UNIVERSIDAD SIMÓN BOLÍVAR

DEPARTAMENTO DE COMPUTACIÓN Y TECNOLOGÍA DE LA INFORMACIÓN

Inteligencia Artificial 2

CI 5438

Proyecto N° 3

Integrantes:

Mauricio Salerno 12-10627

Douglas Torres 11-11027

Sartenejas, marzo 2017

En el presente proyecto se realizó una implementación del algoritmo K-means en el lenguaje de programación python versión 3.4 para la realización de dos actividades que serán explicadas más adelante. En cuanto a la implementación, se guardan los datos en arreglos (vectores) de numpy y se calculan los centroides, para esto se selecciona un punto inicial y se calculan las distancias entre este y todos los demás puntos. Seguidamente, se calcula la probabilidad de cada punto sea escogido como el siguiente centroide inicial, a mayor distancia, mayor probabilidad de ser escogido. Se selecciona aquel punto cuya distancia sea mayor a un número aleatorio generado para ser agregado al arreglo de centroides, esto se hace hasta obtener los k centroides. Luego, se ejecuta el algoritmo k-means que procede a unir aquellos puntos que se encuentren más cerca de los centroides hasta que los clusters que se hayan creado no cambien en iteraciones consecutivas. Cabe destacar que los puntos pertenecientes a cada cluster se encuentran en un arreglo.

El proyecto se dividió en 2 actividades:

Actividad 2:

a) En esta actividad se usó el dataset de “iris-data.txt”, donde se guardaba cada instancia en un arreglo (vector) de numpy. Seguidamente se pasa a ejecutar K-means para agrupar los puntos en tres clusters, uno para cada clasificación posible (“Iris-Virginica” = 2, “Iris-Setosa” = 1 e “Iris-versicolor” = 0). Se corrió el algoritmo kmeans para K = {2,3,4,5}. En las presentes tablas se muestra la cantidad de puntos pertenecientes a cada clase y su agrupamiento de acuerdo a los K específicados

K = 2

	Cluster 1	Cluster 2
Iris-versicolor	3	47
Iris-Setosa	50	0
Iris-Virginica	0	50

K = 3

	Cluster 1	Cluster 2	Cluster 3
Iris-versicolor	3	0	47
Iris-Setosa	0	50	0
Iris-Virginica	36	0	14

K = 4

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Iris-versicolor	3	47	0	0
Iris-Setosa	0	0	23	27
Iris-Virginica	36	14	0	0

K = 5

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Iris-versicolor	0	0	23	27	0
Iris-Setosa	27	0	0	0	23
Iris-Virginica	0	29	20	1	0

Según estas tablas observamos que la distribución de puntos por clase es más o menos equitativa dependiendo de la cantidad de clusters que se pidan, es decir, con K = 2 vemos que la repartición de los puntos es prácticamente 1/3 de los datos en un cluster (todas las instancias de Iris-Setosa y algunas de Iris-versicolor) y 2/3 restante en el otro. Para K = 3 vemos que la repartición es más equitativa con el tercer cluster teniendo pocos puntos más que el resto. Para K = 4 hay una mayor cantidad de puntos entre los primeros dos clusters(3/5 de los puntos) que los dos últimos (2/5 de los puntos), por último con K = 5 todos los clusters salvo el tercero tienen una cantidad de puntos similares. En general podemos observar que “Iris-versicolor” e “Iris-Virginica” son las que más se parecen entre sí ya que casi siempre aparecen juntas en algún cluster y cuando ocurre, dicho cluster es el que presenta mayor cantidad de puntos.

En cuanto a la diferencia entre el agrupamiento entre K = 2 y K = 3 vemos que el agrupamiento hecho en K = 2, al hacer K = 3, se dividen los puntos del segundo cluster y se reparten en clusters distintos, uno para los de Iris-versicolor y algunos Iris-Virginica y otro al que se unen los 3 puntos de Iris-versicolor restantes con la mayoría de los Iris-Virginica, los puntos pertenecientes a Iris-Setosa quedan completamente separados en su propio cluster, dando a entender que a los puntos pertenecientes a Iris-Setosa son aquellos más disimilares a las otras dos clases y esto se puede constatar al ver su clasificación a lo largo de las distintas pruebas con distintos K.

b) Adjunto se pueden observar las distintas imágenes obtenidas al aplicar K-means a los píxeles de las mismas. Esto se logró al cargar la imagen a usar usando el módulo Image de la librería PIL y así poder obtener el valor de cada píxel en escala de (Red, Green, Blue), es decir, cada píxel era de la forma (0-255, 0-255, 0-255). Estos fueron los puntos a ser usados en el K-means, que arrojó K clusters que correspondían a K colores distintos, con K colores, con K = {2,4,8,16,32,64,128}. Por último, se colocaban dichos puntos, agrupados por clusters, en el contorno de la imágen.

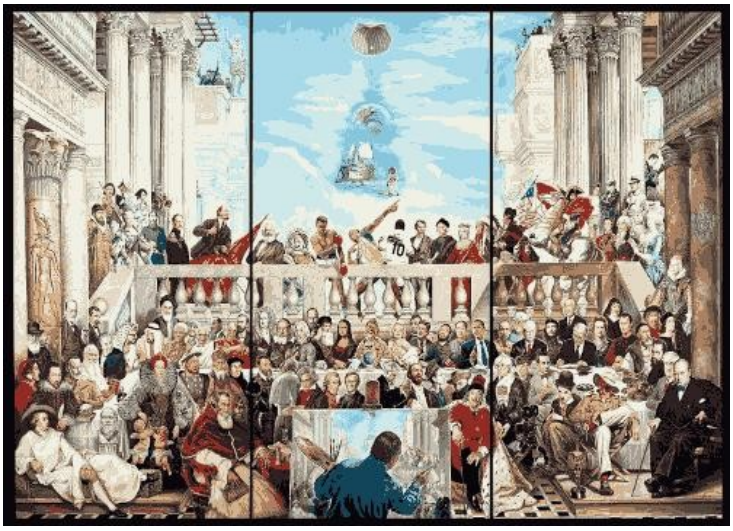
A continuación se muestra la imágen original:

Imágen de 2 colores:

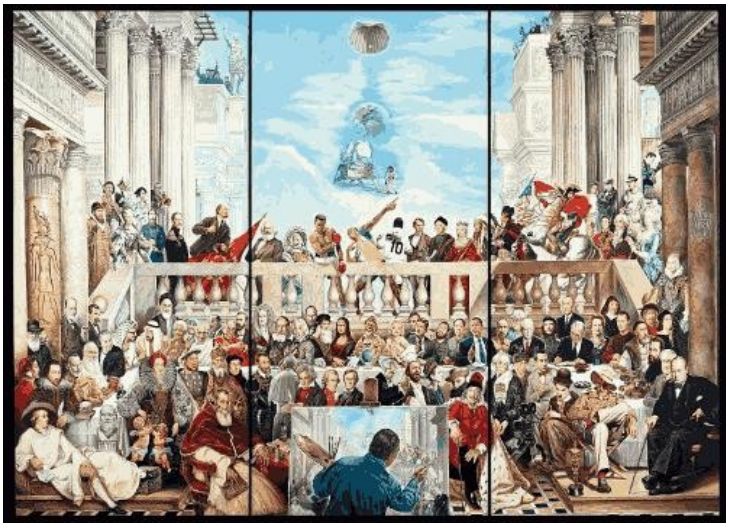
Imágen de 4 colores:

Imágen de 8 colores:

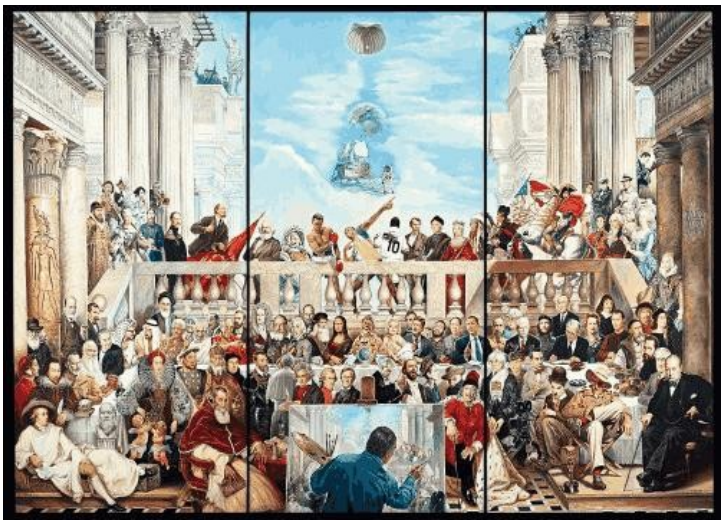
Imágen de 16 colores:



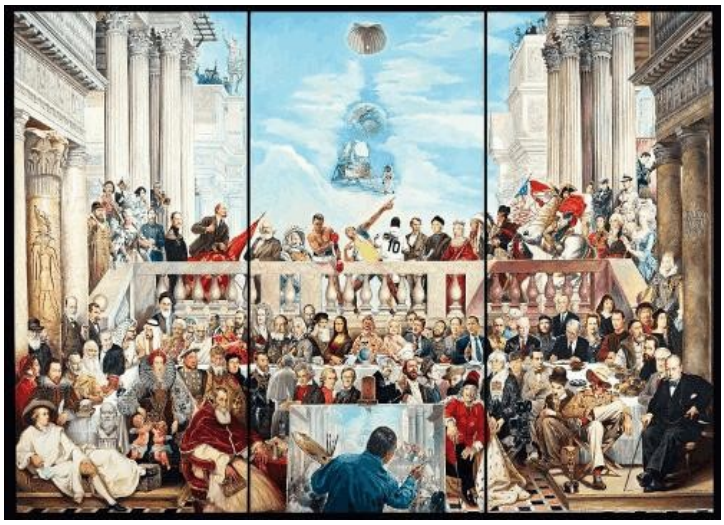
Imágen de 32 colores:



Imágen de 64 colores:



Imágen de 128 colores:



Conclusiones

Podemos concluir que K-means es un algoritmo bastante sencillo y efectivo para el agrupamiento de puntos, pero no es muy eficiente, ya que si se ejecuta con una gran cantidad de puntos tarda mucho tiempo en la ejecución, por lo que puede no ser una opción viable.