Tecnológico Nacional de México

INSTITUTO TECNOLÓGICO CAMPUS TIJUANA

ING. EN SISTEMAS COMPUTACIONALES

Subdirección Académica

Departamento de Sistemas y Computación

BDD-1704 TI9A - 6:00pm-7:00pm

ASIGNATURA:

Datos Masivos

SEMESTRE:

Septiembre- Enero 2020

Tarea:

Investigar el método Elbow para buscar k

MAESTRO:

JOSE CHRISTIAN ROMERO HERNANDEZ

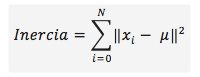
09/12/2020

Uno de los problemas que nos encontramos a la hora de aplicar alguno de los métodos de Clustering (K-means o EM) es la elección del número de Clusters. No existe un criterio objetivo ni ampliamente válido para la elección de un número óptimo de Clusters; pero tenemos que tener en cuenta, que una mala elección de los mismos puede dar lugar a realizar agrupaciones de datos muy heterogéneos (pocos Clusters); o datos, que siendo muy similares unos a otros los agrupemos en Clusters diferentes (muchos Clusters).

Aunque no exista un criterio objetivo para la selección del número de Clusters, si que se han implementado diferentes métodos que nos ayudan a elegir un número apropiado de Clusters para agrupar los datos; como son, el método del codo (elbow method).

**Método del codo (Elbow Method)**

Este método utiliza los valores de la inercia obtenidos tras aplicar el K-means a diferente número de Clusters (desde 1 a N Clusters), siendo la inercia la suma de las distancias al cuadrado de cada objeto del Cluster a su centroide:



Una vez obtenidos los valores de la inercia tras aplicar el K-means de 1 a N Clusters, representamos en una gráfica lineal la inercia respecto del número de Clusters. En esta gráfica se debería de apreciar un cambio brusco en la evolución de la inercia, teniendo la línea representada de una forma similar a la de un brazo y su codo.

El punto en el que se observa ese cambio brusco en la inercia nos dirá el número óptimo de Clusters a seleccionar para ese dataset; o dicho de otra manera: el punto que representaría al codo del brazo será el número óptimo de Clusters para ese data set.

**Implementación**

Para poder ejecutar estos scripts es necesario tener instaladas las librerías de numpy, matplotlib, scipy y scikit-learn. Para descargar e instalar (o actualizar a la última versión con la opción -U) estas librerías; con el sistema de gestión de paquetes pip, se deben ejecutar los siguiente comandos:

pip install -U numpy

pip install -U matplotlib

pip install -U scipy

pip install -U scikit-learn

El script que se muestra a continuación, calcula los valores de la inercia tras aplicar el K-means de 1 a 20 Clusters (para uno de los 3 data sets que tenemos) y los pinta en una gráfica lineal (número de Clusters respecto a la inercia) para poder apreciar **“el codo”** y por tanto determinar el número óptimo de Clusters para el dataset:

# -\*- coding: utf-8 -\*-

\_\_author\_\_ = 'RicardoMoya'

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

# Constant

DATASET1 = "./dataSet/DS\_3Clusters\_999Points.txt"

DATASET2 = "./dataSet/DS2\_3Clusters\_999Points.txt"

DATASET3 = "./dataSet/DS\_5Clusters\_10000Points.txt"

LOOPS = 20

MAX\_ITERATIONS = 10

INITIALIZE\_CLUSTERS = 'k-means++'

CONVERGENCE\_TOLERANCE = 0.001

NUM\_THREADS = 8

def dataset\_to\_list\_points(dir\_dataset):

"""

Read a txt file with a set of points and return a list of objects Point

:param dir\_dataset:

"""

points = list()

with open(dir\_dataset, 'rt') as reader:

for point in reader:

points.append(np.asarray(map(float, point.split("::"))))

return points

def plot\_results(inertials):

x, y = zip(\*[inertia for inertia in inertials])

plt.plot(x, y, 'ro-', markersize=8, lw=2)

plt.grid(True)

plt.xlabel('Num Clusters')

plt.ylabel('Inertia')

plt.show()

def select\_clusters(dataset, loops, max\_iterations, init\_cluster, tolerance,

num\_threads):

# Read data set

points = dataset\_to\_list\_points(dataset)

inertia\_clusters = list()

for i in range(1, loops + 1, 1):

# Object KMeans

kmeans = KMeans(n\_clusters=i, max\_iter=max\_iterations,

init=init\_cluster, tol=tolerance, n\_jobs=num\_threads)

# Calculate Kmeans

kmeans.fit(points)

# Obtain inertia

inertia\_clusters.append([i, kmeans.inertia\_])

plot\_results(inertia\_clusters)

if \_\_name\_\_ == '\_\_main\_\_':

select\_clusters(DATASET1, LOOPS, MAX\_ITERATIONS, INITIALIZE\_CLUSTERS,

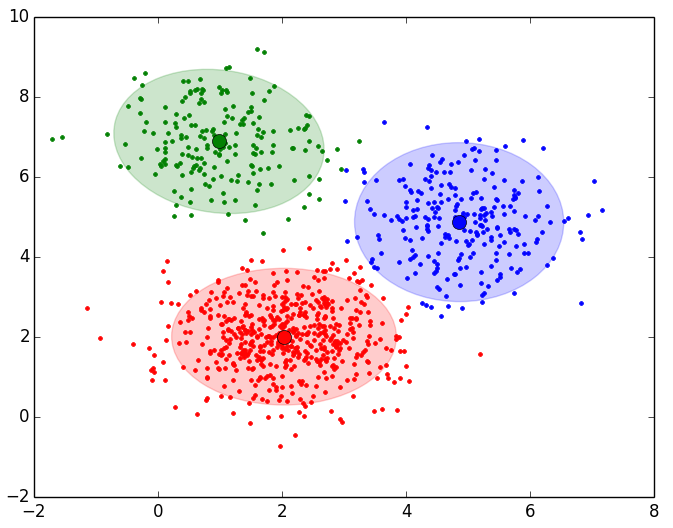
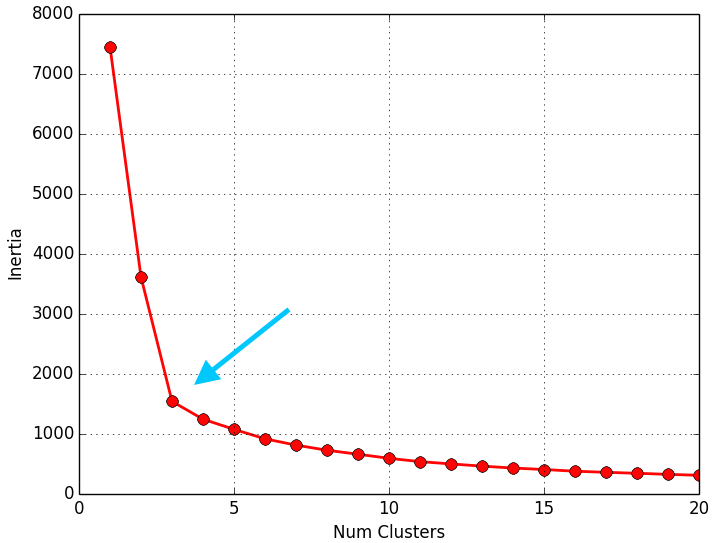
CONVERGENCE\_TOLERANCE, NUM\_THREADS)

A continuación se muestran los resultados obtenidos para cada uno de los tres datasets, El script solo devuelve la gráfica lineal.

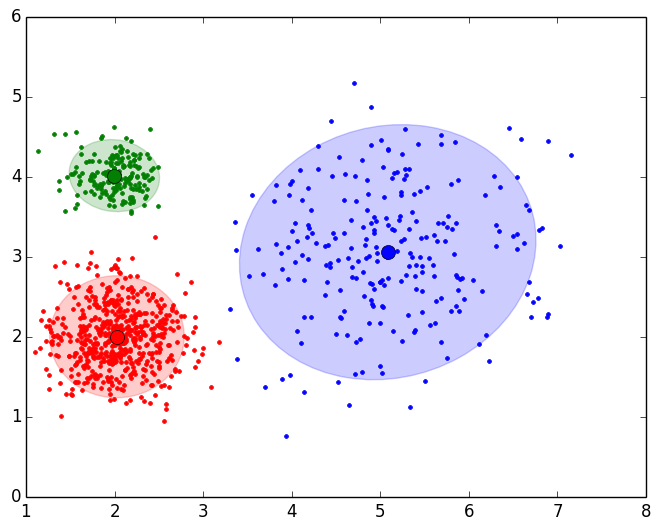
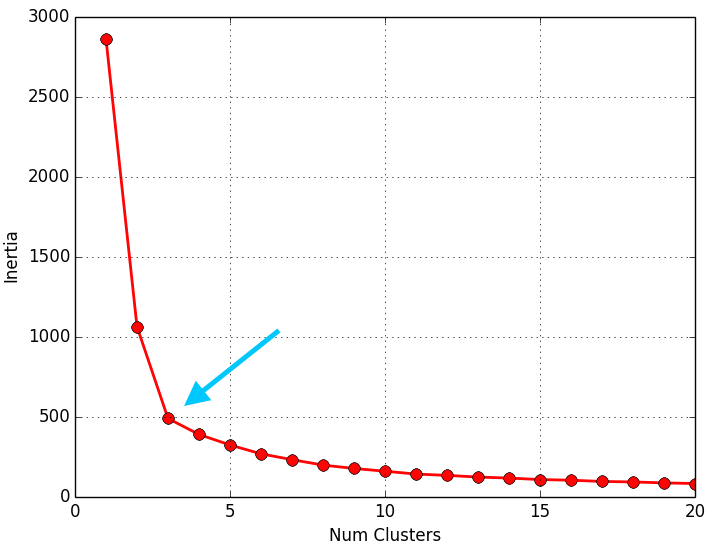
La representación de los Clusters que se muestra al lado de la gráfica lineal se ha obtenido con el [script que implementa el EM](https://jarroba.com/expectation-maximization-python-scikit-learn-ejemplos/) y se muestran para poder apreciar que el número de Clusters que nos indica el método del codo es coherente:

Resultados Gráficos.

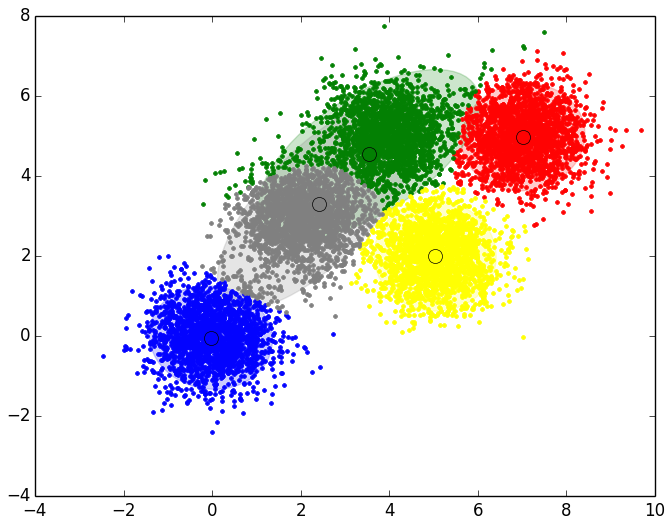
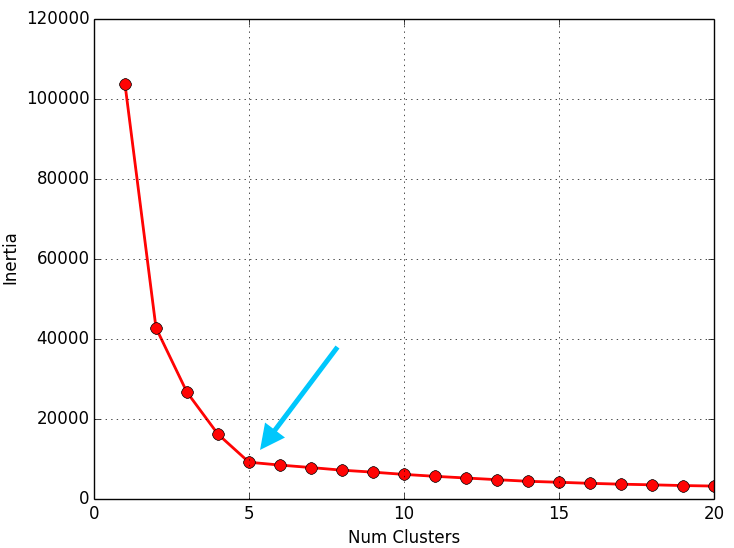
**Primer dataset**



**Segundo dataset**



**Tercer dataset**

****

**Referencias:**

[**https://jarroba.com/seleccion-del-numero-optimo-clusters/**](https://jarroba.com/seleccion-del-numero-optimo-clusters/)