Learning II

Lusi Li
Department of Computer Science
ODU

Reading for This Class:
Not in the Book

Review

- Last Class
  - Learning
  - Decision Tree

- This Class
  - Nearest Neighbors Classification
  - Unsupervised Learning
  - K-means

- Next Class
  - Final Review

Basic Algorithm of Decision Tree

node = root of decision tree

Main loop:

1. A ← the "best" decision attribute for the next node.

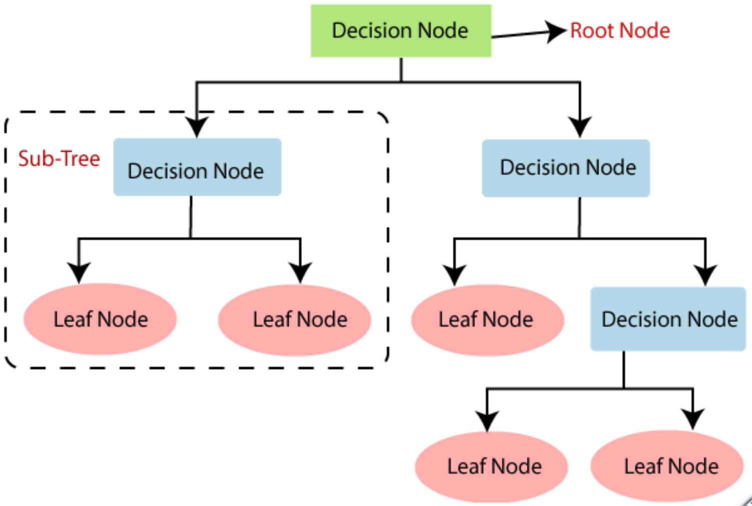
2. Assign A as decision attribute for node.

3. For each value of A, create a new child (sub-tree) of the node.

4. Sort training examples to leaf nodes.

5. If training examples are perfectly classified, stop.

Else, recurse over new leaf nodes.



Disadvantages of Decision-Tree Learning

- **Large-Scale Information**
  - A big decision tree
    - Not efficient
- **Contradictory information**
  - Fail to build a decision tree
  - Decision tree is not robust to contradictory or erroneous information
  - Hard to handle noisy information
- **Missing Information**
  - Not all the attributed values are known in some given examples
  - Hard to classify
- **Adaptability**
  - Learning Decision Tree may not be useful in a changing environment
- **Real Time Response**
  - If the decision tree is large, response time may be long

Outline

- Nearest Neighbor Classification
- Intro to unsupervised learning
- K-means algorithm
- Optimization objective
- Initialization and the number of clusters

Supervised learning

- **Input**: training set (input-output pairs):
  - $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
  - where each pair was generated by an unknown function f

$$f(x^{(i)}) = y^{(i)}, 1 \leq i \leq m$$

- **Goal**: find a hypothesis function h that approximates the true function f

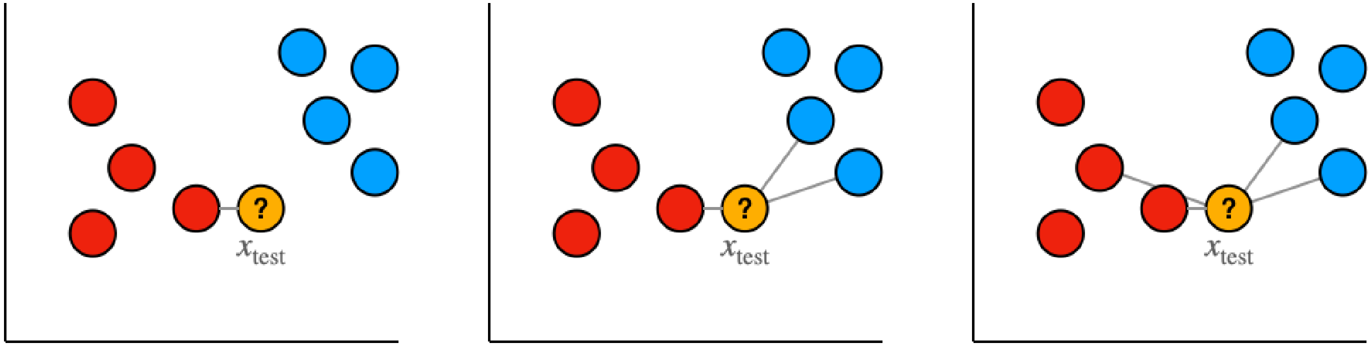
$$h(x^{(i)}) \approx \; y^{(i)}, 1 \leq i \leq m$$

What we know so far

- Decision Trees: how to induce a decision tree from training data
- Other methods?

KNN

- K-Nearest Neighbor (KNN) Classifier
  - Organize and store all training examples
  - Classify new examples based on “most similar” training examples
    - Compute distance to other training examples
    - Identify k nearest neighbors
    - Use class labels of KNN to determine the class label by taking majority vote



$k = 1$

Nearest point is red, so x_{test} classified as red

$k = 3$

Nearest points are {red, blue, blue} so x_{test} classified as blue

$k = 4$

Nearest points are {red, red, blue, blue} so classification of x_{test} is not properly defined

Supervised learning

- Two ways to think about learning

Eager learning

(e.g., decision trees)

- Learn/Train
  - Induce an **abstract model** from data
- Test/Predict/Classify
  - Apply learned model to new data

Lazy learning

(e.g., nearest neighbors)

- Learn
  - **Just store data** in memory
- Test/Predict/Classify
  - Compare new data to stored data
- Properties
  - Retains all information seen in training
  - Complex hypothesis space
  - Classification can be very slow

KNN algorithm

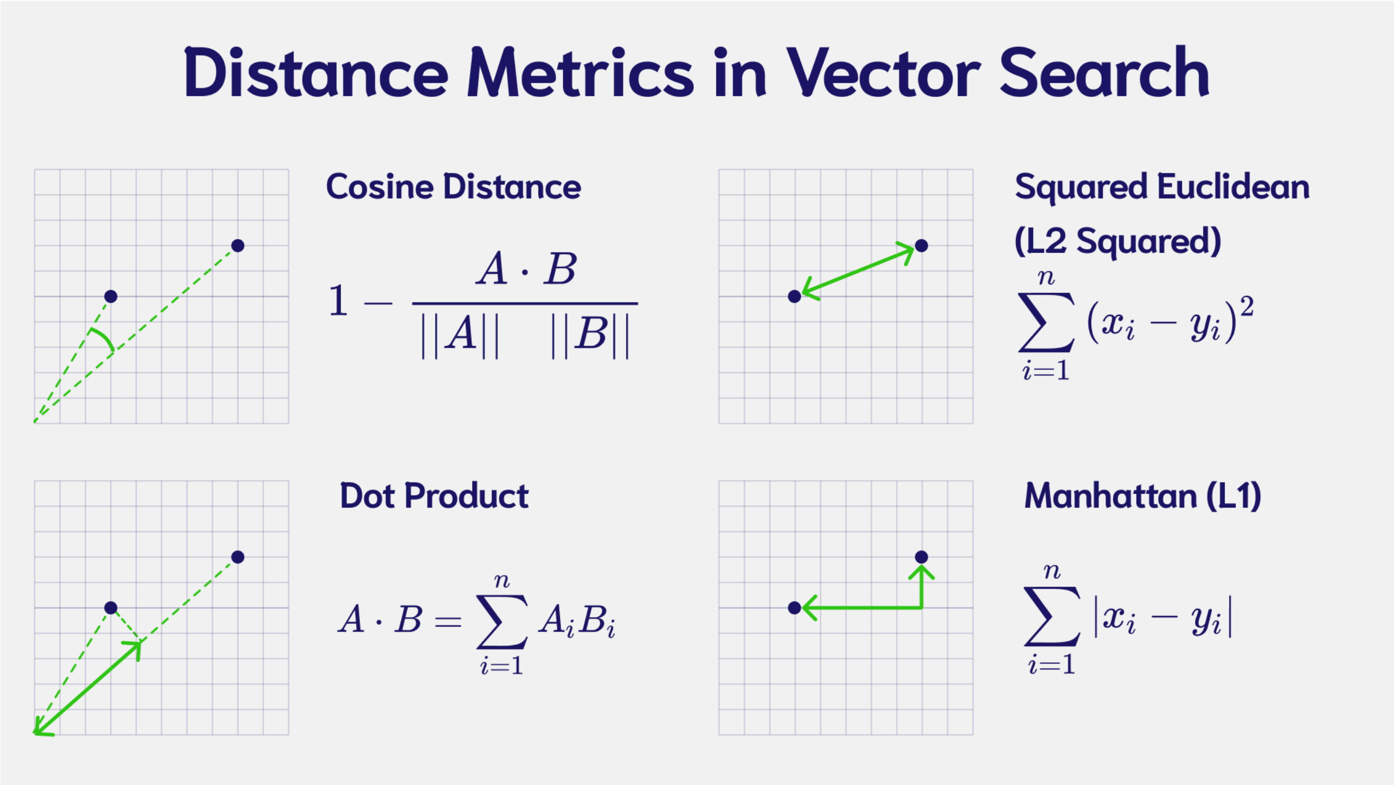
Components of a k-NN Classifier

- **Distance metric**
  - How do we measure distance between instances?
  - Determines the layout of the example space

- **The k hyper-parameter**
  - How large a neighborhood should we consider?
  - Determines the complexity of the hypothesis space

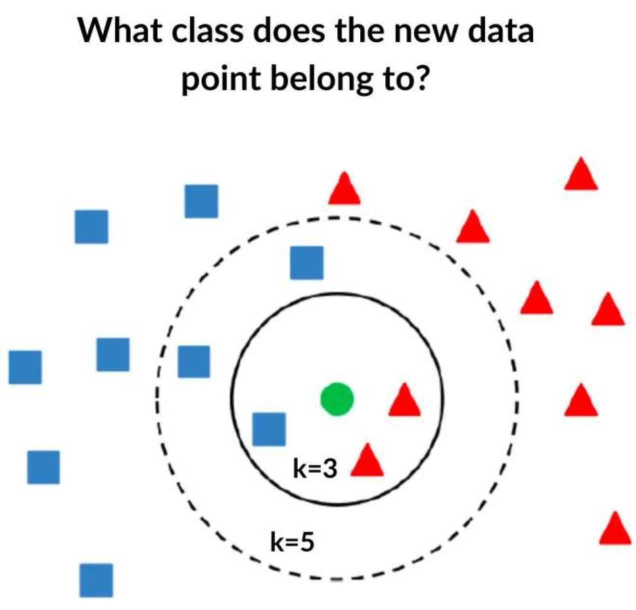
KNN algorithm

- **Distance metric**
  - Any distance function can select nearest neighbors
  - Different distances yield different neighborhoods



KNN algorithm

- **The k hyper-parameter**
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points



How would you set k in practice?

- **Weighted voting**

○ Default:

all neighbors have equal weight

○ Extension:

weight neighbors by (inverse) distance

Unsupervised learning

- **Input**: training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \cdots, x^{(m)}\}$
- **Clustering goal**: automatically partition examples into groups of similar examples

- Why? It is useful for
  - Automatically organizing data
  - Understanding hidden structure in data
  - Preprocessing for further analysis

K-means algorithm

- Input:
  - $\boldsymbol{K}$ (number of clusters)
  - Training set $\{x^{(1)}, x^{(2)}, x^{(3)}, \cdots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$

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

Repeat{

for i = 1 to m

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

Cluster assignment step

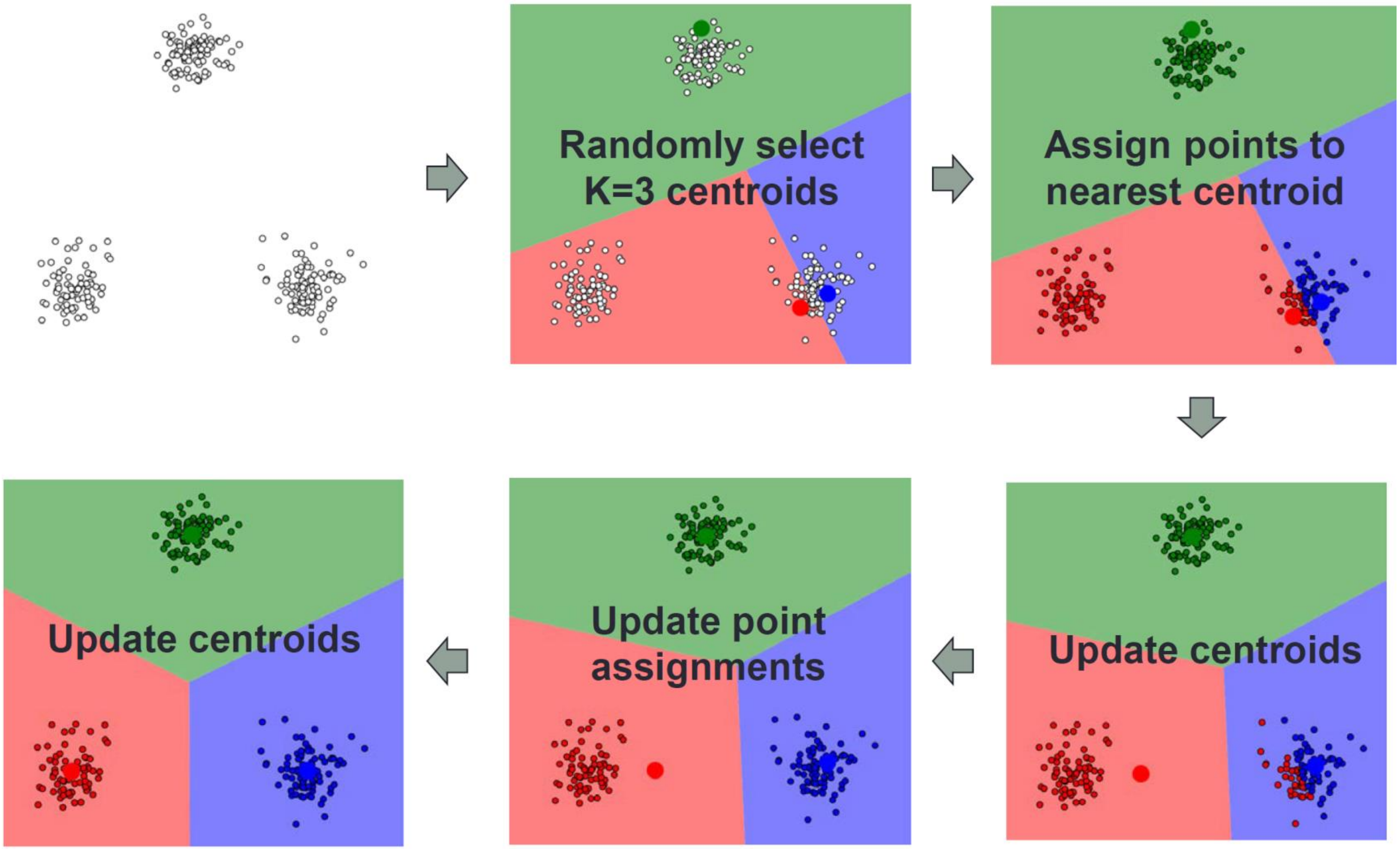
for k = 1 to K

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

Centroid update step

}

K-means algorithm



K-Means example

From https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

K-means optimization objective

- $c^{(i)}$ = Index of cluster (1, 2, … K) to which example $x^{(i)}$ is currently assigned
- μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)
- $\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Example:
For $x^{(i)} \in \mathbb{R}^n$
$c^{(i)} = 2$
$\mu_{c^{(i)}} = \mu_2$

- Optimization objective:

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

$$\min_{\substack{c^{(1)},\cdots,c^{(m)} \\ \mu_1,\cdots,\mu_K}} J\left(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K\right)$$

K-means algorithm

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

Cluster assignment step

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

Repeat{

for i = 1 to m

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

Centroid update step

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

for k = 1 to K

$\mu_k :=$ average (mean) of points assigned to cluster k

}

K-means algorithm

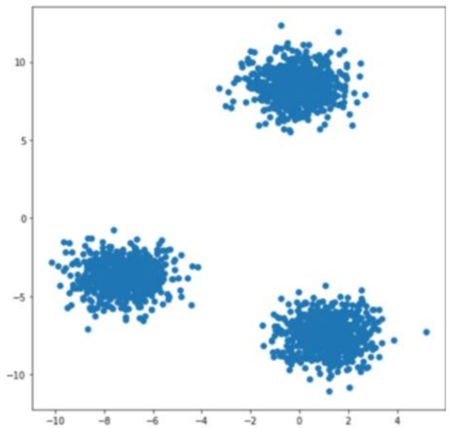
Components of K-means

- **Distance metric**
  - How do we measure distance between instances?
  - Determines the cluster assignment
- **The K hyper-parameter**
  - Needs to be set in advance (as prior knowledge)
- **Cluster-centroid initialization**
  - Different initializations yield different results

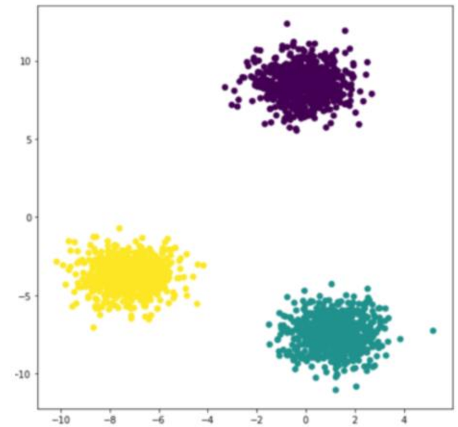
How to choose K?

- Try multiple **K** and use performance from downstream task for selection

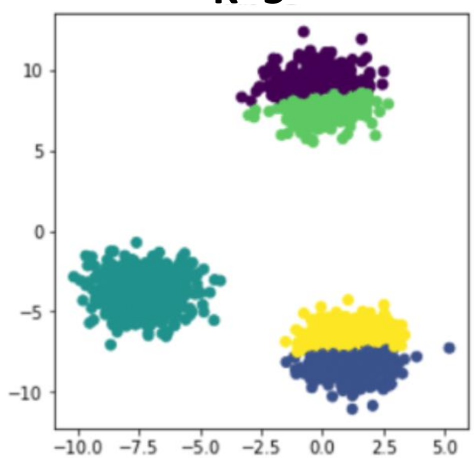
Input



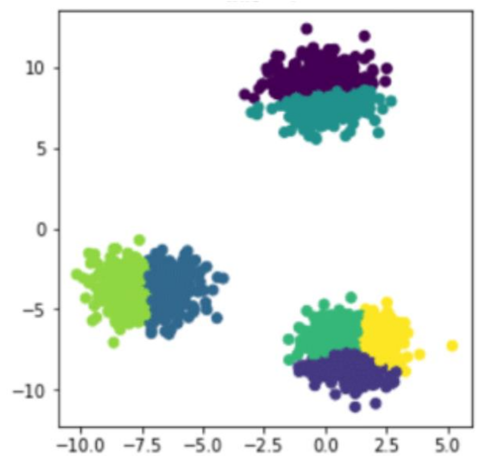
K=3



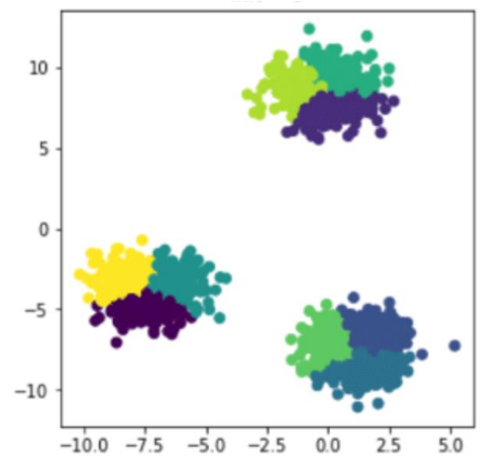
K=5



K=7

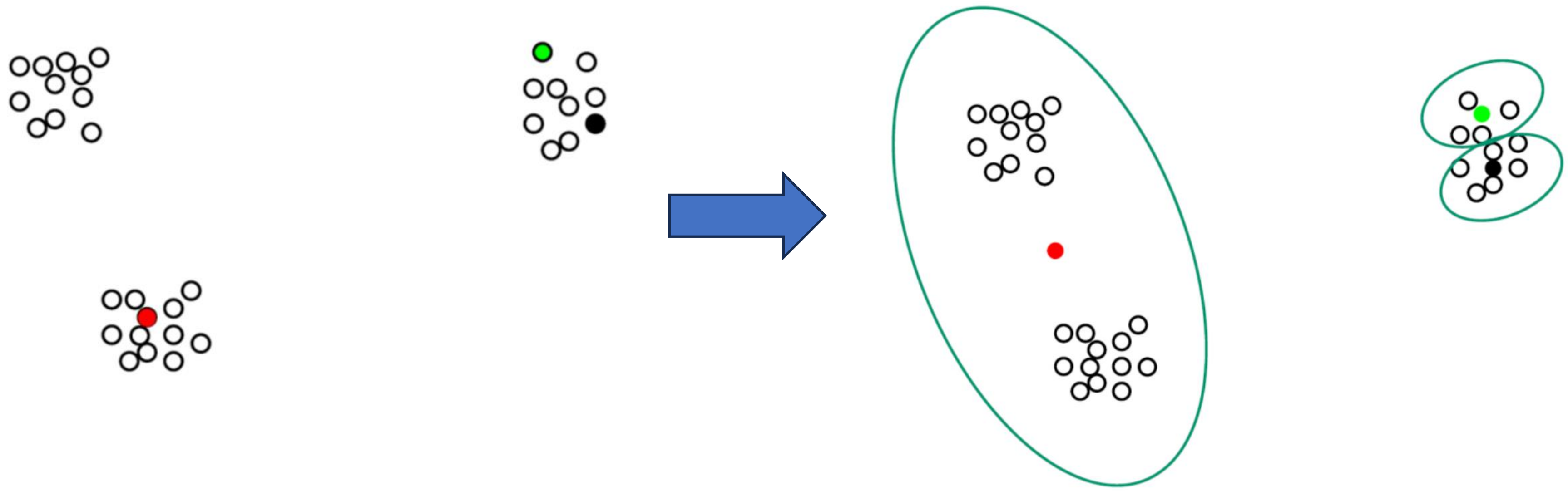


K=9



Impact of initialization

- Randomly pick K training examples
- Set $\mu_1, \mu_2, \cdots, \mu_K$ equal to those K examples



Random initialization

Final clustering results

1) Multiple random initialization

For i = 1 to 100 {

Randomly initialize K-means

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

Compute the cost function (distortion)

$$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K)$$

}

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

2) Furthest point heuristic

Choose μ_1 arbitrarily (or at random)

For *j* = 2 to K

Pick μ_j among data points $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$ that is farthest from previously chosen $\mu_1, \mu_2, \cdots, \mu_{j-1}$

Slide credit: Maria-Florina Balcan

Things to remember

- Nearest Neighbor Classification
- Intro to unsupervised learning
- K-means algorithm
- Optimization objective
- Initialization and the number of clusters

What I want you to do

- **Work on your assignments 3 and 4**