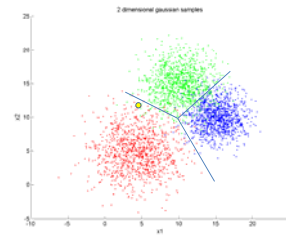


## K-means Clustering

Chen Yu  
Indiana University

## Three types of learning

**Supervised learning:** given training examples of inputs and corresponding outputs  $(x_1, y_1)$ ,  $(x_2, y_2)$ , ...,  $(x_n, y_n)$ , produce the "correct" outputs for new inputs.



## How KNN works?

- The general principle is to find the  $k$  training samples to determine the  **$k$ -nearest neighbors** based on a distance measure. Next, the **majority** of those  $k$  nearest neighbors decide the category of the next instance.

## K nearest neighbors

- Step 1: determine  **$k$**
- Step 2: calculate the **distances** between the new input and **all** the training data
- Step 3: **sort** the distance and determine  $k$  nearest neighbors based on the  $k$ -th minimum distance.
- Step 4: gather the **categories** of those neighbors.
- Step 5: determine the category based on **majority vote**.

### Example

2-dimensional data:

data	label
(3,3)	r
(-1,-4)	b
(2, 3)	r
(0,-5)	b

Now given a new instance (3,4), which category is this new data assigned to?

### Example

- Step 1:  $k = 3$
- Step 2: **distance**

(3,3)	1
(-1,-4) to (3,4)	8.9
(2, 3)	1.4
(0, -5)	9.4

### Step 3: ranking

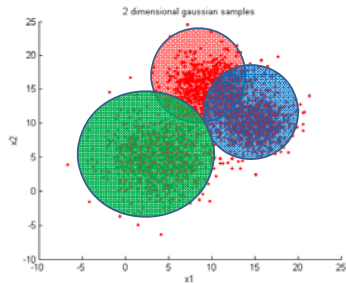
	rank	neighbors
(3,3)	1	Y
(2,3)	2	Y
(-1,-4)	3	Y
(0,-5)	4	N

### Step 4&5: categorization

	rank	neighbors	category
(3,3)	1	Y	r
(2,3)	2	Y	r
(-1,-4)	3	Y	b
(0,-5)	4	N	b

## Three types of learning

**Unsupervised learning:** given only inputs, find structure and regularities in the data. E.g. How to segment images into meaningful regions.



## K-Means

- The algorithm can **group** your data into k number of categories.
- The principle is to minimize the sum of squares of distances between data and the corresponding cluster centroids.

## Example

(3,3)  
(-1,-4)  
(2,3)  
(0,-5)

We know that those data belong to two clusters( $k=2$ ).  
The question is how to determine which data points belong to cluster 1 and which ones belong to cluster 2.

## K-Means Algorithm

- Repeat the following three steps until convergence (stable):
- Step 1: determine the centroid coordinates
  - Step 2: determine the distances of each data point to the centroids.
  - Step 3: group the data points based on minimum distance.

## Example

Initialize the first two centroids

$c1=(3,3)$   $c2=(2,3)$

(3, 3)

(-1,-4)

(2, 3)

(0,-5)

## Measuring distances

- Calculate the distance between a centroid and a data point.
- Euclidean distance

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

## Iteration 1: calculate distances

	$c1=(3,3)$	$c2=(2,3)$
(3, 3)	0	1
(-1,-4)	8.1	7.6
(2, 3)	1	0
(0,-5)	8.5	8.2

## Iteration 1: assign clusters

	$c1=(3,3)$	$c2=(2,3)$
(3, 3)	0	1
(-1,-4)	8.1	7.6
(2, 3)	1	0
(0,-5)	8.5	8.2

Iteration 1: compute new centroids

	$c1=(3,3)$	$c2=(0.3,-2)$
$(3,3)$	0	1
$(-1,-4)$	8.1	7.6
$(2,3)$	1	0
$(0,-5)$	8.5	8.2

Iteration 2: calculate distances with new centroids

	$c1=(3,3)$	$c2=(0.3,-2)$
$(3,3)$	0	5.7
$(-1,-4)$	8.1	2.4
$(2,3)$	1	5.3
$(0,-5)$	8.5	3.0

Iteration 2: assign clusters

	$c1=(3,3)$	$c2=(0.3,-2)$
$(3,3)$	0	5.7
$(-1,-4)$	8.1	2.4
$(2,3)$	1	5.3
$(0,-5)$	8.5	3.0

Iteration 2: compute new centroids

	$c1=(2.5,3)$	$c2=(-0.5,-4.5)$
$(3,3)$	0	5.7
$(-1,-4)$	8.1	2.4
$(2,3)$	1	5.3
$(0,-5)$	8.5	3.0

### Iteration 3

	$c1=(2.5,3)$	$c2=(-0.5,-4.5)$
(3, 3)	0.5	8.2
(-1,-4)	7.8	0.7
(2, 3)	0.5	7.9
(0,-5)	8.3	0.7

The new centroids will be the same!

### K-Means Algorithm

Repeat the following three steps until convergence (stable):

Step 1: determine the centroid coordinates

Step 2: determine the distances of each data point to the centroids.

Step 3: group the data points based on minimum distance.

### Mathematical Description

- A data set of N observations  $\{x_1, \dots, x_n\}$
- Each observation is a D-dimensional  $x_i = \{x_i^1, \dots, x_i^D\}$
- Our goal is to partition the data set into some number K of clusters.
- The standard is that the distances of the data points within a cluster are small compared with the distances to points outside of the cluster.

### K clusters

- The centroids of the clusters are a set of D-dimensional vectors  $\{\mu_1, \dots, \mu_k\}$  where  $k = 1, \dots, K$  and  $\mu_k$  is a prototype of the kth cluster.
- Our goal is to find an **assignment** of data points to clusters, as well as a **set of vectors**  $\{\mu_k\}$ , such that the sum of the squares of the distances of each data point to its center is a minimum.

### Assignment of Data Points to Clusters

- For each data point  $x_n$ , we introduce a set of binary indicator variables  $r_{nk} \in \{0,1\}$ , where  $k = 1, \dots, K$  describing which of the  $k$  clusters the data point  $x_n$  is assigned to:

$$r_{nj} = \begin{cases} 1; & \text{if } x_n \text{ is assigned to the cluster } k \\ 0; & j \neq k \end{cases}$$

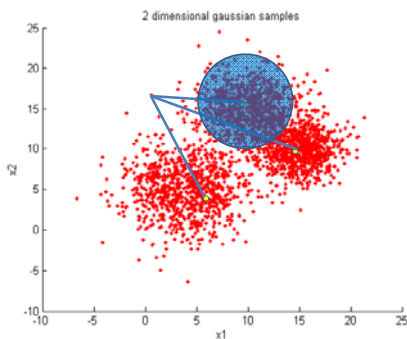
### Objective function

- With the introduction of  $u_k$  and  $r_{nk}$ , we can define our objective as:

$$\sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - u_k\|^2$$

- The sum of squares of the distances of each data points to its assigned vector (center).
- Our goal now is to find values for  $\{r_{nk}\}$  and  $\{u_k\}$

### K-Means Clustering



### A Chicken-and-Egg Problem

- If we know  $\{r_{nk}\}$ , thus, if we know which cluster each data point is assigned to (supervised learning, etc.), then we can easily compute  $u_k$ .

$$u_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

- The denominator is equal to the number of points assigned to cluster  $k$ .
- The mean of all of the data points assigned to that cluster.

### A Chicken-and-Egg Problem

- If we know  $\mu_k$ , then we can easily compute the assignment of each data point based on the distance of this data point to all the centers.

$$r_{nk} = \begin{cases} 1; & \text{if } k = \arg \min_j \|x_n - u_j\|^2 \\ 0; & \text{otherwise.} \end{cases}$$

- Assign the data point to the closest cluster centroid.

### An Iterative procedure

- Each iteration involves two successive steps
- We first choose initial values for  $\mu_k$ .
- Step 1 **data assignment**: we keep  $\mu_k$  fixed, and compute the assignment  $\{r_{nk}\}$
- Step 2 **relocation of centroids**: we keep  $\{r_{nk}\}$  fixed, and compute the centers of clusters.
- Repeat step 1 and step 2 until convergence when the assignments no longer change.

### Hill climbing or greedy descent

- An algorithm starts with a random (potentially poor) solution, and iteratively makes small changes to the solution, each time improving it a little. When the algorithm cannot see any improvement anymore, it terminates.
- Ideally, at that point the current solution is close to optimal, but it is not guaranteed that hill climbing will ever come close to the optimal solution.

### Summary

- The k-means algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters,  $k$ .
- The greedy-descent nature of k-means implies that the convergence is only to a **local** optimal and indeed the algorithm is typically quite sensitive to the initial centroid locations.
- This problem can be counted to some extent by running the algorithm multiple times with different initial centroids.

### Summary-cont.

- The cost of the optimal solution decreases with increasing k till it hits zero when the number of clusters equals the number of distinct data points. One need to find a suitable stopping criterion.
- The algorithm is also sensitive to the presence of outliers. A preprocessing step to remove outliers can be helpful.
- Despite its drawbacks, k-means remains the most widely used partition clustering algorithm in practice. The algorithm is simple, easily understandable and reasonably scalable.

### Image Segmentation

- The goal of segmentation is to partition an image into regions each of which has a reasonably homogeneous visual appearance.
- Each pixel in an image is a point in a 3-dimensional space and we can treat each pixel in the image as a separate data point.

### Image Segmentation



90	87	88	92	96	106	108	109	107	104	101
90	88	89	93	97	106	108	109	107	104	101
91	89	90	95	99	105	107	108	106	103	100
92	90	92	97	101	104	106	107	105	102	99
93	91	93	98	102	103	105	106	105	101	99
94	94	95	98	100	104	103	103	102	101	101
94	94	95	97	99	104	104	104	104	103	101
94	94	95	97	98	104	105	106	106	104	101
94	94	95	96	98	103	104	106	107	105	100
95	95	96	96	97	101	103	105	105	103	99
96	96	96	96	96	99	100	102	102	100	97