**Lab 9. Clustering**

MainScriptToRunKmeansISI:

**Gap[1], [2], [3]:**

* **[CImaNamRed,ThePathRed] = fReadDirNamAndIma13('jpg','Select the folder with the Mixed images','C:\Users\Irene\Desktop\Universidad\Segundo Curso\segundo semestre\ISI\Codigo\_practica9\_Color Images\Color Images\RedImages');**
* **[CImaNamGreen,ThePathGreen]= fReadDirNamAndIma13('jpg','Select the folder with the Mixed images','C:\Users\Irene\Desktop\Universidad\Segundo Curso\segundo semestre\ISI\Codigo\_practica9\_Color Images\Color Images\GreenImages');**
* **[CImaNamMixed,ThePathMixed]= fReadDirNamAndIma13('jpg','Select the folder with the Mixed images','C:\Users\Irene\Desktop\Universidad\Segundo Curso\segundo semestre\ISI\Codigo\_practica9\_Color Images\Color Images\MixedImages');**

Con esta función lee las imágenes y las añade en un cellarray. La función  **fReadDirNamAndIma13,**  la cual recibe 3 parámetros: la extensión, un mensaje y el directorio donde se encuentra.

Por ello, la extensión en todas ellas será “jpg”, ya que se tratan dataset de imágenes.

El mensaje será “Selecciona la carpeta”.

Y el directorio será la ruta absoluta de cada carpeta.

**Gap[4], [5], [6]:**

**RGBMeansMatrixRed= fCellImageToRGBVectors(CImaNamRed);**

**RGBMeansMatrixGreen= fCellImageToRGBVectors(CImaNamGreen);**

**RGBMeansMatrixMixed= fCellImageToRGBVectors(CImaNamMixed);**

Para extraer los vectores de características de las imágenes, se creará un cell mediante la función **fCellImageToRGBVectors(CImaNamColour),**  donde CImaNamColour es un cell-array con las imágenes en la primera fila. Está funcion nos devuelve la media de color RGB de cada imagen.

fKmeansISI:

**GAP[7]:**

**DimensionsDataset= size(Dataset);**

Para guardar la dimensión de nuestro dataset usamos la función “size()” haciendo que reciba nuestro dataset.

**GAP[8]:**

**MuCentroidsMatrix(:,i)= Dataset(:,IndexRandomInitializationSamples(i));**

Creamos una matriz para guardar los primeros centroides, ya que después vamos a ir recalculándolos. Y para ello vamos a extraer los centroides del dataset, que se seleccionan de manera random y almacenandolo en IndexRandomInitializationSamples(i).

**GAP[9]:**

**DistancesVector= fEuclideanDistVectToMatrix(Dataset(:,i),MuCentroidsMatrix);**

Calculamos la distancia euclídea de cada una de las muestras a cada matriz de centroides, que habíamos rellenado en el GAP[8]. Por ello debemos usar  **fEuclideanDistVectToMatrix** y no **fEuclideanDistVectToVect.**

**GAP[10]:**

**[~,IndexMinimum]=min(DistancesVector);**

Obtenemos el índice del centroide mínimo. Como tenemos una matriz con las distancias, buscamos el mínimo y mediante **[~,IndexMinimum],”~”** guarda el valor de ese mínimos y **“IndexMinimum ”** el indice.

**GAP[11]:**

**ClosestCentroidToEachSample(i)=IndexMinimum;**

Asignamos la muestra, la mínima distancia calculada en el GAP[10], al centroide más cercano.

**GAP[12]:**

**NewMuCentroidsMatrix(:,i)= mean(Dataset(:,IndicesClosestToK),2);**

Volvemos a calcular los centroides haciendo las medias y los guardamos en una nueva matriz.

**GAP[13]:**

**DistanceBetweenOldAndNews= fEuclideanDistVectToVect(PreviousCentroids, CurrentCentroids);**

Como en el GAP[12] hemos creado una nueva matriz de centroides, debemos usar **fEuclideanDistVectToVect,** para la distancia entre los viejos centroides y los nuevos.

**GAP[14]:**

**MuCentroidsMatrix= NewMuCentroidsMatrix;**

Actualizamos los nuevos centroides.