A logo of a university

Description automatically generated

UNIVERSIDAD DE SONORA

División de Ingeniería

Departamento de Ingeniería Industrial

**K-Means**

Minería de datos

Grupo 1

**Profesora**  
RAQUEL TORRES PERALTA

**Alumnos**

Chávez Pérez Javier Ariel  
Esquer Monge Abad  
Noriega Robles José Alberto

Hermosillo, Sonora a 25 de marzo, 2024.

1. Analizar los datos

A close-up of a graph

Description automatically generated**Longitud y anchura del sépalo Longitud y anchura del pétalo**

Podemos notar como en el pétalo es mucho más notable la diferencia, pero podemos concluir que en ambas la clase azul se distingue mucho del resto.

1. Seleccionar una representación.

Se usarán todos los datos para procesamiento ya que todas pueden ayudar a ser distinguibles unas clases de otras.

1. Aplicar el algoritmo de K-means.

El resultado del algoritmo se demostrara en los siguientes incisos y el código total se pondrá al final de este documento. Claro que igualmente se anexaran las funciones en archivos aparte.

1. Encontrar la correspondencia entre el cluster que asigna el algoritmo y la clase.

|  |  |
| --- | --- |
| Asignación de clústeres | |
| Real | Calculado |
| 1 | 2 |
| 2 | 1 |
| 3 | 3 |

1. Calcular el acierto del algoritmo por clase (% de aciertos, Precision, Recall y F-Score) y en general (% de aciertos totales).

|  |  |
| --- | --- |
| Clase 1 | |
| Porcentaje de aciertos | 100% |
| Precision | 1 |
| Recall | 1 |
| F-Score | 1 |
| Clase 2 | |
| Porcentaje de aciertos | 94% |
| Precision | 0.77049 |
| Recall | 0.94 |
| F-Score | 0.84685 |
| Clase 3 | |
| Porcentaje de aciertos | 72% |
| Precision | 0.92308 |
| Recall | 0.72 |
| F-Score | 0.80899 |
| **Porcentaje Total** | **88.667%** |

1. Gráfica de los clusters y las predicciones.

**A graph of a number of dots

Description automatically generated with medium confidenceClusters reales**

**Clusters calculados**A graph of a diagram

Description automatically generated with medium confidence

1. Agregar conclusiones.

Al terminar este algoritmo tenemos mucha mas perspectiva de como se deben de manejar los datos si queremos encontrar coincidencias uno con otro. A su vez fuimos conscientes de lo acostumbrados que estamos a trabajar siempre en graficas 2D que fue nuestra limitante visual pero no a nivel datos, en los datos si calculamos la distancia dentro de las 4 dimensiones que nos permite, solo en graficas tuvimos que separar los datos para agregar a este documento.

Nos esperábamos mas distorsión dentro del porcentaje de aciertos, pero realmente estamos sorprendidos de que al menos en la primera clase diera 100% de acierto.

**Código utilizado (igual proporcionaremos los archivos)**

%Agregamos el path del archivo y funciones SIEMPRE AGREGAR PORQUE SE PIERDE SOLO

addpath 'D:\GitHub\KMeansTuto'

%Leer un archivo csv (Comma Separated Values)

filename='iris1.csv'

Casos = csvread( filename );

%Eliminamos la columna de texto que se pone en 0(era texto) pero a su derecha ya dice el numero de clase

Casos(:, 5) = [];

%Declaramos el numero de centroides

numCent=3;

%Necesitamos eliminar la columna de clase a Casos

Datos = Casos(:, [1:5-1, 5+1:end]);

%Para mandar a llamar a la funcion procesoKMeans (matriz, numCentroides)

[centroides,C]=procesoKMeans(Datos,numCent,max\_iter,CReal,centroidesReal);

%Mandamos llamar la funcion para calcular aciertos y demas metricas

calcular\_aciertos(CReal,C,numCent);

%----------De aqui en adelante solamente son las funaciones, al final esta la que manda a llamar a todas-------

%Funcion para normalizar los datos

function norm=normalizar(matriz)

norm=zeros(size(matriz));

for i=1:size(matriz,2)

min\_val = min(matriz(:, i));

max\_val = max(matriz(:, i));

norm(:, i) = (matriz(:, i) - min\_val) / (max\_val - min\_val);

end

end

function centroides = generarCentroides (matriz,numCentroides,numCol)

%Damos tamaño a la matriz

numCol = size(matriz, 2);

centroides= zeros(numCentroides, numCol);

for i=1:numCentroides

for j=1:numCol

%Se genera el centroide de manera random dentro del rango de valores

minimo\_valor=min(matriz(:, j));

rango = max(matriz(:, j)) - minimo\_valor;

valor\_aleatorio = minimo\_valor + rango \* rand;

centroides(i, j) = valor\_aleatorio;

end

end

end

%Funcion para asignar cluster

function asignacionClusters = asignarClusters(datos, centroides)

num\_puntos = size(datos, 1);

num\_clusters = size(centroides, 1);

num\_atributos = size(centroides, 2);

% Inicializar matriz de asignación de puntos a clusters

asignacionClusters = zeros(num\_puntos, 1);

% Calculamos distancias

distancias = zeros(num\_puntos, num\_clusters);

for i = 1:num\_clusters

for j = 1:num\_puntos

for o = 1:num\_atributos

distancias(j, i) = distancias(j, i) + ((datos(j, o) - centroides(i, o)).^2);

end

end

end

% Asignar cada punto al cluster más cercano

[~, asignacionClusters] = min(distancias, [], 2);

end

%funcion para actualizar los centroides

function nuevos\_centroides = actualizar\_centroides(datos, asignacion\_clusters,num\_clusters)

[num\_puntos, num\_atributos] = size(datos);

nuevos\_centroides = zeros(num\_clusters, num\_atributos);

for i = 1:num\_clusters

% Mapeamos los puntos asignados al cluster I

puntos\_en\_cluster = asignacion\_clusters == i;

puntos\_mapeados = datos(puntos\_en\_cluster,:);

for j=1:num\_atributos

%Calculamos la posicion conforme al promedio de sus puntos

nuevos\_centroides(i,j) = mean(datos(puntos\_en\_cluster, j), 1);

end

end

end

%Funcion para calcular aciertos

function calcular\_aciertos(claseReal,clasePredicha,numCent)

%Acumulador de aciertos totales

total\_TP=0;

% Calcularemos para cada clase específica

for i=1:numCent

j=0;

if i == 1

j = 2;

elseif i == 2

j = 1;

elseif i == 3

j = 3;

end

% Verdaderos positivos (TP)

TP = sum(claseReal == i & clasePredicha == j);

%Acumulando aciertos totales de todas las clases

total\_TP = total\_TP + TP;

% Falsos positivos (FP)

FP = sum(claseReal ~= i & clasePredicha == j);

% Verdaderos negativos (TN)

TN = sum(claseReal ~= i & clasePredicha ~= j);

% Falsos negativos (FN)

FN = sum(claseReal == i & clasePredicha ~= j);

% Calcular precisión, recall y F-score

precision = TP / (TP + FP);

recall = TP / (TP + FN);

F\_score = 2 \* ((precision \* recall) / (precision + recall));

% Calcular porcentaje de aciertos totales

total\_clase=claseReal == i;

total\_aciertos = sum(claseReal == i & clasePredicha == j) / sum(claseReal == i) \* 100;

disp(['El resultado de la clase ', num2str(i)]);

disp(['Porcentaje de aciertos: ', num2str(total\_aciertos)]);

disp(['Precision: ', num2str(precision)]);

disp(['Recall: ', num2str(recall)]);

disp(['F-Score: ', num2str(F\_score)]);

end

% Calcular el porcentaje de aciertos totales de todas las clases

total\_aciertos = total\_TP / numel(claseReal) \* 100;

disp(["Porcentaje de aciertos totales: ",num2str(total\_aciertos)]);

end

%Funcion que manda a llamar a las demas funciones :matriz, numCentroides

function [clusters,C]= procesoKMeans(datos,numCent,max\_iter,CReal,centroidesReal)

%Se manda a llamar la normalizacion de los datos

norm=normalizar(datos);

%Mandamos a llamar la funcion generarCentroides: matriz, numCent

centroides=generarCentroides(norm,numCent);

move=true; %Flag para saber si cambiaron de lugar los centroides

newcentroides=[]; %Declaramos la matriz vacia

conteo=0;

%max\_iter es la clausula de escape para que no sea infinito por algun error

while move && conteo < max\_iter

%Asi mandamos a llamar a la funcion asignarClusters:datos, centroides

C=asignarClusters(norm, centroides);

%Asi mandamos a llamar a la funcion actualizar\_centroides:datos, asignacion\_clusters

newcentroides = actualizar\_centroides(norm, C,numCent);

%Corroboramos si se movieron los centroides

move = any(centroides(:) ~= newcentroides(:));

centroides=newcentroides;

conteo++;

disp(['Ciclo #', num2str(conteo)]);

end

if conteo == max\_iter

disp('Se alcanzó el número máximo de iteraciones.');

else

disp(['Numero de ciclos: ', num2str(conteo)]);

end

%Se declara a la variable de salida

clusters=newcentroides;

%por ultimo se grafica

graficar(norm,C,CReal,clusters,centroidesReal);

end