**UNIVERSIDAD SIMÓN BOLÍVAR**

Departamento de Ciencia de la Información

CI-5438 - Inteligencia Artificial II

Profesora: Ivette Martinez

# PROYECTO 3

Elaborado Por:

Edward Fernández 10-11121
Carlos Ferreira 11-10323
Stefani Castellanos 11-11394

Sartenejas, Marzo 2017

## PLANTEAMIENTO DEL PROBLEMA

El algoritmo *k-means* es, sin duda alguna, uno de los más usados para realizar agrupamiento (*clustering)* de ciertos datos debido a que es simple, fácil de comprender y de implementar. Consiste en realizar una separación del conjunto de datos en *k* particiones siguiendo algún criterio previamente establecido. Uno de los elementos más importante de dicho algoritmo es escoger adecuadamente el representante del grupo, el centroide, ya que con este se realizarán las comparaciones para determinar qué ejemplo pertenece a cual grupo. Usualmente se toma como métrica la norma euclidiana del vector de ejemplo la cual calcula su proximidad con el centroide y el ejemplo se agrega al grupo cuyo centroide sea más "cercano".

En el siguiente proyecto se realizan dos experimentos utilizando el algoritmo explicado anteriormente. El primero de ellos utiliza el conjunto de datos sobre la flor Iris que contiene tres variedades de la misma. El segundo, se utiliza una imagen para clasificar sus colores y obtener una versión comprimida de la misma según la cantidad de grupos utilizados.

## ACTIVIDADES

1. Implemente el algoritmo de k-means.

La implementación del algoritmo *k-means* se encuentra en el archivo *k_means.py*, el cual hace uso de la librería *numpy* para manejar arreglos de manera eficiente y generar los primeros *k* centroides, ya que se escogen de manera aleatoria. También se hace uso de las funciones "linalg.norm" para calcular la norma euclidiana del vector y "mean" para calcular la media para cada *cluster* y determinar cuál será su nuevo centroide.

Se considera que el algoritmo converge cuando el centroide de la iteración *i-1* es igual al de la iteración *i* o al alcanzar 20 iteraciones, ya que Emre M (2011) experimentalmente observó que en promedio se puede dibujar la imagen con suficiente precisión luego de aproximadamente 12 iteraciones. Esta decisión no afecta el resultado para el primer experimento.

2. Pruebe su algoritmo con:

a) Para explorar los datos del conjunto Iris Data Set (http://archive.ics.uci.edu/ml/datasets/Iris ).

En la siguiente tabla se muestran los resultados con errores totales más bajos porque son considerados los mejores luego de realizar varias corridas de este experimento:

| **# de *clusters*** | **Etiqueta del *cluster*** | **Aciertos** | **Fallos** |
|---|---|---|---|
| 2 | Iris-setosa | 50 | 0 |
| | Iris-versicolor | 50 | 50 |
| | error total | 0.333333333333 | |
| 3 | Iris-versicolor | 40 | 8 |
| | Iris-virginica | 42 | 10 |
| | Iris-setosa | 50 | 0 |
| | error total | 0.12 | |
| 4 | Iris-versicolor | 27 | 2 |
| | Iris-versicolor | 23 | 19 |
| | Iris-virginica | 29 | 0 |

| | | | |
|---|---|---|---|
| | Iris-setosa | 50 | 0 |
| | error total | 0.14 | |
| 5 | Iris-setosa | 50 | 0 |
| | Iris-virginica | 22 | 2 |
| | Iris-virginica | 24 | 0 |
| | Iris-versicolor | 22 | 2 |
| | Iris-versicolor | 26 | 2 |
| | error total | 0.04 | |

Para K = 2 es evidente que no logra diferenciar los tres tipos de Iris ya que solo hay dos grupos, es por esto que el porcentaje de error es alto en comparación con los demás resultados. Para K = 3 ya distingue todos los tipos de flores existentes, sin embargo no logra hacer una clasificación certera, esto se debe a que los tipos Iris-Versicolor e Iris-Virginica no están completamente aisladas una de la otra. Para K > 3 las etiquetas de los grupos comienzan a repetirse, ya que no hay otro grupo posible.

Para cualquier K Iris-Setosa es clasificado con un 100% de aciertos. Esto se debe a que los ejemplos son suficientemente diferentes de las otras 2 clasificaciones como para que el algoritmo los distinga completamente como se observa en la figura 1.

Note que los resultados de estos experimentos podrían variar debido a las inicializaciones aleatorias del algoritmo.

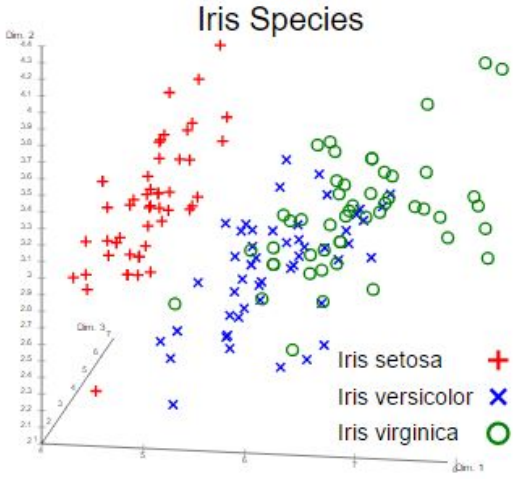


Figura 1. Gráfica aproximada de los tipos de Iris. Fuente: Wikipedia

b) Para realizar la compresión de una imagen de su preferencia. Pruebe para K = {2, 4, 8, 16, 32, 64, 128}. Guarde las imágenes resultantes (y la original) en una carpeta. Si el nombre de la original es Imagen.png , el de las resultantes debe ser ImagenK[NumeroK].png

Para clasificar los colores de la imagen se utiliza la librería *PIL* de Python para la manipulación de la misma. Se obtiene una matriz con los colores (RGB) de los puntos y cada uno de ellos representa un ejemplo. Cuando *k-means* terminar se reconstruye la imagen y se utiliza como color el del centroide, de esta manera todos los ejemplos pertenecientes a un mismo *cluster* tendrán el mismo color.

Inicialmente se decidió utilizar la imagen de "Charlie_Brown.jpg" y *k-means* sin un máximo de iteraciones, resultados se encuentran dentro de la carpeta Charlie_Brown, sin embargo, debido a que al algoritmo le llevaba mucho tiempo converger, no se incluyeron los resultados para K = 128 y se decidió incluir el máximo de iteraciones (20).

Finalmente, se realizó nuevamente el experimento con la cota propuesta y con la imagen de "Charlie_Brown.jpg" nuevamente, estos resultados se encuentran en la carpeta "Charlie_Brown2". Se puede observar que los resultados limitando la cantidad de iteraciones no son muy diferentes de los obtenidos sin límite de iteraciones. Adicionalmente, se realizaron experimentos con otras imágenes:

- "Imagen_up.jpg", una imagen de la película "UP: una aventura de altura" incluidas en la carpeta "Imagen_up".
- "Pokemon.jpg", una imagen de la serie animada "Pokemon" incluidas en la carpeta "Pokemon".
- "Rick_and_Morty.jpg", una imagen de la serie animada "Rick and Morty" incluidas en la carpeta "Rick_and_Morty".

## REFERENCIAS

M. Emre (2011). Improving the Performance of K-Means for Color Quantization.

Russell, S. J., Norvig, P., & Canny, J. (2003). Artificial intelligence: A modern approach.