UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Caracterización de series temporales de tráfico de red mediante la transformada Wavelet

Autor: Álvaro Delgado Sancho
Tutor: Daniel Perdices Burrero

junio 2024

Agradecimientos

Me gustaría mostrar mi agradecimiento, en primer lugar, a mi tutor Daniel Perdices por su gran ayuda y disposición a lo largo de todo el desarrollo de este trabajo, así como a la totalidad del grupo de investigación HPCN (*High Processing Computing and Networking*) de la EPS (Escuela Politécnica Superior) de la UAM (Universidad Autónoma de Madrid).

Además, dar las gracias a todo el equipo docente de la EPS por acompañarme durante estos 4 años de grado poniendo su tiempo a disposición de una formación de calidad, y por su alta exigencia y motivación que me han ayudado a ser un mejor estudiante y profesional.

Por otro lado, agradecer a mis padres, Francisco Javier y María de los Milagros, y hermanos, Francisco Javier y Verónica, junto al resto de familiares y amigos por el afecto y apoyo incondicional mostrado durante todo este proceso de aprendizaje universitario. Finalmente, agradecer a mi pareja Paula por su cariño y motivación.

En resumen, agradecer a todas las personas que hayan podido ayudar o tener un impacto, con mayor o menor intensidad, en la realización de este Trabajo Fin de Grado y que han hecho posible que este trabajo haya llegado a buen término.

Resumen

El panorama actual de los sistemas de monitorización y gestión de redes de comunicaciones y sistemas augura un futuro lleno de novedades y cambios en el paradigma de actuación. Ahora mismo, la mayoría de las organizaciones utilizan sistemas basados en líneas base, los cuales se basan en el comportamiento anterior en el tiempo para dar predicciones futuras.

El objetivo de este trabajo es introducir un nuevo método de caracterización de series temporales de segmentos de red para detectar comportamientos anómalos y evitar problemas de seguridad en la infraestructura de los sistemas. Nuestra propuesta se basa en la eliminación de la componente temporal en las series de cara a poder identificar series temporales con cierta similitud en el tipo de comportamiento, pero muy difíciles de predecir.

A partir de esta idea hemos desarrollado distintos métodos para conseguir este objetivo, junto a una serie de pruebas y experimentos sobre un conjunto masivo de datos reales que nos han permitido observar el comportamiento de los algoritmos implementados.

Como resultado de este trabajo, hemos sido capaces de proveer una nueva manera de modelar el comportamiento de los sistemas y usuarios, lo que permite detectar incidencias de manera automatizada descargando de responsabilidad y carga de trabajo a los administradores de sistemas.

Palabras clave

Ondícula, agrupamiento, detección de anomalías, series temporales, gestión de red.

Abstract

The current landscape of monitoring systems and network communication management anticipates a future full of changes and novelties in the operational paradigm. Currently, most organizations use baseline-based systems, which rely on previous behavior over time to make future predictions.

The objective of this work is to introduce a new characterization method for time series in network segments to detect anomalous behavior and prevent security issues in system infrastructure. Our proposal is based on the removal of the temporal component from the series in order to identify time series with certain similarities in their behavior, which otherwise are very difficult to predict.

Based on this idea, we have developed various methods to achieve this goal, along with a series of tests and experiments on a massive dataset that have allowed us to observe the behavior of the implemented algorithms.

As a result of this work, we have been able to provide a new way of modeling system and user behaviors, which allows automated detection of incidents, thereby reducing the responsibility and workload of system administrators.

Keywords

Wavelet, clustering, anomaly detection, time series, network management.

Índice

Índice de figuras

1 Introducción

En los últimos años, investigadores y administradores de sistemas han tomado una gran relevancia en el ámbito de la monitorización de servicios de red con el objetivo de detectar comportamientos extraños y generar alertas para dichas situaciones. Esto sumado a la creciente capacidad y carga de los servidores de empresas modernas y CPD (Centro de Procesamiento de Datos), ha hecho de este proceso algo bastante complejo y laborioso, debido a la variedad en el comportamiento de los servidores y subredes de dichas organizaciones. El objetivo principal de esta labor es encontrar soluciones que se adapten al comportamiento concreto de los servidores o segmentos de red a analizar. La gran cantidad de información y datos de series temporales, así como el limitado tiempo de los investigadores y administradores, dificulta su manejo y nos sugiere poner la vista en soluciones basadas en soluciones automáticas que puedan hacer frente al procesado masivo de datos.

La monitorización de segmentos de red es una herramienta muy útil para caracterizar las series temporales del tráfico de segmentos de red, y poder así agruparlas respecto a un comportamiento similar. La agrupación de estas series nos permite ver un comportamiento común sobre el que poder detectar anomalías o situaciones de cambio relevantes en la dinámica de comportamiento del segmento de red.

La detección de estos cambios o anomalías suponen una gran ventaja para las organizaciones y empresas de cara a poder distribuir sus recursos de manera eficiente y lógica, y tener capacidad de previsión acerca de posibles subidas o bajadas de carga en sus servidores. La incapacidad para prever este tipo de situaciones podría suponer la pérdida de disponibilidad de servicios o el derroche de estos en una red infrautilizada, lo que se traduce en problemas económicos para la empresa.

Para la caracterización de estas series temporales vamos a utilizar métodos de *clustering*, que de manera automática agrupan los datos en el número de clústeres elegido teniendo en cuenta posibles relaciones entre los datos de su mismo clúster y diferencias más o menos significativas frente a datos de otros clústeres.

Para el proceso de agrupación de las series temporales, teniendo en cuenta la alta dimensión que poseen estas, sería necesario hacer uso de alguna técnica de reducción de componentes como PCA (*Principal Component Analysis*) [1], t-SNE (*t-Distributed Stochastic Neighbor Embedding*) [2] o UMAP (*Uniform Manifold Approximation and*

Projection) [3]. Posteriormente, se utilizarían métodos de *clustering* como *k-means* [4] o DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [5] para generar los resultados en base a la reducción de componentes previamente realizada.

Esta reducción de componentes nos facilita la visualización de los datos para su posterior interpretación, cuyo concepto surge de la idea de que la representación de una secuencia o grupo de datos puede basarse en un subconjunto más pequeño de variables respecto al inicial, manteniendo siempre la medida de la varianza entre los datos. La varianza es la métrica que ayuda a distribuir los datos sobre los distintos clústeres y que mide la diferencia o variabilidad de los datos respecto al resto, por lo que, al reducir el número de componentes se debe mantener dicha variabilidad para que la distribución de los datos se mantenga correctamente, que es justamente la idea que implementan los métodos de visualización como PCA.

Una vez agrupados los datos en los distintos clústeres, la idea es poder categorizar las relaciones o patrones que puedan existir entre las series temporales pertenecientes a un mismo clúster, de manera que podamos encontrar información útil para utilizarla posteriormente en detección de anomalías o similares.

1.1. Motivación

Una manera inteligente de poder afrontar y detectar comportamientos anómalos es la de realizar predicciones, lo más precisas posibles, del comportamiento futuro de la carga de los segmentos de red. La longitud en el tiempo (horas, días, meses…) de estas predicciones puede variar dependiendo del tipo de comportamiento a detectar y del nivel de cambio en la carga de los segmentos de red.

Esta manera de actuar es eficaz en sistemas donde la carga del segmento de red está bastante relacionada con la componente temporal, es decir, segmentos de red en los que la carga durante un periodo temporal se repite de manera periódica y estable. Por lo contrario, en sistemas donde la carga es muy variable y caótica, la predicción de anomalías futuras sería mucho más compleja, lo cual abre otro posible campo de investigación para mitigar este tipo de problemática.

Sin embargo, queremos encontrar una forma de eliminar la temporalidad en el proceso de caracterización de las series temporales para el posterior intento de detección de anomalías en dichas series. El hecho de eliminar la componente temporal de los datos nos da la posibilidad de dejar a un lado los comportamientos periódicos y estacionales que puedan entorpecer la visualización de patrones de mayor relevancia. Esta idea puede suponer avances en la caracterización de series temporales, las cuales son aplicables a distintos campos de dentro y fuera de la informática.

1.2. Objetivos

El principal objetivo de este trabajo es el de investigar y encontrar nuevas estrategias de caracterización de series temporales. De manera más concreta, queremos investigar acerca de una estrategia de caracterización de series temporales basada en la eliminación de la componente temporal de las señales para poder encontrar patrones o similitudes que no tengan relación con la temporalidad de la señal. El desarrollo de un método de estas características que funcione de manera eficiente supondría un gran avance en materia de detección y caracterización de anomalías en los sistemas. Este tipo de anomalías no están completamente identificadas actualmente debido a la dificultad de detectarlas, por lo que esta investigación podría suponer un ámbito de estudio futuro.

Concretando en la lista de objetivos, uno de ellos es poder llevar a cabo la representación gráfica de las series temporales en crudo, y, en base al método desarrollado, poder realizar una posterior comparación de las similitudes y diferencias entre series temporales que se hayan clasificado de manera similar.

Otro objetivo marcado es aplicar la transformada *Wavelet* a dichas series temporales para la identificación de patrones de comportamiento. De este modo podremos llevar a cabo la comparación previamente mencionada de manera visual.

Finalmente, mostrar a lo largo de una serie de experimentos y pruebas la validez y eficiencia del método que hemos desarrollado y contrastarlos a través de sus resultados. De esta manera, podremos comprobar si este método debe ser desarrollado en mayor profundidad o simplemente resulta en una hipótesis errónea que no proporciona los resultados esperados.

1.3. Estructura del documento

Este documento se encuentra estructurado en capítulos de la siguiente forma: en el capítulo 1 se proporciona una introducción acerca del tema a tratar por el trabajo, exponiendo las motivaciones y objetivos. Seguido a esto, en el capítulo 2 se expone de manera resumida el estado del arte relacionado con el tema del trabajo, aportando información sobre técnicas utilizadas y artículos relacionados. En tercer lugar, en el capítulo 3 se explica el desarrollo e implementación de los algoritmos y métodos utilizados. En el capítulo 4 se realizan una serie de pruebas y experimentos en un conjunto masivo de datos de cara a comprobar la validez de los métodos desarrollados. Finalmente, se presentan una serie de conclusiones e ideas para posibles trabajos futuros en el capítulo 5.

2 Estado del Arte

El uso de las series temporales se ha extendido en los últimos años a múltiples disciplinas dentro del campo de las ciencias, como pueden ser la medicina, la economía o la misma ingeniería. Las series temporales son esencialmente secuencias de información medidas a lo largo de un periodo de tiempo, y su análisis puede ser muy útil para detectar patrones de comportamiento y realizar predicciones futuras. A diferencia de otro tipo de estructuras, la importancia de estas secuencias de datos reside en la existencia de una temporalidad asociada a los datos.

En el apartado de la ingeniería, más concretamente en el análisis de sistemas o segmentos de red, la caracterización de series temporales supone una herramienta para la detección y predicción de anomalías y comportamientos no deseados. Por ello, actualmente se investiga mucho acerca de este tema, ya que el desarrollo de sistemas eficientes y automáticos que realicen esta tarea supondría un gran cambio en términos de seguridad y eficiencia operativa. Una rápida capacidad de detección podría prevenir fallas en un sistema o la propagación de ataques externos.

2.1. Artículos relacionados

La caracterización de series temporales en redes de comunicaciones es un tema aún candente en el estado del arte. Por lo general, las investigaciones más recientes buscan la automatización de esta tarea de caracterización, intentando llegar a este objetivo de diferentes maneras.

Por ejemplo, en el caso de la caracterización de series para detección de anomalías, existen artículos que explican métodos innovadores [6], [7]. El artículo [6] explica la implementación de una unidad de alarma que convierte una tarea supervisada, como la detección de anomalías, en una tarea automatizada. Otra idea que desarrolla es la adecuación del umbral de detección de alertas en base a la carga de trabajo del administrador, para no generar alertas que finalmente no sean atendidas. Finalmente se evalúan las condiciones de cada alerta y se les asigna un nivel de relevancia, así como el tiempo que se empleará en atenderlas para generar un calendario de tareas.

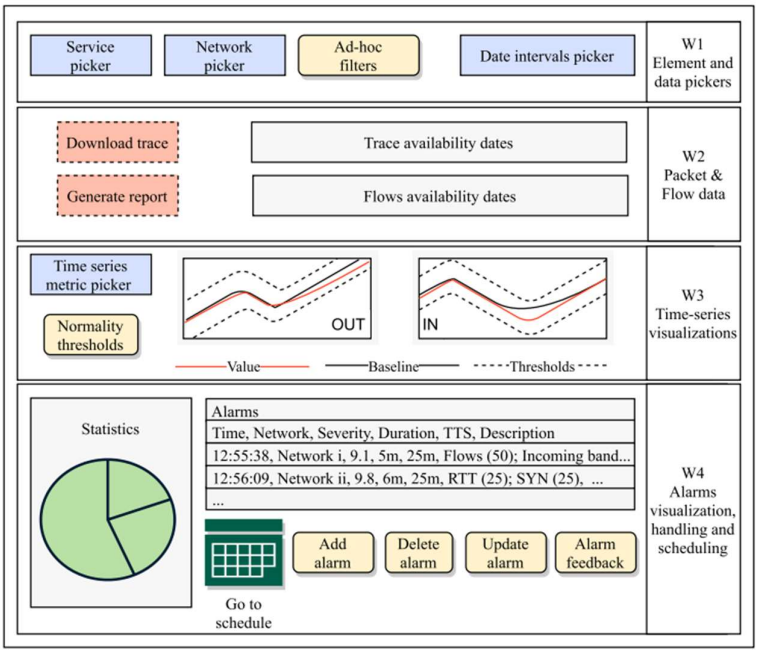


Figura 2.1: Estructura general del sistema de monitorización. *Extraído de* [6].

La figura 2.1 representa la estructura general de sistema de monitorización tratado en este trabajo, en el que se muestran las distintas funcionalidades que puede realizar. Existen distintas unidades de actuación que se encargan de tareas muy variadas y modularizadas. Para algunas de estas tareas se utilizan redes neuronales recurrentes LSTM (*Long Short-Term Memory*) [8], que permiten introducir información y resultados pasados como *input* en otras predicciones. La estructura de la red neuronal explicada y utilizada en el trabajo se muestra en la figura 2.2.

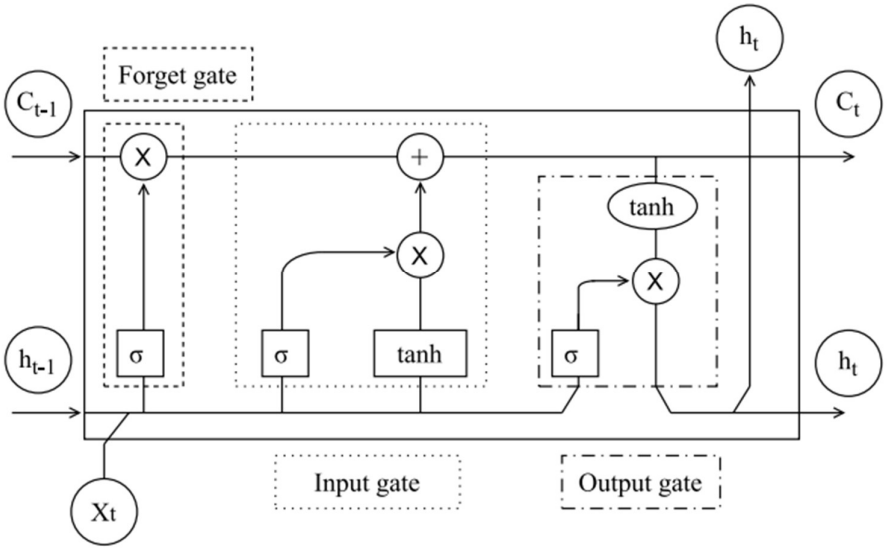


Figura 2.2: Diagrama de la red neuronal LSTM. *Extraído de* [6].

Otro estudio [7] relacionado con la caracterización de series temporales utiliza redes neuronales en conjunto con *Deep-FDA* (*Deep Functional Data Analysis*) para automatizar

dicha tarea. Los autores utilizan métodos clásicos como PCA o *k-means*, y a su vez, utilizan herramientas como los *autoencoders*, que son redes neuronales pensadas para comprimir la información de entrada a su estado esencial para luego reconstruirla para obtener una salida. La estructura se explica perfectamente en la figura 2.3.

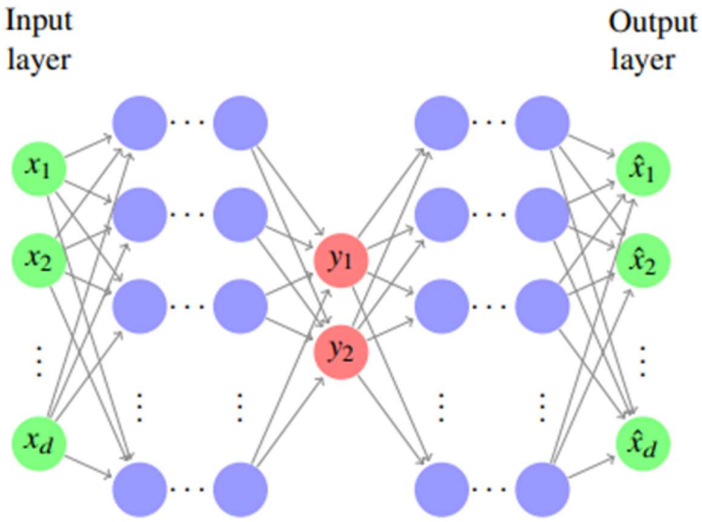


Figura 2.3: Estructura de un autoencoder (AE). *Extraído de* [7].

De manera conjunta al uso de estas estructuras se utiliza FDA (*Functional Data Analysis*) con herramientas de aprendizaje profundo. El FDA analiza funciones que pueden ser analizados como si fueran datos o vectores. De esta manera, en vez estructuras de datos clásicas, se pueden realizar análisis sobre funciones o curvas de manera más sencilla. Alguno de los métodos que se utilizan para ello son PCA y FPCA (*Functional Principal Component Analysis*) [9].

La parte del aprendizaje profundo se implementa a través de redes neuronales con varios niveles de capas, ya que este modelo de red neuronal es muy eficiente detectando patrones de comportamiento complejos. La unión de estos 2 métodos ofrece una manera moderna y potente de analizar conjuntos grandes de datos y obtener mediciones y resultados muy útiles a partir de ellos.

Finalmente, existen investigaciones [10], [11] en las que se relaciona el proceso de clasificación de series temporales con el uso del *deep learning.* El autor del artículo [11] se sorprende del poco uso que se les ha dado a las redes neuronales para el análisis de series temporales, por lo que hace un repaso de diferentes tecnologías relacionadas para comprobar su comportamiento con las series. Según comenta, pocas investigaciones han abordado el uso de DNNs (*Deep Neural Networks)* [12], [13] para la automatización de estas tareas, teniendo en cuenta la creciente moda del uso de este tipo de herramientas.

En el artículo [11], más que una investigación como tal, se realiza un repaso de todas las herramientas que utilizan el aprendizaje profundo para el trabajo con las series

temporales. En este trabajo, el autor muestra la capacidad de distintas tecnologías para el análisis de las series y proporciona los resultados que ofrecen cada uno de ellos.

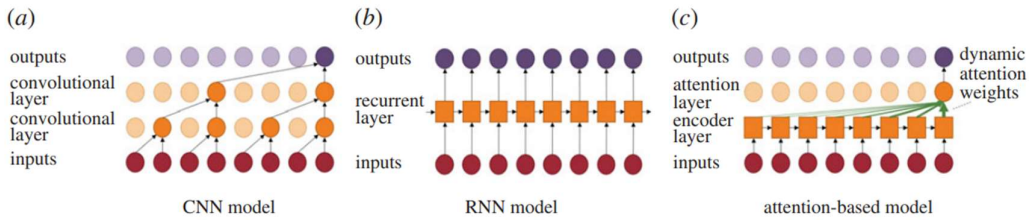


Figura 2.4: Estructura de diferentes arquitecturas codificadores. *Extraído de* [11].

En la figura 2.4 se muestran las arquitecturas de varios modelos de redes neuronales. La primera de ellas es CNN (*Convolutional Neural Networks)* [14], las cuales utilizan capas de convolución para el aprendizaje de comportamientos o dinámicas de los datos para mostrar resultados. Este tipo de redes son muy potentes en su uso con series temporales e imágenes.

Otro de los modelos mostrados es el RNN (*Recurrent Neural Networks)* [8], cuya peculiaridad reside en su uso sobre datos secuenciales, como es el caso de las series temporales. Este modelo trabaja con capas recurrentes que actúan como una especie de "memoria". Esto es así gracias a su estructura en la que se utilizan los *inputs anteriores* como si fuera otra entrada en el estado actual, por lo que se introduce información de estados pasados para influir en los futuros.

Finalmente, el último modelo que estudia es el de redes neuronales basada en atención [15]. Este modelo de red neuronal es curioso, ya que posee la capacidad de centrar todo su atención y potencia de cómputo en los puntos de entrada más importantes. Esto se implementa a partir de un método de puntuación por el que se evalúa cada uno de los elementos de entrada para encontrar los más influyentes de cara a la salida.

2.2. Métodos preliminares

En el ámbito del análisis y la caracterización de las series existen numerosos métodos destinados a estudiar diferentes características y dinámicas. Una técnica muy utilizada en este ámbito es la transformada de Fourier, que descompone una señal en sus componentes de frecuencia, es decir, representa las ondas sinusoidales complejas que componen la señal original. Este método es muy útil para señales estacionarias, estas son las que no tienen ningún cambio a lo largo del tiempo y se mantienen "constantes". Por el contrario, este tipo de transformación no tiene en cuenta la componente temporal, por lo que es bastante ineficiente para el análisis de señales no estacionarias o, mejor dicho, aquellas que tienen variación de onda a lo largo del periodo temporal.

Otro de los métodos más utilizados para este objetivo es la transformada Wavelet, la cual, si posee la capacidad de analizar las distintas frecuencias a lo largo del tiempo, lo que la hace muy eficiente frente a señales no estacionarias, a diferencia de la transformada de Fourier. Esta transformada utiliza diferentes escalas para estudiar patrones o comportamientos a distinto plazo, esto es debido a la capacidad de la señal para estirarse o comprimirse en base al valor de la escala.

A la hora de representar las transformaciones realizadas por alguno de estos métodos, son muy utilizadas técnicas como PCA o t-SNE para la reducción de la dimensionalidad de las series temporales. Esta reducción es necesaria de cara a la visualización gráfica de estas transformaciones que, por lo general, suelen poseer numerosas componentes en su haber. Gracias a estas herramientas podemos observar de manera gráfica, normalmente en 2 dimensiones, y poder realizar un mejor análisis de los datos.

3 Desarrollo

A lo largo del proyecto, la hipótesis general se ha elaborado de forma organizada, siguiendo una serie de pasos claramente estructurados. La primera tarea al comenzar un proyecto de investigación es siempre familiarizarse con las herramientas de trabajo que se van a utilizar, la información y datos que se manejarán y proponer un método de trabajo, y en este caso no ha sido diferente.

3.1. Familiarización con los datos

Desde el comienzo del proyecto, tenía conocimiento de que los datos e información utilizados para llevar a cabo diversas pruebas y análisis provenían de los servidores de una empresa energética. Dichos datos son administrados por el grupo de investigación HPCN (*High Processing Computing and Networking*) de la Escuela Politécnica Superior. La cantidad de datos proporcionados es bastante elevada, concretamente, corresponden al tráfico de red de los servidores de dicha empresa a lo largo de varios años. De cara a trabajar con los datos proporcionados de manera más flexible y manejable, hemos acotado el periodo del que hemos tomado los datos. Por ello, hemos decidido centrarnos en los datos recogidos durante 6 meses, entre el 1 de enero de 2021 y el 30 de septiembre de ese mismo año. Para poder trabajar de manera uniforme se ha realizado un preprocesado de los datos para eliminar discontinuidades en las mediciones o datos erróneos.

Adentrándonos en la estructura de los datos, estos se disponen en archivos nombrados con el día correspondiente, donde se encuentra la información diaria acerca de la carga de los servidores. En cada uno de los archivos se proporciona una tabla con distintas columnas, como número de conexiones, *bits* enviados/recibidos por segundo, referencias de tiempo, etc. Cada fila de estas tablas contiene la información de las distintas conexiones realizadas a un servidor sobre diferentes direcciones IP.

Para proteger la confidencialidad de las direcciones IP y los datos pertenecientes a la empresa, ambos utilizados en este trabajo, se ha eliminado el último octeto en la nomenclatura de las IP de todas las representaciones gráficas donde aparezcan este tipo de datos. A modo de sustitución, los dígitos eliminados por privacidad se han sustituido por los caracteres XY, siendo XY números enteros que, en ningún caso, poseen el mismo valor para varias direcciones.

3.2. Representación gráfica

Una vez estudiada la estructura y amplitud de los datos que se han utilizado para probar la hipótesis, la siguiente tarea que tenemos que realizar es una representación gráfica de las distintas series temporales.

Para ello, y debido a la gran variedad de datos en las distintas tablas, se consideró oportuno agrupar los datos según la dirección IP a la que se conectaban, recordando que cada fila de las tablas representa una conexión a un servidor. De esta manera, logramos agrupar las conexiones a direcciones concretas y estudiar las más relevantes, es decir, en nuestro caso, ordenando los datos a partir del número de conexiones realizados a dichas direcciones IP.

Una vez obtenidos los datos más relevantes (direcciones IP con mayor número de conexiones), la idea es representarlos y poder observar diferencias de la carga de red sobre las diferentes direcciones. Para evitar representaciones con información demasiado escasa que impidiera un posible análisis o estudio, se restringió el proceso de representación gráfica a aquellas direcciones que tenían al menos 1000 conexiones.

Excluyendo las direcciones mencionadas, el resto de ellas que cumplían con el número mínimo de conexiones se han representado gráficamente. En el eje horizontal se representa la componente temporal, y en el eje vertical se indican los *bits* totales recibidos, a los que podríamos denominar como “carga”. Sin embargo, para poder realizar comparaciones entre direcciones con distinta carga, se ha aplicado una normalización al eje vertical dividiendo cada valor por el valor máximo de *bits* recibidos. De esta manera, todas las gráficas generadas tienen valores verticales comprendidos en el intervalo [0, 1], lo cual hace que podamos tener a una misma escala todas las representaciones gráficas independientemente del tamaño de unas u otras.

Como es habitual en el entorno profesional, cabe comentar que los servidores pueden experimentar periodos de carga mínima o nula, ya sea por mantenimiento, por problemas de funcionamiento o simplemente por el ciclo de trabajo de la propia empresa. Para evitar distorsionar en exceso las representaciones gráficas de las direcciones IP, los periodos en los que existe una falta de datos se han sustituido por valores nulos, es decir, por ceros.

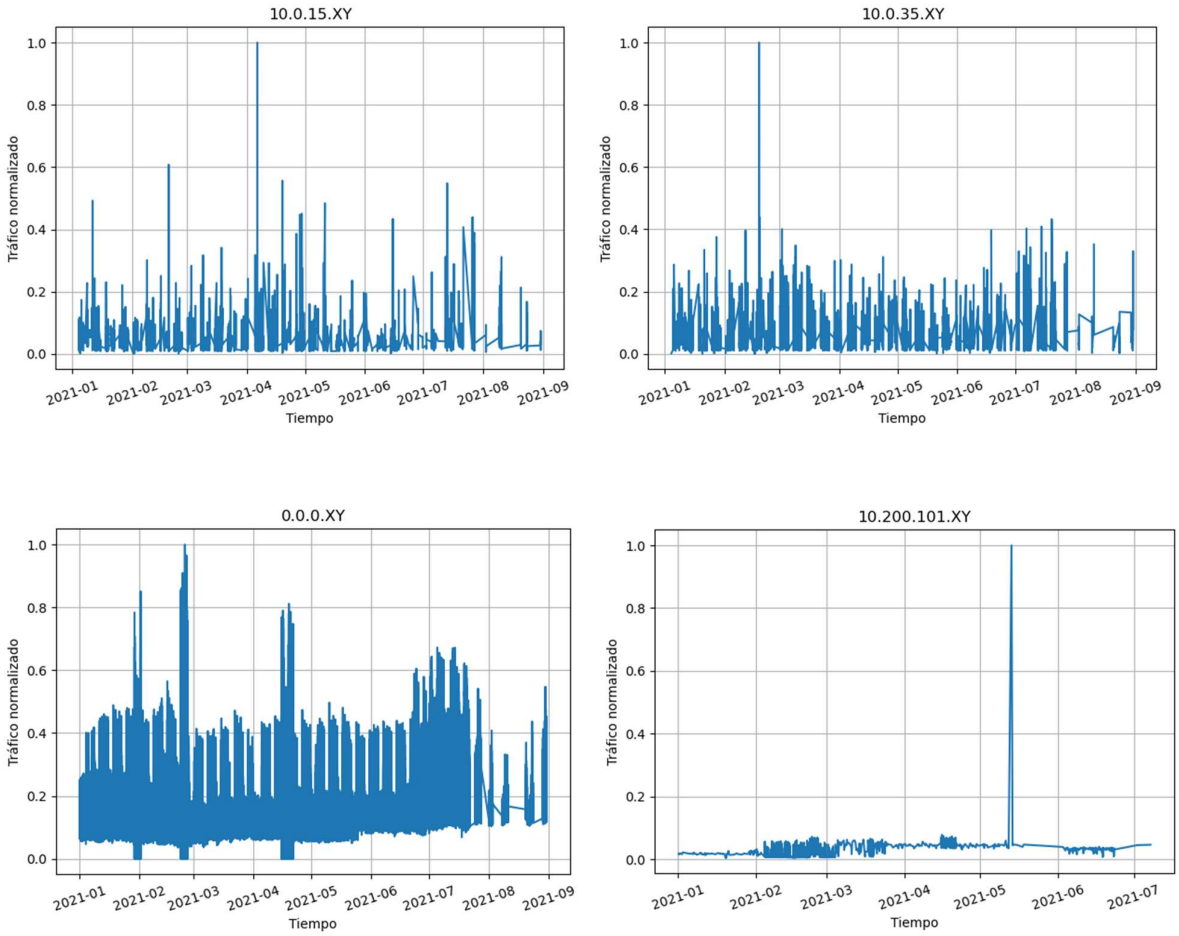


Figura 3.1: Ejemplo de series temporales

Las gráficas de la figura 3.1 muestran varios ejemplos de las posibles representaciones del tráfico de red de distintas direcciones IP, en las que se pueden diferenciar de manera clara un comportamiento variable en cada una de ellas. Existen direcciones con una carga mayor y más continuada como puede ser la de la dirección 0.0.0.XY que difieren mucho de otras como la 10.200.101.XY, que posee una carga bastante baja a lo largo de todo su periodo temporal, excepto en un momento muy concreto que recibe un pico de carga bastante más alto de lo habitual.

Cabe comentar que la IP 0.0.0.XY corresponde realmente a la dirección 0.0.0.0 que, en la mayoría de los casos, incluido el nuestro, suele representar el agregado de la monitorización de todo el sistema, es decir, la totalidad de todo el tráfico del sistema.

El comportamiento de las series temporales es muy diverso y cambiante entre unas y otras, y son precisamente esas diferencias las que, tras la transformación de las series y su caracterización eliminando el eje temporal, queremos poder estudiar y catalogar.

3.3. Transformada Wavelet

3.3.1. Definición

La transformada *Wavelet* [16], [17] es una herramienta matemática utilizada en el análisis y procesamiento de datos, principalmente de señales. Es especialmente potente con señales no estacionarias, que son aquellas que poseen algún tipo de variación en su frecuencia a lo largo del tiempo. La potencia de la transformada *Wavelet* reside en su capacidad para retener información, tanto del tiempo, como de la frecuencia, para poder así realizar análisis locales del comportamiento de la señal.

La diferencia respecto a otro tipo de transformaciones, como pueden ser la de *Fourier* [18], [19] o la STFT (*Short-time Fourier Transform*), que utilizan bases sinusoidales, reside en la estructura de sus propias funciones base, las cuales poseen una estrecha relación entre la frecuencia y el tiempo. Estas funciones base ayudan a descomponer las señales y obtener sus diferentes niveles de resolución para poder analizar comportamiento multiescalar. Los autores de [20] definen la transformada *Wavelet* de la siguiente forma:

"La Transformada Wavelet no es estrictamente un método estadístico de reconocimiento de patrones, sino que es un método de preprocesamiento que permite que los datos sean expresados más sucintamente." Montoya J. R. A.

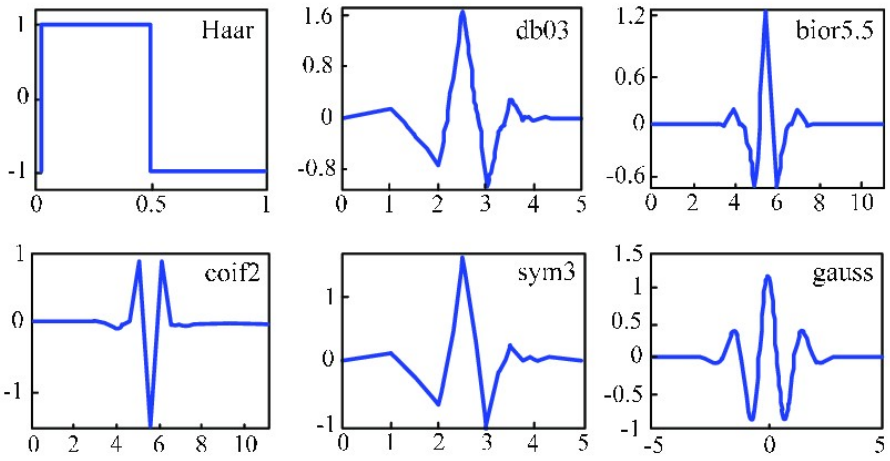


Figura 3.2: Ejemplos de funciones base Wavelet

En la figura 3.2 se muestran varias familias de *Wavelets* como la *Wavelet* Haar, la *Wavelet Daubechies*, la *Wavelet* Gauss, etc. Según el tipo de datos que se manejan o los resultados que se desean obtener con esta técnica, se seleccionan diferentes funciones base. Normalmente, la función *Wavelet* elegida como base se la denomina *Wavelet* madre. La *Wavelet* madre sufre distintas modificaciones respecto a los valores de los conceptos de escala y desplazamiento. La fórmula de la wavelet madre se muestra en la ecuación 3.1.

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\,\psi\left(\frac{t-b}{a}\right)$$

(3.1)

donde **a** se corresponde al concepto de escala y **b** al de desplazamiento o traslación. La escala es la responsable de comprimir o estirar la *Wavelet* madre de cara a analizar distinto nivel de detalle acerca de una señal. Por otro lado, el desplazamiento define el camino de la *Wavelet* madre a lo largo de la señal que se quiere estudiar. Haciendo uso de estos conceptos, la transformada *Wavelet* descompone las señales en sus distintos niveles y analiza cada uno de manera separada.

La transformada *Wavelet* continua [21], como su nombre indica, utiliza conjuntos continuos de escala y desplazamiento, permitiendo analizar detalladamente todo el espectro de dichos conjuntos. La fórmula de la transformada continua se muestra en la ecuación 3.2.

$$CWT(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} x(t)\psi\left(\frac{t-b}{a}\right)dt$$

(3.2)

donde **x(t)** se corresponde con la señal a estudiar. La transformada nos proporciona una serie de coeficientes en base a los valores de escala y desplazamiento, los cuales nos otorgan una idea acerca de la similitud entre la *Wavelet* madre seleccionada y la señal estudiada. A partir de estos coeficientes es posible realizar diversos análisis matemáticos o científicos mucho más interpretables. Esta operación transforma una señal completa a un conjunto de valores numéricos, los cuales son mucho más sencillos de manejar.

La CWT realiza un análisis exhaustivo de la señal completa, generando cantidades de datos abundantes y la necesidad de mayor capacidad de procesamiento que el otro tipo de transformada wavelet que se explica, la discreta.

La transformada *Wavelet* discreta [22] se corresponde con una versión de la transformada continua a la que se le aplica un proceso de discretización de los valores de escala y desplazamiento. De esta manera, se convierte el proceso de análisis continuo a otro con un conjunto de valores finito, dando paso a la representación de la señal a partir de sus funciones base.

Una de las mejores formas de discretizar la escala y el desplazamiento, es a partir de un conjunto de valores dado por el uso de una función exponencial. Los valores

discretizados de escala y desplazamiento se toman a partir de las ecuaciones 3.3.

$$\boldsymbol{e = a^{-j}}$$

$$\boldsymbol{d = kna^{-j}}$$

(3.3)

donde nuevamente, **a** se corresponde al concepto de escala, b al de traslación y **kn** a un valor entero. Aunque es posible utilizar distintos valores para el factor *a*, un valor habitual para dicho factor es 2^{-j}. De esta manera, al sustituirla en la *Wavelet* madre se obtiene la ecuación 3.4.

$$\boldsymbol{\psi_{j,k}(t) = 2^{\frac{j}{2}}\psi(2^j t - kn) \quad j,k \in \mathbb{Z}}$$

(3.4)

Es por ello por lo que, a partir de esta *Wavelet* madre discretizada, la fórmula final de la transformada *Wavelet* discreta se presenta en la ecuación 3.5.

$$\boldsymbol{DWT_{j,k} = 2^{\frac{j}{2}} \int_{-\infty}^{\infty} x(t)\,\psi(2^j t - kn)}$$

(3.5)

donde **x(t)** se corresponde con una señal discreta. Tomando la ecuación 3.5, tenemos una forma de generar un análisis discretizado sobre una señal. Este método es mucho menos exigente en términos de capacidad de procesamiento y almacenamiento de datos, ya que reduce considerablemente el conjunto de valores de escala y desplazamiento a utilizar.

A modo de resumen, podríamos decir que la transformada *Wavelet* continua proporciona un análisis con mayor grado de detalle, e incluso de redundancia, ya que se generan muchas repeticiones de datos, todo ello sacrificando la eficiencia y capacidad computacional. En cambio, la transformada discreta aplica un subconjunto de los valores que se utilizan en la continua sobre los conceptos de escala y desplazamiento, proporcionando un análisis con mayor rango de amplitud (resolución) y menor detalle, a cambio de un menor uso computacional y la capacidad de generar análisis en distintos niveles de resolución.

3.3.2. Aplicación

Una vez comprendida la idea general de la transformada Wavelet, empezamos a trabajar con la librería *pywt* de Python, que proporciona una gran cantidad de funcionalidad para este tipo de transformadas.

Para realizar una transformada *Wavelet* con esta librería existen varios métodos, sin embargo, nosotros hemos elegido el método continuo mediante la función *cwt(),* la cual necesita de una serie de parámetros para su configuración. Inicialmente, se le pasa la señal que deseamos transformar, que en nuestro caso se corresponde al eje vertical de nuestras representaciones, es decir, a los *bits* recibidos. También se debe indicar el tipo de transformada que se le aplica a la señal. Previamente se han mencionado varias familias de transformadas, y en mi caso he utilizado la gaussiana para realizar esta transformación, debido a su forma suave que otorga una convergencia veloz. Finalmente, se le pasa un conjunto de escalas sobre las que itera la propia transformada para proporcionar distintos niveles de detalle.

La función mencionada devuelve 2 estructuras con valores diferentes. La primera es un vector de 2 dimensiones, donde se devuelven los valores de escala y tiempo de la señal original. La segunda estructura devuelta es otro vector con las frecuencias respectivas a cada una de las escalas utilizadas en las transformaciones.

El concepto de escala en la transformada *Wavelet* se relaciona con la capacidad de estirar o comprimir la señal en búsqueda de diferentes características. Utilizando escalas bajas podemos comprimir la señal y realizar un análisis más detallado de la señal. Por lo contrario, con escalas altas podemos estirar la señal y recoger características o patrones de índole general. Debemos tener en cuenta que la escala es inversamente proporcional a la frecuencia, por lo que en escalas pequeñas estamos estudiando frecuencias altas, y viceversa.

A su vez, el concepto de tiempo, también denominado traslación o desplazamiento, permite analizar el cambio de las frecuencias a lo largo de la señal. El desplazamiento se compara con una ventana temporal que se mueve a lo largo de la señal para analizar las partes de la señal en distintas resoluciones temporales. Es esta segunda lista de frecuencias la que nos interesaba para poder formar los descriptores numéricos para la señal. Dado que para cada una de las direcciones se generaba una lista distinta de descriptores, la manera de agrupar estas listas en un solo valor ha sido realizando una media aritmética de todos sus valores.

Como se ha comentado previamente, la primera estructura devuelta contiene los coeficientes para las distintas escalas y tiempos de la señal. El objetivo de esta transformada es obtener descriptores numéricos pertenecientes a las distintas direcciones

IP, por lo que al tener una matriz de 2 dimensiones se complica la manipulación de estos datos. De cara a facilitar dicho manejo de los datos y cumplir los objetivos propuestos, se planteó realizar una suma del eje Y de dicha matriz para así obtener simplemente una lista de valores. En resumen, tras realizar la operación de la transformada *Wavelet* sobre los datos de todas las direcciones IP, se obtienen listas de valores numéricos, es decir, los descriptores deseados, que identifican a cada una de estas direcciones. Para su posterior estudio y análisis, estos descriptores se han guardado en ficheros de texto plano.

Todo este proceso de la transformación de las señales, aunque pudiera parecer que simplemente se ha realizado en una ocasión, se ha tenido que realizar en varias ocasiones, esto es debido a los diferentes estudios que hemos querido realizar sobre los datos. Para realizar un estudio más completo, se ha realizado este proceso para 3 escalas diferentes siendo estas diaria, semanal y exponencial.

Debemos tomar en cuenta que cada unidad de escala corresponde aproximadamente a 5 minutos de tiempo real, por lo que dependiendo de la escala que deseamos estudiar, tenemos que calcular el equivalente a un día o una semana. En el caso de la exponencial, el concepto es algo diferente. La escala introducida va saltando por las potencias de 2 hasta la escala máxima introducida, que en este caso ha sido 1024 (2^10). Con esto se consigue utilizar los datos más relevantes sin tener que ir pasando por todas las escalas.

Una vez terminados dichos procesos de transformación es el momento de procesar los descriptores de todas las direcciones IP y caracterizar dichos descriptores para buscar similitudes y diferencias entre series temporales.

3.4. Clustering

3.4.1. Definición

El *clustering* consiste en una técnica o conjunto de procesos cuyo objetivo es el de dividir conjuntos de datos en distintos grupos. Es una herramienta muy utilizada en aprendizaje automático para la detección de patrones en conjuntos masivos de datos, los cuales serían difíciles de identificar de manera manual debido a su tamaño. Concretamente, el *clustering* es un algoritmo de aprendizaje automático no supervisado, el cual no dispone de un entrenamiento previo con conjuntos de prueba y que tiene el objetivo de crear una organización de los datos en grupos que compartan algún tipo de estructura o patrón.

Los distintos grupos en los que se dividen los datos se denominan clústeres, los cuales concentran conjuntos de datos que poseen una determinada similitud entre ellos.

Para el cálculo de dicha similitud entre datos existen distintas técnicas que brindan resultados diferentes. Algunas de las técnicas más utilizadas para esto son k-means, *clustering* jerárquico o DBSCAN.

En el ámbito de estos algoritmos, los conceptos de similitud y distancia se disponen esenciales para la comprensión de su funcionamiento. En el caso de k-means, que ha sido el algoritmo mayormente utilizado en este trabajo, estos dos conceptos se relacionan íntimamente. Para determinar el clúster al que pertenece un punto concreto se utiliza la distancia, generalmente la euclidiana, que calcula la distancia entre 2 puntos, en este caso, el centroide de un clúster y el punto pendiente de asignar. Tras el cálculo de las distancias con todos los centroides, se elige aquel clúster cuyo centroide tenga la menor distancia de entre todas las existentes y se asigna el punto a dicho clúster.

Inicialmente, se deben elegir las componentes o propiedades sobre las que el algoritmo debe centrarse para discriminar los datos y el número de grupos o clústeres en los que se quiere dividir una muestra, aunque este número debería ir variando para encontrar la mejor separación de los datos posible. Una vez elegidas esas propiedades, entra en la conversación el concepto de centroide, que se define como el punto medio en el que la suma de las distancias al resto de puntos del mismo clúster sea mínima.

Al inicio del algoritmo estos centroides se pueden asignar manualmente, ya que de manera iterativa se irán actualizando hasta dar con los centroides reales. Tras la asignación inicial de los centroides, se van añadiendo puntos del conjunto total de datos y clasificándolos al clúster que posea su centroide a menor distancia de dicho punto. Después de estas clasificaciones se procede a recalcular los centroides de todos los clústeres de manera que se cumpla la condición descrita previamente.

Estos pasos se repiten de manera iterativa hasta llegar al número límite de iteraciones o hasta llegar a un punto donde los centroides no cambian de manera significativa.

3.4.2. Aplicación

El proceso completo de *clustering* explicado previamente necesita de distintas herramientas para poder trabajar de manera eficiente. Para ello, se ha hecho uso de la librería *sklearn* de Python, la cual contiene herramientas como *k-means* o PCA, necesarias para el procesamiento y agrupación de los datos, así como para la reducción del número de dimensiones de estructuras, respectivamente.

El algoritmo de *k-means* necesita un conjunto de datos completo para realizar la separación en distintos clústeres, por lo que, al disponer nosotros de los descriptores de las direcciones IP separados, un primer paso necesario sería generar una lista de

descriptores que contenga todos ellos. Sin embargo, al realizar esta unión de los datos en un solo conjunto, perdemos la capacidad de distinguir direcciones concretas. Debido a esto, la mejor manera de poder identificar puntos concretos dentro de los clústeres es creando un diccionario, donde la clave se corresponde con la dirección IP representada, y el valor con su descriptor. Una vez se hayan representado gráficamente los clústeres, mediante las coordenadas de los puntos que queremos estudiar podemos extraer sus descriptores que a su vez están asociados a una dirección IP en el diccionario.

Una vez creados, tanto la lista con los descriptores como el diccionario, conviene transformar de manera inteligente los datos para trabajar con ellos de la manera más eficiente posible. Dado que cada uno de los descriptores está compuesto por una lista de valores numéricos, eso supone que la lista completa de descriptores es realmente una lista de listas, la cual sería más fácil de tratar si la convertimos en una matriz donde cada descriptor se corresponda con una fila. Además, conviene eliminar los puntos donde no tenemos datos medidos y que están representados con *NaN* y reemplazarlos con ceros para poder tratar más cómodamente con ellos.

Ahora el siguiente paso es definir una serie de parámetros que nos ayuden a identificar las diferencias entre datos. Lo primero es fijar un número de clúster en los que repartir los distintos puntos, que en nuestro caso se ha decidido que sean 3. Este número se ha elegido tras diferentes pruebas, ya que la representación habitual de los datos se compone de un clúster central con gran densidad de puntos, y un clúster a cada lado de este con mayor dispersión de puntos en ellos. Aunque se ha comentado que se deberían fijar unos centroides iniciales para comenzar el algoritmo, al crear un objeto KMeans, la propia librería genera unos centroides por defecto, por lo que no es necesario introducir nuevos.

Para facilitar la visualización de los datos y reducir el número de dimensiones, utilizamos el algoritmo PCA, al cual le indicamos el número de componentes al que queremos reducir los datos y los transforma. Una vez transformados los datos solo queda representar gráficamente los clústeres y sus puntos de manera ordenada, y utilizando un color diferente para los puntos de cada clúster. En la figura 3.3 podemos encontrar un ejemplo de *clustering* con escala exponencial utilizando las herramientas mencionadas (*k-means* y PCA).

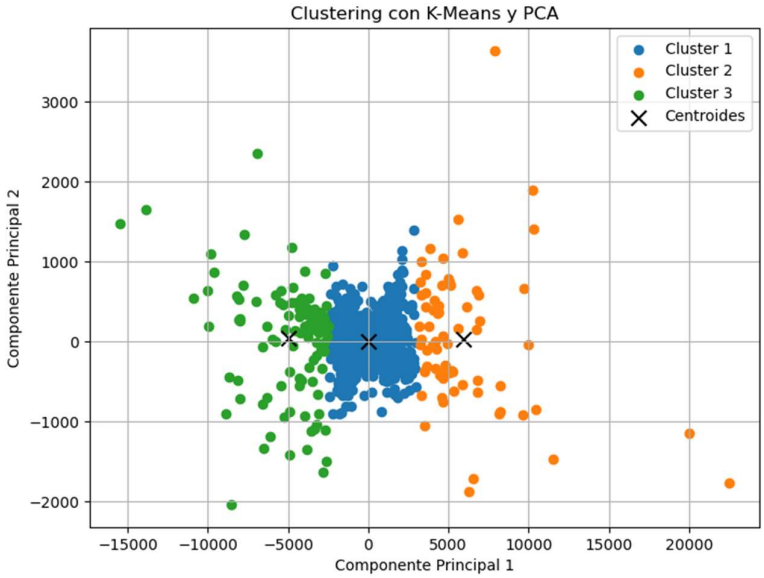


Figura 3.3: Ejemplo de agrupación con PCA y k-means

Como ya se comentaba previamente, la elección del número de clústeres que se han definido viene dado por la distribución propia de los datos, ya que el clúster central (azul) posee una mayor densidad de puntos que los clústeres que tiene a sus 2 lados (verde y naranja) que tiene datos más dispersos, e incluso algunos puntos que distan mucho de cualquiera de los clústeres, denominados habitualmente como *outliers*. Este tipo de puntos serán uno de los principales motivos de estudio, por lo que tienen una gran importancia.

Además de este tipo de representación gráfica, se han probado otras diferentes que han mostrado resultados no tan útiles como las utilizadas. Sin embargo, estas pruebas nos han dado la posibilidad de generar ciertas conclusiones acerca de los datos.

Finalmente, y tras haber representado los datos visualmente, debemos tener la posibilidad de escoger una serie de puntos de la gráfica y observar las series temporales relacionadas con ellos. Por ello se ha implementado un método de búsqueda que, introduciendo unas coordenadas, podemos obtener la dirección IP de los N puntos más cercanos a dichas coordenadas, siendo N el número de puntos que queremos encontrar.

4 Experimentos

Con el objetivo de contrastar la hipótesis general del proyecto, se han realizado una serie de pruebas para corroborar o desmentir dicha idea. Teniendo en cuenta que la interpretación de observar gráficas “similares” es algo subjetiva a pesar de utilizar métricas objetivas, no podemos definir unos resultados completamente precisos acerca de estos experimentos, aunque cabe decir que, como valoración general, los resultados han sido positivos.

La realización de estos experimentos se ha realizado a partir de la fase de *clustering* de los datos, ya que es la fase donde comienzan a poder observarse de manera gráfica y visual los datos, y la que ha comprendido la gran mayoría de las horas invertidas en este trabajo.

4.1. Transformada Wavelet

En esta primera fase de pruebas, hemos experimentado con distintas maneras de realizar las transformaciones. Por un lado, se han realizado estas transformaciones proporcionando diferentes escalas y tipos de *Wavelet* para poder estudiar diferentes opciones.

Como ya se ha comentado previamente, lo correspondiente a cada unidad de escala son aproximadamente 5 minutos en tiempo real, por lo que para las escalas lineales se han introducido 12*24 unidades para la escala diaria y 12*24*7 para la semanal, de manera que cada una toma el valor correspondiente a ese tiempo real.

Dado que nosotros poseemos datos correspondientes a 9 meses completos, se han realizado intentos para poder realizar una escala que analice el periodo entero. Sin embargo, la escala diaria y sobre todo la escala semanal han sido muy costosas computacionalmente para realizar las transformaciones respectivamente, por lo que se descartó la idea de realizarla con una escala mucho mayor, como sería una escala que comprendiera esos 9 meses. Para poner en contexto, el tiempo necesario para realizar la transformación con escala diario era aproximadamente de 2 horas, y en el caso de la semanal, de casi 10 horas. Si llevamos este aumento del tiempo a horas necesarias para una escala mensual o mayor, los tiempos necesarios serían demasiados grandes como para realizar estas transformaciones de manera dinámica. Recordemos que estas

transformaciones no se realizan una única vez, sino que se han ido realizando pruebas con distintas escalas y familias de *Wavelet* para obtener los resultados más convenientes de cara a la investigación.

Una vez tratado el concepto de escala y sus límites computacionales, toca comentar las diferentes Wavelet base utilizadas en dichas pruebas. Existen diferentes familias de *Wavelet* que podemos utilizar como base para nuestras transformaciones, como pueden ser la de *Daubechies*, *Mexican Hat*, *Gauss* o *Haar*.

Dado que las transformaciones solo nos proporcionaban descriptores numéricos y de esta manera era difícil comparar la conveniencia de utilizar unas u otras, tras obtener los descriptores numéricos era necesario realizar un proceso de *clustering* para poder observar los resultados de manera visual y realizar dicha comparación.

Dado que las familias mencionadas previamente son las más importantes y utilizadas, son aquellas con las que hemos realizado transformaciones experimentales. Inicialmente, se realizó la prueba con la *Wavelet Daubechies* en sus diferentes versiones, sin embargo, los resultados tras la fase de *clustering* eran demasiado dispersos para poder comparar series temporales, caso que sucedía también con la *Wavelet Haar*, ya que ambas son muy parecidas en cuanto a su estructura.

Las *Wavelet* de Gauss y *Mexican Hat* tuvieron mucho mejores resultados, proporcionando clústeres con una representación gráfica muy útil de cara a su posterior caracterización. Dado que la *Mexican Hat* es una *Wavelet* algo más peculiar que la gaussiana, nos decidimos por utilizar esta segunda de cara al resto de experimentos de *clustering* y caracterización del proyecto, aunque como se comenta, los resultados con la *Wavelet Mexican Hat* habrían sido muy positivos igualmente.

De manera uniforme y global, las salidas de las transformaciones con diferentes escalas y familias de *Wavelet* proporcionaban la misma estructura de salida, la cual corresponde a un descriptor numérico, el cual es en sí mismo una lista de valores.

4.2. Clustering

En la fase de agrupación de las series temporales se han tomado en cuenta distintos aspectos de cara a la realización de pruebas. Dichos aspectos por examinar pueden ser las diferentes escalas que se quieren probar (diaria, semanal y exponencial), así como los distintos métodos de *clustering* existentes para realizar este tipo de experimentos.

En la fase de caracterización, debemos separar las herramientas utilizadas en 2 tipos: *clustering* y visualización. Las herramientas de *clustering* propias que hemos utilizado han sido: *k-means*, la cual forma clústeres asignando los datos respecto a la

distancia más corta a un centroide, y DBSCAN, la cual basa su estrategia de agrupación en la densidad de los datos.

4.3. Tipos de clustering

4.3.1. K-Means

Como se ha explicado previamente, el algoritmo *k-means* basa su funcionamiento en la asignación de puntos de datos en base al centroide que se encuentre más cerca. Este algoritmo es sencillo de utilizar y es bastante eficiente computacionalmente, sin embargo, es sensible al número de clústeres inicializados y a la asignación inicial de los centroides, que pueden otorgar resultados adversos dependiendo de los valores introducidos. Otro punto a tener en cuenta es su forma, ya que se suelen obtener clústeres con distribución esférica, en los que pueden ser muy importantes sus puntos atípicos o *outliers*.

Simplemente, este método ha otorgado resultados muy buenos desde el principio, debido a su facilidad para implementarse y la capacidad de adaptación a los datos introducidos. En primer lugar, se intentó trabajar con un número variado de clústeres para observar el distinto comportamiento del método. Dado que la forma habitual del conjunto de clústeres suele parecerse a una esfera, se querían recoger los datos más céntricos y diferenciarlos del resto, sin embargo, nos hemos encontrado con 3 grupos muy diferenciados: uno central, con mayor densidad de puntos; y otros 2, uno a cada lado del clúster central, con coordenadas negativas y positivas respectivamente. Teniendo en cuenta esta distribución de los datos, la asignación del número de clústeres se ha fijado en 3. Un ejemplo de esta distribución puede ser la figura 4.1.

Otro aspecto a tener en cuenta es el método de visualización de los datos a utilizar. En este caso, se ha trabajado con PCA y con t-SNE, siendo ambos muy diferentes entre ellos. Mientras que PCA tiene como objetivo reducir la dimensionalidad intentando mantener el mayor valor de varianza de los datos posible, t-SNE hace lo mismo intentando mantener la similitud entre datos.

Teniendo en cuenta que la figura 4.1 está representada por PCA, este método ha mostrado una distribución de los datos bastante sencilla de interpretar y de trabajar con ella. En la imagen se pueden observar claramente los distintos clústeres, así como los *outliers*.

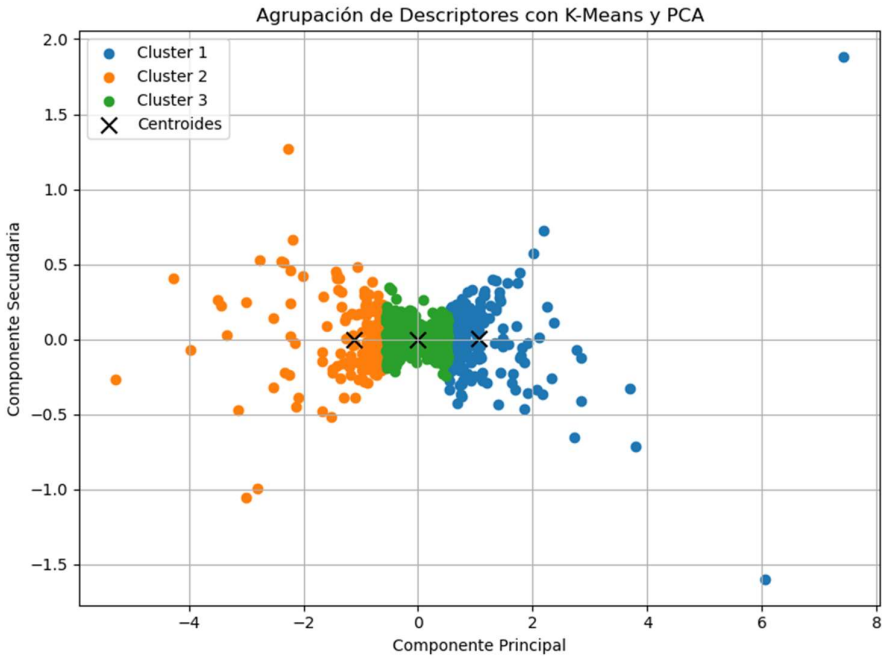


Figura 4.1: Ejemplo de agrupación con PCA y k-means

En cambio, la representación gráfica de los clústeres con el método t-SNE se muestra difícilmente interpretable. Esto se debe a su peculiar forma de "serpiente" que agrupa los datos de una manera poco intuitiva y en la que es difícil diferenciar entre puntos. Además, no queda muy clara la manera en la que diferencia las características de los 3 clústeres, ya que no se observan diferencias sustanciales entre los puntos de unos y de otros, como se puede ver en la figura 4.2.

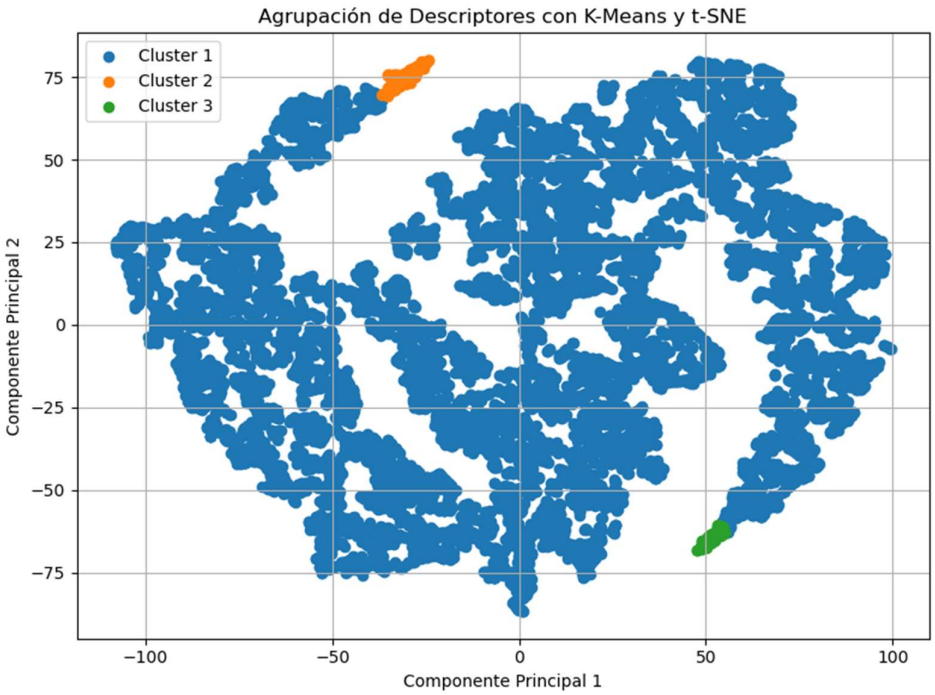


Figura 4.2: Ejemplo de agrupación con t-SNE y K-Means

Como se comentaba anteriormente, la separación de los distintos clústeres que ha realizado el método de visualización t-SNE no deja muy clara la razón por la cual discrimina los puntos entre un clúster u otro, ya que existen puntos muy cercanos que pertenecen a otro clúster. La conclusión que hemos sacado de este método es que al tener que generar 3 clústeres exactamente, el algoritmo se ve obligado a mostrar esa separación, aunque no tenga una característica suficientemente discriminatoria para hacerlo. Es por estas razones por las que hemos decidido trabajar con el método de visualización PCA, el cual es bastante amigable visualmente y nos ayuda a distinguir diferencias entre puntos.

4.3.2. DBSCAN

El método DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*), como su propio nombre indica, basa su sistema de agrupación en la densidad de los datos. Como principal diferencia respecto a PCA, este algoritmo no necesita de una configuración inicial con el número de clústeres en los que se desea separar la muestra, sino que el propio algoritmo detecta automáticamente el número necesario. Esto supone una ventaja en cuanto a complejidad de uso, aunque se posee un menor control sobre los grupos en los que se quieren dividir los datos.

Aunque este método es muy utilizado para realizar operaciones de *clustering* con diferentes objetivos, no nos ha proporcionado los resultados esperados de cara a poder comparar series temporales con comportamiento similar. La problemática surgida con este método es la aparición de los puntos demasiado dispersos a lo largo de la gráfica y la dificultad para poder observar la separación entre distintos clústeres. Otro motivo viene relacionado con este último, y es el gran número de clústeres generados por el algoritmo en comparación con los 3 grupos en los que dividimos los datos con PCA, que en este caso ha sido de 10 clústeres diferentes. La división de los datos en tantos grupos diferentes complica claramente su estudio y comparación, por lo que finalmente se ha optado por utilizar PCA como método principal de *clustering* para el resto de las pruebas y experimentos.

Como se puede observar, no se muestran representaciones gráficas de los experimentos realizados con esta metodología. Esto se debe a la baja calidad de distribución y representación que nos ha proporcionado este método, por lo que se ha optado por no mostrar dichos resultados.

4.4. Escalas

Una vez probadas diferentes herramientas, tanto de agrupación como de visualización, se ha optado por utilizar el método *k-means* para la división de los datos y el método PCA para su visualización gráfica, que por supuesto conlleva la reducción del número de dimensiones de los datos. En esta sección vamos a ver los resultados obtenidos para las distintas escalas que hemos utilizado en la investigación.

4.4.1. Escala diaria

En este primer apartado, el objetivo es realizar un análisis detallado de la aplicación de las transformaciones y metodologías previamente explicadas sobre una escala diaria. Haciendo uso de escalas “pequeñas” como esta podemos detectar características con mayor detalle de las señales, es decir, comportamientos breves en el tiempo y que tengan el suficiente impacto.

En la siguiente imagen podemos observar la representación obtenida con los datos divididos en los 3 clústeres deseados. Dado que la componente principal es la que posee una mayor cantidad de información a la hora de discriminar los datos, tiene mucho sentido que la división de los clústeres sea principalmente respecto al eje horizontal.

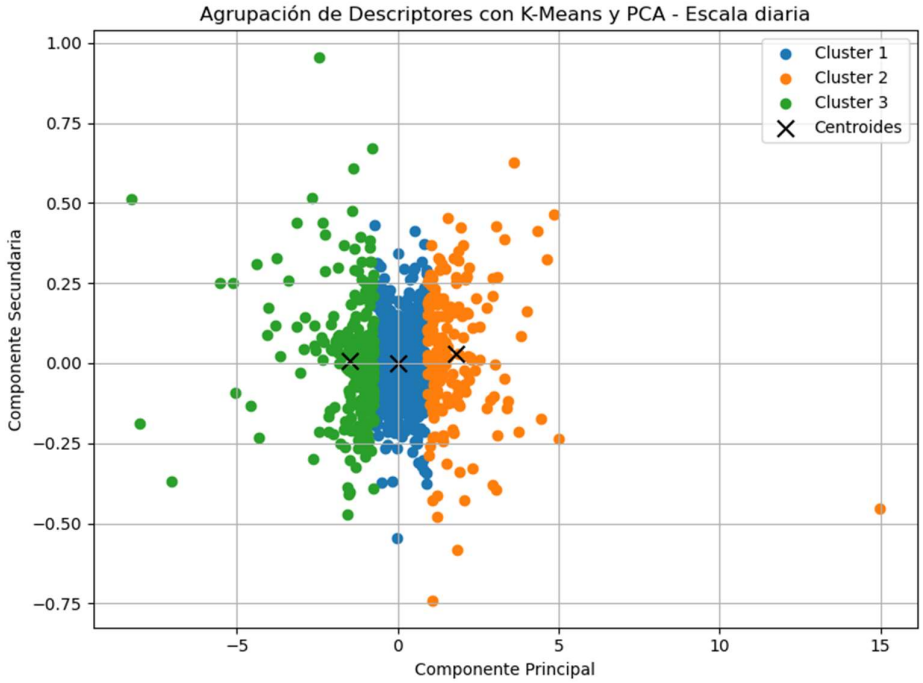


Figura 4.3: Agrupación con PCA y K-Means utilizando escala diaria

Analizando la imagen nos encontramos con una zona céntrica muy poblada con forma redondeada donde encontramos la gran mayoría de los puntos. Aunque es la zona

de mayor densidad y merece un análisis propio, también daremos mucha importancia a los puntos más alejado o atípicos de la gráfica, llamados *outliers*, ya que al ser más diferentes al resto nos pueden proporcionar un aporte de información diferente. Además, podemos ver la cercanía que tienen los centroides entre sí, lo que nos indica la gran densidad en torno a las coordenadas (0, 0) y la similitud entre los puntos de diferentes clústeres.

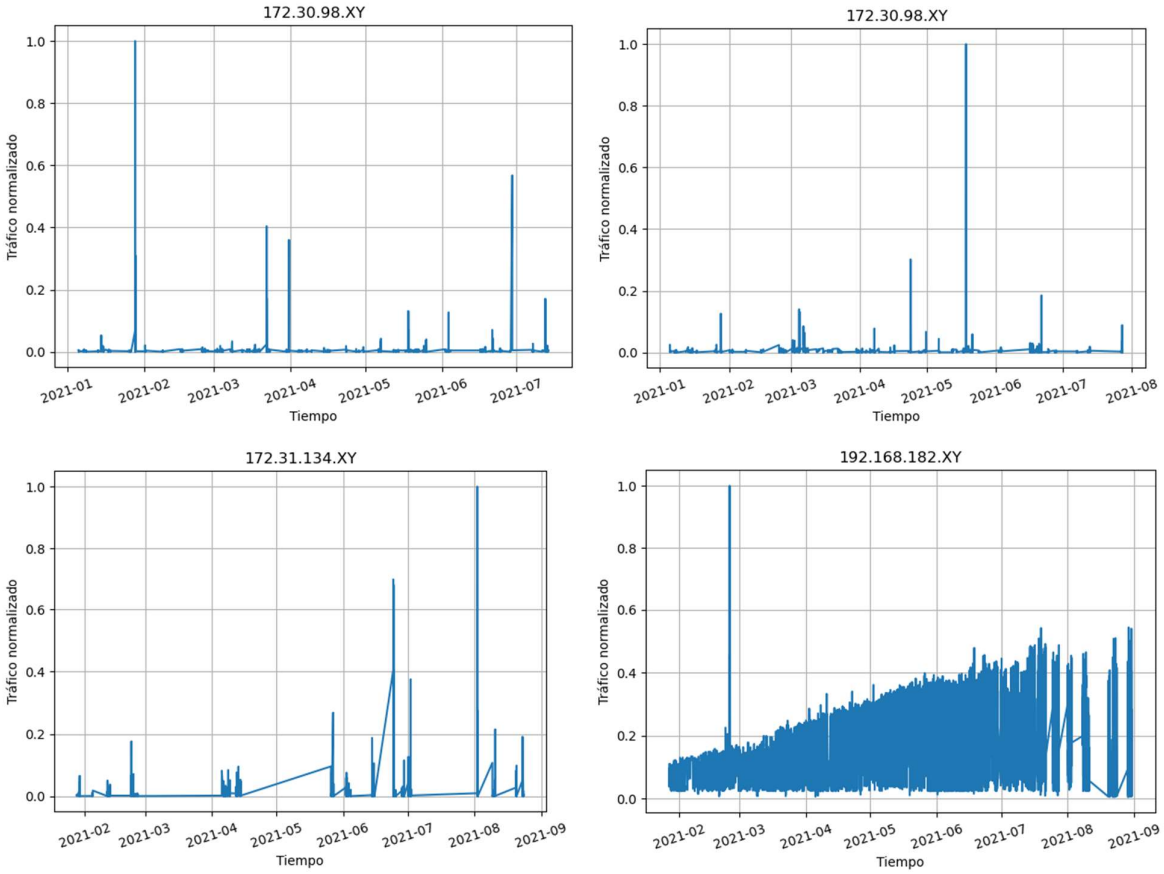


Figura 4.4: Series temporales más cercanas a la coordenada (0,0)

Este grupo de imágenes representa las 4 series temporales más cercanas al punto (0, 0) de la representación de los clústeres, por lo tanto, pertenecientes al clúster central (color azul). Como podemos ver, el nombre de cada una de las direcciones aparece como título de las gráficas.

A la hora de realizar un análisis sobre estas y futuras gráficas debemos tener en cuenta que el objetivo es buscar series temporales que tengan cierto parecido o similitud a simple vista y puede que, de manera desordenada. Esto es, la posible existencia de un pico en un momento inicial de una gráfica tiene cierta similitud a otra serie que tenga ese mismo pico en un momento más tardío. Esto se debe al método que estamos investigando, ya que la temporalidad queda en un segundo plano.

Observando las gráficas podemos observar que existen ciertos parecidos al igual

que diferencias. Por ejemplo, tomando las 2 primeras gráficas, podemos adivinar la existencia de un comportamiento bastante estable y con poca carga media a lo largo de ambas señales. Tenemos además la existencia en ambas de varios picos puntuales de carga en distintos momentos del tiempo, por lo que este sería un ejemplo de lo que estamos buscando. La tercera gráfica tiene también comportamientos puntuales de mayor carga, sin embargo, posee saltos de unos puntos a otros debido a falta de información en esos momentos. Finalmente, tenemos una cuarta gráfica que difiera totalmente del resto, ya que posee una carga habitual mucho mayor y con carácter ascendente, lo cual no es el caso de las demás. Esta diferencia es lógica teniendo en cuenta la gran variedad de puntos que existe en una zona tan densa de puntos como es la central, unido a la forma en la que están calculados los descriptores, que es realizando la media entre el número de descriptores existentes. Este comportamiento es común, tanto al clúster central con centroide en las coordenadas centrales (0, 0), como para los otros 2 clústeres ubicados 1 a cada lado de este, por lo que se suponen las mismas conclusiones de los puntos más centrados de los 3 clústeres.

Cabe comentar que la obtención de las direcciones IP relativas a los puntos representados en la gráfica se realiza gracias al diccionario donde se guardan los descriptores de cada dirección, como se había comentado previamente. Una vez identificada la dirección IP podemos comparar su serie temporal con la del resto. Aunque el estudio de la zona central es necesario e interesante, dado que existe una cierta dispersión de puntos alrededor del núcleo central, es importante realizar un análisis de los puntos más alejados de los centroides, que a su vez se encuentran relativamente cerca de otros puntos. El estudio de este tipo de puntos llamados *outliers* es importante de cara a identificar posibles series temporales anómalas o distintas al resto, cuya existencia es el propósito de este trabajo. Tomando una pareja de puntos externos a la zona central podemos hacer un estudio de este tipo de *outliers*. Cogiendo siempre como referencia la representación gráfica del clúster, hemos querido comparar los puntos más cercanos a la zona inferior izquierda, más concretamente los puntos alrededor de la coordenada (-8, -0.35).

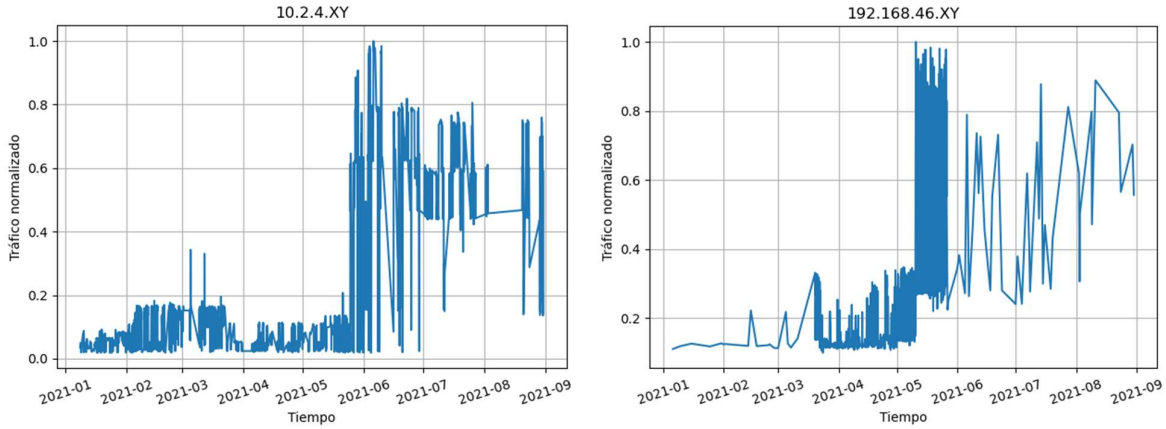


Figura 4.5: Series temporales más cercanas a la coordenada (-8, -0.35)

Como podemos ver en este análisis, esto es un claro ejemplo del comportamiento que estamos buscando y tenemos como objetivo. Se ve claramente las diferencias entre ambas gráficas, sin embargo, vemos como se producen cambios en el comportamiento de las series en momentos temporales y situaciones diferentes. La principal característica es el cambio radical que presentan ambas series, pasando de unos niveles de carga bajos y estables hacia unos mucho más altas y fluctuantes.

Este tipo de cambio en el nivel de carga de un servidor es bastante notable, y el hecho de que ambos descriptores de las series temporales tengan similitud nos hace pensar en la posibilidad de detección de este tipo de cambios y variaciones.

4.4.2. Escala semanal

Al contrario que en la escala diaria, con esta escala semanal el objetivo es detectar características y patrones de largo plazo o comportamientos que afecten a la generalidad de la señal. Aunque es cierto que el periodo temporal de una semana no es un periodo muy a largo plazo, es lo máximo que podemos tratar de forma manejable y computacionalmente.

Al igual que con la escala diaria, en la figura 4.6 se muestra la gráfica generada por la operación de *clustering* sobre los datos bajo una escala semanal.

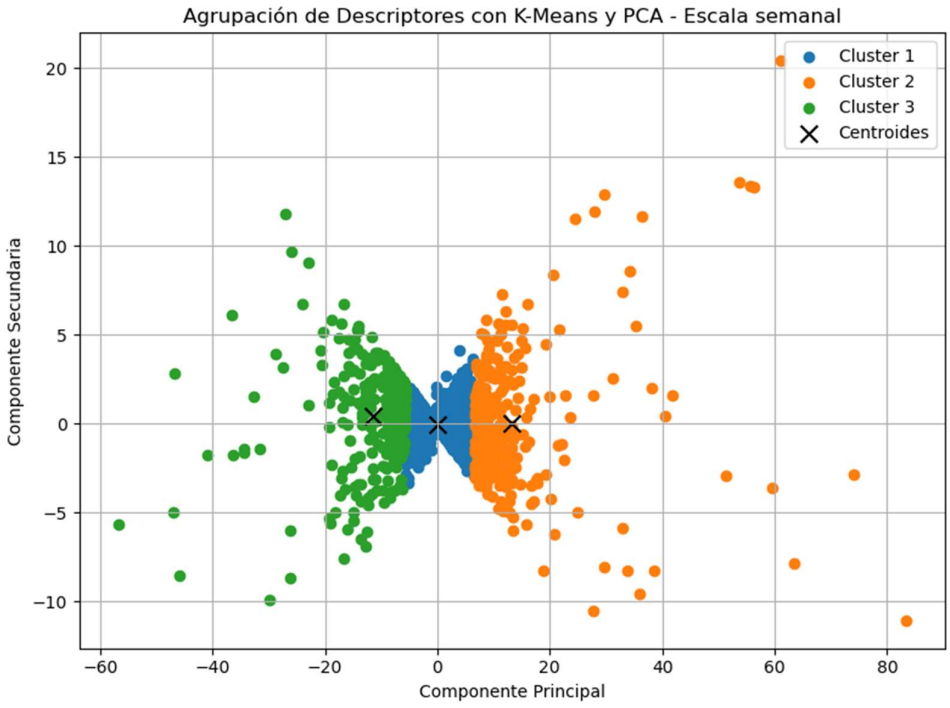


Figura 4.6: Agrupación con PCA y K-Means utilizando escala semanal

Analizando la figura 4.6 podemos ver un comportamiento de agrupación semejante a la escala diaria, donde la gran mayoría de los puntos se concentra sobre la zona central de una forma muy densa. También es cierto que existe un mayor grado de dispersión de los puntos en los clústeres laterales, los cuales vamos a estudiar por separado.

Otra característica a tener en cuenta del clúster es la forma en la que se distribuyen los puntos. A pesar de que estos clústeres suelen tener una representación redondeada, en este caso tenemos una forma parecida algo distinta.

Ahora pasamos a analizar las series más cercanas al centroide ubicado en el origen, para de esta manera observar si se cumplen las relaciones de similitud entre series temporales representadas de manera cercana bajo esta escala.

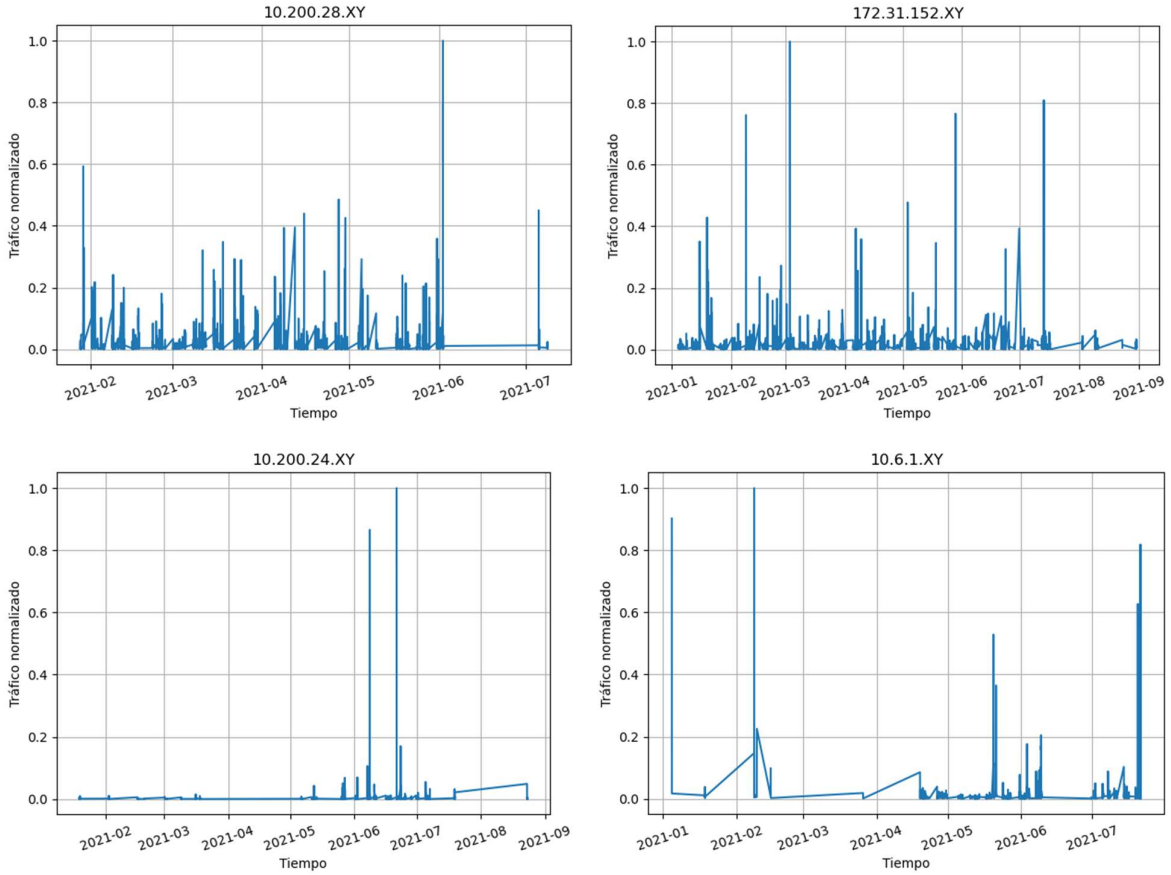


Figura 4.7: Series temporales cercanas a la coordenada (0, 0)

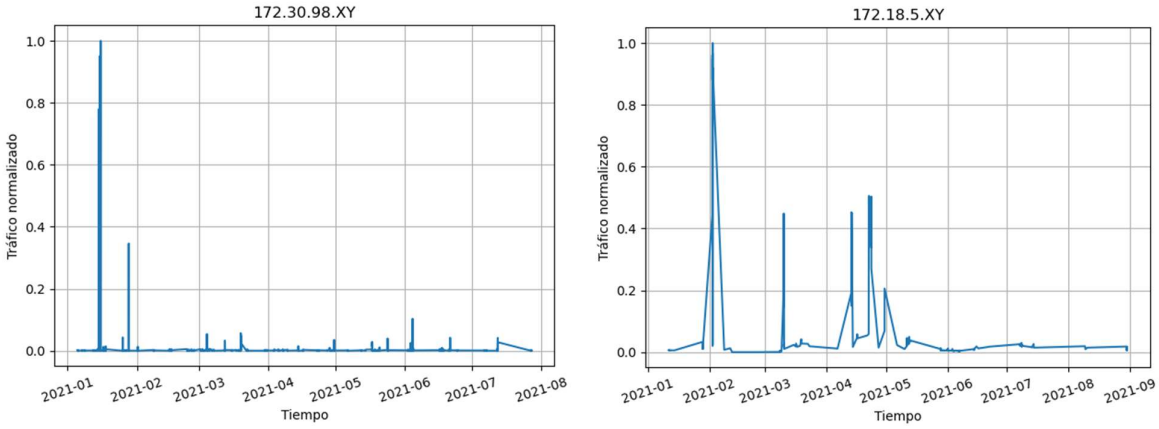
Tomando las 2 gráficas superiores de la figura 4.7 nos damos cuenta de que tienen un comportamiento bastante parecido a lo largo de todo el periodo temporal. Existe en ambas un periodo de tráfico variable con momentos de mayor y menor carga alternativamente, lo que las hace tener una cierta similitud. Además, existen picos de carga más elevados de lo habitual en distintos momentos del tiempo, lo cual verifica una vez el objetivo al que queremos llegar identificando las series.

Incluso podemos comentar la parte final del periodo temporal de ambas gráficas, donde encontrar una bajada considerable de la carga de los servidores junto a épocas cortas donde se observa una falta de mediciones, lo que genera picos muy pronunciados o siluetas muy rectas. Esa falta de datos a lo largo de un periodo más o menos largo de tiempo se podría considerar como una anomalía o patrón de comportamiento, y el método desarrollado ha sido capaz de detectar este mismo en ambas series temporales, lo que nos lleva a pensar que cumple positivamente con su función.

Por otro lado, las 2 gráficas restantes de la figura 4.7 poseen también un comportamiento muy similar entre ellas, con la presencia habitual de un bajo tráfico en los servidores y la aparición puntual de algunos picos de tráfico ubicados en distintos momentos temporales.

Cabe mencionar que todas las gráficas se han obtenido de manera conjunta al obtener los puntos más cercanos al origen, sin embargo, las 4 gráficas no se parecen entre todas ellas, sino que podemos intuir una relación de similitud dos a dos, es decir, las 2 gráficas superiores y las 2 inferiores. Esto tiene explicación debido a la gran densidad que existe en la zona donde se han tomado los datos, por lo que pueden existir descriptores que tengan valores numéricos similares sin tener una relación real de similitud entre las propias series temporales.

Ahora vamos a estudiar el comportamiento de los *outliers* presentes en la gráfica, ya que estos nos suelen proporcionar más información, al ser diferentes a la mayoría. En este primer caso, veremos las gráficas de los 4 puntos más cercanos a las coordenadas (-35, -2).



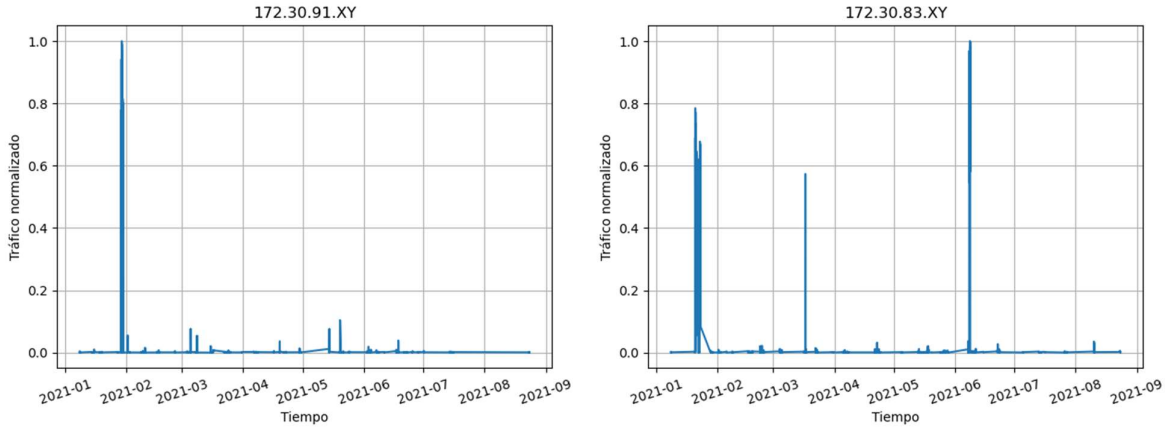


Figura 4.8: Series temporales cercanas a la coordenada (-35, -2)

Todas las gráficas de la figura 4.8 tienen un comportamiento base muy similar entre ellas, dado que poseen un tráfico habitual bastante bajo acompañado de momentos de mayor carga en los servidores. Como ya se ha comentado anteriormente en otras comparaciones, estos picos de trabajo ocurren en momentos temporales diferentes en cada una de las direcciones IP, sin embargo, sirven para poder dar más validez al método desarrollado.

Una de las gráficas, concretamente la superior derecha, contiene algunas peculiaridades o mejor dichos, discontinuidades en cuanto a la medición de los datos. La existencia de líneas rectas entre puntos algo lejanos nos indica la falta de datos durante ese periodo de tiempo. Este tipo de situaciones son muy habituales en servidores reales debido a mantenimiento o caídas de los servidores, periodos vacacionales donde la empresa produce una menor carga o el abandono del uso de un servidor. No por ello debemos de ignorar estos comportamientos, ya que están dentro de las situaciones habituales y se deben tratar como tal.

Centrándonos de nuevo en las gráficas, vamos a analizar los puntos atípicos de la zona derecha de la figura 4.6, pertenecientes al clúster representado por el color naranja. Este clúster posee una mayor dispersión de los puntos que el representado por el color verde en la zona izquierda, por lo que parece interesante observar el comportamiento de estos puntos. En la prueba anterior hemos centrado las coordenadas en una zona donde existen 4 puntos muy cercanos, por lo que ahora haremos lo opuesto y centraremos la prueba en una zona más dispersa. La coordenada elegida para esta prueba es la (60, 0) y las gráficas se muestran en la figura 4.9.

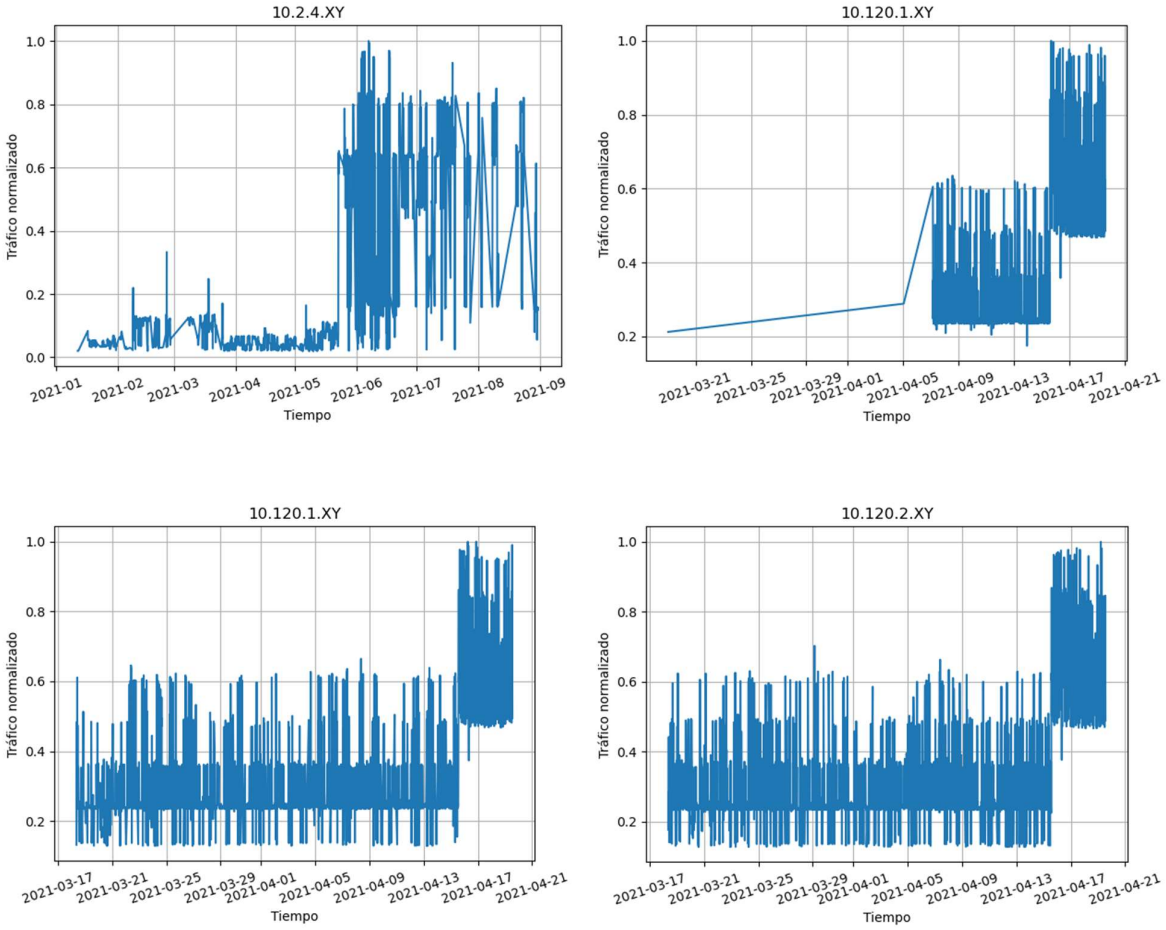


Figura 4.9: Series temporales cercanas a la coordenada (60, 0)

En la figura 4.9 podemos observar las 4 gráficas de las series temporales más cercanas a las coordenadas elegidas. Hay que recordar que, al hacer uso de una escala semanal, la motivación es encontrar patrones de comportamientos a largo plazo, al contrario que en escalas más pequeñas donde se buscaba a menor plazo.

En todas las gráficas de la figura 4.9 observamos un gran cambio a largo plazo en el tráfico de los servidores. Se puede apreciar fácilmente que todos consisten en servidores con una carga media bastante alta, lo cual es muy positivo de cara a analizarlos debido a la existencia de una gran cantidad de datos en su interior. De todos modos, aunque todos poseen una gran carga de tráfico, existe un momento en el tiempo donde el tráfico aumenta considerablemente de manera estabilizada. Esta situación es diferente a lo que habíamos visto anteriormente en otras pruebas, ya que la aparición de picos de tráfico era momentánea (tomando un momento como un periodo corto de tiempo), todo lo contrario al comportamiento de estas gráficas que tras sufrir un cambio al alza se mantienen a ese nivel de carga durante un periodo amplio de tiempo.

4.4.3. Escala exponencial

Este tipo de escala tiene cierta peculiaridad y difiere de las anteriores en la manera de trabajar en la transformada Wavelet. En las escalas anteriores introducíamos un límite de escala y la transformación se ocupaba de generar un análisis para cada una de las escalas entre 1 y el límite, sin embargo, en el caso de una escala exponencial se permite realizar un análisis más amplio. Dado que la escala va saltando de manera exponencial, en nuestro caso por las potencias de 2, podemos llegar a escalas mucho más altas a la vez que estudiamos un menor número de coeficientes. De esta manera, podemos captar de una manera más precisa patrones de comportamiento más detallados junto a algunos más generales. Además, mejoramos notablemente la complejidad computacional de estos métodos, ya que no se utilizan todas las escalas, sino tan solo las potencias de 2 hasta el límite superior, que en nuestro caso está fijado en 2^10. Finalmente, cabe comentar que este tipo de escala podría servir para detectar patrones de comportamientos logarítmicos que pasen desapercibidos con el uso de escalas lineales.

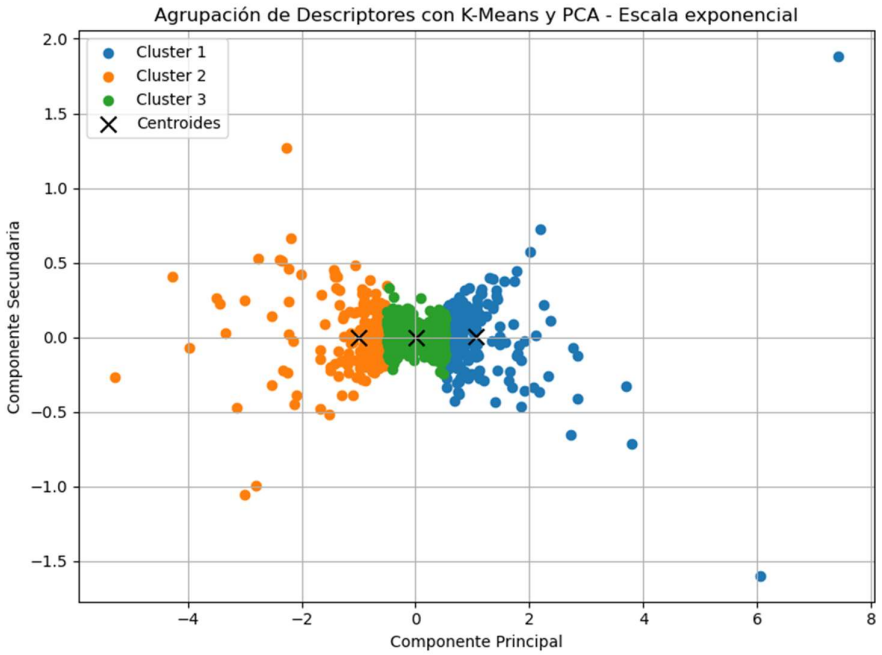


Figura 4.10: Agrupación con PCA y K-Means utilizando escala exponencial

Dado que en la figura 4.10 existe una zona central llena de puntos muy concentrados, y teniendo en cuenta resultados anteriores donde se demostraba la capacidad del método desarrollado para agrupar series temporales similares, obviamos la comparación de los puntos en esta zona y vamos directamente con los outliers de la coordenada (-4, 0).

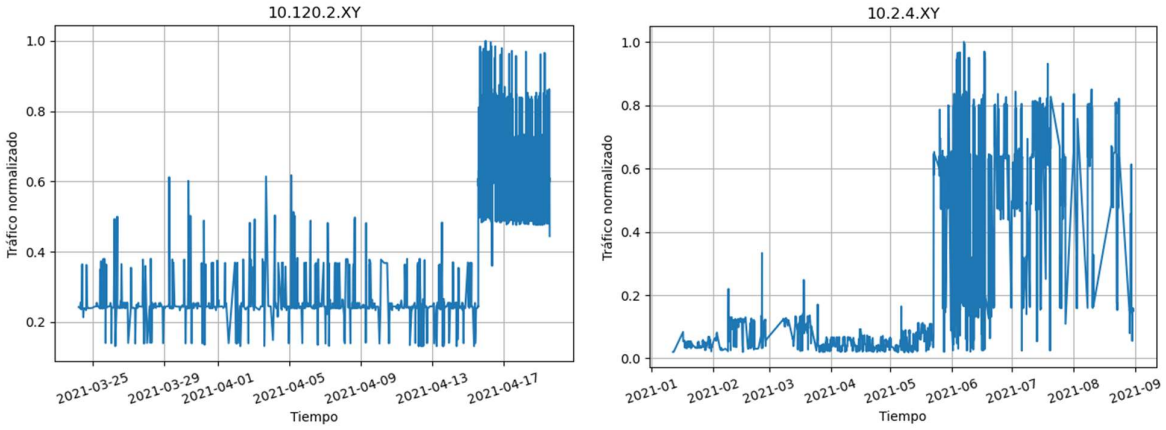


Figura 4.11: Series temporales cercanas a la coordenada (-4, 0)

Como hemos comentado previamente, la escala exponencial tiene la capacidad de llegar a un rango más amplio de estudio, por lo que podemos observar comportamiento detallado y algo más general simultáneamente. Dadas las gráficas de la figura 4.11, podemos observar que existen varios comportamientos a distinta frecuencia que podemos comentar.

Observando la primera de las gráficas vemos que el periodo en el que se han medido todos los datos es de alrededor de 1 mes, mientras que la otra tiene datos de 9 meses. Dado que nuestro método no tiene en cuenta la temporalidad para ver la similitud entre series temporales, no debería ser un problema, aunque es cierto que al representarse gráficamente se generan menos picos. Dicho esto, los picos observados en la primera de las gráficas pueden deberse a la carga de trabajo diaria, por lo que durante las noches baja mucho el tráfico y seguidamente, al iniciar el día, aumenta enormemente.

Este tipo de comportamiento a corto plazo se corresponde con el análisis de los datos a alta frecuencia, en la que ambas series poseen un patrón de comportamiento similar. Pero esto no queda ahí, ya que las gráficas también poseen un gran cambio de comportamiento a gran escala a partir de un momento significativo en el tiempo, donde el tráfico de los servidores aumenta considerablemente y estabilizándose en esos niveles durante un periodo largo. Este patrón a baja frecuencia es fácilmente identificable a simple vista.

La capacidad de la transformada Wavelet junto a una escala exponencial le otorga una potencia de análisis añadida capaz de detectar patrones a distinta escala, por lo que proporciona unos resultados mucho más completos respecto al uso de una escala lineal.

Finalmente, y como refuerzo a los resultados de las pruebas de estas series temporales, se muestran las gráficas de las series de los puntos atípicos de la coordenada (2, 0) del otro clúster en la figura 4.12.

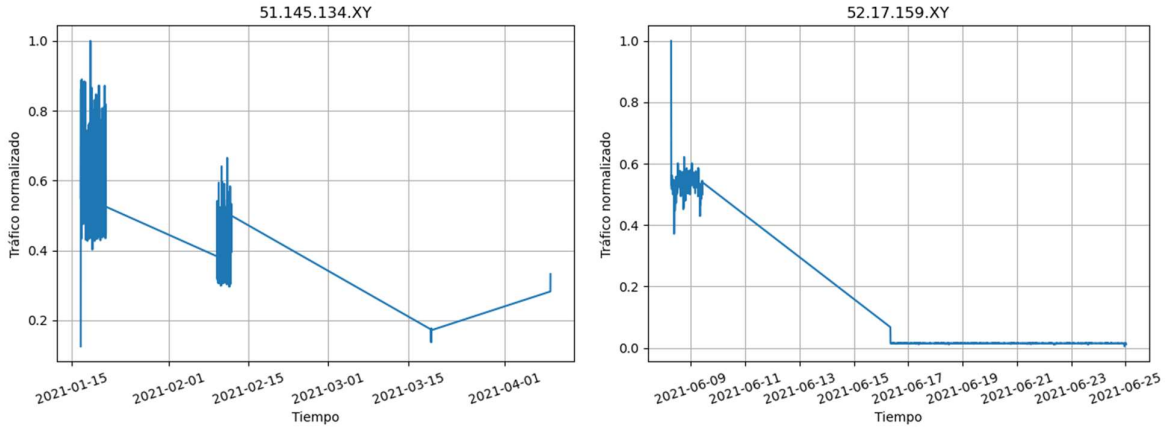


Figura 4.12: Series temporales cercanas a la coordenada (2, 0)

A simple vista se observa que las gráficas de la figura 4.12 son muy diferentes a las gráficas con los que hemos estado trabajando en el resto de las pruebas. Esto sucede por la falta de medición de datos en estas series. Como se ha comentado en secciones anteriores, la presencia de líneas rectas en las gráficas se debe a huecos o momentos en el tiempo donde no existen datos acerca de ese servidor, por lo que se producen saltos entre los momentos donde sí existen datos generando este tipo de rectas.

En otro tipo de investigaciones, la falta de datos podría ser un punto negativo de cara a realizar un análisis, sin embargo, en este caso no tendría tanto impacto. Si recordamos la manera en la que se agrupan los datos, los descriptores de las series temporales no tienen en cuenta la cantidad de datos que contiene, simplemente genera un valor numérico. Por ello, las series temporales que contienen huecos en la medición de los datos pueden caracterizarse de la misma manera.

En el caso de estas gráficas, se observa una gran falta de datos a lo largo de la mayoría del tiempo medido, sin embargo, el comportamiento general de ambas es muy similar. En ellas existe inicialmente un tráfico en niveles intermedios que, en periodos posteriores, sufre un cambio drástico de comportamiento, provocado por la falta de datos mencionada. Podríamos decir que este cambio en el comportamiento se produce a largo plazo, y a pesar de producirse de manera excepcional, el método que hemos desarrollado detecta correctamente este tipo de situaciones anómalas.

5 Conclusiones

El objetivo principal de este trabajo era realizar una investigación sobre un posible método de caracterización de series temporales basado en la eliminación de la temporalidad. Para efectuar todas las etapas de la investigación se han hecho uso de múltiples herramientas que nos han ayudado durante todo este proceso.

5.1. Objetivos cumplidos

En la fase inicial del trabajo se fijaron una serie de objetivos a cumplir a lo largo del proyecto, por lo que vamos a analizar cuáles de ellos se han completado.

De manera general, el objetivo relacionado con el aprendizaje de todo el ciclo de vida de una investigación se ha cumplido con creces. Durante la duración total del proyecto se ha aprendido a tomar posibles soluciones a diferentes problemáticas, analizar la validez de los resultados obtenidos, así como la redacción de todos estos pasos para dejar reflejado todo el trabajo realizado.

En cuanto al tema técnico, pienso que se han obtenido resultados muy positivos e interesantes de cara a una investigación futura más completa. Es evidente que la capacidad de investigación en este trabajo es limitada y puede ser desarrollada mucho más a fondo para poder obtener resultados más valiosos, sin embargo, los resultados proporcionados nos hacen pensar en la posibilidad de grandes avances en el área de la caracterización de series temporales. Es por ello que creo firmemente que el propósito general del trabajo ha sido muy positivo y se han cumplido los objetivos previamente fijados de manera positiva.

5.2. Aprendizaje

A lo largo de todo el trabajo he aprendido el proceso completo del desarrollo de una investigación. Después de haber encontrado una idea sobre la que basar todo el trabajo, se ha trabajado sobre ella de manera organizada y completando una serie de pasos. En el apartado más técnico, se ha trabajado con conceptos matemáticos, como la transformada Wavelet, así como con el análisis y comparación de conjuntos masivos de datos. Además, se han aprendido métodos de clustering y de visualización de datos previamente desconocidos, por lo que se han adquirido gran variedad de conocimientos

de distinto ámbito, lo cual es bastante enriquecedor de cara al desarrollo personal y profesional.

5.3. Trabajo futuro

Tomando el trabajo de investigación realizado en este proyecto, sería interesante continuar de manera más profunda con dicha investigación y adentrarse en la capacidad de detección de anomalías a partir de metodologías como la desarrollada en este trabajo.

Como ejemplo, una de las mayores promesas actuales en el campo de la ciberseguridad es el desarrollo de IDS (Intrusion Detection System) basado en eventos y anomalías. En estos casos, podría ser interesante utilizar el método que hemos investigado acerca de la eliminación de la temporalidad de la carga de los servidores, o en otros posibles casos, del tráfico de red, para poder realizar avances en el campo de la detección de anomalías o ataques en sistemas.

Este trabajo está basado en la caracterización de las series temporales, sin embargo, no se ha profundizado en materia de detección de las anomalías. La manera de proceder ha sido la de realizar una caracterización de las series y una posterior búsqueda de similitudes entre las series. Esta búsqueda se ha realizado de manera visual y “humana”, por lo que automatizar esta tarea sería un posible campo de investigación hacia el futuro.

Bibliografía

[1] S. Marukatat, “Tutorial on PCA and approximate PCA and approximate kernel PCA,” *Artif Intell Rev*, vol. 56, no. 6, 2023, doi: 10.1007/s10462-022-10297-z.

[2] G. C. Linderman and S. Steinerberger, “Clustering with t-SNE, Provably,” *SIAM J Math Data Sci*, vol. 1, no. 2, 2019, doi: 10.1137/18m1216134.

[3] L. McInnes, J. Healy, N. Saul, and L. Großberger, “UMAP: Uniform Manifold Approximation and Projection,” *J Open Source Softw*, vol. 3, no. 29, 2018, doi: 10.21105/joss.00861.

[4] K. P. Sinaga and M. S. Yang, “Unsupervised K-means clustering algorithm,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2988796.

[5] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN,” *ACM Transactions on Database Systems*, vol. 42, no. 3, 2017, doi: 10.1145/3068335.

[6] D. Perdices, J. L. Garcia-Dorado, J. Ramos, R. De Pool, and J. Aracil, “Towards the Automatic and Schedule-Aware Alerting of Internetwork Time Series,” *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3073598.

[7] D. Perdices, J. E. De Vergara López, and J. Ramos, “Deep-FDA: Using Functional Data Analysis and Neural Networks to Characterize Network Services Time Series,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, 2021, doi: 10.1109/TNSM.2021.3053835.

[8] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Physica D*, vol. 404, 2020, doi: 10.1016/j.physd.2019.132306.

[9] W. K. Seo, “Functional principal component analysis for cointegrated functional time series,” *Journal of Time Series Analysis*, vol. 45, no. 2. 2024. doi: 10.1111/jtsa.12707.

[10] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller, “Deep learning for time series classification: a review,” *Data Min Knowl Discov*, vol. 33, no. 4, 2019, doi: 10.1007/s10618-019-00619-1.

[11] B. Lim and S. Zohren, “Time-series forecasting with deep learning: A survey,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194. 2021. doi: 10.1098/rsta.2020.0209.

[12] R. M. Cichy and D. Kaiser, “Deep Neural Networks as Scientific Models,” *Trends in Cognitive Sciences*, vol. 23, no. 4. 2019. doi: 10.1016/j.tics.2019.01.009.

[13] P. Tarigopula, S. L. Fairhall, A. Bavaresco, N. Truong, and U. Hasson, “Improved prediction of behavioral and neural similarity spaces using pruned DNNs,” *Neural Networks*, vol. 168, 2023, doi: 10.1016/j.neunet.2023.08.049.

[14] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, “A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN,” *Electronics (Switzerland)*, vol. 12, no. 1, 2023, doi: 10.3390/electronics12010232.

[15] S. Sakib *et al.*, “Attention-Based Models for Multivariate Time Series Forecasting: Multi-step Solar Irradiation Prediction,” *Heliyon*, vol. 10, no. 6, 2024, doi: 10.1016/j.heliyon.2024.e27795.

[16] C. Torrence and G. P. Compo, “A Practical Guide to Wavelet Analysis,” *Bull Am Meteorol Soc*, vol. 79, no. 1, 1998, doi: 10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2.

[17] T. Guo, T. Zhang, E. Lim, M. Lopez-Benitez, F. Ma, and L. Yu, “A Review of Wavelet Analysis and Its Applications: Challenges and Opportunities,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3179517.

[18] D. Arrigo, “Fourier Transform,” in *Synthesis Lectures on Mathematics and Statistics*, 2023. doi: 10.1007/978-3-031-22087-6_6.

[19] Z. Zhao, “Fourier Analysis and Its Application,” *Highlights in Science, Engineering and Technology*, vol. 38, 2023, doi: 10.54097/hset.v38i.5943.

[20] J. R. A. Montoya, “La transformada wavelet,” *Revista de la Universidad de Mendoza*, 2001.

[21] J. A. Cortés, H. B. Cano-Garzón, and J. A. Chaves, “Del análisis de Fourier a las Wavelets - Transformada continua wavelet (CWT),” *Scientia et Technica*, vol. XIII, no. 37, 2007.

[22] N. Nieto and D. Orozco, “El uso de la transformada wavelet discreta en la reconstrucción de señales senosoidales,” *Scientia Et Technica*, vol. 14, no. 38, 2008.

Apéndices

A

Apéndice A

Este apéndice muestra el código utilizado para la representación gráfica de los datos separados por direcciones IP. A partir de este método se genera una carpeta donde se cargan todas las imágenes de las series temporales.

```
import pandas as pd
import matplotlib.pyplot as plt
import glob

# Time interval - 6 months
# procesa2-2021-01-01.csv - procesa2-2021-06-30.csv
files = glob.glob("shared-nvme/flujos/procesa2-2021-0*.csv")

# Read CSV files
dfs = []
for file in files:
    df = pd.read_csv(file)
    dfs.append(df)

# DataFrame with all the data
combined_df = pd.concat(dfs, ignore_index=True)

# Plot every IP with more than 1000 points
for ip, filtered_df in combined_df.groupby("targetIP"):

    if filtered_df.shape[0] < 1000:
        continue

    part1, separator, part2 = ip.rpartition('.')
    xx = filtered_df.sort_values("tref_start", ascending=True)
    X = xx.tref_start
    Y = xx.bpsPhyRcv / xx.bpsPhyRcv.max()
    plt.figure()
    plt.grid(True)
    plt.xlabel("Tiempo")
    plt.xticks(rotation=20)
    plt.ylabel("Tráfico [bit/s]")
    plt.title(part1 + ".XY")
    plt.tight_layout()
    plt.plot(X,Y)
    plt.savefig(f"shared-hdd/TFGAlvaro/formatted_plots/{ip}_pro-
cessed_plot.png")
    plt.close()
```

Para la aplicación de la transformada *Wavelet* sobre las series temporales creadas se ha utilizado el siguiente fragmento de Código. En este caso se muestra en la línea 33 el uso de una escala exponencial cuyo valor se asigna en la línea 20. Para los casos donde se utilizan escalas lineales, simplemente se elimina la función de *np.power()* y se simplifica con el rango normal.

```
1  import pywt
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import glob
5  import pandas as pd
6
7  files = glob.glob("shared-nvme/flujos/procesa2-2021-0*.csv")
8
9  # Read CSV files
10 # Save it in a CSV or parquet??
11 dfs = []
12 for file in files:
13     df = pd.read_csv(file)
14     dfs.append(df)
15
16 # DataFrame with all the data
17 combined_df = pd.concat(dfs, ignore_index=True)
18
19 wt='gaus1'
20 scaleMax = 10+1 # 2^10=1024
21
22 # Plot every IP with more than 1000 points
23 for ip, filtered_df in combined_df.groupby("targetIP"):
24
25     if filtered_df.shape[0] < 1000:
26         continue
27
28     print(ip)
29     xx = filtered_df.sort_values("tref_start", ascending=True)
30     X = xx.tref_start
31     Y = xx.bpsPhyRcv / xx.bpsPhyRcv.max()
32     plt.figure()
33     scales = np.power(2, np.arange(scaleMax))
34     coef, freqs=pywt.cwt(Y, scales, wt)
35
36     # Calculate descriptor adding the Y axis
37     descriptor = coef.sum(axis=1)
38
39     # Save the descriptor matrix in
40     file = f"descriptors/exp/descriptor_{ip}_exp.npy"
41     with open(file, 'wb') as f:
42         np.save(file, coef)
43
44     plt.matshow(coef)
45     plt.title(f"DWT({wt}) {ip}")
46     plt.savefig(f"dwt/exp/{ip}_cwt_{wt}_{scaleMax}_plot.pdf")
47     plt.close()
```

A continuación, se muestra el Código utilizado para realizar la representación gráfica a partir de la herramienta PCA.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import glob

# Example
descriptors = []
ip_list = []
ip_desc = {}

files = glob.glob("descriptors/exp/descriptor*.npy")
for file in files:
    partes = file.split("/")
    if len(partes) > 1:
        file_name = partes[-1]
        palabras = file_name.split("_")
        if len(palabras) > 1:
            ip_name = palabras[1]

    descriptor = np.load(file)
    avg_desc = descriptor.sum(axis=1) / descriptor.shape[1]

    ip_desc[ip_name] = avg_desc
    descriptors += [avg_desc]

descriptors = np.asarray(np.matrix(descriptors))
descriptors = np.nan_to_num(descriptors)

# Number of clusters to user
n_clusters = 3

# Kmeans object with n_clusters
kmeans = KMeans(n_clusters=n_clusters, n_init=10)

# Adjust k-means to our descriptor
kmeans.fit(descriptors)

# Get the labels assigned to every
labels = kmeans.labels_

# Get the centroids coordinates of the cluster
centroids = kmeans.cluster_centers_

pca = PCA(n_components=2)
descriptor_pca = pca.fit_transform(descriptors)

# Obtener coordenadas de los centroides de los clústeres en el espacio PCA
centroids_pca = pca.transform(kmeans.cluster_centers_)

# Visualizar los clústeres en el espacio PCA
plt.figure(figsize=(8, 6))

# Dibujar cada punto de datos con un color correspondiente a su clúster
for i in range(n_clusters):
```

```
    plt.scatter(descriptor_pca[labels == i, 0],
                descriptor_pca[labels == i, 1], label=f'Cluster {i+1}')

# Dibujar los centroides de los clústeres en el espacio PCA
plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1],
            s=100, c='black', marker='x', label='Centroides')

plt.title('Agrupación de Descriptores con K-Means y PCA')
plt.xlabel('Componente Principal')
plt.ylabel('Componente Secundaria')
plt.legend()
plt.tight_layout()
plt.grid(True)
plt.savefig(f"descriptors/visual_desc/exp/pca_exp_final.png")
plt.close()
```

De manera adicional, se muestra en el siguiente fragmento de código el método utilizado para encontrar el nombre de las direcciones IP más cercanas respecto a una coordenada exacta.

```
# CENTROID (0, 0)

# Analize the point in (X, Y)
x = 0 # X coord
y = 0 # Y coord

# Find the nearest index in the PCA space
distancias = np.sqrt((descriptor_pca[:, 0] - x)**2 +
                     (descriptor_pca[:, 1] - y)**2)
indice_punto_interes = np.argsort(distancias)[1:5]

# Access the descriptor
descriptor_interes = descriptors[indice_punto_interes]

for i in range(0,4):
    for clave, val in ip_desc.items():
        if str(val) == str(descriptor_interes[i]):
            print(clave)
```

A continuación, se muestra el código utilizado para realizar la representación gráfica a partir de la herramienta DBSCAN.

```
import numpy as np
import numba
import glob
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA

# Example
descriptors = []

```

```
files = glob.glob("descriptors/exp/descriptor*.npy")
for file in files:
    partes = file.split("/")
    if len(partes) > 1:
        file_name = partes[-1]

    descriptor = np.load(file).sum(axis=1)
    descriptors += [descriptor]

descriptors = np.asarray(np.matrix(descriptors))
descriptors = np.nan_to_num(descriptors)

# Number of clusters to user
eps = 0.5
min_samples = 3

# DBSCAN object with eps and min_samples
dbscan = DBSCAN(eps=eps, min_samples=min_samples)

# Adjust DBSCAN to our descriptor
dbscan.fit(descriptors)

# Get the labels assigned to every
labels = dbscan.labels_

pca = PCA(n_components=2)
descriptor_pca = pca.fit_transform(descriptors)

# Plot the clusters
plt.figure(figsize=(8, 6))

# Plot each data point with a color corresponding to its cluster
for i in range(len(np.unique(labels))):
    plt.scatter(descriptor_pca[labels == i, 0],
                descriptor_pca[labels == i, 1], label=f'Cluster {i}')

plt.title('Clustering de Descriptores con DBSCAN y PCA')
plt.xlabel('UMAP Dimension 1')
plt.ylabel('UMAP Dimension 2')
plt.legend()
plt.grid(True)
plt.savefig("descriptors/visual_desc/exp/dbscan_week_cluster.png")
plt.close()
```

A continuación, se muestra el código utilizado para realizar la representación gráfica a partir de la herramienta t-SNE.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
import glob

# Example
descriptors = []
ip_list = []
```

```
ip_desc = {}

files = glob.glob("descriptors/exp/descriptor*.npy")
for file in files:
    partes = file.split("/")
    if len(partes) > 1:
        file_name = partes[-1]
        palabras = file_name.split("_")
        if len(palabras) > 1:
            ip_name = palabras[1]

    descriptor = np.load(file).sum(axis=1)
    descriptors += [descriptor]

    ip_desc[ip_name] = descriptor

descriptors = np.asarray(np.matrix(descriptors))
descriptors = np.nan_to_num(descriptors)

# Number of clusters to user
n_clusters = 3

# Kmeans object with n_clusters
kmeans = KMeans(n_clusters=n_clusters, n_init=10)

# Adjust k-means to our descriptor
kmeans.fit(descriptors)

# Get the labels assigned to every
labels = kmeans.labels_

# Get the centroids coordinates of the cluster
centroids = kmeans.cluster_centers_

tsne = TSNE(n_components=2)
descriptor_tsne = tsne.fit_transform(descriptors)

# Visualizar los clústeres en el espacio PCA
plt.figure(figsize=(8, 6))

# Dibujar cada punto de datos con un color correspondiente a su clúster
for i in range(n_clusters):
    plt.scatter(descriptor_tsne[labels == i, 0],
                descriptor_tsne[labels == i, 1], label=f'Cluster {i+1}')

plt.title('Agrupación de Descriptores con K-Means y t-SNE')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.legend()
plt.tight_layout()
plt.grid(True)
plt.savefig(f"descriptors/visual_desc/exp/tsne_exp_cluster.png")
plt.close()
```

Glosario

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*

FPCA *Functional Principal Component Analysis*

HPCN *High Processing Computing and Networking*

IP *Internet Protocol*

LSTM *Long Short-Term Memory*

PCA *Principal Component Analysis*

STFT *Short-Time Fourier Transform*

t-SNE *t-Distributed Stochastic Neighbor Embedding*

UMAP *Uniform Manifold Approximation and Projection*