Práctica 7:
Clustering

Grupo 13:
David Ortiz Fernández.
Andrés Ortiz Loaiza.

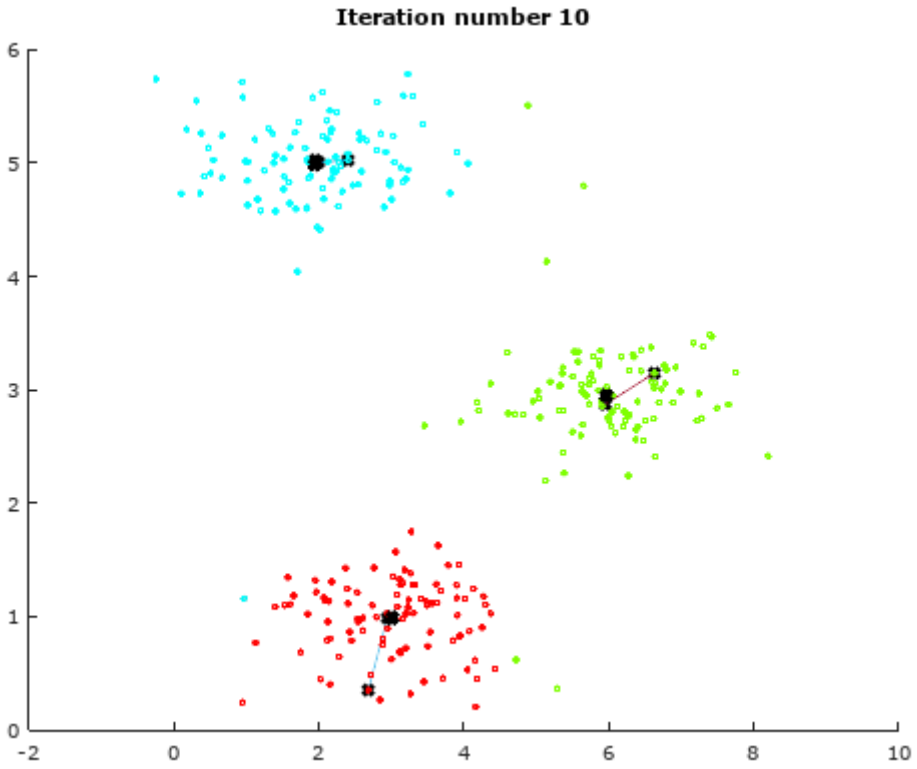
Primera parte: K-means

Esta práctica está destinada a familiarizarnos con el clusterin, para ello implementamos k-means y comprobaremos gráficamente su funcionamiento. Para implementar k-means ha sido necesario la creación de una función.

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. A continuación, hemos implementado la función computeCentroids, que calcula la nueva posición de los centroides.

Para que comprobar todo funciona bien, hemos cargado el fichero ex7data2.mat que e invocado a la función runkMeans.

Hemos ido mostrando el resultado iteración a iteración para un total de 10 iteraciones. El resultado ha sido el siguiente en una de las pruebas, recordemos que depende de la aleatoriedad puede cambiar:



Código:

Flujo principal:

```
clear;
close all;
warning('off','all');

load('ex7data2.mat');

K = 3; % 3 Centroides
inicial_centroides = [3 3; 6 2; 8 5];
idx = findClosestCentroids(X, inicial_centroides);

centroids = computeCentroids(X, idx, K);

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

Función findClosestCentroids.m

```
function idx = findClosestCentroids(X, centroids)

%bucle para cada centroide
for i=1:rows(X)
  normaMin = norma(X(i, :), centroids(1, :)) ^ 2;
    idx(i,:) = 1;
    for j=2:rows(centroids)
      c = norma(X(i, :), centroids(j, :));
      if (c < normaMin)
        idx(i,:) = j;
        normaMin = c;
      endif
    end
    %devolvemos el minimo
end
end
```

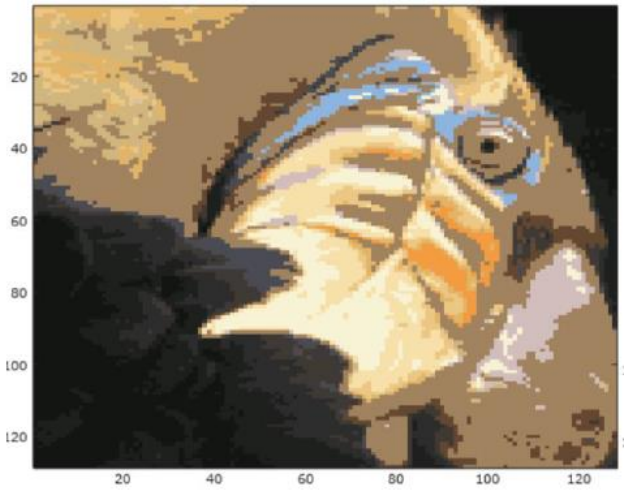
Funcion computeCentroids.m

```
function centroids = computeCentroids(X, idx, K)
[m n] = size(X);

for i = 1:K
 points = idx == i;
 suma = zeros(1,columns(X) );
    for j = 1:rows(points)
      if(points(j, :) == 1)
        suma = suma + X(j, :);
      endif
    endfor
    centroids(i, :) = suma./sum(points);
  endfor
endfunction
```

```
function centroids = computeCentroids(X, idx, K)
[m n] = size(X);

for i = 1:K
 points = double(idx == i);
 suma = zeros(1, n);
    for j = 1:rows(points)
      if(points(j, :) == 1)
        suma = suma + X(j, :);
      endif
    endfor
    centroids(i, :) = suma./sum(points);
  endfor
```

Segunda parte: Compresión de imágenes Kmeans

En la segunda parte de la práctica se utiliza lo desarrollado en la primera parte para que nuestro algoritmo encuentre tantos centroides como colores hemos elegido y luego encontraremos el centroide más cercano para cada píxel. El resultado fue el siguiente para la imagen comprimida de 16 colores tras 10 iteraciones.



Código:

Flujo principal:

```
clear;
close all;
warning('off','all');

%las posiciones (i; j; 1); (i; j; 2) e (i; j; 3) representan respectivamente
%los porcentajes de rojo, verde y azul del pixel de la posicion (i,j)
A = double(imread('bird_small.png'));

%dividimos en 255 todos los pixeles para reresentar en rangos
%entre 0 y 1
A = A / 255;
imagesc(A);

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

K = 32;

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

  for i = 1:rows(X)
    X_comprimida(i, :) = centroides(idx(i),:);
  endfor

X_comprimida = reshape(X_comprimida,rows(A) ,columns(A), 3);
imagesc(X_comprimida);
```