Unsupervised learning

K-means Algorithm

Cluster analysis is fundamental in data science , notably in deep learning , it is used to extract models out of a given dataset , a model is a group of data that share certain attributes , k-means algorithm helps us achieve this .

How does it work ?

let's suppose we have a data set of numeric values , denoted by

$D = \{a_1, a_2, a_3, ..., a_n\}$ where $a_i(x_{i1}, x_{i2}, ..., x_{ip})$ is a data point with its cordinates x_{ij} in p-dimensions space vector , $1 \leq i \leq n$, $1 \leq j \leq p$

The goal is to build data models by clustering our data set into k partitions . each partition must have a centroid . let $P_k(c_k, D_k = \{\emptyset\})$ our k-th partition , c_k and D_k are respectively the centroid and dataset (which is initially empty) .
First of all , we need to select k centroids for our models , the k-th partition will include the datapoints that minimize its distances with the k-th centroid . to be more accurate , the datapoints will search for the closest centroid and join its partition .
once we get our models , we need to calculate the new position of our centroids , since the initial positions might not lead to a proper and correct clustering . then we iterate with the same process until our partitions become stable . stability occurs when an iteration keeps the previous partitions unchanged .

We are going to need the following mathematical operators :

Euclidian distance : $d(a_i, a_j) = \sqrt{\sum_{k=1}^{p} (x_{jk} - x_{ik})^2}$

Mean value : $avg(a_1, a_2, ..., a_n) = \frac{1}{n} \sum_{k=1}^{n} a_k$

k-means Algorithm

Environment Variables :

n : dataset size

k : integer , //number of clusters $k < n$

$D = \{a_1, a_2, a_3, ..., a_n\}$, // denotes our dataset

$a_i(x_{i1}, x_{i2}, x_{i3}, ..., x_{ip})$ $p \geq 1$

$P_1, P_2, P_3, ..., P_k$, // our k-clusters to initialize, centroids could be chosen randomly but it won't give the same clusters every time, for efficiency, we use an optimization technique (kmeans++) explained below.

Initialization / pre-conditions :

for each $1 \leq j \leq k$

$P_j = (c_j, \{\emptyset\})$ where c_j is the center of gravity (can be considered as user input for now)

Start :

let isStable = false
WHILE (isStable = false) DO

BEGIN loop
for each a_i where $1 \leq i \leq n$
let minValue = $minDistance(a_i, \{c_1, c_2, ..., c_k\})$
let matchingCentroid = $getMatchingPointByDistance(a_i, minValue)$
let matchingPartition = $getPartitionByCentroid(matchingCentroid)$
$addDataPointToCluster(a_i, matchingPartition)$
END loop

isStable = $checkPartitionStability()$

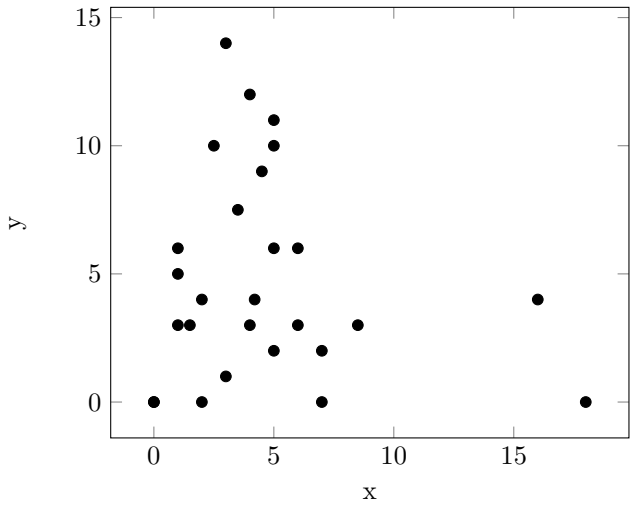
BEGIN loop for each P_j where $1 \leq j \leq k$
$updateCentroid(P_j)$
END loop

END WHILE

Kmeans ++ optimization

the kmeans++ algorithm decides how to pick the initial centroids so that the kmeans algorithm costs less iterations and always produces accurate results. the algorithm will take any point from the dataset as a centroid then calculates the probability that the next point in the dataset will be the next centroid.the highest probabilities will be considered in picking the centroids.

Visualization of the data input



Visualization of the data output