



Universitat
de les Illes Balears

Evaluación de Comportamiento de Sistemas Informáticos

Práctica 6 - Evaluación y modelado del rendimiento de los sistemas informáticos

Alberto Pérez Ancín

Professores:

Dr. Carlos Juíz García

Dra. Belén Bermejo González



APLICACIÓN PRÁCTICA

1.1 Objetivo

El objetivo principal es la evaluación práctica, real y completa de un sistema informático. Para ello se aplicarán todos los conceptos, metodologías y técnicas vistos a lo largo de todo el curso. Se tratarán desde los aspectos relacionados con la monitorización y el benchmarking, pasando por el modelado y llegando finalmente a la predicción de la carga.

Un ejemplo real del camino expuesto se ve en el artículo “The Performance Evaluation Journey of a Flight Seats Availability Service: A Real-world Business Case Study of Transactional Workload Running in Virtual Machines”, el cual se recomienda encarecidamente que se tenga a disposición y se comprenda en su totalidad.

1.2 Sexta parte

El objetivo de esta parte 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.

1. APLICACIÓN PRÁCTICA

- 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”.

Con los datos proporcionados se pide caracterizar la carga haciendo uso del algoritmo de Kmeans y responder a las siguientes preguntas:

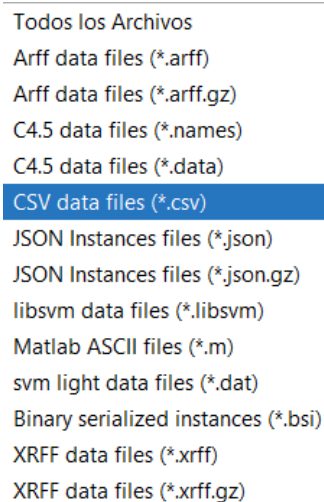
- **Aplicando el algoritmo con 100 iteraciones y agrupando los datos en 3 clases, ¿qué resultados se obtienen? Muéstralo gráficamente.**

Antes de comenzar con la práctica, se han realizado ciertas tareas de preparación con respecto al archivo “data.txt”. En primer lugar, se ha realizado una revisión del archivo para comprender su estructura y los tipos de datos que contiene. Esto implica verificar las columnas y su orden, así como los posibles valores incorrectos o formatos inesperados.

Además, se ha llevado a cabo un proceso de limpieza y transformación de los datos. Se ha corregido el error de representación en la columna de ancho de banda, reemplazando las comas por puntos para asegurar una representación coherente de los valores decimales.

También se ha realizado un tratamiento adecuado de los valores “-1” en la columna de “size” del fichero accedido. Estos valores indican fallos de acceso, por lo que se han identificado y se ha tomado decisión de eliminarlos durante el análisis. Finalmente, se han realizado tareas de limpieza y transformación de los datos mediante un script creado a propósito para esta práctica en Python.

Formatos que dan soporte Weka para introducir los datos:



Todos los Archivos
Arff data files (*.arff)
Arff data files (*.arff.gz)
C4.5 data files (*.names)
C4.5 data files (*.data)
CSV data files (*.csv)
JSON Instances files (*.json)
JSON Instances files (*.json.gz)
libsvm data files (*.libsvm)
Matlab ASCII files (*.m)
svm light data files (*.dat)
Binary serialized instances (*.bsi)
XRFF data files (*.xrff)
XRFF data files (*.xrff.gz)

El formato CSV (Comma-Separated Values) es la opción que he elegido para almacenar datos, ya que conocía de antes su funcionamiento y lo más importante que es uno de los formatos que soporta Weka.

Script creado para solventar los problemas del fichero data.txt

```
1
2 import argparse
3 import csv
4
5 parser = argparse.ArgumentParser()
6
7 # Definimos el argumento de entrada que se puede pasar cuando se ejecuta el
  programa
8 parser.add_argument("-i", "--input", help="Input file", default="data.txt",
  type=str)
9 args = parser.parse_args()
10
11 # Esta función toma un número en forma de cadena y reorganiza los
  componentes antes y después del punto.
12 def fractor_number(number):
13     temp = number.split(".")
14     # Si hay tres componentes, los reorganiza como específica en el
  enunciado de la P6.
15     return f"{temp[0]}{temp[1]}.{temp[2]}" if len(temp) == 3 else number
16
17 # Esta función procesa una línea de datos
18 def process_line(line):
19     # Si el primer elemento de la línea es -1, la función devuelve None y
  la línea se ignora.
20     if line[0] == "-1":
21         return None
22
23     # Arregla el tercer elemento de la línea.
24     try:
25         line.pop(2)
26     except IndexError:
27         pass
28
29     try:
30         temp = line[1].split("\t")
31         line[1] = temp[0]
32         # Añade el segundo componente después de procesarlo a través de la
  función fractor_number.
33         line.append(fractor_number(temp[1]))
34     except IndexError:
35         print(line)
36         return None
37
38     return line
39
40 def main():
41     # Abre el archivo de entrada y lee sus datos
42     with open(args.input, 'r') as file:
43         reader = csv.reader(file)
```

1. APLICACIÓN PRÁCTICA

```
44         data = list(reader)
45
46     # Elimina la cabecera del archivo de entrada
47     data.pop(0)
48
49     # Abre un nuevo archivo csv para escribir los datos procesados
50     with open("data.csv", "w", newline='') as output:
51         writer = csv.writer(output)
52         # Escribe la cabecera del nuevo archivo csv
53         writer.writerow(["size", "hour", "MB/s"])
54
55     # Procesa cada línea de datos y las escribe en el nuevo archivo csv
56     for line in data:
57         processed_line = process_line(line)
58         if processed_line is not None:
59             writer.writerow(processed_line)
60
61 if __name__ == "__main__":
62     main()
```

En Weka, el algoritmo de agrupamiento que selecciono es SimpleKMeans para clasificar un conjunto de datos en diferentes grupos o clústeres. En esta práctica, se ha configurado el algoritmo con ciertos parámetros específicos.

El parámetro “distanceFunction” se establece en “EuclideanDistance”. Esto significa que se utilizará la distancia euclidiana como medida para calcular la similitud entre los puntos de datos. La distancia euclidiana es una métrica utilizada en el análisis de agrupamiento y se basa en la geometría euclidiana para medir la distancia entre dos puntos en un espacio multidimensional.

El parámetro “maxIterations” se establece en 100. Esto indica que el algoritmo realizará un máximo de 100 iteraciones para ajustar los centroides de los clústeres y mejorar la asignación de los puntos de datos a los clústeres. Las iteraciones son pasos repetitivos en los que el algoritmo ajusta y actualiza los centroides para obtener una clasificación más precisa.

El parámetro “numClusters” se establece en 3. Esto indica que se van a obtener tres clústeres o grupos a partir de los datos de entrada de “datos.csv”. El algoritmo asignará cada punto de datos a uno de los tres clústeres en función de su similitud y ubicación en el espacio multidimensional.

En resumen, la configuración mencionada del algoritmo SimpleKMeans en Weka indica que se utilizará la distancia euclidiana como medida, se realizarán un máximo de 100 iteraciones para ajustar los centroides y se formarán tres clústeres a partir de los datos de entrada.

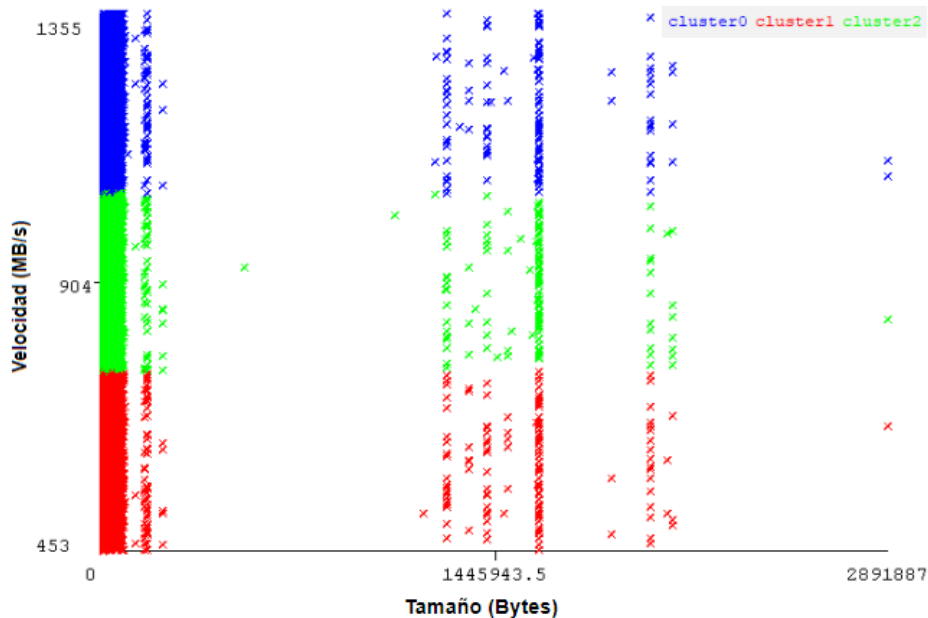


Figura 1.1: Realizo con Weka 3 Clusters mediante SimpleKMeans con EuclideanDistance

Al visualizar el gráfico, noto que la columna de tiempo no proporciona información relevante debido a la falta de datos suficientes. Por lo tanto, decido omitirla en la representación de los datos, enfocándome en graficar la velocidad de transferencia frente al tamaño. A partir de la agrupación de Weka en tres grupos, observo que la mayoría de los valores se concentran cerca de cero, disminuyendo drásticamente a medida que aumenta el tamaño. Esto sugiere que el sistema de almacenamiento utiliza principalmente archivos de tamaño pequeño antes que los de mayor tamaño, probablemente debido a los archivos pequeños son más fáciles de transmitir.

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

Al realizar con Weka una nueva agrupación de datos en 5 clases en lugar de 3, debería obtener una distribución de datos con más detalles. Esto significa que los datos se dividirán en más grupos, lo que puede llegar proporcionar una visión más detallada de los datos.

En mi análisis inicial, noté que la mayoría de las transferencias de archivos se realizaban con archivos más pequeños y que la frecuencia de estas transferencias disminuía a medida que el tamaño del archivo aumentaba. Esto me llevó a concluir que mi sistema de almacenamiento tiende a priorizar la transferencia de archivos más pequeños, probablemente por razones de eficiencia o rendimiento.

Entonces al reagrupar los datos en 5 clusters en lugar de los 3 originales. Observé que el patrón general se mantenía y las transferencias de archivos más pequeños seguían siendo más frecuentes. Esto reafirmó mi conclusión inicial de que mi sistema de almacenamiento favorece la transferencia de archivos más pequeños. La principal diferencia observable es que el Weka realiza cluster de 5 en vez de 3. En resumen, esta nueva agrupación en 5 clusters me proporciona una visión muy similar a la anterior de mi sistema de almacenamiento.

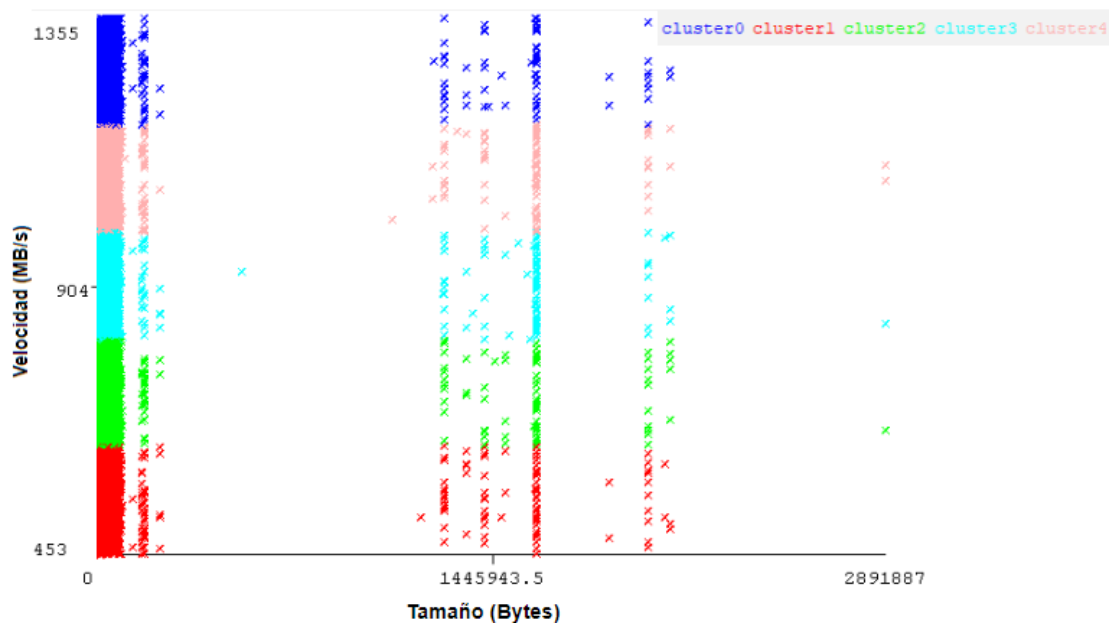


Figura 1.2: Realizo con Weka 5 Clusters mediante SimpleKMeans con EuclideanDistance

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

La “carga proporcionada” se ve en la cantidad de trabajo o demanda que se impone al sistema. En el contexto de este sistema de almacenamiento de datos o transferencia de archivos, implica la cantidad de datos que se están transfiriendo o la cantidad de operaciones de lectura y escritura que se están llevando a cabo.

Una característica especial que he observado en la carga proporcionada es la existencia de patrones. Por ejemplo, el tamaño de los archivos es una consideración importante. Como mencione anteriormente, la mayoría de las transferencias se realizan con archivos más pequeños. Esto sugiere una característica especial en la carga proporcionada es que está compuesta en su mayoría por transferencias de archivos de menor tamaño.

En resumen, al analizar la carga proporcionada en un sistema, es importante considerar varios factores. Además, la distribución de los datos es bastante igualada entre los clústeres. Este patrón inusual sugiere que los datos introducidos en Weka tal vez no eran los más apropiados para obtener los resultados que se esperaban. Es evidente que otro factor característico en la distribución es tener la mayoría de los datos concentrados en los valores más cercanos al cero, disminuyendo a medida que aumenta el eje de las abscisas.

Por último, para obtener una mejor representación y análisis de los clústeres, sería útil realizar una agrupación cualitativa previa basada en el tamaño de los archivos. Esto permitiría realizar un clúster para cada grupo de datos, lo que probablemente revelaría patrones más interesantes en los datos. Esto remarca la importancia del preprocesamiento de datos y construir los grupos o clusters.

Información relevante para la práctica:

- Enlace a la web oficial de Weka: <https://www.cs.waikato.ac.nz/ml/weka/>