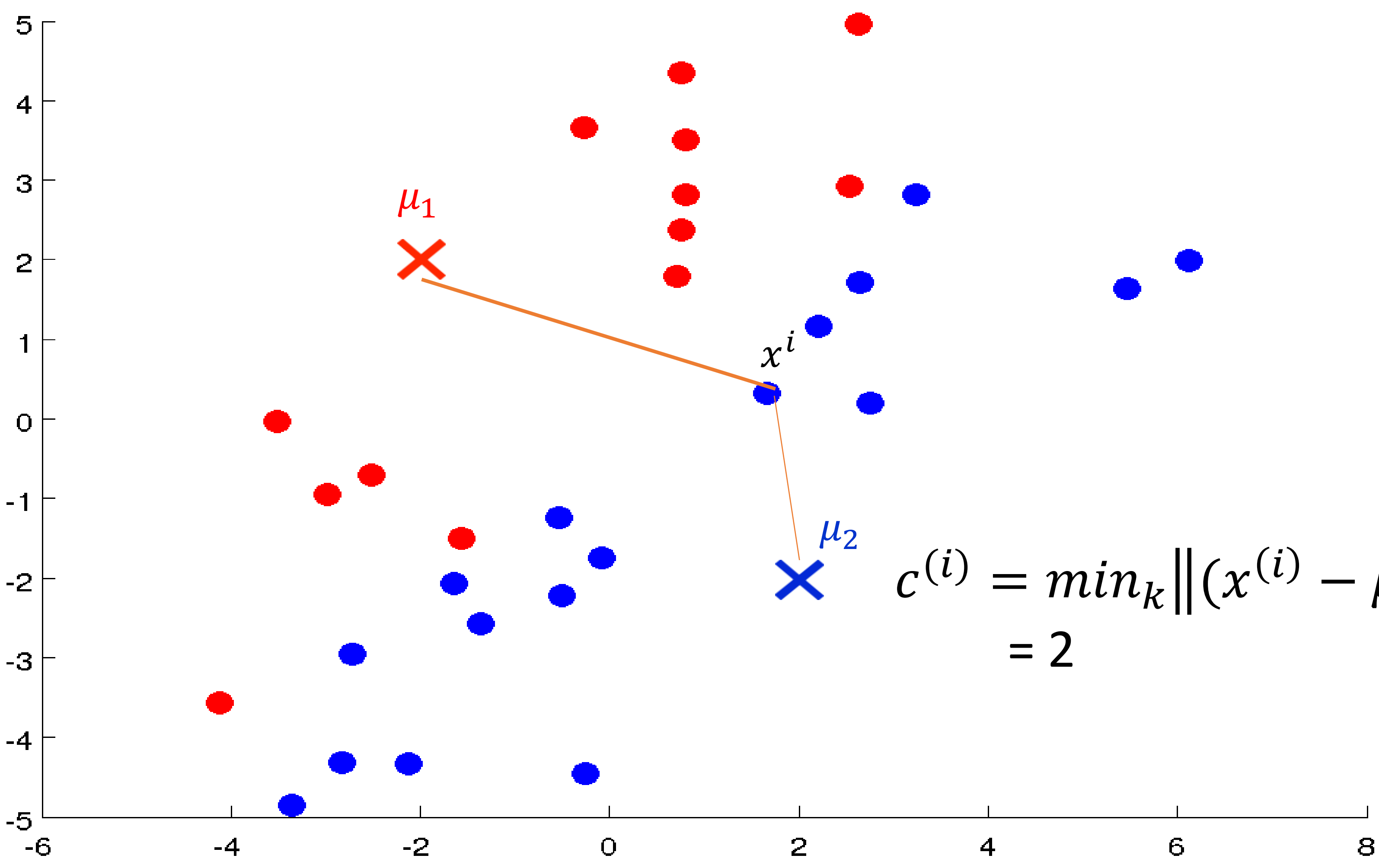
= number of clusters that you want

K-means Algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

K-means Steps: Cluster Assignment

Assign each point to the cluster with the ***closest*** centroid



$$c^{(i)} = min_k \left\|(x^{(i)} - \mu_k)\right\|^2, k = 1, .. K$$

= 2

K-means Algorithm: Cluster Assignment

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster Assignment Step

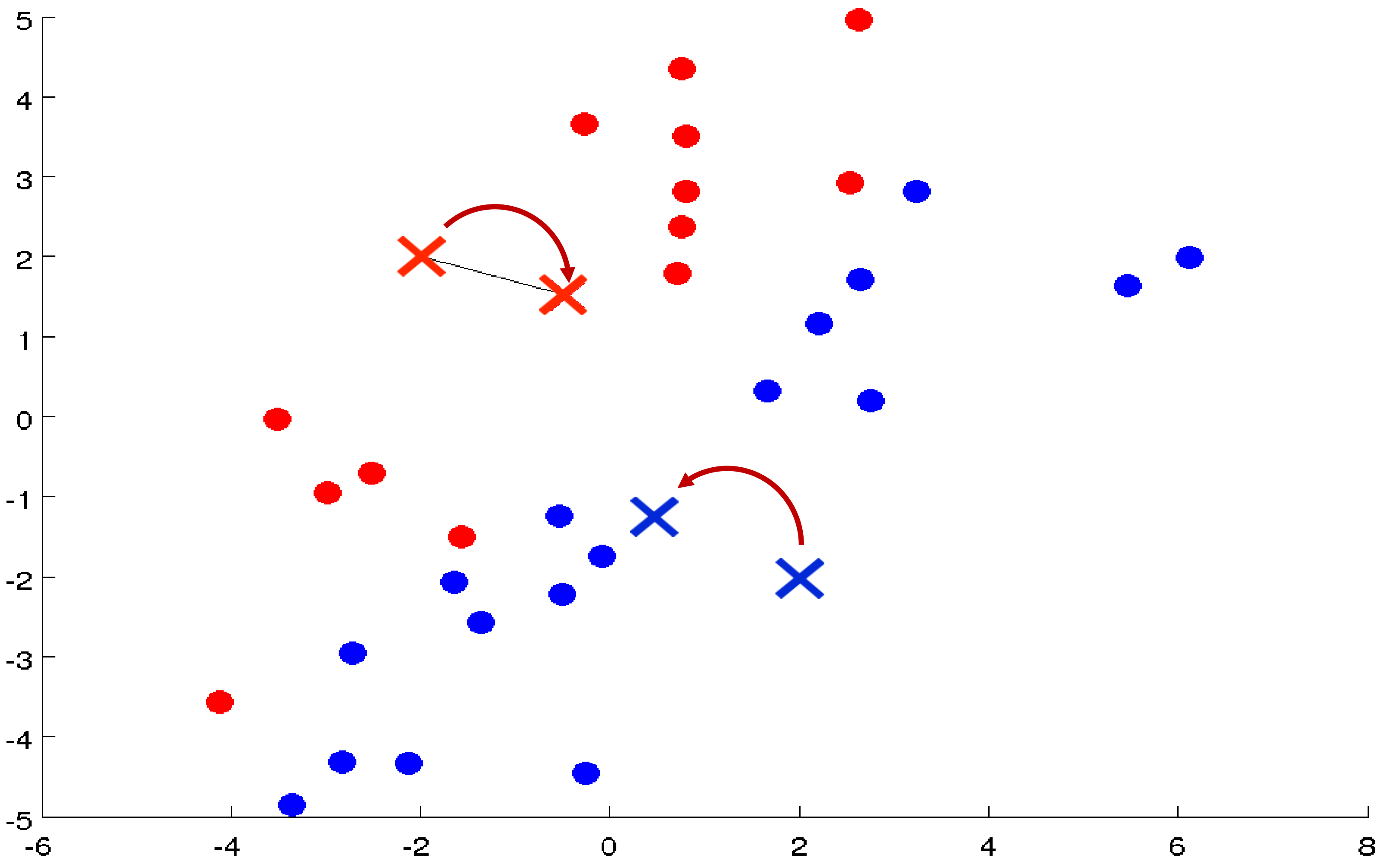
for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

It means: Calculate difference $c^{(i)} = min_k \left\| (x^{(i)} - \mu_k) \right\|^2, k = 1, \ldots, K$

$for\ example$: $c^{(1)} = 2, c^{(2)} = 1, c^{(3)} = 2, c^{(4)} = 1, \ldots, c^{(m)} = 1$

K-means Steps: Move centroids



K-means Algorithm: Cluster Assignment

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster Assignment Step

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

Move centroid Step

for k = 1 to K

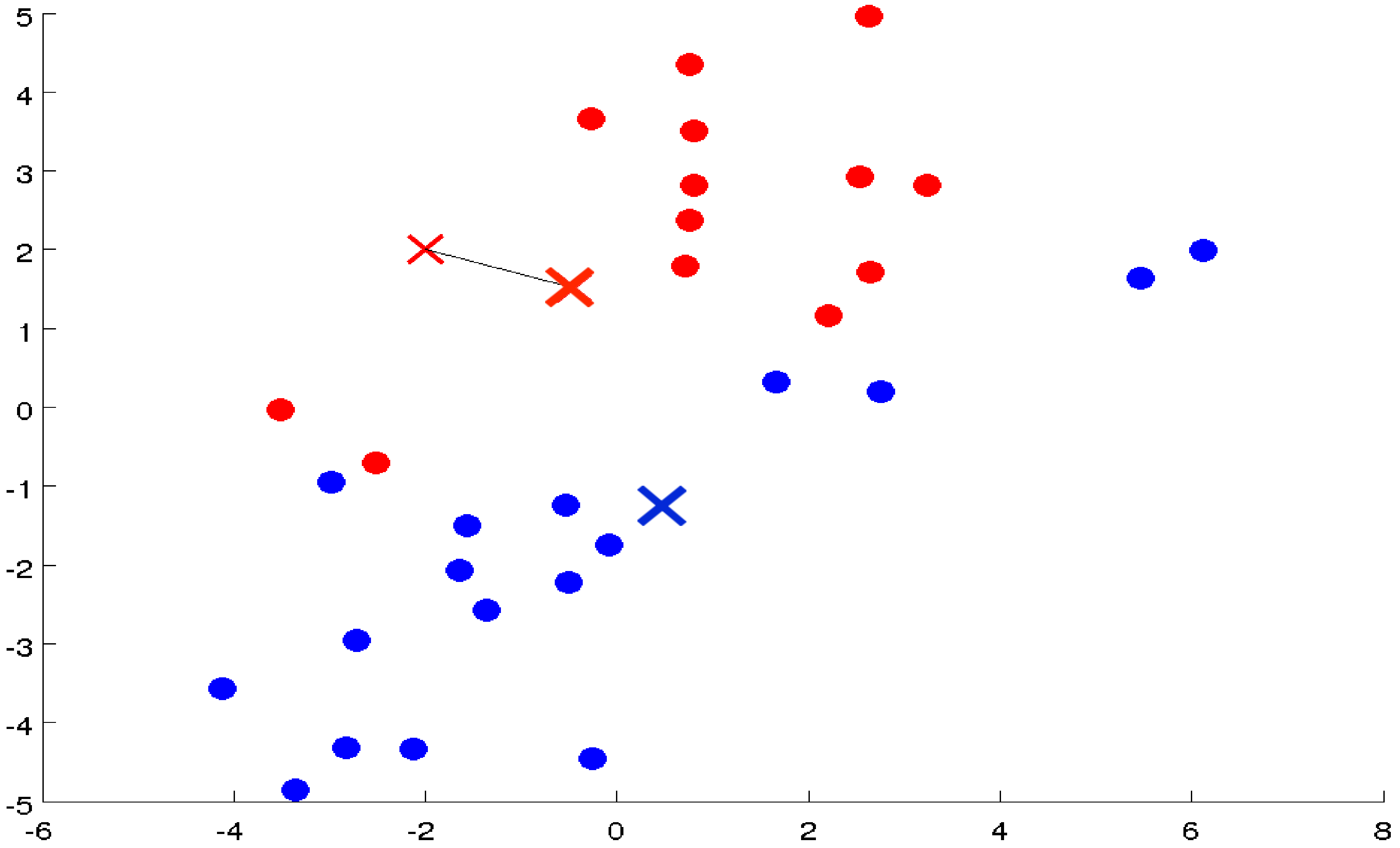
μ_k := average (mean) of points assigned to cluster k

$if\ for\ example$: $c^{(1)} = 2, c^{(3)} = 2, c^{(11)} = 2, c^{(20)} = 2$

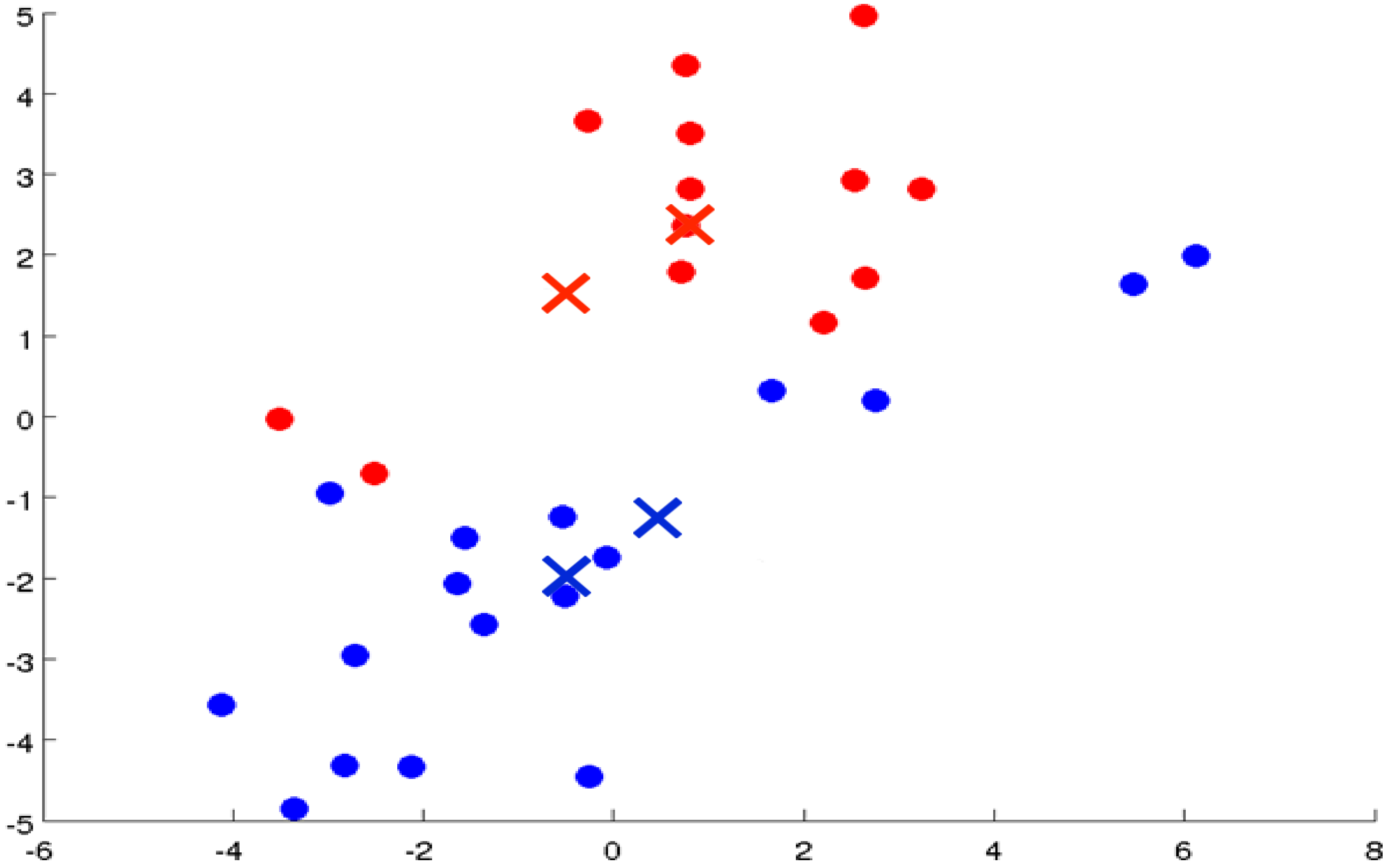
}

So, $\mu_2 = \frac{1}{4}[x^{(1)} + x^{(3)} + x^{(11)} + x^{(20)}]$ $\quad \in \boldsymbol{R^n}$

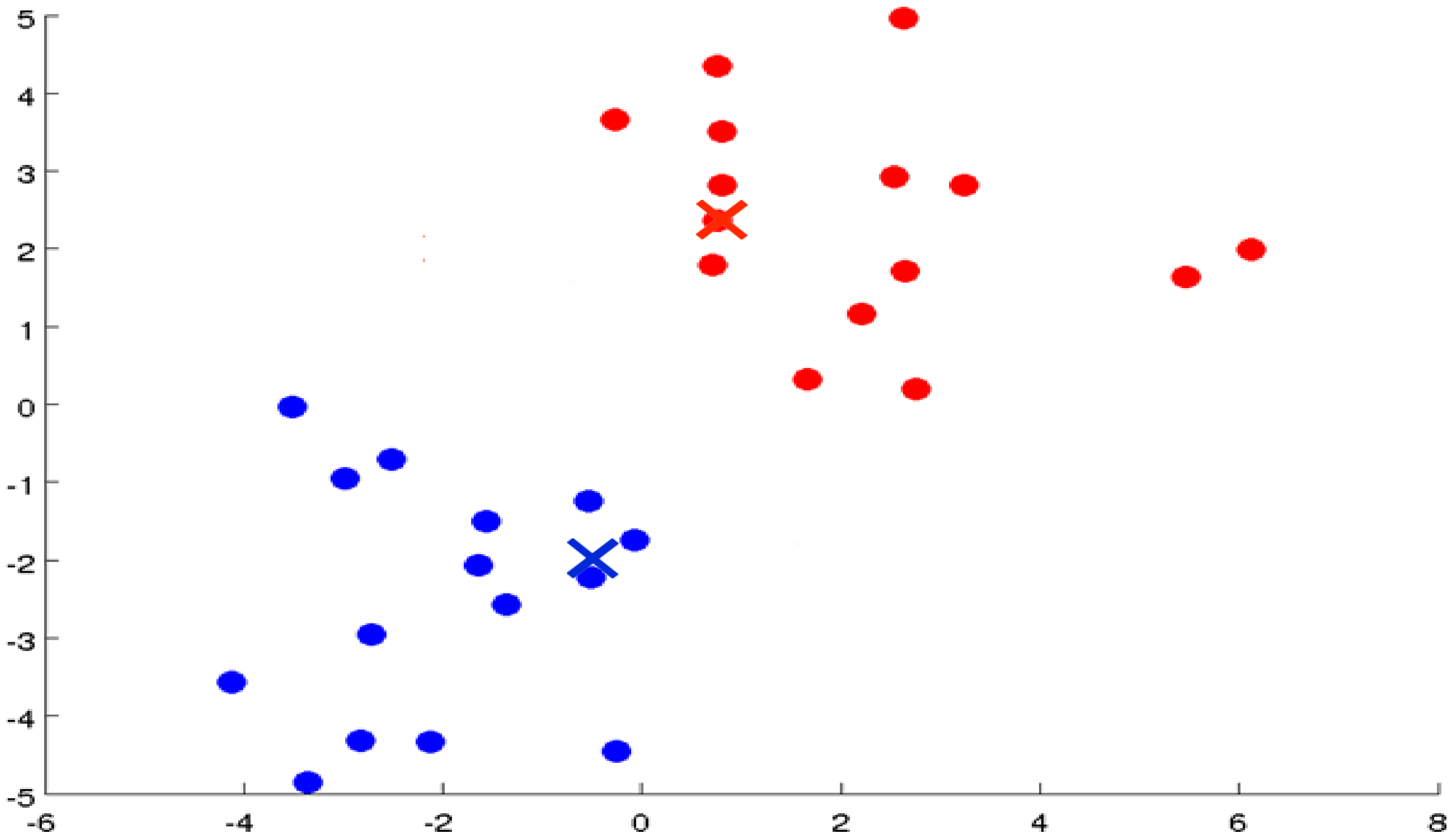
K-means Steps: Cluster Assignment



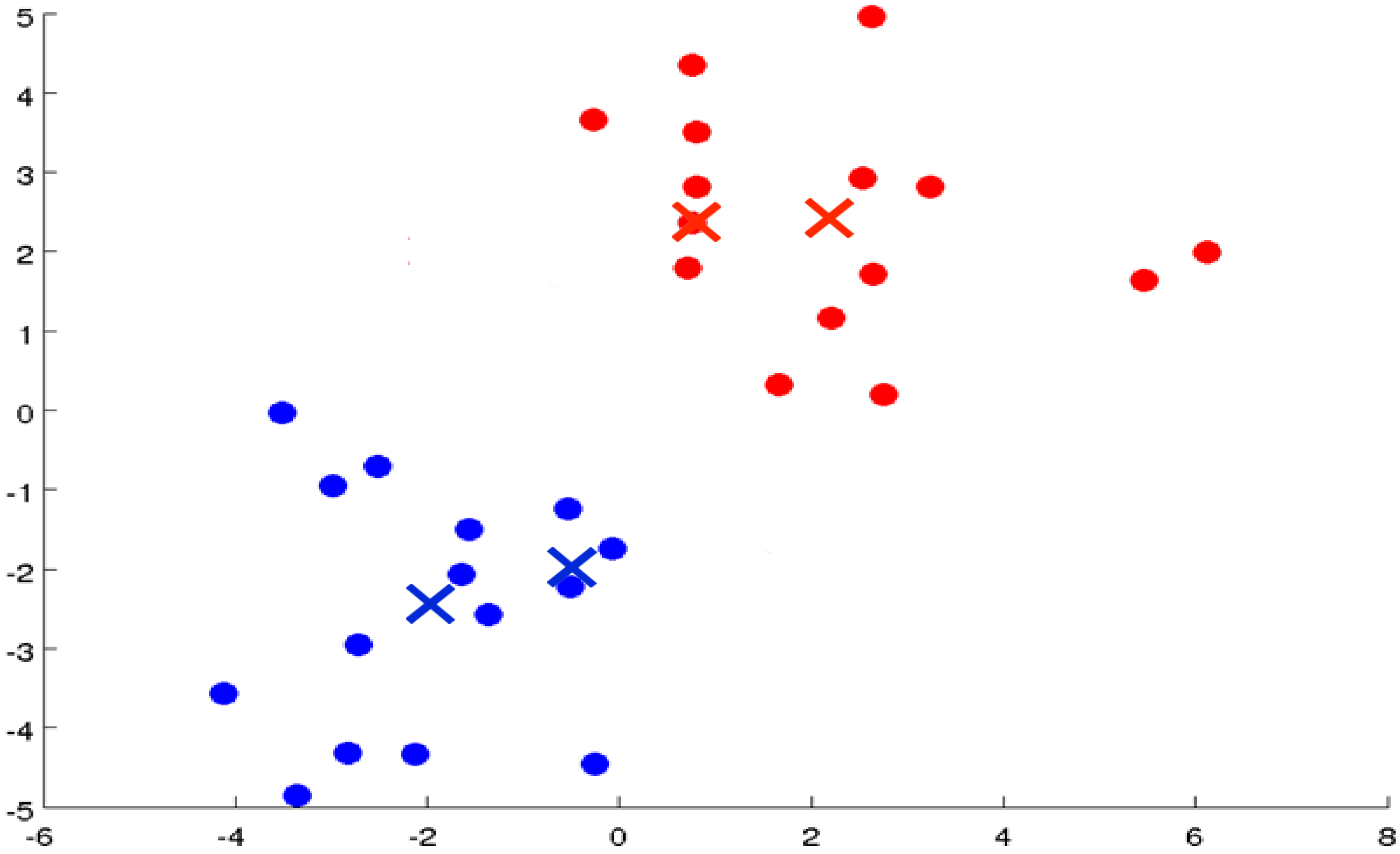
K-means Steps: Move centroids



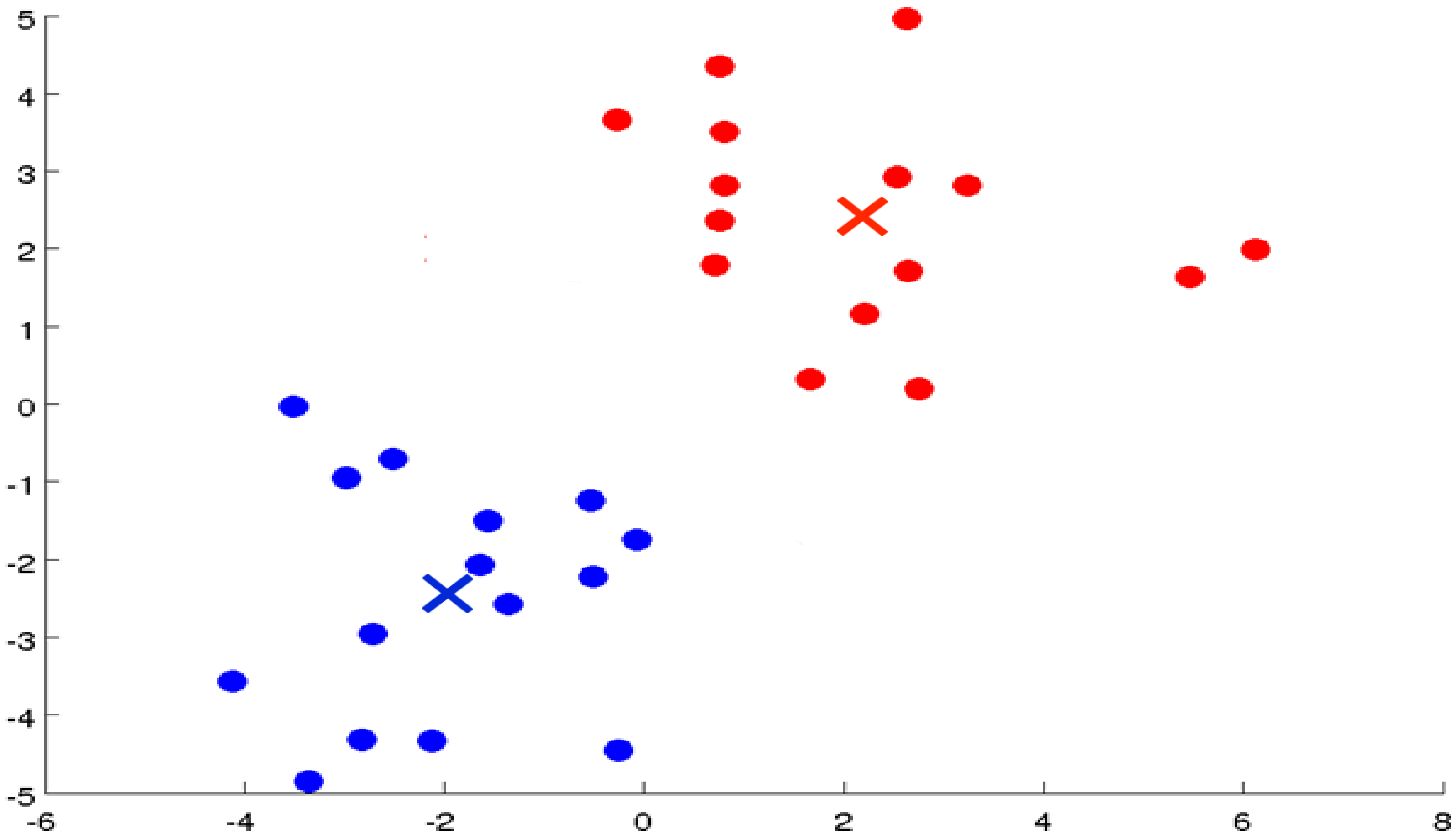
K-means Steps: Repeat until no change



K-means Steps: Repeat until no change



K-means Steps: Repeat until no change



K-means Algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster Assignment Step

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

Move centroid Step

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

}

K-means Properties

1- **Different initializations** yield different results! Doesn’t necessarily converge to the best partition

2- **K is a hyperparameter.** It needs to be set in advance.

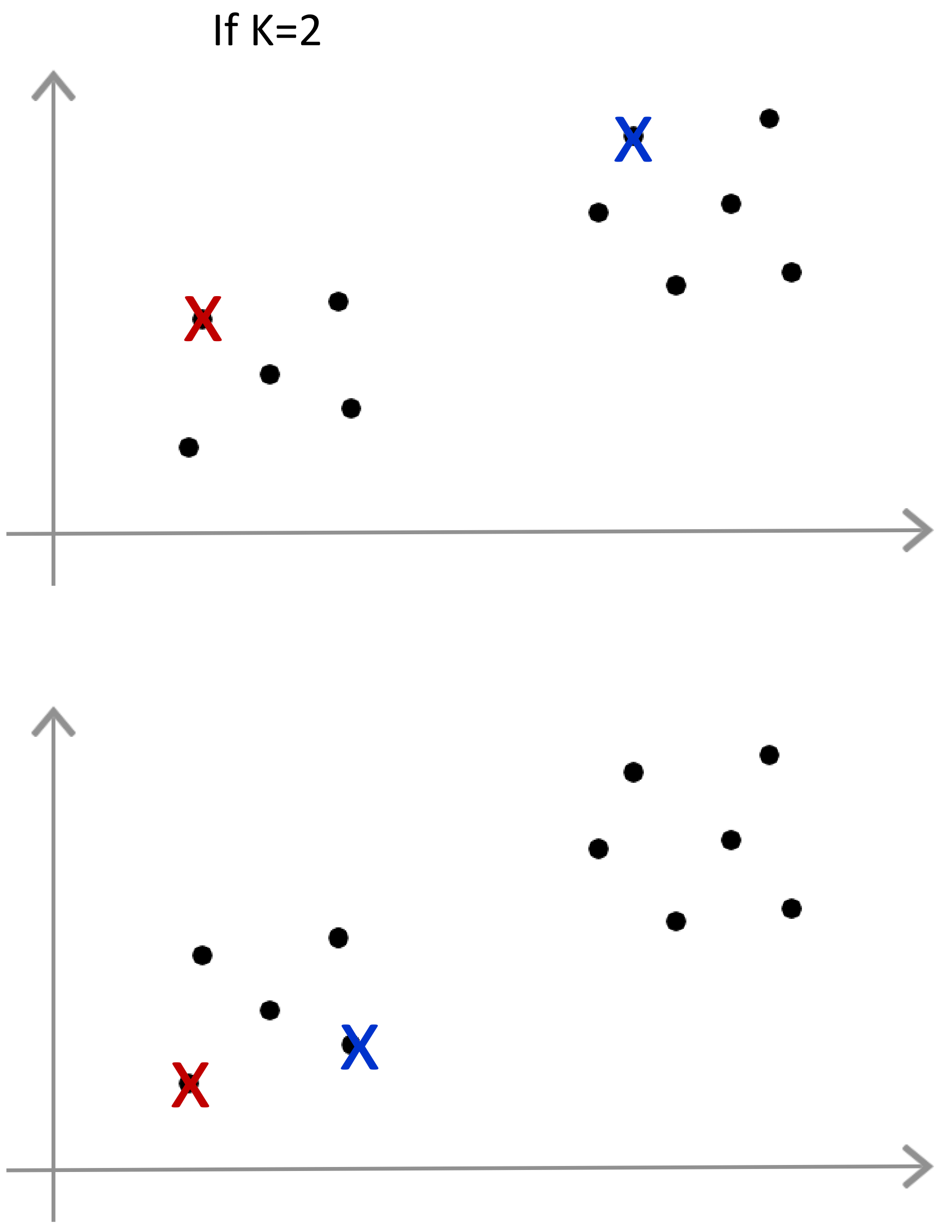
Random Initialization

Random initialization

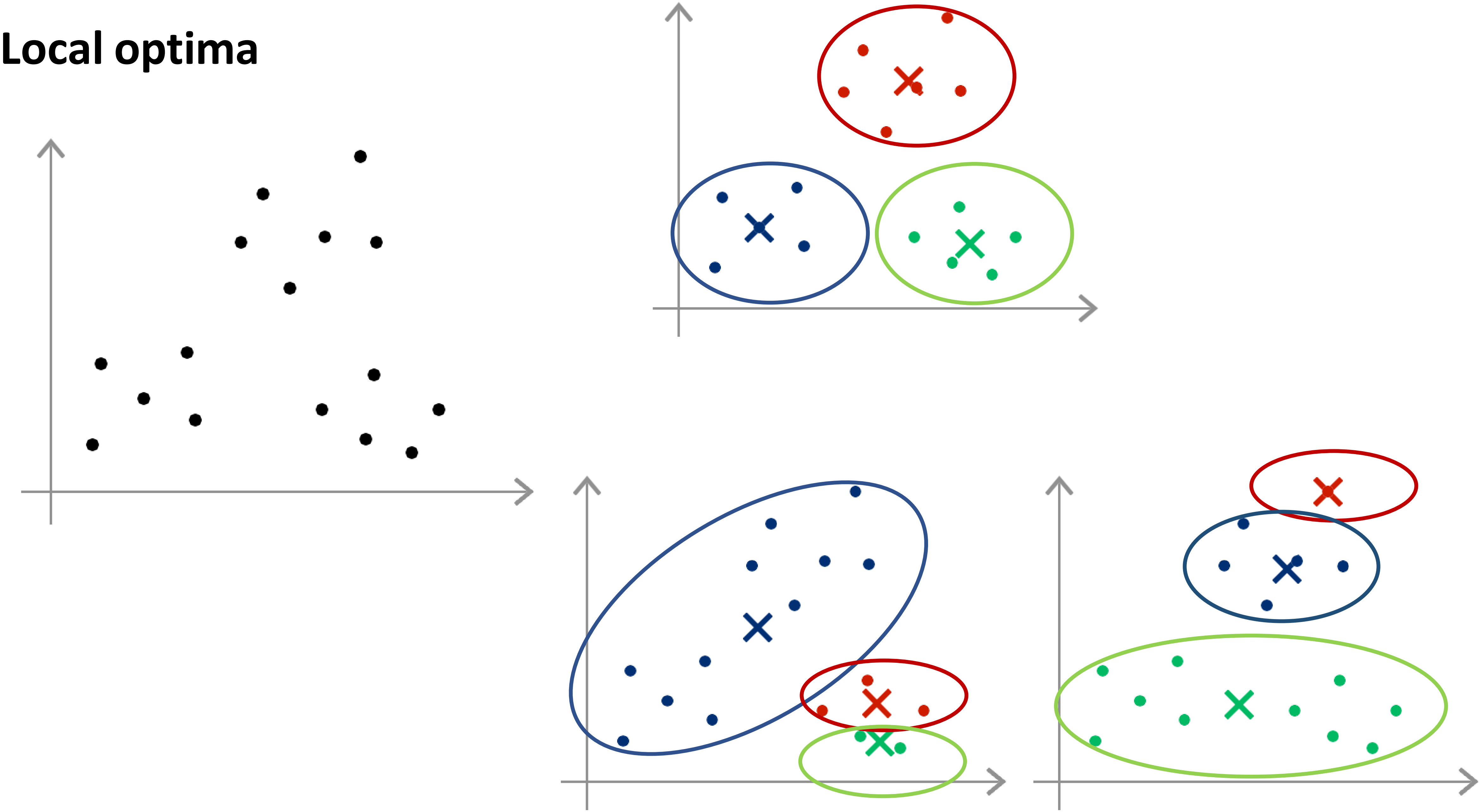
Should have $K < m$

Randomly pick $\underline{K}$ training examples.

Set $\mu_1, \ldots, \mu_K$ equal to these K examples.



Local optima



Random Initialization

For i = 1 to 100 {

Randomly initialize K--means.

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.

Compute cost function (distortion)

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

}

Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

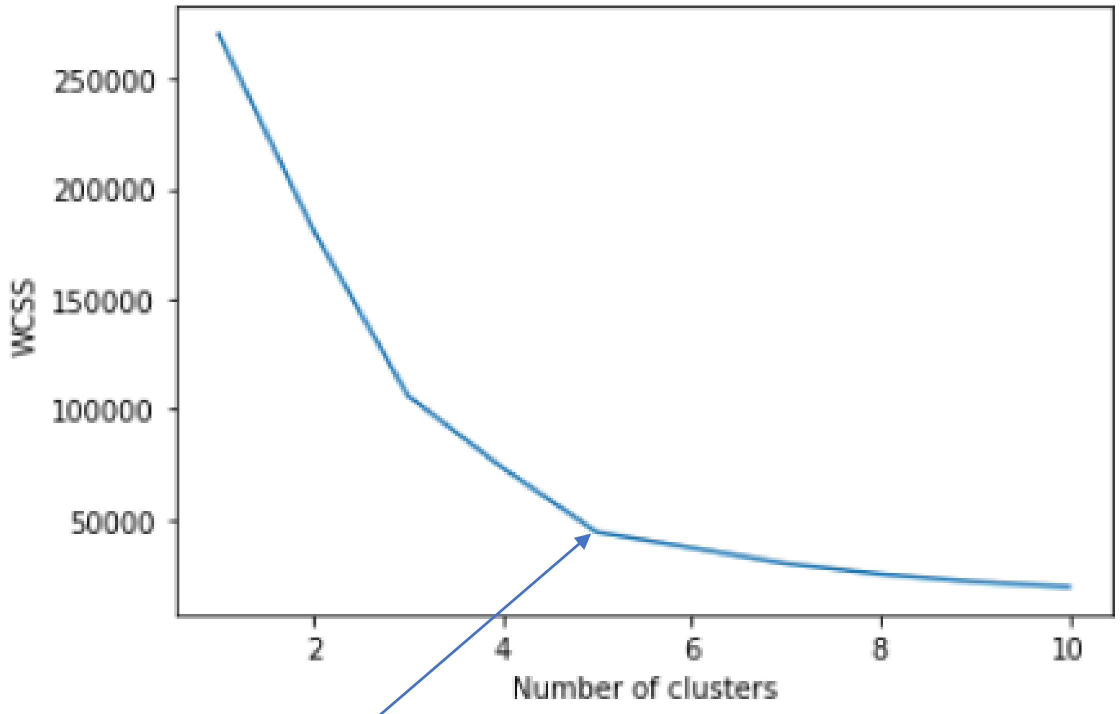
Choosing K (number of clusters)

Choosing the value of K

- Heuristic: find the "**elbow**" of the **within-sum-of-squares**(wss) plot as a function of K, here K=5.

$$\text{wss} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} |x_{ij} - c_i|^2$$

k: # of clusters
n_i: # points in i^{th} cluster
c_i: centroid of i^{th} cluster
x_{ij}: j^{th} point of i^{th} cluster

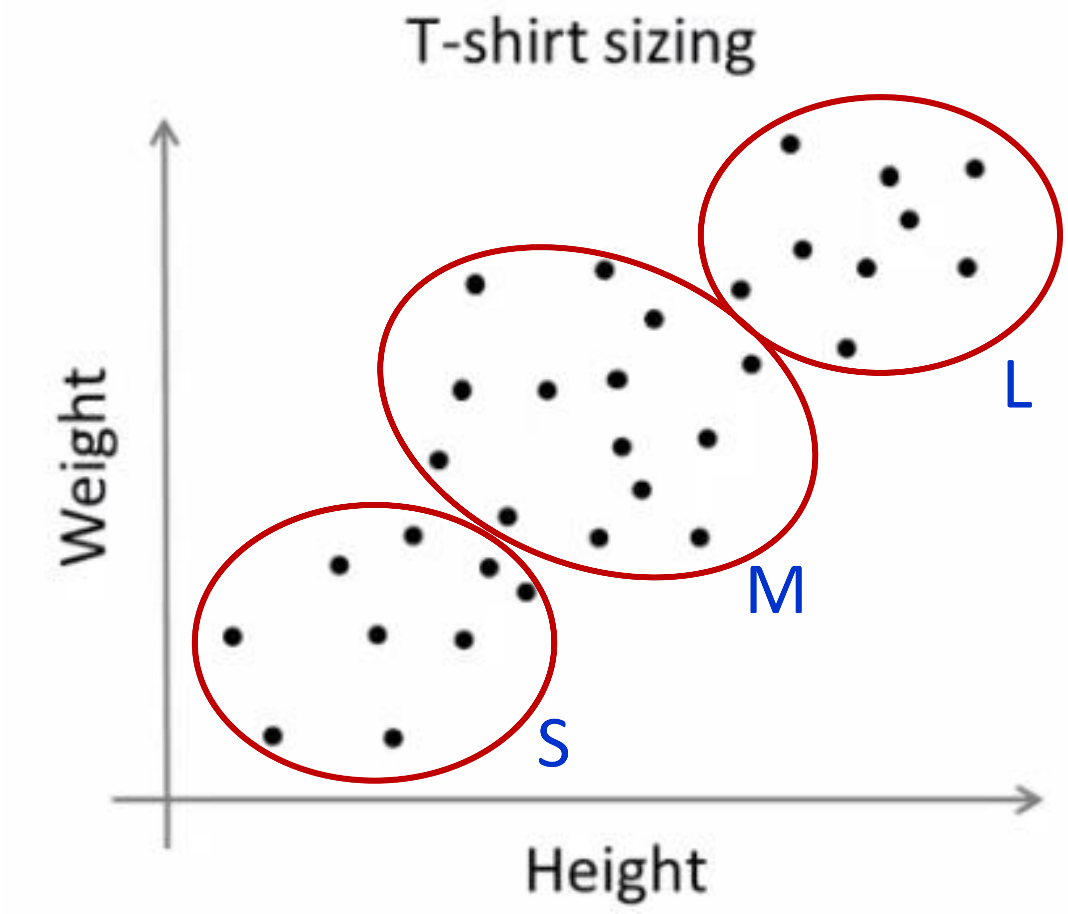


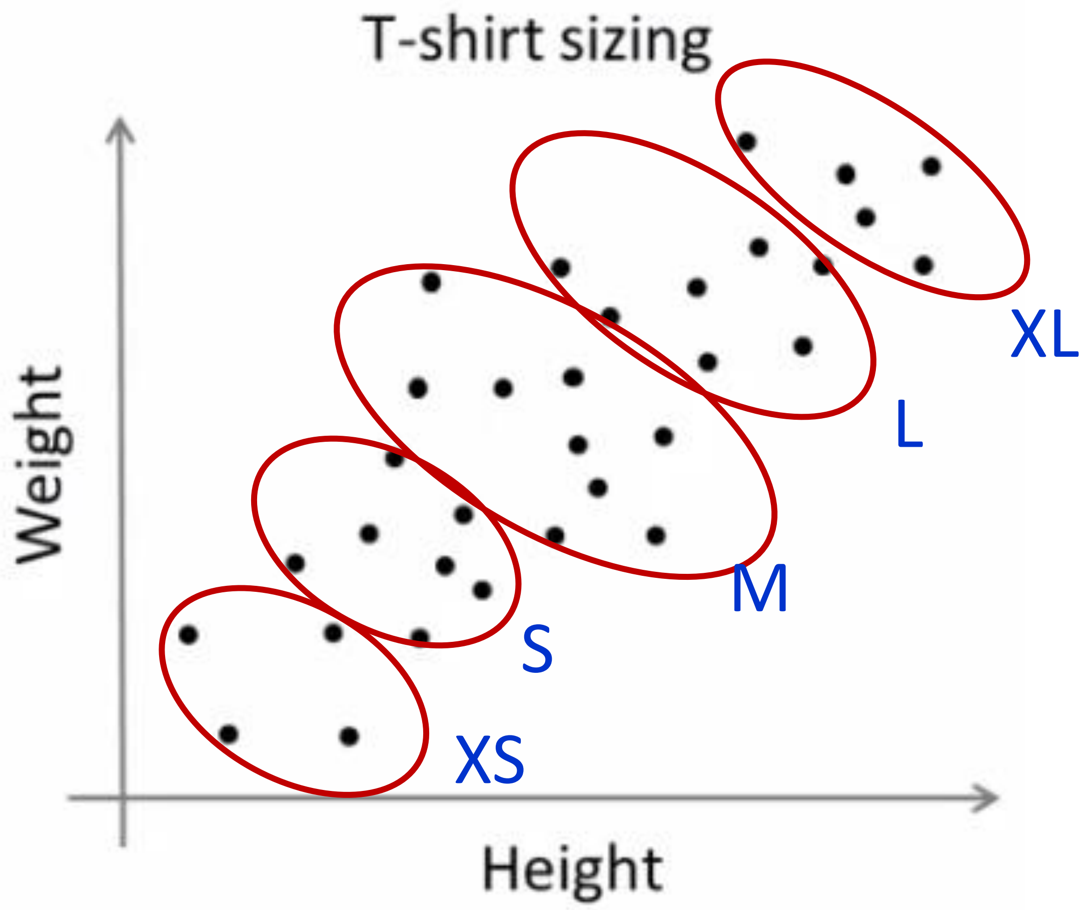
elbow

Choosing the value of K

- Sometimes, you're running K–means to get clusters to use for some later/downstream purpose. Evaluate K–means based on a metric for **how well it performs for that later purpose**.
- Example: T-shirt sizing
  - You need 3 sizes (S,M,L) or 5 (XS, S, M, L ,XL)

K-means based on T-shirt Business





Thanks