Universitat de les Illes Balears

Escola Politécnica Superior

ACSI Cuaderno de Prácticas

Pablo Núñez Pérez - 43199702K

Práctica 6

El objetivo de esta práctica es la comprensión del concepto de caracterización de la carga. Para ello, se hará uso de la herramienta Weka. De la monitorización de un sistema de almacenamiento, se ha obtenido se proporciona un fichero de datos llamado data.txt. En el fichero se almacenan tres columnas con la siguiente información:

- El tamaño del fichero accedido (en MB). Los valores que correspondan con "-1" quieren decir que el acceso al fichero ha fallado.

- La hora a la que se hizo el acceso. El valor 22 representan las 22h, el valor 01 representan las 1h (a.m.), etc.

- El ancho de banda consumido (en MS/s). Los valores de esta columna están entre 453 y 1355, por lo tanto, los valores de esta columna deberán ser tratados. Es decir, el valor crudo de "1258.84,", corresponde con "1258,84".

1. Aplicando el algoritmo con 100 iteraciones y agrupando los datos en 3 clases, ¿qué resultados se obtienen? Muéstralo gráficamente.

Una vez procesados los datos con un **script de Python** y con los datos obtenidos tras la clusterización con Weka con los parámetros indicados (100 iteraciones, 3 clusters), dibujamos la siguiente figura:

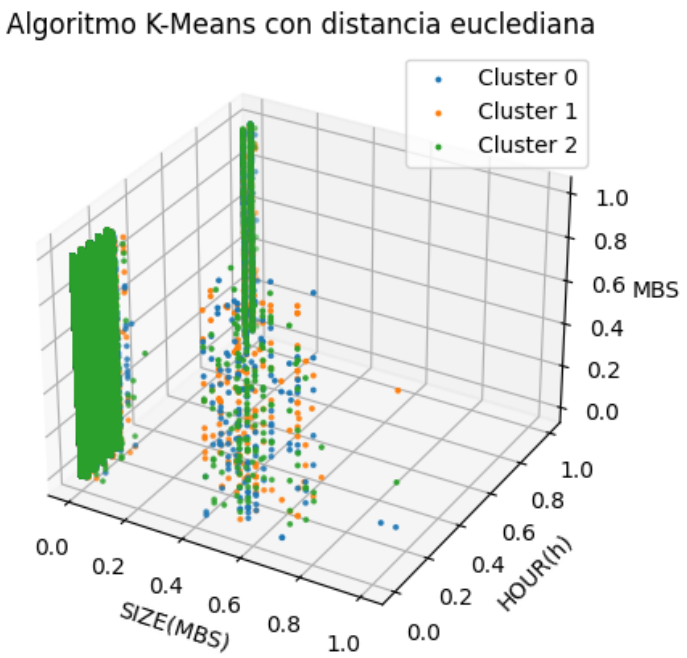

Figura 1: Clustering 3D

En esta figura se muestran los clusters detectados por Weka. Los parámetros de los componentes son: el ancho de banda (MB/s) el cual es un ratio, el tamaño en MB el cual es un valor númerico flotante y la hora, un valor entero. A causa de la gran diferencia entre el rango de valores de los parámetros se ha decidido normalizar para tener una mejor representación de los datos.

Lo primero que podemos apreciar es un gran hueco en el plano de la ***hora***. Esto se debe a que de las 24 horas que tenemos disponibles, **solo se han medido 7**. De las 22 PM hasta las 5 AM (saltandonos las 00h). Esto nos hace sospechar que este parámetro pueda ser perjudicial a la hora de hacer el clustering, ya que mancha los datos y realmente no nos aporta información por la falta de valores.

Además, haciendo clustering con todos los pares de atributos posibles (Ex: (hora, size), (size, MB/s)) comprobamos que el **atributo más importante** es el del ancho de banda ***MB/s*** y que el resto influencian en menor medida. Teniendo en cuenta este hecho, y lo explicado en el párrafo anterior, se decide suprimir el atributo *Hora* a la hora de hacer los clusters.

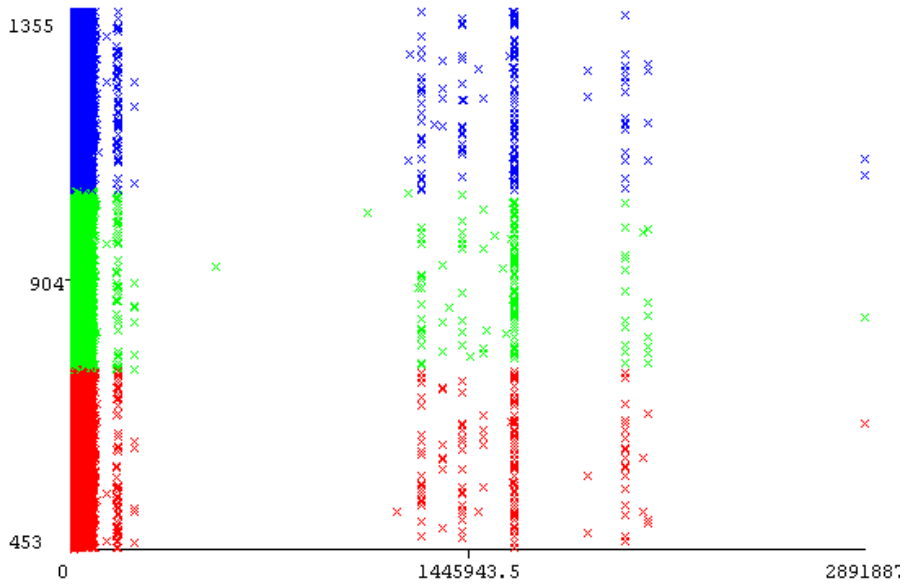

Figura 2: Clustering sin Hora

Siendo el eje X *size* y el eje Y *MB/s*, Weka nos ofrece este clustering. Es un clustering algo extraño, ya que no encontramos nubes de puntos. Esto se debe a la gran densidad de datos que tenemos por la izquierda, a diferencia que en resto del intervalo. Es por ello que se puede decir que los datos siguen una distribución heavy long tailed.

Lo que podemos sacar en claro de estos datos es que el sistema de almacenamiento ha sido usado durante el período de monitorización para acceder a muchos ficheros de muy poco tamaño respecto al resto de tamaños y a una gran variedad de velocidades.

En cuanto a los representantes de las clases, Weka ya nos da los centroides de cada una en su pestaña de resultados:

```
kMeans
======

Number of iterations: 37
Within cluster sum of squared errors: 6035.828250563902

Initial starting points (random):

Cluster 0: 269,1336.661
Cluster 1: 1308,457.982
Cluster 2: 8790,683.954

Missing values globally replaced with mean/mode

Final cluster centroids:
                        Cluster#
Attribute    Full Data          0          1          2
            (635742.0) (212581.0) (211090.0) (212071.0)
=========================================================
size         4728.2753  4805.8846  4754.3041  4624.5708
MB/s          903.9707  1204.1532   602.7878   902.8561
```

Figura 3: Resultados 3 Clusters

Traduciendo estos resultados a una tabla:

Clase	Size	MB/s	Componentes
Alta velocidad	4805.88	1204.15	212581
Velocidad media	4624.57	902.85	210090
Baja velocidad	4754.30	602.78	212071

En la tabla de arriba se pueden ver las clases de accesos que hemos encontrado, con sus parámetros respectivos y el número de componentes por grupo.

Los nombres de las clases coinciden con lo dicho al principio de este documento y a la gráfica de la clusterización: el parámetro dominante es el ancho de banda.

2. Con el mismo número de iteraciones y agrupando los datos en 5 clases, ¿qué resultados se obtienen? ¿Cómo difieren de los anteriormente obtenidos?

En este clustering también hemos decidido obviar la variable *hour*, porqué nos hemos encontrado los problemas mencionados en el apartado anterior. Como podemos ver en la figura siguiente, el único cambio que podemos ver es en el número de clusters. La distribución de estos sigue siendo simétrica, cada cluster contiene un 20 % de las instancias.

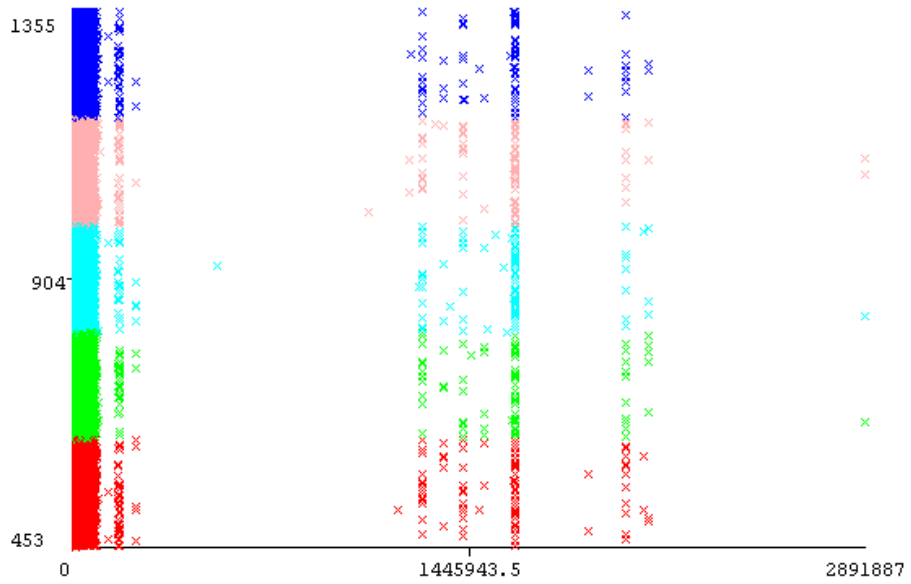

Figura 4: Clustering sin Hora (5 Clusters)

Los datos obviamente no han sido modificados, así que siguen el mismo patrón: cuanto mayor el tamaño de los ficheros, menos accesos se han llevado a cabo.

En cuanto a la caracterización de la carga, la podemos ver en la siguiente tabla:

Clase	Size	MB/s	Componentes
Muy alta velocidad	4660.83	1264.96	126860
Alta velocidad	4779.70	1084.78	127318
Velocidad media	4755.79	904.08	127171
Velocidad baja	4605.04	724.09	127221
Velocidad muy baja	4839.67	543.21	127172

Como en el apartado anterior, aquí tenemos las clases con los parámetros correspondientes y su número de componentes correspodentes. Viendo ambas tablas, nos podemos percatar que el tamaño del fichero no ha afectado en la velocidad de transmisión de los datos.

3. ¿Hay alguna característica especial en la carga proporcionada? Explícala con detalle.

Como ya hemos ido adelantando durante la práctica, una de las características especiales de esta carga es que es *heavy long tailed*, es decir mucha densidad de datos en un punto alejado del centro de los datos. Para tener un mejor entendimiento de esta distribución he aquí una imagen.

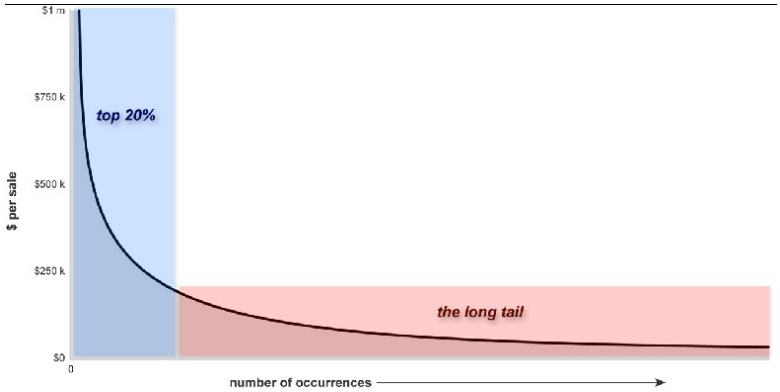

Figura 5: Gráfica Long Tailed

En el caso actual, la región azul de la imagen representa la gran cantidad de datos en *data.txt* que corresponden a ficheros de bajo tamaño, y la región roja el resto.

La segunda característica que podemos ver en los datos es que estan divididos en tres pilares bien diferenciados. Este hecho es consecuencia de la distribución explicada.

Como última característica, podemos ver que los datos no estan distribuidos en una nube, como solemos estar acostumbrados. De hecho la clusterización ha sido casi simétrica. Estos hechos puede que nos esten indicando que los datos de *data.txt* no sean los idóneos para el análisis directo de estos. De hecho, en una primera instancia, este clustering crea bastante confusión, ya que vemos puntos muy cercanos en clusters distintos, y otros bastante alejados en el mismo cluster.

Para finalizar, comentar que se deberían de agrupar cualitativamente los datos (por tamaño de fichero) para tener una mejor representación, clustering y por ende análisis de los datos monitorizados. Así, con los datos agrupados, podríamos usar Weka para clusterizar cada grupo y tener unos clusters sustancialmente mejor distribuidos.

Scripts Python

Dibujado de Cluster

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
def main():
    plt.figure()
    #Iteracion entre todos los clusters (1,2,3)
    i = 0
    while True:
        try:
            cluster = create_cluster(i)

            cluster_x = [float(data[0]) for data in cluster]
            cluster_y = [float(data[1]) for data in cluster]


            minAndMax_X = [np.min(cluster_x), np.max(cluster_x)]
            minAndMax_Y = [np.min(cluster_y), np.max(cluster_y)]

            normalized_X = [normalization(data, minAndMax_X[0], minAndMax_X[1]) for data in cluster_x]
            normalized_Y = [normalization(data, minAndMax_Y[0], minAndMax_Y[1]) for data in cluster_y]

            plt.scatter(normalized_X, normalized_Y, s=3, label=f"Cluster {i}")
            i += 1
        except:
            break

    plt.xlabel('SIZE(MBS)')
    plt.ylabel('MBS')
    plt.legend(loc="upper right")
    plt.title(f"Algoritmo K-Means con distancia euclediana ")
    plt.show()
#Crear Cluster
def create_cluster(value):
    with open("cluster_noTime.arff") as file:
        data = file.read().splitlines()

    #Quitamos header del archivo arff
        data = data[9:]
    #Cluster es una lista unidimensional de datos
    cluster = []
    for line in data:
        line = line.split(",")
        if line[-1] == f"cluster{value}": #Si encontramos una fila con el cluster correspondiente
                                          #al valor "-1", lo suprimimos
            cluster.append([line[1], line[2]])

    file.close()
    return cluster
#Normalizar
def normalization(value, min, max):
    return (value - min)/(max - min)
if __name__ == "__main__":
    main()
```

Procesador de datos

```
    def main():
    with open("data.csv") as fichero:
        data = fichero.read().splitlines() # Leemos el fichero en formato de una lista de líneas

    # Quitamos el header
    data.pop(0)
    #Cerramos enlace
    fichero.close()
    with open(f"dataProcessed.csv", "x") as dataProcessed: #Creamos un fichero nuevo procesado.

        dataProcessed.write("size,hour,MB/s\n")

        for fila in data:
            fila = fila.split(",")

            # Si encontramos una fila errónea
            if fila[0] == "-1": continue #No concatenaremos esta fila en el fichero final

            # Quitamos espacio inutil de la última columna
            try:
                fila.pop(2)
            except:
                pass #NOP

            # Dividimos la segunda columna en dos y lo guardamos en un temporal
            temp = fila[1].split("\t")
            fila[1] = temp[0] #Ponemos el valor de la hora en la segunda columna de la fila

            # Formateamos el número (lo que seria la ultima columna de la lína final)
            fila.append(formateo(temp[1]))

            dataProcessed.write(f"{fila[0]},{fila[1]},{fila[2]}\n")
        dataProcessed.close() #Cerramos enlace

def formateo(numero):
    temp = numero.split(".") #Spliteamos con el punto.
                        #Ya que no hay distincion entre el punto que separa miles y los decimales:
    #Si tiene más de 3 valores el número, significa que hay que poner un punto entre el 2 y 3er numero
    if len(temp) == 3:
        return f"{temp[0]}{temp[1]}.{temp[2]}"
    else:
        return numero

if __name__ == "__main__":
    main()
```

Destruir columna

```
from this import d
import pandas as pd
def main():
    data = pd.read_csv('dataProcessed.csv')
    # delete de columna "Hour"
    data.drop('hour', inplace=True, axis=1)
    data.to_csv('dataNoTime.csv', index=False)
if __name__ == "__main__":
    main()
```