Práctica 7

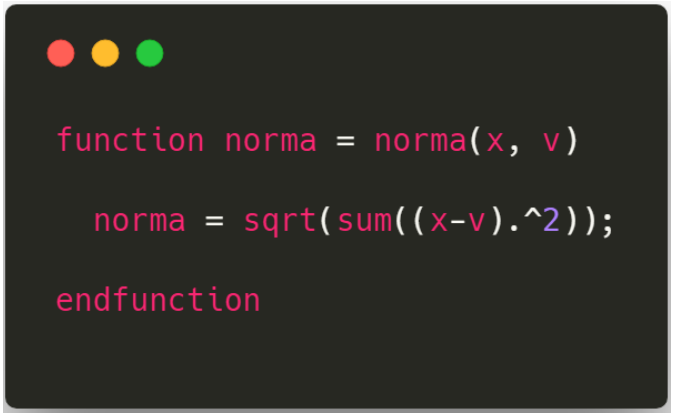
Pablo Arranz Ropero
Juan Alberto Camino Sáez
Grupo 2

Práctica 7: Clustering

En esta práctica implementaremos el algoritmo de clustering k-means y verificaremos su funcionamiento. Después, se aplicará para reducir el tamaño de una imagen disminuyendo el número de colores que utiliza.

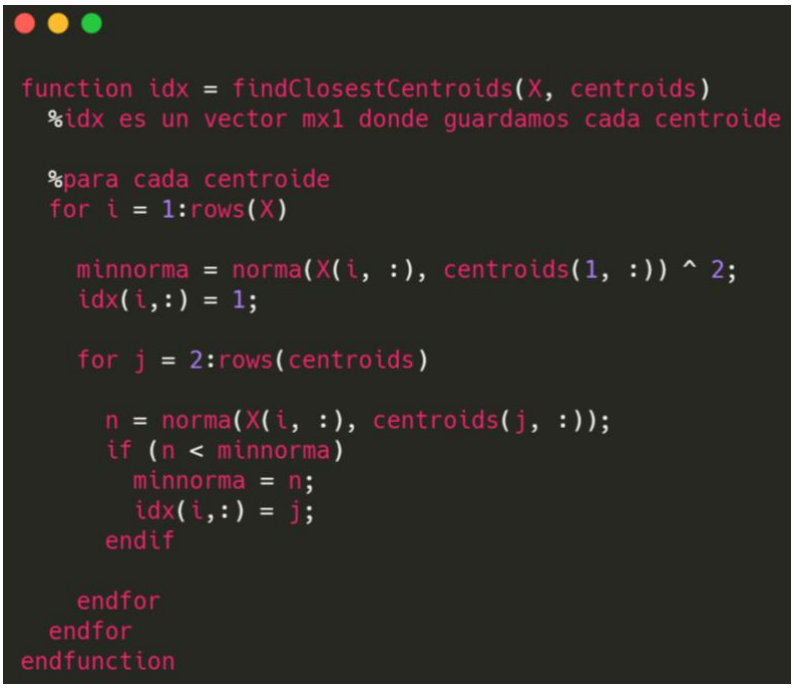
IMPLEMENTACIÓN DE K-MEANS

Para implementar el algoritmo, hemos tenido que realizar dos funciones. La primera, llamada *findClosestCentroids.m*, el cual devuelve el índice del centroide más cercano por cada punto de los datos. Para su realización, nos hemos ayudado de la función *norma.m*, que calcula la norma entre un centroide y un punto de los datos. Este es el código de la función *norma.m:*



```
function norma = norma(x, v)

  norma = sqrt(sum((x-v).^2));

endfunction
```

Y este de la función *findClosestCentroids.m:*

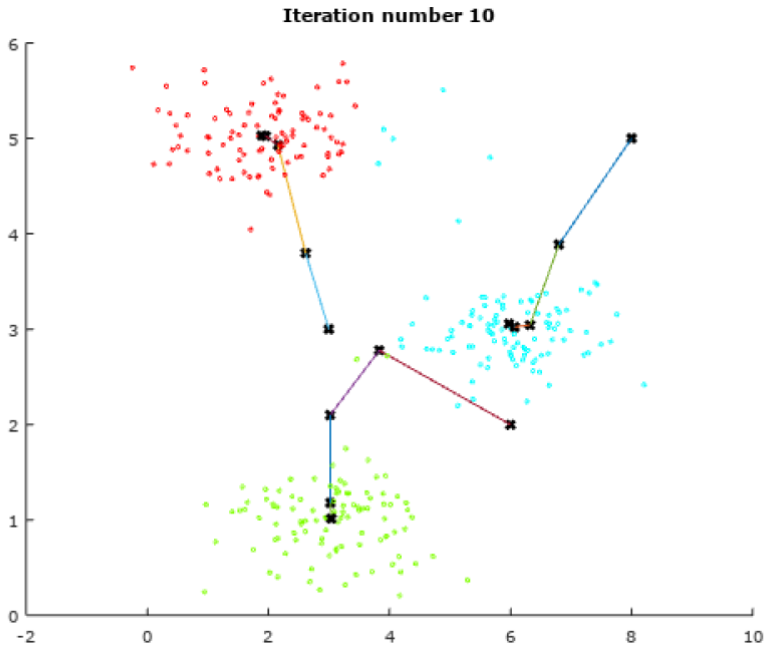


```
function idx = findClosestCentroids(X, centroids)
  %idx es un vector mx1 donde guardamos cada centroide

  %para cada centroide
  for i = 1:rows(X)

    minnorma = norma(X(i, :), centroids(1, :)) ^ 2;
    idx(i,:) = 1;

    for j = 2:rows(centroids)

      n = norma(X(i, :), centroids(j, :));
      if (n < minnorma)
        minnorma = n;
        idx(i,:) = j;
      endif

    endfor
  endfor
endfunction
```

A continuación, hemos implementado la función *computeCentroids.m*, que se encarga de calcular los nuevos centroides para la siguiente iteración. Este es el código de la función:

```
function centroids = computeCentroids(X, idx, K)

  for i = 1:K

    Ck = idx == i;
    sumatorio = zeros(1, columns(X));
    for j = 1:rows(Ck)
      if(Ck(j, :) == 1)
        sumatorio = sumatorio + X(j, :)
      endif
    endfor


    centroids(i, :) = sumatorio./sum(Ck);
  endfor

endfunction
```

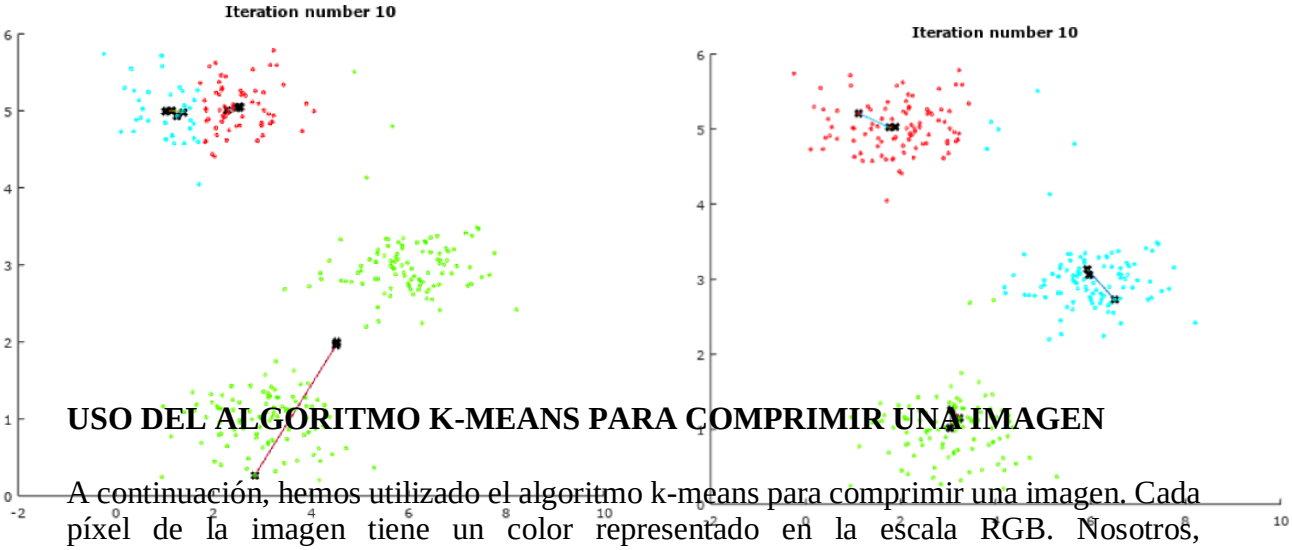
Tras ejecuar el algoritmo k-means con los centroides iniciales de la práctica y 10 iteraciones, obtenemos la siguiente gráfica:



Tras ejecutar el siguiente código, inicializando los centros de manera aleatoria.

```
function ex7kmeans()
  load("ex7data2.mat");
  K = 3;
  randidx = randperm(size(X, 1));
  centroids = X(randidx(1:K), :);
  [centroids, idx] = runkMeans(X, centroids, 10, true);
endfunction
```

Al inicializar los centros de manera aleatoria, es posible que ocurran resultados como los siguientes:



USO DEL ALGORITMO K-MEANS PARA COMPRIMIR UNA IMAGEN

A continuación, hemos utilizado el algoritmo k-means para comprimir una imagen. Cada píxel de la imagen tiene un color representado en la escala RGB. Nosotros, especificaremos en cuantos colores queremos representar la imagen para que nuestro algoritmo encuentre tantos centroides como colores hemos elegido y luego encontraremos el centroide más cercano para cada píxel. De esta manera reducimos el número de bits utilizados en una imagen resultando en un tamaño menor de la misma.

Para esta parte de la práctica hemos utilizado el siguiente código:

```
function ex7img
  A = double (imread('bird_small.png')) ;

  A = A / 255;

  imagesc(A);
  X = reshape(A, rows(A) * columns(A), 3);

  K = 32;
  randidx = randperm(size(X, 1));
  centroids = X(randidx(1:K), :);
  [centroids, idx] = runkMeans(X, centroids, 10, true);

  for i = 1:rows(X)
    X_compressed(i, :) = centroids(idx(i),:);
  endfor

  X_compressed = reshape(X_compressed, rows(A), columns(A), 3);

  imagesc(X_compressed);
endfunction
```

Este es el resultado de comprimir la imagen dada en 16 y en 32 colores, respectivamente: