UCAM

UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Programa de Doctorado en Tecnologías de la Computación e Ingeniería Ambiental

TESIS DOCTORAL

Desarrollo eficiente de algoritmos de clasificación difusa en entornos *Big Data*

Autor:
Dña. Isabel María Timón Pérez

Directores:
Dr. D. José María Cecilia Canales
Dr. D. Jesús A. Soto Espinosa

Murcia, septiembre de 2018

UCAM

UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Programa de Doctorado en Tecnologías de la Computación e Ingeniería Ambiental

TESIS DOCTORAL

Desarrollo eficiente de algoritmos de clasificación difusa en entornos *Big Data*

Autor:
Dña. Isabel María Timón Pérez

Directores:
Dr. D. José María Cecilia Canales
Dr. D. Jesús A. Soto Espinosa

Murcia, septiembre de 2018

UCAM

UNIVERSIDAD CATÓLICA
DE MURCIA

AUTORIZACIÓN DEL DIRECTOR DE LA TESIS PARA SU PRESENTACIÓN

El Dr. D. José María Cecilia Canales y el Dr. D. Jesús Antonio Soto Espinosa como Directores[1] de la Tesis Doctoral titulada "Desarrollo eficiente de algoritmos de clasificación difusa en entornos *Big Data*" realizada por Dña. Isabel María Timón Pérez en la Escuela Internacional de Doctorado (EIDUCAM), **autoriza su presentación a trámite** dado que reúne las condiciones necesarias para su defensa.

Lo que firmo, para dar cumplimiento a los Reales Decretos 99/2011, 1393/2007, 56/2005 y 778/98, en Murcia a 28 de Septiembre de 2018.

Dr. D. José Mª Cecilia Canales

Dr. D. Jesús A. Soto Espinosa

[1] Si la Tesis está dirigida por más de un Director tienen que constar y firmar ambos.

A mi marido, Emilio. A mis padres, Carmen y Santiago. A mis hermanas, Carmen Mª y Mª Mercedes. Y a mis abuelos, los que están y los que ya se fueron, Isabel y Sotero, Carmen y Agustín.

A vosotros os dedico esta tesis, por serlo todo en mi vida.

Agradecimientos

"La familia es base de la sociedad y el lugar donde las personas aprenden por vez primera los valores que les guían durante toda su vida."

Papa Juan Pablo II

Me gustaría dedicar estas palabras a todas las personas que componen "mi familia" pues son aquellas a través de las cuáles he aprendido valores y han hecho de guía a lo largo de mi vida.

Comenzaré con la familia en la que el destino me ha situado: la Universidad Católica San Antonio de Murcia, UCAM.

Quiero hacer una referencia especial a mis directores: el Dr. D. José Mª Cecilia Canales y el Dr. D. Jesús A. Soto Espinosa. Tanto en el ámbito profesional como investigadores de reconocido prestigio y apasionados docentes, como en el ámbito personal con su dedicación, paciencia y esfuerzo, han conseguido generar en mí el gusanillo por la investigación y han ayudado a completar mi formación académica para poder culminar con éxito esta tesis. Gracias por compartir vuestro tiempo, vuestra sabiduría y vuestros consejos.

Me gustaría nombrar también a la Dra. Dña. Belén López Ayuso y al Dr. D. Manuel C. Ruiz González quienes, a través de su esfuerzo y compromiso, se han convertido en dos de los pilares más importantes de la Universidad, ocupando sendos puestos de Vicerrectora de Enseñanza Virtual y Vicerrector de Calidad y Ordenación Académica. Podemos estar tranquilos, la UCAM está en buenas manos con vosotros. A ambos quería agradecer la confianza depositada en mí y haberme brindado la oportunidad de formar parte de esta gran familia.

A mis compañeros de departamento: a Encarni, Tano y Toñi, de Calidad; a Mª Luisa, Almudena, Eva, Ginés, Vero, Fran y Cristina, de Ordenación Académica; y a Dolo, Lola y Tomás, de SOIL. Por los ánimos en los momentos de dudas, los buenos ratos que hemos pasado y los que están por llegar,

gracias.

Continuaré con la familia que se elige: los amigos.

Un día te elegí a ti como amigo, como compañero de vida, como confidente. Y tú me elegiste a mí. Emilio, gracias por todo. Porque la vida nos ha dado algún que otro "golpecito", pero ahí estaremos para superarlo y para celebrar todo lo bueno que nos tengan preparado. Viniste acompañado de una familia maravillosa, de la que bien orgulloso puedes estar. A tus padres, Teresa y Emilio, que siempre me han hecho sentir como una hija, gracias. Y gracias también a tus tíos: Rosa, siempre atenta y detallista; y Paco, con sus sutiles y discretas preguntas, sus palabras de ánimo y aliento, y su inestimable disposición.

No pasa un día, dos como mucho, sin recibir un mensaje que te haga recordar que siempre están ahí, Diana y Miriam. Gracias por vuestros ánimos, los momentos de confidencias y de risas, hacéis que cada día me sienta afortunada por tener unas amigas como vosotras.

Cuando la trayectoria emocional se une a la académica, se crean unos lazos difíciles de romper. Puede que pasen semanas, meses o, incluso, años pero la complicidad y el cariño perduran y, siempre que nos volvemos a juntar, me siento como si el día siguiente tuviéramos que ir a la facultad, como si tuviéramos que entregar unos ejercicios (*sencillicos, sencillicos...*) de Ecuaciones en Derivadas Parciales, como si nos quedáramos a comer en la cantina de químicas... como si no hubiera pasado el tiempo. Ana, Rober, Pichí, mis compañeros de batallas, de risas, de inventos, mis "otros tres fantásticos", gracias.

A lo largo de mi vida académica he tenido grandes docentes pero quiero resaltar la figura del Profesor Víctor Jiménez. Impecable es la palabra que más se adapta a su persona: Impecable como docente, con sus clases minuciosamente hiladas y los materiales dignos de enmarcar; impecable como orientador, con su despacho siempre abierto para cualquier cuestión que le quisieras plantear, ya fuera académica o personal; e impecable como amigo, siempre dispuesto a compartir ratos de risas y anécdotas e, incluso, a dejarse engañar, de buen agrado, cuando se nos ocurre llevarlo a un *escape room*. Es

por ello que lo menciono en este apartado, gracias por ser amigo.

Y terminaré mencionando a la familia que no elegí, la familia que Dios dispuso para mí: mis abuelos, mis padres y mis hermanas.

Nací, hace casi 30 años, en el seno de una familia humilde, pero plena de valores y amor. Junto con mis hermanas, he crecido rodeada de seres queridos que, con una generosidad envidiable, han dedicado su vida a formarnos y educarnos como personas íntegras y capaces de dar lo mejor de nosotras mismas. Siempre han tenido la paciencia y el tesón de escucharme, de establecer los límites cuando consideraban que descarriaba del camino y, también, de encontrar los consejos y las palabras adecuadas para cada momento. Por todo lo que habéis hecho hasta ahora y por lo que, estoy segura, seguiréis haciendo, quiero daros las gracias. Sin vosotros, jamás sería lo que soy.

Resumen

Estamos presenciando una época de transición donde los "datos" son los principales protagonistas. En la actualidad, cada día se genera una ingente cantidad de información en la conocida como era del *Big Data*. La toma de decisiones basada en estos datos, su estructuración, organización así como su correcta integración y análisis, constituyen un factor clave para muchos sectores estratégicos de la sociedad. En el tratamiento de cantidades grandes de datos, las técnicas de almacenamiento y análisis asociadas al *Big Data* nos proporcionan una gran ayuda. Entre estas técnicas predominan los algoritmos conocidos como *machine learning*, esenciales para el análisis predictivo a partir de grandes cantidades de datos. Dentro del campo del *machine learning*, los algoritmos de clasificación difusa son empleados con frecuencia para la resolución de una gran variedad de problemas, principalmente, los relacionados con control de procesos industriales complejos, sistemas de decisión en general, la resolución y la compresión de datos. Los sistemas de clasificación están también muy extendidos en la tecnología cotidiana, por ejemplo en cámaras digitales, sistemas de aire acondicionado, etc.

El éxito del uso de las técnicas de *machine learning* está limitado por las restricciones de los recursos computacionales actuales, especialmente, cuando se trabaja con grandes conjuntos de datos y requisitos de tiempo real. En este contexto, dichos algoritmos necesitan ser rediseñados e, incluso, repensados con la finalidad de aprovechar al máximo las arquitecturas masivamente paralelas que ofrecen el máximo rendimiento en la actualidad. Esta tesis doctoral se centra dentro de este contexto, analizando computacionalmente el actual panorama de algoritmos de clasificación y proponiendo algoritmos de clasificación paralelos que permitan ofrecer soluciones adecuadas en un intervalo de tiempo reducido. En concreto, se ha realizado un estudio en profundidad de técnicas bien conocidas de *machine learning* mediante un caso de aplicación práctica. Esta aplicación predice el nivel de

ozono en diferentes áreas de la Región de Murcia. Dicho análisis se fundamentó en la recogida de distintos parámetros de contaminación para cada día durante los años 2013 y 2014. El estudio reveló que la técnica que obtenía mejores resultados fue *Random Forest* y se obtuvo una regionalización en dos grandes zonas, atendiendo a los datos procesados.

A continuación, se centró el objetivo en los algoritmos de clasificación difusa. En este caso, se utilizó una modificación del algoritmo *Fuzzy C-Means* (FCM), μFCM, como técnica de discretización con el objetivo de convertir los datos de entrada de continuos a discretos. Este proceso tiene especial importancia debido a que hay determinados algoritmos que necesitan valores discretos para poder trabajar, incluso técnicas que sí trabajan con datos continuos, obtienen mejores resultados con datos discretos. Esta técnica fue validada a través de la aplicación al bien conocido conjunto de *Iris Data* de Anderson, donde se comparó estadísticamente con la técnica de *K-Means* (KM), proporcionando mejores resultados.

Una vez realizado el estudio de los algoritmos de clasificación difusa, se detecta que dichas técnicas son sensibles a la cantidad de datos, incrementando su tiempo computacional. De modo que la eficiencia en la programación de estos algoritmos es un factor crítico para su posible aplicabilidad al *Big Data*. Por lo tanto, se propone la paralelización de un algoritmo de clasificación difusa a fin de conseguir que la aplicación sea más rápida conforme aumente el grado de paralelismo del sistema. Para ello, se propuso el algoritmo de clasificación difusa *Parallel Fuzzy Minimals* (PFM) y se comparó con los algoritmos FCM y *Fuzzy Minimals* (FM) en diferentes conjuntos de datos. En términos de calidad, la clasificación era similar a la obtenida por los tres algoritmos, sin embargo, en términos de escalabilidad, el algoritmo paralelizado PFM obtenía una aceleración lineal con respecto al número de procesadores empleados.

Habiendo identificado la necesidad de que dichas técnicas tengan que ser desarrolladas en entornos masivamente paralelos, se propone una infraestructura de *hardware* y *software* de alto rendimiento para procesar, en tiempo real, los datos obtenidos de varios vehículos en relación a variables que

analizan problemas de contaminación y tráfico. Los resultados mostraron un rendimiento adecuado del sistema trabajando con grandes cantidades de datos y, en términos de escalabilidad, las ejecuciones fueron satisfactorias. Se visualizan grandes retos a la hora de identificar otras aplicaciones en entornos *Big Data* y ser capaces de utilizar dichas técnicas para la predicción en áreas tan relevantes como la contaminación, el tráfico y las ciudades inteligentes.

Palabras clave: Minería de datos, Aprendizaje automático, Clasificación difusa, Computación de alto rendimiento, Contaminación, Sistemas de transporte inteligentes, Ciudades inteligentes

Abstract

We are witnessing a time of transition where "data" is the main protagonist. Today, an enormous amount of information is generated every day in what is known as the Big Data era. Decision making based on this data, its structuring, organization as well as its correct integration and analysis, constitute a key factor for many strategic sectors of society. In the treatment of large amounts of data, the storage and analysis techniques associated with Big Data provide us with great help. These techniques are dominated by algorithms known as machine learning, essential for predictive analysis from large amounts of data. In the field of machine learning, fuzzy clustering algorithms are often used to solve a wide variety of problems, mainly those related to the control of complex industrial processes, decision systems in general, data resolution and compression. Classification systems are also widely used in everyday technology, for example in digital cameras, air conditioning systems, etc.

The success of the use of machine learning techniques is limited by the constraints of current computational resources, especially when working with large datasets and real-time constraints. In this context, such algorithms need to be redesigned and even rethought in order to take full advantage of massively parallel architectures that offer maximum performance today. This doctoral thesis focuses on this context, analyzing computationally the current panorama of clustering algorithms and proposing parallel clustering algorithms that allow us to offer adequate solutions in a reduced time interval. Specifically, an in-depth study of well-known machine learning techniques has been carried out by means of a practical application case. This application predicts the ozone level in different areas of the Region of Murcia. This analysis was based on the collection of different pollution parameters for each day during the years 2013 and 2014. The study revealed that the technique that obtained the best results was Random Forest and regionalisa-

tion was obtained in two large areas, based on the processed data.

The focus was then on the fuzzy clustering algorithms. In this case, a modification of the Fuzzy C-Means algorithm (FCM), μFCM, was used as a discretization technique with the aim of converting input data from continuous to discrete. This process is especially important because there are certain algorithms that need discrete values to work, even techniques that work with continuous data, get better results with discrete data. This technique was validated through the application to Anderson's well-known set of Iris Data, where it was statistically compared with the technique of K-Means (KM), providing better results.

Once the study of the fuzzy clustering algorithms has been carried out, it is detected that these techniques are sensitive to the amount of data, increasing their computational time. So the efficiency in the programming of these algorithms is a critical factor for their possible applicability to Big Data. Therefore, the parallelization of a fuzzy clustering algorithm is proposed in order to make the application faster as the degree of parallelism of the system increases. For this purpose, the fuzzy clustering algorithm Parallel Fuzzy Minimals (PFM) was proposed and compared with the FCM and Fuzzy Minimals (FM) algorithms in different data sets. In terms of quality, the results of the classification was similar to that obtained by the three algorithms, however, in terms of scalability, the parallelized PFM algorithm obtained a linear acceleration with respect to the number of processors used.

Having identified the need for such techniques to be developed in massively parallel environments, a high-performance hardware and software infrastructure is proposed to process, in real time, data obtained from various vehicles in relation to variables analysing pollution and traffic problems. The results showed adequate system performance working with large amounts of data and, in terms of scalability, the executions were satisfactory. Great challenges are visualized when it comes to identifying other applications in environments Big Data and being able to use these techniques for prediction in areas as relevant as pollution, traffic and smart cities.

Key words: Data mining, Machine learning, Fuzzy clustering algorithms, High performance computing, Air pollution, Intelligent transport systems, Smart cities

Índice

Capítulo 1

Introducción

Estamos presenciando la siguiente "revolución industrial", cuyo ingrediente principal es la ingente generación de datos. Motivados, principalmente, por la autogeneración de contenidos en redes sociales, el internet de la cosas y la continua digitalización de servicios, la cantidad de datos disponibles y almacenados crece de manera exponencial. Sin lugar a dudas, el análisis de estos datos puede ofrecer, no sólo nuevos servicios, sino también un nuevo modelo empresarial que garantice la viabilidad económica de nuestro tejido productivo.

La búsqueda de patrones en datos no es algo nuevo; son procedimientos anhelados desde el principio de nuestros días para analizar cosas tan dispares como el comportamiento de las migraciones animales, la optimización del crecimiento en cultivos agrícolas o la toma de decisiones sobre la intención de voto en campañas electorales. El trabajo de los especialistas, conocidos en la actualidad como científicos de datos, es dar sentido a los datos, descubrir los patrones que gobiernan el mundo físico y encajarlos en teorías que puedan ser utilizadas para predecir lo que puede suceder en nuevas situaciones.

Han *et al.* explican en [19] que el rápido crecimiento, la tremenda cantidad de datos recolectados y almacenados en grandes y numerosos repositorios, ha superado con creces nuestras capacidades cognitivas. Se han he-

cho esfuerzos para desarrollar sistemas expertos y tecnologías basadas en conocimiento, que normalmente dependen de los usuarios o expertos en la materia para introducir (manualmente) información en las bases de datos. Sin embargo, el procedimiento manual de generación de conocimiento es propenso a sesgos, errores y muy costoso en tiempo, lo que dificulta su aplicación real. La brecha, cada vez mayor, entre datos e información exige el desarrollo sistemático de herramientas de minería de datos que respondan a los requisitos de tiempo establecidos en el contexto de aplicación.

En el área de la minería de datos se han realizado diferentes trabajos con la idea de que pueden buscarse, identificarse y validarse automáticamente patrones en los datos y, posteriormente, estos pueden ser empleados para la predicción. Así es como Witten *et al.* definen en [43] la minería de datos y Hand, en [20], la establece como el descubrimiento de estructuras interesantes, inesperadas o valiosas en grandes conjuntos de datos. La minería de datos se compone de varias técnicas, incluyendo el aprendizaje automático y el análisis estadístico, sólo por mencionar algunas de ellas. Sin embargo, el objetivo principal de la minería de datos es apuntar a un escenario específico que se modela con un conjunto particular de datos con el fin de tratar un problema o situación concreta [34].

La adquisición, el almacenamiento, la gestión y el análisis de datos son áreas en las que surgirán grandes retos debido al marcado aumento de manejo de datos en la era del *Big Data*. La gestión de datos tradicional y los sistemas de análisis se basan en el *Relational database management system* (RDBMS). Sin embargo, tales sistemas sólo se aplican a los datos estructurados, excluyéndose los semiestructurados o no estructurados.

Los RDBMS tradicionales se han visto superados con el enorme volumen de datos y la heterogeneidad de los mismos, presentes en la era del *Big Data*. Para aliviar esto, se han propuesto algunas soluciones desde diferentes perspectivas por parte de la comunidad investigadora. Un ejemplo de ello sería el uso de la computación en la nube con la finalidad de cubrir los requisitos de infraestructura para grandes volúmenes de datos, entre otros, la eficiencia de costes o la adaptabilidad. Para soluciones de almacenamiento

y gestión de grandes bases de datos desordenados se han desarrollado buenas aproximaciones de sistemas de archivo distribuidos [25] y NoSQL [10]. Dichos sistemas han logrado un gran éxito en el procesamiento de tareas agrupadas, especialmente para el posicionamiento de páginas *web* [3,36,37]. Muchas aplicaciones de *Big Data* se pueden desarrollar sobre este tipo de plataformas emergentes pero el despliegue de estos sistemas de análisis de datos no es trivial. En la literatura [1,11,33] se discuten algunos de los obstáculos en el desarrollo de aplicaciones para *Big Data*. Los principales desafíos se enumeran a continuación:

- **Representación de datos:** Muchos conjuntos de datos tienen ciertos niveles de heterogeneidad en el tipo, la estructura, la semántica, la organización y la accesibilidad. La representación tiene como objetivo que los datos sean más significativos para el análisis computacional y la interpretación del usuario. Sin embargo, una representación de datos incorrecta puede reducir el valor de los datos originales e, incluso, obstruir un análisis de datos eficaz. La representación de datos eficiente reflejará la estructura de los mismos, clase y tipo, así como las tecnologías integradas, con la finalidad de permitir diversas operaciones en diferentes bases de datos.

- **Reducción de la redundancia y compresión de los datos:** En general, los conjuntos de datos contienen un alto nivel de redundancia. La reducción de ésta, junto con la compresión de los datos, permitiría que el sistema pudiera reducir los costes indirectos, teniendo en cuenta siempre que no se vieran afectados los posibles valores de los datos. A modo de ejemplo, cuando los datos son generados por redes de sensores, la mayoría de ellos son altamente redundantes, sin embargo, es posible que puedan ser filtrados y comprimidos substancialmente.

- **Gestión de datos cíclicos:** En comparación con la relativa lentitud de los avances de los sistemas de almacenamiento, se están generando datos sin precedentes y a gran escala. Nos enfrentamos a una gran

cantidad de desafíos apremiantes, uno de los cuales es que el sistema de almacenamiento actual no puede soportar estos datos masivos. En términos generales, existen dependencias de los valores ocultos en *Big Data* con otros datos. Por lo tanto, es importante el desarrollo de un principio relacionado con el valor analítico de los datos para decidir cuáles se almacenarán y cuáles se descartarán.

- **Mecanismos de análisis:** Los sistemas de análisis de *Big Data* deberán ser capaces de procesar grandes cantidades de datos heterogéneos en un tiempo limitado. Sin embargo, los tradicionales RDBMSs están estrictamente diseñados con falta de escalabilidad y capacidad de expansión, hecho que impide cumplir con los requisitos de rendimiento. Por contra, las bases de datos no relacionales han demostrado sus ventajas únicas en el tratamiento de datos no estructurados y comenzaron a convertirse en la principal corriente para el análisis de grandes volúmenes de información. Aun así, todavía hay algunos problemas de estas bases de datos en su rendimiento y aplicaciones particulares.

- **Confidencialidad de los datos:** Los servicios de proveedores de datos o los propietarios tienen una limitada capacidad efectiva, lo que no les permite, en la actualidad, mantener y analizar esos enormes conjuntos de datos. Estas tareas se han tenido que ir delegando en profesionales o en herramientas para el análisis de esos datos, lo que supone un aumento de los riesgos potenciales de seguridad. El hecho de que el análisis de *Big Data* pueda ser entregado a un tercero para procesar, implica que se tomen las medidas preventivas adecuadas, con el fin de proteger dichos datos sensibles y garantizar su seguridad.

- **Expansión y escalabilidad:** Los sistemas de análisis de *Big Data* deben ser compatibles con los conjuntos de datos actuales y futuros. Las herramientas empleadas para el análisis deberían ser capaces de procesar la creciente expansión y los conjuntos de datos más complejos.

- **Cooperación:** El análisis de grandes conjuntos de datos es una inves-

tigación interdisciplinar. Es por ello que, para extraer el potencial de grandes volúmenes de datos, se requiere de expertos en diferentes campos que cooperan entre sí. Para ayudar a los científicos y especialistas en diversos campos a acceder a diferentes tipos de datos y utilizar plenamente su experiencia, se debe establecer una gran arquitectura de red de datos, con el fin de cooperar para alcanzar objetivos comunes.

- **Gestión del gasto energético:** El consumo de energía de los sistemas de computación ha alertado a la comunidad desde la perspectiva tanto de la economía como del medioambiente [41]. Con el aumento del volumen de datos y necesidades de análisis, el procesamiento, almacenamiento y transmisión de datos consumen, inevitablemente, cada vez más energía eléctrica. Por lo tanto, se debería establecer controles en los niveles de potencia de los sistemas y mecanismos de gestión de *Big Data* teniendo aseguradas las capacidades de expansión y accesibilidad.

1.1 Estrategias algorítmicas de minería de datos

1.1.1 Técnicas de *machine learning*

En la práctica, la mayoría de las tareas que requieren una habilidad similar a la inteligencia humana también deben ser capaces de inducir nuevos conocimientos basándose en las experiencias realizadas. Se dice que un programa aprende alguna tarea de la experiencia si su rendimiento en la tarea mejora con dicha experiencia, de acuerdo con alguna medida de rendimiento [34]. El aprendizaje automático (o *machine learning*) investiga cómo los ordenadores pueden aprender, o mejorar su rendimiento, basándose en la información que proporciona el tratamiento de datos. Una de las principales áreas de investigación dentro de este campo es aprender automáticamente, lo que supone reconocer patrones complejos y tomar decisiones inteligentes, basadas en datos, de manera automática. Han *et al.* clasifican en [19] el *machine*

learning en cuatro categorías definidas como sigue:

- **Aprendizaje supervisado.** El algoritmo deduce una función, a partir de datos de entrenamiento, la cual establece una correspondencia entre las entradas del sistema y las salidas deseadas. Un ejemplo de este tipo de algoritmos es el problema de la clasificación, donde el sistema de aprendizaje trata de clasificar una serie de vectores en una o varias categorías (clases). La base de conocimiento del sistema está formada por ejemplos que han sido etiquetados anteriormente, los datos de entrenamiento.

- **Aprendizaje no supervisado.** En este caso, el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado por entradas del sistema. No se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas. Típicamente, podemos usar las técnicas de agrupación para descubrir clases dentro de los datos.

- **Aprendizaje semisupervisado.** Estos algoritmos combinan los dos tipos de algoritmos anteriores, supervisados y no supervisados, para poder obtener la clasificación. Se tienen en cuenta tanto los datos marcados como los no marcados.

- **Aprendizaje activo.** Es un enfoque que permite a los usuarios desempeñar un papel activo en el proceso de aprendizaje. El algoritmo aprende mediante la observación del mundo que le rodea. Su información de entrada es el *feedback* o retroalimentación que obtiene como respuesta a sus acciones. Por lo tanto, el aprendizaje del sistema se lleva a cabo a base de ensayo-error.

1.1.2 Estrategias de aprendizaje (supervisado vs. no supervisado)

La clasificación se puede definir, en términos generales, como la realización de algún pronóstico o la toma de decisiones en base a la información que

se posee en un momento dado. Diferentes métodos de clasificación han sido propuestos en el aprendizaje automático y el reconocimiento de patrones [26]. La mayoría de los algoritmos trabajan asumiendo un gasto computacional inevitable, a causa de la cantidad de datos procesados. Sin embargo, las investigaciones recientes en el campo de la minería de datos se han basado en el desarrollo de técnicas escalables de clasificación y predicción capaces de manejar grandes cantidades de datos [32]. La clasificación tiene numerosas aplicaciones, entre las que se incluyen la detección de fraudes, el *marketing*, la predicción del rendimiento, el diagnóstico médico y la explotación de datos a tiempo real en ciudades inteligentes como, por ejemplo, control del tráfico y la contaminación [13, 21, 38].

En el aprendizaje supervisado, la clasificación se puede definir como la construcción de procedimientos a los que se les aplica una secuencia continua de ejemplos, donde a cada nuevo ejemplo se le debe asignar una de las clases ya predefinidas dentro de un conjunto, en base a las observaciones de sus atributos [14]. Todo algoritmo que trate de resolver la tarea de la clasificación, intenta construir un modelo (clasificador) a partir de un conjunto de ejemplos o datos de entrada, denominado conjunto de entrenamiento. Posteriormente, estos clasificadores son usados para inferir la clase de ejemplos desconocidos. Se distinguen dos fases en el aprendizaje, el entrenamiento y el test del modelo aprendido [34].

En la primera fase se trata de hacer que la técnica de aprendizaje extraiga las conclusiones apropiadas del conjunto de ejemplos de entrenamiento y devuelva un modelo que sea capaz de mostrar lo aprendido. En la segunda fase se estudia la precisión del modelo aprendido probándolo con un conjunto de datos diferente y estimando, en la mayoría de los casos, unos coeficientes de error. Una vez que está disponible un modelo, se puede inferir la clase de dichos ejemplos al resto de valores del sistema.

El aprendizaje es "supervisado" en el sentido de que se le proporciona el conocimiento previo de a qué clase pertenece cada elemento. Éste contrasta con el aprendizaje no supervisado, en el que no se extrae previamente un modelo y, en algunos casos, es posible que no se conozca de antemano el

número o conjunto de clases en las que se puede clasificar.

Como técnicas de aprendizaje no supervisado, los algoritmos de agrupación (o *clustering*) han sido aplicados al análisis de conjuntos de datos de varios campos como el procesamiento de imágenes, el reconocimiento de patrones, el análisis de datos de microarrays en bioinformática, etc. [2,8,12,44], con el fin de proporcionar un conocimiento valioso dentro de los mismos. Dependiendo de las propiedades de los datos o del objetivo de la agrupación se han ido desarrollando diferentes tipos de algoritmos. Además, muy raramente los datos presentan una solución de *clustering* única y también es difícil interpretar la representación del grupo (o *cluster*). Por lo tanto, requieren muchas experimentaciones con diferentes algoritmos o con diferentes características del mismo conjunto de datos y resultan computacionalmente costosos, lo que supone que el ahorro de complejidad computacional sea un problema significativo para los algoritmos de *clustering*.

El análisis de *clustering* se define como la organización de una colección de patrones en grupos basados en la similitud. Algunas características de este tipo de algoritmos están descritas en [31]. El objetivo de este tipo de algoritmos es separar o dividir un conjunto de datos no etiquetados en varios grupos, basándose en las propiedades de los conjuntos de datos de entrada. Además, al tratarse de un aprendizaje no supervisado, divide los conjuntos de datos en grupos sin necesidad de introducir un conocimiento previo. Un algoritmo de agrupación requiere varios pasos esenciales. El procedimiento de agrupación se define en [45] como: 1) selección de características (o atributos); 2) diseño o selección de algoritmo de clasificación; 3) validación de la agrupación; y 4) interpretación de los resultados.

En general, hay un conjunto de características deseables para los algoritmos de agrupación, que están definidas en [5,31]. Entre otras: escalabilidad al trabajar con grandes conjuntos de datos; capacidad para detectar valores atípicos en el conjunto; insensibilidad al orden de los datos de entrada; que el número de parámetros introducidos por el usuario sea el mínimo posible; o que la forma de los grupos no esté predeterminada.

1.1.3 Tipos de clasificación (*fuzzy* vs. *hard*)

En la literatura [5, 26, 27, 45] clasifican los algoritmos de *clustering* según diferentes puntos de vista. Siguiendo la categorización de Andreopoulos *et al.* en [5], los algoritmos de *clustering* pueden ser clasificados hasta en seis tipos: particionales, jerárquicos, evolutivos, basados en densidades, basados en modelos y basados en gráficos. Sin embargo, la mayoría de los algoritmos podrían clasificarse dentro de uno de los dos métodos siguientes: algoritmos jerárquicos y de partición.

Los algoritmos de agrupación jerárquica dividen los datos en un árbol de nodos, donde cada nodo representa un mapa de ruta del *cluster*. En algunas aplicaciones, como las bioinformáticas, los métodos de agrupación jerárquica son más populares, ya que pueden tener varios niveles de subconjuntos. Por contra, los métodos jerárquicos son lentos, los errores no son tolerables y las pérdidas de información son comunes al mover los niveles. Algunos de los algoritmos de agrupación jerárquica más empleados son: BIRCH [46], CURE [17], Spectral Custering [42], ROCK [18] y LIMBO [6].

Por su parte, los métodos de partición son útiles para las aplicaciones donde se requiere un número fijo de grupos, ya que asignan los elementos del conjunto de datos a *k clusters* [45]. Como algoritmos particionales se sitúan, entre otros, K-Means, K-Modes, Fuzzy K-Modes, Squeezer o COOLCAT, Farthest First Traversal K-Center (FFT) [23], K-Medoids [28], PAM (Partitioning Around Medoids), CLARA (Clustering Large Applications) [29], CLARANS (Clustering Large Applications Based Upon Randomized Search) [30] y Fuzzy K-Means [24].

Dentro de los métodos particionales, es posible dividir los algoritmos atendiendo a la lógica empleada en su clasificación: *fuzzy* (difusa) o *hard*. La diferencia reside en el resultado de la clasificación realizada, más concretamente en la pertenencia de cada elemento a cada uno de los grupos obtenidos. Cuando la clasificación es *fuzzy* los grupos se superponen, de forma que un elemento puede pertenecer a varios grupos diferentes. Por el contrario, cuando es *hard*, los grupos son disjuntos y cada elemento pertenece a uno, y

sólo uno, de los grupos obtenidos. Para ilustrar la diferencia entre las lógicas *hard* y *fuzzy*, a continuación se explica cómo funcionan dos de los algoritmos particionales anteriormente mencionados: *K-Means* (KM) y *Fuzzy C-Means* (FCM).

El algoritmo KM es uno de los métodos clásicos de clustering más populares [22]. Este algoritmo comienza por determinar los centros de los k grupos y, luego, continúa con los siguientes pasos, hasta que se logre la convergencia. Cada grupo se construye asignando cada elemento al centro de grupo más cercano y cada centro es, posteriormente, reemplazado por la media de los elementos pertenecientes a ese grupo. Luego, el algoritmo KM asigna cada elemento a un grupo de forma inequívoca.

Por su parte, el algoritmo FCM es una extensión del algoritmo KM desarrollada por Dunn en [15] y mejorada por Bezdek en [9] a principios de los ochenta y es uno de los métodos de clasificación *fuzzy* más utilizados. El análisis de *clustering* FCM tiene como objetivo encontrar los patrones en los datos mediante el cálculo de distancias entre los elementos a los centros de grupos. De esta forma, un elemento x puede pertenecer a diferentes grupos al mismo tiempo, y el grado en que pertenece al grupo k-ésimo se expresa en términos de un grado de pertenencia.

1.2 Desarrollo eficiente de aplicaciones

El análisis de datos masivos recolectados, por ejemplo, en una ciudad, se ha vuelto cada vez más relevante para identificar nuevas percepciones y ofrecer soluciones novedosas para problemas futuros. De hecho, estos grandes conjuntos de datos deben procesarse normalmente con requisitos en tiempo real para poder aportar soluciones en tiempo y forma. En este contexto, la computación de alto rendimiento (*High Performance Computing* o HPC) es fundamental para poder ofrecer garantías de viabilidad a estas soluciones y es obligatorio pensar en esta área desde las primeras etapas del desarrollo de la infraestructura para hacer frente a dichos requisitos.

El mercado de HPC está siendo testigo de la transición constante de sis-

temas de computación homogéneos a heterogéneos, donde los nodos combinan arquitecturas multinúcleo (CPU) y aceleradores tradicionales (representados principalmente por el movimiento de computación en la GPU o las tarjetas *Intel Xeon Phi*). En sistemas informáticos heterogéneos los programadores juegan un papel fundamental, ya que están llamados a rediseñar aplicaciones para maximizar el rendimiento con el paralelismo como ingrediente obligatorio. En pocos años, el campo de las GPGPU (*General Purpose application on GPU*) ha crecido muy rápidamente y se ha convertido en una de las mejores formas de lograr un alto rendimiento de cálculo desde plataformas heterogéneas novedosas. En particular, *Compute Unified Device Architecture* (CUDA) es uno de los principales ejemplos de GPGPU y ofrece una plataforma para GPU que incluye tanto *software* como *hardware*.

Incluso con estos grandes avances en los campos de HPC, la cantidad de datos a procesar crece exponencialmente cada año. Por lo tanto, el análisis de datos debe realizarse con técnicas que no requieran grandes tiempos de cálculo. Para ello se deben buscar algoritmos que se adapten al nuevo paradigma de computación en HPC y que exploten todos los recursos computacionales disponibles. Muchos de los algoritmos de *Soft Computing* son intrínsecamente paralelos por su definición, y la combinación con HPC es obligatoria para proporcionar nuevas aplicaciones para entornos inteligentes donde la clasificación difusa pueda aportar un cierto beneficio.

Con la idea de potenciar el rendimiento de una aplicación paralela, un primer factor a tener en cuenta es identificar su patrón de cómputo y estudiar la idoneidad para cada tipo de arquitectura. Uno de los mayores obstáculos para innovar en computación paralela es que no está claro cómo es mejor expresar dicha computación. Por tanto, existe una necesidad de encontrar un nivel más alto de abstracción para razonar sobre los requisitos de una aplicación paralela. En [7] se describen un conjunto de *benchmarks*, llamados *dwarfs*, que definen un patrón de cómputo y de comunicaciones comunes para un conjunto importante de aplicaciones para, posteriormente, describir las mejores plataformas donde ejecutar dichas aplicaciones en base a esos patrones.

La escalabilidad de las aplicaciones para sistemas paralelos y la portabilidad entre sistemas de distinta naturaleza son también otros factores críticos para el rendimiento de las aplicaciones paralelas. Debido al continuo incremento de *cores* de los sistemas paralelos actuales, el objetivo principal subyacente es conseguir que la aplicación tenga una cierta escalabilidad, es decir, sea más rápida conforme aumente el grado de paralelismo del sistema. Además, otro parámetro fundamental para el desarrollo de aplicaciones que mejoren el rendimiento es la portabilidad de las aplicaciones paralelas entre sistemas paralelos de distinta naturaleza.

Finalmente, el rendimiento de cualquier aplicación y, en concreto, de las aplicaciones paralelas, está estrechamente ligado al modelo de programación subyacente de la propia aplicación. Se han propuesto muchos modelos de programación paralelos durante las últimas décadas [35]. Los más usados para *clusters* de procesadores con un modelo de memoria distribuida son *Message Passing Interface* y, para multiprocesadores de memoria compartida, OpenMP. Estos dos modelos de programación son estándares de programación paralela, especialmente para aplicaciones codificadas según el paradigma un único programa trabajando sobre múltiples datos (*Single Program, Multiple Data*; SPMD). CUDA ofrece un nuevo modelo de programación desarrollado principalmente para un paradigma donde una misma instrucción es ejecutada sobre múltiples datos (*Single Instruction, Multiple Data*; SIMD), aunque para describir el problema a alto nivel también permite utilizar el modelo SPMD, y se ha demostrado su eficacia a la hora de codificar aplicaciones paralelas de propósito general en el entorno de las GPUs.

1.3 Objetivos

Esta tesis doctoral está fundamentada en base a un conjunto de objetivos que se enumeran y se detallan a continuación. Estos objetivos son los pilares sobre los que se asienta la investigación presentada en esta tesis doctoral, y corresponden a tres líneas interrelacionadas entre sí. La primera línea de investigación está orientada a avanzar el estado del arte de la paralelización de

algoritmos de clasificación difusa. La segunda línea de trabajo se encamina a mejorar la eficiencia y rendimiento de estos algoritmos en plataformas de cómputo emergentes. Finalmente, la tercera línea se centra en la aplicación de los avances conseguidos en las líneas de investigación anteriores a diversos campos, donde el análisis de grandes cantidades de datos puede aportar un gran beneficio para la sociedad.

- **Objetivo 1.** Estudio de las técnicas de *Soft Computing* que permita mejorar los algoritmos existentes, con el fin de evitar posibles arbitrariedades y problemas con superposición de datos.
- **Objetivo 2.** Mejora del rendimiento de los algoritmos de clasificación difusa para ofrecer tiempos de respuesta reducidos en entornos de grandes cantidades de datos.
- **Objetivo 3.** Desarrollo de aplicaciones inteligentes en diferentes contextos sociales basadas en las técnicas de clasificación desarrolladas.

1.4 Estructura del documento de Tesis

A continuación se describe de forma resumida el contenido de cada uno de los capítulos que la componen, para guiar al lector y estructurar mejor el contenido de esta tesis doctoral:

- *Capítulo 1: Introducción*. En este primer capítulo se contextualiza la tesis doctoral analizando el campo de estudio, describiendo el problema al que nos enfrentamos y citando las soluciones planteadas por otros investigadores para resolverlo. También se enumeran los objetivos a alcanzar.
- *Capítulo 2: Publicaciones que componen la tesis doctoral*. En este capítulo se comienza con una fundamentación de la tesis y, a continuación, se incluyen las publicaciones que forman parte del compendio. Esta fundamentación consiste en demostrar que con el conjunto de publicacio-

nes científicas presentadas se obtienen los resultados que satisfacen los objetivos propuestos en el capítulo 1.

- *Capítulo 3: Conclusiones y líneas de trabajo futuras*. En este capítulo se comentan las conclusiones que se derivan de las publicaciones y se plantean posibles líneas de continuación de las investigaciones, con el fin de reafirmar la consecución de los objetivos propuestos.

- *Capítulo 4: Publicaciones y calidad de las revistas*. En este capítulo se resume una serie de indicadores de calidad que las revistas en las que se han publicado los artículos que forman el compendio tienen asociadas, con la finalidad de constatar la calidad de las publicaciones aportadas en esta tesis.

- *Bibliografía*. Se aportan las referencias empleadas en la introducción y fundamentación de la tesis doctoral, donde profundizar y conseguir más información de los estudios planteados.

Capítulo 2

Artículos que componen la tesis doctoral

2.1 Fundamentación de la tesis doctoral

La consecución de los objetivos planteados en el Capítulo 1 a través de publicaciones científicas en revistas de reconocido prestigio aporta solidez a la hora de justificar su cumplimiento. A continuación se va a identificar cada una de las cuatro publicaciones aportadas en esta tesis por compendio con el objetivo que aborda, desde el estudio y diseño de algoritmos eficientes hasta su posterior aplicación a entornos de *Big Data*.

En la línea del primer objetivo propuesto, surgen las dos primeras publicaciones. En el primer artículo, 2.2, se hizo un amplio análisis de las técnicas de *machine learning* y se realizó una comparativa de ellas, aplicándolas a la predicción del nivel de ozono (O_3) en diferentes zonas de la Región de Murcia (España). El O_3 es uno de los principales peligros para la salud cuando alcanza ciertos niveles; de hecho, contar con modelos precisos de predicción de la calidad del aire es un paso previo para tomar medidas de mitigación, actividades que pueden beneficiar a las personas con enfermedades respiratorias como asma, bronquitis o neumonía en el ámbito que actualmente conocemos como *smart cities* o ciudades inteligentes.

Se dispuso de un conjunto de datos real, obtenido de cinco estaciones atmosféricas situadas en cuatro ciudades diferentes de la Región (Alcantarilla, Aljorra, Lorca y Caravaca), que contenía el promedio por hora de diferentes parámetros climáticos y elementos químicos que afectan a la calidad del aire, para cada día durante los años 2013 y 2014. La predicción se centró en la medición del O_3 y se desarrolló en dos fases: en primer lugar, se evaluaron los niveles de O_3 a través de las técnicas de *machine learning* seleccionadas; y, en segundo lugar, se utilizó la técnica de agrupación jerárquica para valorar cuántos modelos serían necesarios para predecir el nivel de O_3 en la Región de Murcia.

Los resultados obtenidos fueron considerablemente satisfactorios para los modelos de predicción aplicados en las ciudades de la Región de Murcia implicadas en el estudio. La técnica que obtuvo los mejores resultados, en general, fue *Random Forest*, habiéndose validado la ejecución estadísticamente. Además, a través de la aplicación de la agrupación jerárquica, se obtuvo que la monitorización de la contaminación del aire podía dividirse en dos áreas, agrupando por un lado las ciudades de Lorca, Alcantarilla y Aljorra y, por otro lado, Caravaca.

En el segundo trabajo, 2.3, se combinó la clasificación con técnicas supervisadas y no supervisadas de *machine learning*. Se utilizó una modificación el algoritmo de *fuzzy clustering* FCM que se desarrolló en [40], denominada μFCM, como técnica de discretización y se emplea con el objetivo de convertir en discretos los valores de datos continuos a través de la utilización de particiones difusas.

En el artículo se explica que algunas técnicas de minería de datos necesitan trabajar con valores discretos. Es más, a pesar de que haya técnicas que sí puedan manipular valores continuos, obtienen resultados más satisfactorios al introducir valores discretos. Este hecho hace que el proceso de discretización sea una técnica fundamental para el pre-procesamiento de datos en el análisis inteligente de datos, ya que esta técnica tiene como objetivo principal convertir los atributos de un conjunto de datos de continuos a discretos.

Con el fin de evaluar el comportamiento de la propuesta, la técnica se

aplicó al conjunto *Iris Data* de Anderson, [4], que contiene las medidas de longitud y ancho de pétalos y sépalos de 150 iris (50 del tipo iris Versicolor, 50 del tipo iris Setosa y 50 del tipo iris Virgínica). Para estos experimentos se empleó un clasificador de *Extreme Learning Machine* y un comité de *Extreme Learning Machine*, además de comparar los resultados de la discretización obtenida utilizando el algoritmo KM. Estos experimentos fueron validados estadísticamente, donde el enfoque propuesto (μFCM) obtenía mejores resultados con un 97 % de nivel de confianza.

Ya establecido el mapa de los algoritmos de *machine learning*, se centra el estudio en el área de la clasificación difusa y, más concretamente, en los algoritmos *fuzzy* desarrollados en entornos paralelos. Tal y como se ha indicado con anterioridad, este tipo de algoritmos están enmarcados en el aprendizaje no supervisado, ya que no es necesaria la introducción de conocimiento previo. Es por esto que son de gran utilidad a la hora de clasificar conjuntos de datos y encontrar estructuras interesantes, hecho que sería difícil de conseguir sin la ayuda de la computación. Se detectó que la mayoría de estos algoritmos dependen de procesos iterativos con el objetivo de encontrar una solución óptima local o global en grandes conjuntos de datos. La capacidad de encontrar dichas soluciones habitualmente requiere la realización de diferentes experimentos con diferentes algoritmos para encontrar el adecuado. Esta cuestión desemboca en la necesidad de desarrollar los algoritmos de clasificación en entornos paralelos en la era actual, la era del *Big Data*.

Se han realizado grandes esfuerzos para obtener una paralelización de uno de los algoritmos de clasificación difusa más empleado, el FCM, con el objetivo de aplicarlo a grandes conjuntos de datos. Sin embargo, los resultados demuestran que los tiempos de ejecución crecen exponencialmente en relación al tamaño de la muestra, además de que necesita la introducción previa del número de grupos en los que debe dividirse el conjunto. Esta última cuestión implica que deban realizarse varias ejecuciones para poder encontrar el número óptimo de grupos. Como alternativa al algoritmo FCM y teniendo en cuenta las trabas encontradas a la hora de programarlo en un entorno paralelo, decidimos utilizar el algoritmo de clasificación difusa

Fuzzy Minimals (FM), que fue propuesto en [16].

En el artículo 2.4 se presenta la versión paralela del algoritmo FM, *Parallel Fuzzy Minimals* (PFM). Dicha paralelización se inicia en el procesador principal que realiza el cálculo del factor *r* (parámetro que mide la isotropía del conjunto de datos) y divide la muestra, equitativamente, entre todos los procesadores (secundarios) que participan en la ejecución. A continuación, cada uno de los procesadores aplica el algoritmo FM sobre el subconjunto de datos que le ha sido asignado y devuelve, al procesador principal, la clasificación hallada. Finalmente, el procesador principal unifica todos los grupos encontrados a través de la técnica de agrupación jerárquica de dendrograma [39] para obtener la clasificación definitiva.

La paralelización del algoritmo PFM fue validada a través de la comparación del resultado de la clasificación con los algoritmos FCM y FM en tres conjuntos diferentes de datos. En primer lugar, utilizamos el conocido conjunto de *Iris Data* de Anderson [4]. El segundo conjunto fue creado artificialmente para testear la escalabilidad del algoritmo PFM. Consistía en 5000 datos tridimensionales que representaban tres elipsoides disjuntos, pero lo suficientemente próximos para comprobar, a la vez, la calidad de la clasificación. Por último, se probó en un conjunto bidimensional de 15506 registros, que contenía las coordenadas geográficas de los hospitales de Estados Unidos. Este conjunto fue desordenado aleatoriamente para evitar cualquier tipo de correlación. La evaluación del rendimiento en los tres conjuntos de datos reveló que el enfoque proporcionado obtenía una aceleración lineal, mientras que la calidad de la agrupación se mantenía estable en comparación con la versión secuencial.

Identificada la necesidad de que el análisis de datos debe realizarse con técnicas que no requieran grandes tiempos de cálculo, esto es, mejorar en términos de rendimiento y escalabilidad y con el objetivo de aplicarlo a entornos *Big Data*, se publica el último artículo, 2.5. En éste se presentó una infraestructura de hardware y software de alto rendimiento para recopilar información de una serie de vehículos y procesarla eficientemente para proporcionar servicios basados en Sistemas de Transporte Inteligente (*Intelligent*

Transport Systems o ITS) novedosos.

En este trabajo se propuso un enfoque de paralelización de la técnica desarrollada en el artículo 2.4 en servidores heterogéneos basados en CPU y varias GPUs, adaptado a los problemas de clasificación en los servicios de comunicación Vehículo-a-Infraestructura (V2I). Dicha infraestructura fue probada empíricamente para ofrecer un servicio de información sobre contaminación geolocalizada a través de la recogida periódica de datos de posición y estado del vehículo. Esto permitió ofrecer un servicio a tiempo real que identificaba correctamente áreas de tráfico contaminadas y prácticas en conductores altamente contaminantes.

Esta infraestructura fue aplicada a la monitorización en dos escenarios diferentes. En el primer escenario se evaluó el consumo de combustible de un vehículo a través de la aplicación del algoritmo PFM a 843 muestras que contenían información acerca de la velocidad del vehículo, RPM, temperatura del motor y tasa de combustible. En el segundo escenario se midió la contaminación del aire en diversas áreas mediante el análisis de 4100 muestras de cuatro vehículos, conteniendo información de los mismos parámetros que en el escenario 1 además de las coordenadas GPS, con el objetivo de monitorizar la contaminación en el aire de varias áreas específicas en la Región de Murcia (España).

Los resultados indicaron un buen rendimiento del sistema bajo altas cargas, y el análisis de escalabilidad reveló una buena ejecución en un entorno realmente ambicioso gracias al uso tanto de CPU como de múltiples GPUs. Este hecho demostró que la propuesta puede albergar eficientemente servicios cooperativos que impliquen un alto grado de procesamiento en el contexto de los ITS.

2.2 Air-pollution prediction in smart cities through machine learning methods: A case of study in Murcia, Spain

Título	*Air-pollution prediction in smart cities through machine learning methods: A case of study in Murcia, Spain*
Autores	Raquel Martínez-España, Andrés Bueno-Crespo, Isabel Timón, Jesús Soto, Andrés Muñoz y José M. Cecilia
Revista	Journal of Universal Computer Science
Año	2018
Estado	Publicado

Contribución de la doctoranda

Isabel María Timón Pérez declara, con el visto bueno del resto de autores, ser el principal autor y la principal contribuidora del artículo *Air-pollution prediction in smart cities through machine learning methods: A case of study in Murcia, Spain*.

Journal of Universal Computer Science, vol. 24, no. 3 (2018), 261-276
submitted: 15/10/17, accepted: 5/3/18, appeared: 28/3/18 © *J.UCS*

Air-Pollution Prediction in Smart Cities through Machine Learning Methods: A Case of Study in Murcia, Spain

Raquel Martínez-España, Andrés Bueno-Crespo, Isabel Timón, Jesús Soto, Andrés Muñoz, José M. Cecilia
(Department of Computer Engineering, Universidad Católica de Murcia, Spain
{rmartinez, abueno, imtimon, jsoto, amunoz, jmcecilia}@ucam.edu)

Abstract: Air-pollution is one of the main threats for developed societies. According to the World Health Organization (WHO), pollution is the main cause of deaths among children aged under five. Smart cities are called to play a decisive role to improve such pollution by first collecting, in real-time, different parameters such as SO_2, NO_x, O_3, NH_3, CO, PM_10, just to mention a few, and then performing the subsequent data analysis and prediction. However, some machine learning techniques may be more well-suited than others to predict pollution-like variables. In this paper several machine learning methods are analyzed to predict the ozone level (O_3) in the Region of Murcia (Spain). O_3 is one of the main hazards to health when it reaches certain levels. Indeed, having accurate air-quality prediction models is a previous step to take mitigation activities that may benefit people with respiratory disease like Asthma, Bronchitis or Pneumonia in intelligent cities. Moreover, here it is identified the most-significant variables to monitor the air-quality in cities. Our results indicate an adjustment for the proposed O_3 prediction models from 90% and a root mean square error less than 11 μ/m^3 for the cities of the Region of Murcia involved in the study.

Key Words: Air-pollution monitoring, Ozone, Smart cities, Machine learning, Random forest, Hierarchical clustering.
Category: I.2, H.3.5, H.4

1 Introduction

Smart cities have become an endless source of urban data. These data range from traffic events to data related to the management of public resources, through indicators about the citizens' quality of life. Among the latter, one of the most important indicators is related to air quality. According to the World Health Organization (WHO), the air pollution is a leading cause of chronic or non-communicable diseases (NCDs), causing over one-third of deaths from stroke, lung cancer and chronic respiratory disease, and one-quarter of deaths from ischaemic heart disease [WHO, 2018]. In fact, this issue is included by the European Union as one of the challenges for smart cities in its H2020 programme, recently debated in the European Forum on Eco-Innovation[1].

Air quality is affected by several factors including airborne particulate matter (PM), sulfur dioxide (SO_2), Nitrogen dioxide (NO_2) and Ozone (O_3), just to

[1] `http://ec.europa.eu/environment/ecoinnovation2018/1st_forum/case-studies_02_en.html`

mention a few. Several works have shown that short-term O_3 exposures within a period of 1 to 2 days may be directly related to acute coronary events in middle-aged adults without heart disease [Ruidavets et al., 2005, Brook et al., 2002]. Indeed, the continuous monitoring of these variables can provide firm foundation for creating models to predict hypothetical high-level concentrations of a determinate polluted factor. Several works have been done in developing IoT infrastructures for air pollution monitoring [Al-Ali et al., 2010, Shaikh et al., 2017, Afshar-Mohajer et al., 2018]. However, the fast-growing, tremendous amount of data, collected and stored in large and numerous data repositories, have far exceeded our human ability for comprehension without computational tools. Some efforts have been made to develop expert system and knowledge-based technologies, which typically rely on users or domain experts to *manually* input knowledge into knowledge databases. However, the manual knowledge input procedure is prone to biases and errors and it is extremely costly and time-consuming. The widening gap between data and information calls for the systematic development of *data mining tools*.

In the data mining area, the data is stored electronically and the analysis is automated, or at least augmented, by computers. Some efforts have been done with the idea that patterns in data can be automatically sought, identified, validated, and used for prediction. Data mining is defined in [Witten et al., 2016] as the process of discovering patterns in data and, in [Hand, 2007], it is stated as the discovery of interesting, unexpected or valuable structures in large datasets. Indeed, the process must be automatic or, more usually, semiautomatic and the patterns discovered must be meaningful in a practical sense. Data mining is composed of several techniques, including machine learning and statistics analysis, just to mention few of them. However, the data mining main goal is to target a specific scenario which is modeled with a particular data set in order to deal with a specific problem or situation.

In practice, most tasks that require intelligence also require an ability to induce new knowledge from experiences. A computer program is said to learn some task from experience if its performance at the task improves with experience, according to some performance measure. *Machine learning* investigates how computers can learn, or improve their performance, based on data. A main research area for computer programs is to automatically learn to recognize complex patterns and make intelligent decisions based on data. In [Han et al., 2011] machine learning is classified in four categories: supervised learning, unsupervised learning, semi-supervised learning and active learning.

One of the aims of smart cities is to act on the basis of data obtained through sensors. However, it must be noted that the sensors may cause failures and errors when obtaining data, hence it is necessary to develop a model to predict values of interest in order to control air quality. The main goal of this paper is to analyze

different machine learning techniques to predict ozone levels. The aim of this analysis is to obtain the best model for predicting ozone. Thus, in the event of a sensor failure, the model can predict with the least possible error the amount of ozone in the air in order to create an alert if the recommended thresholds are exceeded and take the appropriate measures. Specifically, this analysis has been performed in four cities at the Region of Murcia (Spain) taking real data from 4 stations for air quality measurement.

The rest of the paper is structured as follows. Section 2 describes work related to air quality and ozone prediction in cities. Section 3 presents the machine learning techniques used for ozone prediction as well as for the study of the most important variables to consider for such prediction. Section 4 shows the assessment performed to obtain the best models for the ozone prediction and a clustering of stations according to the similarity of the data. Finally, Section 5 describes the conclusion and future work of this paper.

2 Related Work

Smart cities have attracted considerable attention in the context of urban development policies. The Internet and broadband networking technologies are seen as enablers of e-services and are becoming increasingly important for urban development, while cities increasingly assume a key role as drivers of innovation in areas such as health, inclusion, environment and business [Schaffers et al., 2011]. Thus, for example, there are many studies related to traffic control in cities [Bui and Jung, 2017, Shaghaghi et al., 2017], and this topic is related to the environment in smart cities. In this same area and related to traffic control is the topic of air quality monitoring. Many of the works in the area of air pollution in smart cities focus on the monitoring of parameters considered as pollutants. Thus, in [Al-Ali et al., 2010] a GPRS-Sensor array system is proposed to report real-time pollution level in a Google map. Basically, this system is empowered with CO, NO_2 and SO_2 pollution sensors augmented with GPS data including location, date and time. Authors show a proof of concept for the city of Sharjah in the United Arab Emirates. A more advanced approach is presented in [Jin et al., 2014], where an IoT-based infrastructure called IDRA offers several environmental monitoring services. Among them, authors highlight the vigilance of parameters such as hydrocarbons and oxides of nitrogen. Due to the rapid evolution of Information Technology (IT), we have entered the age of Big Data in multiple areas of research (see for example [Jung, 2017a, Jung, 2017b]). A Big Data analytics-based approach [Rathore et al., 2016] uses ozone, CO, NO_2 and SO_2 levels to, along with data from smart homes, traffic, time, surveillance, etc., assist in urban planning decision making. However, these works do not propose any technique for predicting pollution for the next days or identifying related factors.

With an ever-increasing air pollution ratios, it is necessary to implement efficient air quality monitoring models, obtained from the data collected by pollution sensors, that help to predict the concentration of air pollutants and provide assessment of air pollution in each area. Hence, air quality evaluation and prediction have become an important research area. In relation to works in the literature that do take into account the prediction of air pollution, there is a clear majority of use of artificial neural networks (ANN) compared to other models such as multiple linear regression(MLR) [Kurt et al., 2008, Azid et al., 2013, Ahmad et al., 2017, Azid et al., 2017]. Being this the general trend, it is important to observe that ANNs also present some weakness for this topic, as identified by [Zhang and Ding, 2017]: They have poor generalization, falling in local minimum with relative ease; they do not have an analytical method for model selection; and they follow a long-running process to obtain the most accurate model. Finally, it is worth mentioning new approaches that combine machine learning techniques with the use of social media data to understand and improve predictions on air pollution [Ravi et al., 2017].

In [Kang et al., 2018] various big-data and machine learning based techniques for air quality forecasting are investigated. The paper reviews the published research results relating to air quality evaluation using methods of artificial intelligence, decision trees, deep learning, etc. On the other hand, in [Reid et al., 2017] is explored the implementation of an Internet of Things multiagent system distributed along the roadway to collect and share vehicular data among its nodes and then process the data using a machine learning algorithm for inference of vehicle types. In [Li et al., 2017] is proposed a deep learning model to estimate air pollution throughout the city, utilizing the readily available urban data as proxy data. As with many big data driven approaches, the proxy data may be sparse/missing. The authors propose the M-BP algorithm to recover/fill in such missing data. The potential usage of machine learning and reduced-order modeling techniques to mitigate some of these limitations is discussed in [Keller et al., 2017], where the authors find that this approach shows promising initial results for important air pollutants such as Ozone, predicting concentrations that deviate less than 10% from the values computed by the traditional model.

The specific problem of the prediction of Ozone and PM10 is addressed in [Corani, 2005], using to this end several statistical approaches. In particular, they use feed-forward neural networks (FFNNs), which have been extensively used for the prediction of air quality, and they are compared to two different machine learning approaches: lazy learning and pruned neural networks.

Application of a novel classifier ($\sigma - FLNMAP$) [Athanasiadis et al., 2003] is introduced for estimating the ozone concentration level in the atmosphere. The $\sigma - FLNMAP$ classifier gets better results (with only a few rules) compared to

FFNNN and C4.5 algorithm.

3 Machine learning methods for air-pollution monitoring

In this work several automatic learning techniques are evaluated to predict the level of ozone in smart cities. For this purpose, several techniques have been selected taking as criteria the interpretability of the models they obtain. The techniques used have been extensively tested and are capable of providing good performance. The techniques used are: Bagging, Random Committee, Random Forest, a decision tree and an instance-based technique. A summary of the basic fundamentals of the techniques is explained below:

- **Bagging**

  Bagging is a multi classifier that learns several classifiers and output is a composition of the result that each of them [Breiman, 1996]. The base classifier can be based on different techniques, for example, trees, rules, instances, etc. In this case, the classifier used in this paper is REPTree. REPTree builds a decision/regression tree using information gain and prunes it using reduced error pruning (with adjustment). This tree classifies the values of the numerical attributes only once. This tree has a behavior similar to the decision tree C4.5 [Quinlan, 2014].

- **Random Committe**

  Random Committe is an ensemble of random base classifiers. Each base classifier is constructed using a different random number of seeds (but using the same data). The final prediction is a direct average of the predictions generated by the individual base classifiers. The base classifier used in this paper is the Random Tree. Random Tree [Kalmegh, 2015] is a decision tree that considers K randomly selected attributes at each node. It does not prune. It also has an option to estimate the target mean for regression based on a hold-out set (backfitting).

- **Random Forest**

  Random forest [Breiman, 2001] is an ensemble composed of decision trees where each tree depends on the values of a random vector sampled independently and with the same distribution for all the trees in the forest. The error for the forest tends to stabilize from a certain elevated number of trees. This is because the error depends on the quality of each individual tree and the correlation between those trees.

- **Decision Tree**

The decision tree used in this case is the M5P [Wang and Witten, 1997]. This tree is an improved version of [Quinlan et al., 1992]. The basic idea for building this tree model is quite straightforward. In a first stage an induction decision tree is constructed where instead of maximizing the gain of information within each node a division criteria is used to minimize the intra-subset variation. In the second stage, a pruning is carried out inside the nodes, replacing the node if necessary with a regression plane.

– **k Nearest Neighbors (kNN)**

The technique (kNN) [Aha et al., 1991] is a type of instance-based learning, or lazy learning, in which the function is only approached locally and all calculations are postponed until classification. This technique is used in both classification and regression, it has no training phase and it calculates the nearest neighbours to a given instance using distance or similarity functions. For regression the output consist of the average of the values of its k nearest neighbors.

– **Hierarchical cluster**

Hierarchical clustering technique [Langfelder et al., 2007] defines the cluster distance between two clusters to be the maximum distance between their individual components. At every stage of the clustering process, the two nearest clusters are merged into a new cluster. In this case, the technique is based on the Discrete Wavelet Transform (DWT) [Wickerhauser, 1996], which is a useful feature extraction technique often used to measure dissimilarity between Time Series. DWT performs a scale-wise decomposing of the time series in such a way that most of the energy of the time series can be represented by only a few coefficients. The basic idea is to replace the original series by their wavelet approximation coefficients in an appropriate scale, and then to measure the dissimilarity between both.

4 Evaluation

The techniques explained in section 3 are now evaluated by means of a series of datasets obtained from four air-quality measurement stations in four different cities at the Region of Murcia, Spain. The prediction is performed on O_3, being one of the most polluting agents in the air. The assessment is divided into two parts. Firstly, the O_3 prediction results are calculated for the different techniques explained in section 3. The Weka tool has been used for this evaluation [Hall et al., 2009]. Secondly, a hierarchical clustering is carried out to evaluate how many models would be necessary to predict ozone in the Region of Murcia. To obtain the endogram through the hierarchical cluster the R language is used,

specifically the TSclust package [Manso et al., 2017]. For the two experimental approaches the datasets are described next.

4.1 Dataset

The data used in this experiment are real data obtained from four atmospheric stations [2] in the Region of Murcia. These stations are located in the cities of Alcantarilla, Aljorra, Lorca and Caravaca. The data used covers the average per hour of different climatic parameters and chemical elements affecting air quality each day for the years 2013 and 2014 for each station. Not all stations have the same measuring instruments and therefore not all stations analyze all elements affecting air quality. Moreover, as measuring instruments do not always work properly, when the dataset contained missing data in any of the input variables used, the instance has been discarded, so each dataset contains a different number of instances.

Table 1 shows the description of the datasets used to predict ozone in the aforementioned four cities of the Region of Murcia. In this table, the "N.Inst." column indicates the number of instances that each dataset contains whereas the "Inputs" column indicates the variables that are taken into account in the datasets to predict ozone. For the "Alcantarilla" station, new air quality sensors were added in 2014, hence for 2013 there are 10 input variables whereas for 2014 there are 13 input variables. The datasets called "Alcantarilla2" consist in the fact that for 2014 those new variables have been eliminated and the same input variables have been left as in 2013 in order to establish if those new variables incorporated in 2014 are significant or not.

The inputs used to predict Ozone, shown in an abbreviated form in Table 1, are measured in microgram per cubic meter (μ/m^3) and consist of Nitrogen Monoxide (NO), Nitrogen Dioxide (NO_2), Sulfur Dioxide(SO_2), Total Nitrogen Oxides (NOX), Particulate matter in suspension $< 10\mu\ g(PM_{10})$, Benzeno (C_6H_6), Toluene (C_7H_8) and Xileno (XIL). The rest of the elements are Temperature (TMP) measured in degrees Celsius (^oC), Relative Humidity (HR) measured in %, wind direction (DD) in grades, Wind speed (VV) in meters per second (m/s), Atmospheric pressure (PRB) in bar and Solar Radiation (RS) in watts per square meter(w/m^2). Finally, the variable to be predicted O_3 is also measured in μ/m^3.

4.2 Parameters

The machine learning techniques have been validated using different parameters where the best results are shown in Table 2. For this table it should be clarified

[2] `https://sinqlair.carm.es/calidadaire/Default.aspx`

Table 1: Description of the datasets for ozone prediction.

	2013		2014	
Datasets	N.Inst.	Inputs	N.Inst.	Inputs
Alcantarilla	8496	NO, NO_2, SO_2, TMP, HR, NOX, DD, VV, PM_{10}	8496	NO, NO_2, SO_2, TMP, HR, NOX, DD, VV, C_6H_6, C_7H_8, XIL, PM_{10}
Alcantarilla2	-	-	8496	NO, NO_2, SO_2, TMP, HR, NOX, DD, VV, PM_{10}
Aljorra	6093	NO, NO_2, SO_2, TMP, HR, NOX, DD, PRB, RS, VV, PM_{10}	8348	NO, NO_2, SO_2, TMP, HR, NOX, DD, PRB, RS, VV, PM_{10}
Lorca	7982	NO, NO_2, SO_2, TMP, HR, NOX, DD, RS, VV, PM_{10}	7865	NO, NO_2, SO_2, TMP, HR, NOX, DD, RS, VV, PM_{10}
Caravaca	8341	NO, NO_2, TMP, HR, NOX, DD, PRB, RS, VV, PM_{10}	8492	NO, NO_2, TMP, HR, NOX, DD, PRB, RS, VV, PM_{10}

that in the Random Committee technique the value of K for Random Tree consists of $log_2 A + 1$, where A is the number of input values. Likewise for the decision tree, the parameter "Min Number Samples" refers to a node being a leaf when it contains that minimum number of examples.

4.3 Ozone prediction

In this section, the machine learning techniques proposed for predicting O_3 are assessed and analyzed. This assessment is performed by a 3-fold cross validation; i.e. the database is divided into three groups where one group is selected for the evaluation and the other two for training. The three groups are eventually used for evaluation. Moreover, the quality and reliability of our models are measured by the Mean Absolute Error (MAE) and Root Mean Squared Error

Table 2: Relevant parameters for the selected machine learning techniques.

Techniques	Parameters	
Bagging	Base Classifier:	REPTree
	Iterations:	20
	% Bag Size:	100%
Random Committe	Base Classifier:	Random Tree
	K Value for Random Tree:	$log_2 A + 1$
	Iterations:	20
Random Forest	Base Classifier:	C4.5
	Number Trees:	150
	Minimum Features:	1
Decision Tree	Min Number Samples:	4
KNN	K value:	2
Hierarchical cluster	Distance:	DWT Euclidean

(RMSE). Finally, the robustness and suitability are evaluated through the determination coefficient R^2. The measurements MAE, RMSE and R^2 are defined in the equations 1, 2 and 3, respectively.

$$MAE = \frac{\sum_{i=1}^{n} |v_p - v_r|}{n} \tag{1}$$

where n is the number of instances, v_p is the value predicted by the model and v_r is the actual ozone value.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_p - v_r)^2} \tag{2}$$

$$R^2 = \frac{\sigma^2_{v_p,v_r}}{\sigma^2_{v_p}\sigma^2_{v_r}} \tag{3}$$

where σ_{v_p,v_r} is the covariance of v_p, v_r and σ_{v_p} and σ_{v_r} is the standard deviation of the variable v_p and v_r respectively.

Table 3 shows the error metrics of the proposed machine learning techniques in terms of RMSE and MAE in the cities involved in the study. It is noteworthy to highlight that the RMSE and MAE of the Alcantarilla2 dataset ("Alcant.2") is relatively similar to the Alcantarilla ("Alcant.") dataset for the year 2014. This means the new sensors (i.e. variables) introduced at Alcantarilla station during 2014 do not have a great influence on achieving a better ozone prediction.

Furthermore, the random forest algorithm obtains, in general, a lower RMSE and MAE than the other machine learning strategies. The Random Committee

Table 3: RMSE and MAE obtained by machine learning techniques in the prediction of O_3 for the years 2013 and 2014.

Techniques		Bagging		Random Committee		Random Forest		M5P Tree		KNN	
Data	Years	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Alc	2013	8.31	10.99	8.01	10.83	7.65	10.20	8.48	11.19	8.87	12.33
	2014	8.03	10.65	7.66	10.26	7.33	9.77	8.66	11.47	9.43	13.07
Alc2	2014	8.22	10.97	8.00	10.78	7.68	10.24	8.58	11.41	8.87	12.33
Alj	2013	9.17	12.14	8.62	11.68	8.34	11.11	8.57	11.36	9.32	12.98
	2014	7.63	9.79	7.35	9.59	7.10	9.16	8.06	10.29	7.70	10.37
Lorca	2013	8.95	11.72	8.61	11.50	8.25	10.89	9.46	12.31	10.04	13.64
	2014	8.53	11.09	8.13	10.77	7.87	10.30	9.25	11.92	9.05	12.36
Car	2013	8.50	11.07	8.13	10.80	7.90	10.35	9.19	11.94	8.93	12.33
	2014	8.54	11.12	7.96	10.68	7.77	10.29	9.30	12.06	9.71	13.27

is placed in second position according with the error ratios. The overall comparison of the targeted machine learning techniques is shown in Figure 1, where Figure 1(a) refers to 2013 and Figure 1(b) to 2014. These two figures shows several relevant points. In 2013, the city with the least error is Caravaca followed by Aljorra. In 2014, the city with the best prediction was Aljorra followed by Alcantarilla, the random forest predicted. The worst performance machine learning technique studied (i.e. having the highest RMSE and MAE) was KNN. This fact happened using all datasets for the two years studied, so this technique obtains unsatisfactory results to predict O_3 and we would not advise the use of this technique to predict O_3 whenever this sensor may fail in a smart city environment.

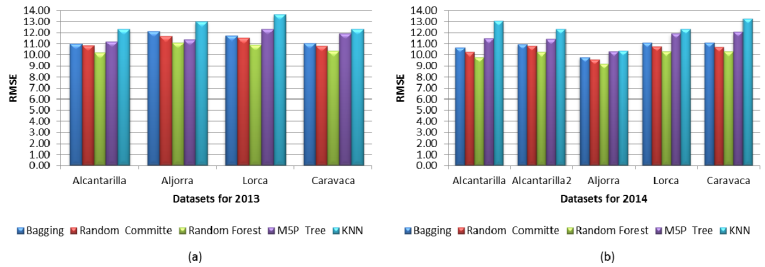


Figure 1: Comparison of the RMSE for the years 2013 (a) and 2014 (b) of the different machine learning techniques.

Table 4 shows the determination coefficient R^2 to assess the quality of the results for our models. Above $0.75 R_2$ could be considered as satisfactory (let us remind the reader that $R_2 = 1$ means a perfect fit). Therefore, Table 4 shows that most of the models generally obtain a good fit and therefore they obtain satisfactory and reliable results. However, it should be underlined that for 2013 and for the city of Caravaca the adjustment of the different models is worse.

Table 4: Adjustment of the models obtained by machine learning techniques for the years 2013 and 2014.

Techniques		Bagging	Random Committee	Random Forest	M5P Tree	KNN
Datasets	Years	R^2	R^2	R^2	R^2	R^2
Alcantarilla	2013	0.895	0.898	0.910	0.891	0.870
	2014	0.908	0.911	0.920	0.900	0.870
Alcantarilla2	2014	0.913	0.919	0.927	0.899	0.871
Aljorra	2013	0.792	0.807	0.826	0.897	0.766
	2014	0.801	0.808	0.826	0.780	0.779
Lorca	2013	0.832	0.839	0.856	0.815	0.780
	2014	0.849	0.858	0.870	0.835	0.817
Caravaca	2013	0.679	0.695	0.722	0.626	0.616
	2014	0.742	0.761	0.780	0.752	0.645

Figure 2 shows that the prediction for the Caravaca city is always lower on average than the other cities. Moreover, although the best model studied is random forest, for the year 2013 and for the city of Aljorra, the technique with the best suitability is the M5P decision tree. Regarding the prediction error, the technique with the worst suitability is KNN.

Although Figures 1 and 2 state the random forest technique obtains the best models to predict O_3, a non-parametric statistical test has been performed to validate this statement; the Wilcoxon Signed Ratings Test is used [Kruskal, 1957]. This test compares two paired groups and thus it can be used to test when the null hypothesis indicates that two populations have the same continuous distribution. The Wilcoxon test confirms that the Random Forest technique provides better results and a better fit than the other machine learning techniques with 99% confidence level. The second and third techniques with a better fit for the O_3 prediction are Random Committee and Bagging respectively.

Finally, it is noteworthy to highlight that the techniques used in this analysis enables the result interpretation in a simple manner. Therefore, we are going to use decision tree techniques to analyze which variables are the most important

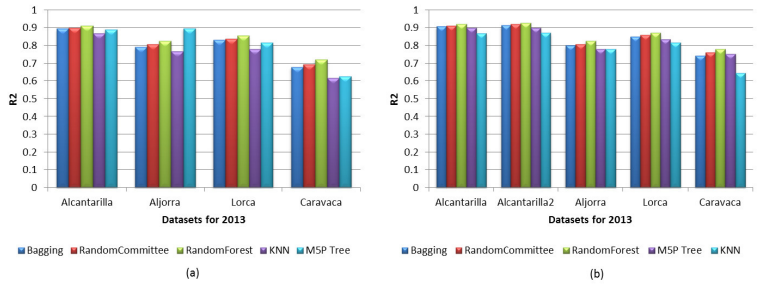


Figure 2: Suitability of the models according to R^2 for the cities of the Region of Murcia for the years 2013 (a) and 2014 (b).

in predicting O_3. Thus, and taking into account the variables that are measured in all cities, the most influential variables to predict O_3 are: NO_X, TMP, DD, VV, HR, SO_2, NO and PM_{10}.

4.4 Clustering cities according to their O_3

Once the best model for predicting ozone has been selected, a hierarchical clustering technique is applied to identify how many models would be needed to predict the ozone in the Region of Murcia. This analysis has been carried out for the year 2014. Two different measures have been used for this purpose: on the one hand, DWT (Dissimilarity for Time Series Based on Wavelet Feature Extraction) and on the other hand, Euclidean distance measurement. The reason for using DWT is to analyze the results obtained by treating the air quality data collected as time series.

Finally, Figure 3 shows the endogram obtained with the 2014-data for the four cities studied in this paper. The endogram shows similar results whenever the euclidean distance or the DWT distance are used. These results show that it is not necessary to deal with the data as a time series. The endogram also unifies the cities of Lorca, Alcantarilla and Aljorra, putting off the city of Caravaca. Caravaca has different air quality ratio than the other 3 cities and therefore it needs a specific model in order to provide an accurate ozone prediction. Thus, by placing the cities on the map of the Region of Murcia, the behaviour of air quality in Murcia is referenced as shown in Figure 4.

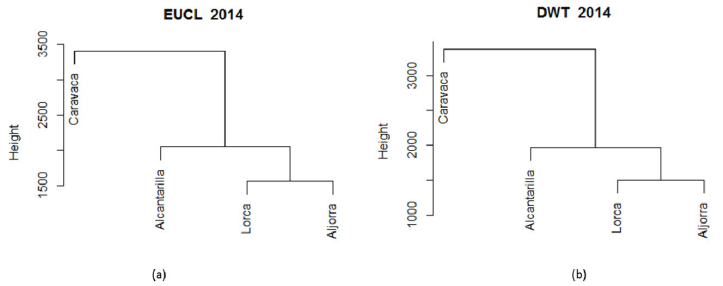


Figure 3: Endogram using the two distances for hierarchical clustering: (a) Using the Euclidean distance; (b) Using DWT distance.

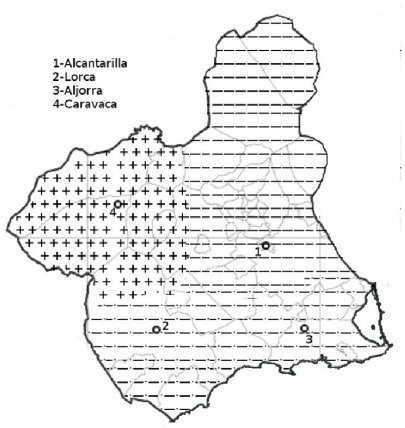


Figure 4: Map of the Region of Murcia by air quality zones classified by the clustering technique.

5 Conclusions and Future Work

We are witnessing the steady transaction to smart cities where machine learning is called to play a decisive role. Among the main issues that are currently worth of attention in developed countries is the pollution concentration in city areas causing chronic diseases. Indeed, the and analysis of pollutants such as SO_2, NO_x, O_3, NH_3, CO and PM_10 through machine learning techniques may help to predict pollution peaks as well as to establish thresholds and action plans

for local authorities, drivers, industries, etc. In this paper, we analyze different machine learning techniques to predict the O_3 levels, one of the more harmful air-pollution parameter. The machine learning techniques studied in this work have been Random Forest, Decision Tree, Random Committee, Bagging and KNN. The technique that obtains the best fit in general is Random Forest, being this assertion validated by statistical tests. The results indicate an R^2 setting between 80% and 90% overall and an O_3 prediction error less than 11 μ/m^3. It is also important to note that among the parameters that most influence the ozone prediction we have found climatic variables related to temperature, humidity and wind. In addition, hierarchical clustering indicates that the air-pollution monitoring areas in the Region of Murcia can be divided into two zones only so as to create two general O_3 prediction models for the entire Region. These two areas would be the cities of Lorca, Alcantarilla and Aljorra on one side and Caravaca on the other.

As future work, new parameters such as PM_{10} and SO_2, that seriously affect air quality as well, must be analyzed and studied to create models that help to predict them. Another extension will be the automatic generation of recommendations to local authorities, drivers and other related actors in the influence of air-quality factors according to the alerts raised by our system.

Acknowledgements

This work is supported by the Spanish MINECO under grant TIN2016-78799-P (AEI/FEDER, UE).

References

[Afshar-Mohajer et al., 2018] Afshar-Mohajer, N., Zuidema, C., Sousan, S., Hallett, L., Tatum, M., Rule, A. M., Thomas, G., Peters, T., and Koehler, K. (2018). Evaluation of low-cost electro-chemical sensors for environmental monitoring of ozone, nitrogen dioxide and carbon monoxide. Journal of occupational and environmental hygiene.

[Aha et al., 1991] Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. Machine learning, 6(1):37–66.

[Ahmad et al., 2017] Ahmad, Z., Rahim, N. A., Bahadori, A., and Zhang, J. (2017). Air polluiton index prediction using multiple neural networks. IIUM Engineering Journal, 18(1):1–12.

[Al-Ali et al., 2010] Al-Ali, A. R., Zualkernan, I., and Aloul, F. (2010). A mobile gprs-sensors array for air pollution monitoring. IEEE Sensors Journal, 10(10):1666–1671.

[Athanasiadis et al., 2003] Athanasiadis, I. N., Kaburlasos, V. G., Mitkas, P. A., and Petridis, V. (2003). Applying machine learning techniques on air quality data for real-time decision support. In First international NAISO symposium on information technologies in environmental engineering (ITEE'2003), Gdansk, Poland.

[Azid et al., 2013] Azid, A., Juahir, H., Latif, M. T., Zain, S. M., and Osman, M. R. (2013). Feed-forward artificial neural network model for air pollutant index prediction in the southern region of peninsular malaysia. Journal of Environmental Protection, 4(12):1.

[Azid et al., 2017] Azid, A., Rani, N., Samsudin, M., Khalit, S., Gasim, M., Kamarudin, M., Yunus, K., Saudi, A., and Yusof, K. (2017). Air quality modelling using chemometric techniques. Journal of Fundamental and Applied Sciences, 9(2S):443–466.

[Bello-Orgaz et al., 2016] Bello-Orgaz, G., Jung, J. J., and Camacho, D. (2016). Social big data: Recent achievements and new challenges. Information Fusion, 28:45-59.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2):123–140.

[Breiman, 2001] Breiman, L. (2001). Random forests. Machine learning, 45(1):5–32.

[Brook et al., 2002] Brook, R. D., Brook, J. R., Urch, B., Vincent, R., Rajagopalan, S., and Silverman, F. (2002). Inhalation of fine particulate air pollution and ozone causes acute arterial vasoconstriction in healthy adults. Circulation, 105(13):1534–1536.

[Bui and Jung, 2017] Bui, K.-H. N. and Jung, J. J. (2017). Internet of agents framework for connected vehicles: A case study on distributed traffic control system. Journal of Parallel and Distributed Computing 116:89-95.

[Corani, 2005] Corani, G. (2005). Air quality prediction in milan: feed-forward neural networks, pruned neural networks and lazy learning. Ecological Modelling, 185(2-4):513–529.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. ACM SIGKDD explorations newsletter, 11(1):10–18.

[Han et al., 2011] Han, J., Pei, J., and Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[Hand, 2007] Hand, D. J. (2007). Principles of data mining. Drug safety, 30(7):621–622.

[Hong and Jung, 2018] Hong, M. and Jung, J. J. (2018). Multi-Sided recommendation based on social tensor factorization. Information Sciences, 447:140-156.

[Jin et al., 2014] Jin, J., Gubbi, J., Marusic, S., and Palaniswami, M. (2014). An information framework for creating a smart city through internet of things. IEEE Internet of Things Journal, 1(2):112–121.

[Jung, 2017a] Jung, J. E. (2017b). Discovering Social Bursts by Using Link Analytics on Large-Scale Social Networks. Mobile Networks & Applications, 22(4):625–633.

[Jung, 2017b] Jung, J. J. (2017a). Computational collective intelligence with big data: Challenges and opportunities. Future Generation Computer Systems, 66:87–88.

[Kalmegh, 2015] Kalmegh, S. (2015). Analysis of weka data mining algorithm reptree, simple cart and randomtree for classification of indian news. Int. J. Innov. Sci. Eng. Technol, 2(2):438–446.

[Kang et al., 2018] Kang, G. K., Gao, J. Z., Chiao, S., Lu, S., and Xie, G. (2018). Air quality prediction: Big data and machine learning approaches. International Journal of Environmental Science and Development, 9(1):8–16.

[Keller et al., 2017] Keller, C. A., Evans, M. J., Kutz, J. N., and Pawson, S. (2017). Machine learning and air quality modeling. In Big Data (Big Data), 2017 IEEE International Conference on, pages 4570–4576. IEEE.

[Kruskal, 1957] Kruskal, W. H. (1957). Historical notes on the wilcoxon unpaired two-sample testhistorical notes on the wilcoxon unpaired two-sample test. Journal of the American Statistical Association, 52(279):356–360.

[Kurt et al., 2008] Kurt, A., Gulbagci, B., Karaca, F., and Alagha, O. (2008). An online air pollution forecasting system using neural networks. Environment International, 34(5):592–598.

[Langfelder et al., 2007] Langfelder, P., Zhang, B., and Horvath, S. (2007). Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r. Bioinformatics, 24(5):719–720.

[Li et al., 2017] Li, V. O., Lam, J. C., Chen, Y., and Gu, J. (2017). Deep learning model to estimate air pollution using m-bp to fill in missing proxy urban data. In GLOBECOM 2017-2017 IEEE Global Communications Conference, pages 1–6. IEEE.

[Manso et al., 2017] Manso, P. M., Vilar, J. A., and Montero, M. P. (2017). Package 'TScluster'. 62(1):1–43.
[Quinlan, 2014] Quinlan, J. R. (2014). C4. 5: programs for machine learning. Elsevier.
[Quinlan et al., 1992] Quinlan, J. R. et al. (1992). Learning with continuous classes. In 5th Australian joint conference on artificial intelligence, volume 92, pages 343–348. Singapore.
[Rathore et al., 2016] Rathore, M. M., Ahmad, A., Paul, A., and Rho, S. (2016). Urban planning and building smart cities based on the internet of things using big data analytics. Computer Networks, 101:63–80.
[Ravi et al., 2017] Ravi, N., Manoranjani, R., Seshadri, K., et al. (2017). Leveraging social networks for smart cities: A case-study in mitigation of air pollution. In International Conference on Intelligent Information Technologies, pages 179–193. Springer.
[Reid et al., 2017] Reid, A. R., Pérez, C. R. C., and Rodríguez, D. M. (2017). Inference of vehicular traffic in smart cities using machine learning with the internet of things. International Journal on Interactive Design and Manufacturing (IJIDeM), pages 1–14.
[Ruidavets et al., 2005] Ruidavets, J.-B., Cournot, M., Cassadou, S., Giroux, M., Meybeck, M., and Ferrières, J. (2005). Ozone air pollution is associated with acute myocardial infarction. Circulation, 111(5):563–569.
[Schaffers et al., 2011] Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., and Oliveira, A. (2011). Smart cities and the future internet: Towards cooperation frameworks for open innovation. In The future internet assembly, pages 431–446. Springer.
[Shaghaghi et al., 2017] Shaghaghi, E., Jabbarpour, M. R., Noor, R. M., Yeo, H., and Jung, J. J. (2017). Adaptive green traffic signal controlling using vehicular communication. Frontiers of Information Technology & Electronic Engineering, 18(3):373–393.
[Shaikh et al., 2017] Shaikh, F. K., Zeadally, S., and Exposito, E. (2017). Enabling technologies for green internet of things. IEEE Systems Journal, 11(2):983–994.
[Wang and Witten, 1997] Wang, Y. and Witten, I. H. (1997). Inducing model trees for continuous classes. In Proceedings of the Ninth European Conference on Machine Learning, pages 128–137.
[WHO, 2018] (2018). World health Organization (WHO) air pollution programme. `http://www.who.int/airpollution/en/`. Accessed: 2018-03-26.
[Wickerhauser, 1996] Wickerhauser, M. V. (1996). Adapted wavelet analysis: from theory to software. AK Peters/CRC Press.
[Witten et al., 2016] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.
[Zhang and Ding, 2017] Zhang, J. and Ding, W. (2017). Prediction of air pollutants concentration based on an extreme learning machine: the case of hong kong. International journal of environmental research and public health, 14(2):114.

2.3 An unsupervised technique to discretize numerical values by fuzzy partitions

Título	*An unsupervised technique to discretize numerical values by fuzzy partitions*
Autores	Andrés Bueno-Crespo, Raquel Martínez-España, Isabel Timón y Jesús Soto
Revista	Journal of Ambient Intelligence and Smart Environments
Año	2018
Estado	Publicado

Contribución de la doctoranda

Isabel María Timón Pérez declara, con el visto bueno del resto de autores, ser el principal autor y la principal contribuidora del artículo *An unsupervised technique to discretize numerical values by fuzzy partitions*.

Journal of Ambient Intelligence and Smart Environments 10 (2018) 289–300
DOI 10.3233/AIS-180488
IOS Press

An unsupervised technique to discretize numerical values by fuzzy partitions

Andrés Bueno-Crespo, Raquel Martínez-España *, Isabel Timón and Jesús Soto
Dept. of Computer Engineering, Catholic University of Murcia, Murcia, Spain
E-mails: abueno@ucam.edu, rmartinez@ucam.edu, imtimon@ucam.edu, jsoto@ucam.edu

Abstract. The numerical value discretization is a process that is performed in the data preprocessing phase of intelligent data analysis. Preprocessing phase is very relevant because the quality of the models obtained in data mining step depends on this phase. Value discretization is an important task in data preprocessing because not all data mining techniques can handle continuous values. In this paper an unsupervised technique to discretize continuous data values using fuzzy partitions is proposed. Specifically a clustering technique that gets fuzzy partitions is presented. In addition, to evaluate the behavior of the proposed technique a series of experiments have been proposed using a Extreme Learning Machine classifier and a committee of Extreme Learning Machine. Beside comparing with the K-means discretization technique. These experiments have been validated statistically obtaining the best results the approach proposed.

Keywords: Discretization, fuzzy clustering, membership function, extreme learning machine, data mining

1. Introduction

The discretization is a relevant task of data preprocessing within Intelligent Data Analysis. This task has as objective to transform the attributes with continuous values into discrete values, either through intervals or through fuzzy partitions. This discretization process is essential because there are data mining techniques that can not manage continuous values, these techniques can only handle categorical/discretized values some of these are: association rules, induction rules, Bayesian networks, some techniques of decision trees, random forest, etc. There are even data mining techniques, that being able to work with continuous values, obtain more satisfactory models working with discrete values. Between the advantages of dicretization it can be highlighted the capability to reduce the number of store data, to improve the interpretability both initial data and results obtained and the effectiveness and efficiency that will be established later in the application of data mining techniques. The effectiveness and efficiency caused by discretization favors to a certain extent the reduction of the computational cost [4,8].

In literature several taxonomies to classify the different discretization techniques can be found [22,35]. Without being exhaustive some of these classifications are analyzed taking into account the characteristics of them:

- Static versus Dynamic: A static discretizer performs the learning task independent from the learning algorithm. A dynamic technique performs the discretization during the learning phase of the data mining technique.
- Global or Local: Global discretization is performed when all the information is used to discretize. Local discretization technique only uses part of the information to perform the discretization process.
- Supervised or Unsupervised: Unsupervised techniques do not consider the label class to perform the discretization. Supervised techniques need to evaluate the label class to perform the partition.
- Univariate or Multivariate: Multivariate techniques create the initial cut point with all at-

*Corresponding author. E-mail: rmartinez@ucam.edu.

tributes simultaneously. The univariate techniques only consider an attribute each time.
- Direct or Incremental: Direct techniques divide the range into k intervals simultaneously, requiring a criterion to determine the k value. By contrast, incremental discretizers begin with a simple discretization, and perform improvement process until a stop criterion is fulfill.
- Splitting or Merging: Splitting discretizers create new intervals dividing the domain of the continuous values. Merging techniques create a number of intervals which are removing and merging to obtain the final partitions. Some techniques can be considered hybrid because they can alternate splits with merges in running time [8].
- Evaluation measure: Discretizers can also classify considering the measure used to assess the partitions. Thus, some measure used are: information measure such as entropy and Gini Index, rough set measure, statistical measures, error in classification or binning that is the absence of measure.
- Fuzzy or crisp: This type of discretization depends on the logic used (classical or fuzzy logic). The difference between a fuzzy discretization or a crisp discretization is the result of the intervals. When the discretization is fuzzy, the intervals overlap and a value may belong to more than one interval. On the contrary in a crisp discretization, the intervals are disjoint and a value can only belong to an interval.
- Parametric or Nonparametric: A parametric discretizers need that users fix a maximum number of intervals. However, a nonparametric discretizer determines automatically the number of intervals to discretize a continuous attribute.
- Top-Down or Bottom Up: This characteristic is usual in incremental discretizers, [8]. Top-down techniques start with an empty discretization and add new cut point during the discretization process. On the contrary, bottom up techniques start with all possible intervals, and during the discretization process, they are removing cut point, that means, merging intervals.
- Stopping criteria: This characteristic is referred to the stop conditions of the discretization process, some of this conditions are confidence thresholds [17], inconsistency ratios [5] or the Minimum Description Length measure [6].
- Disjoint or Non-disjoint: Disjoint techniques divide the domain in crisp partitions, that means, in intervals. Non-disjoin techniques allow the overlapping between partitions. Usually, fuzzy discretization techniques are non-disjoin.
- Nominal or Ordinal: Nominal discretizers divide the domain in nominal qualitative values. On the contrary, ordinal discretizers transform the domain in ordinal qualitative values. This last type of discretization is not very common.

The assessment of the partitions can be performed using different method such as number of intervals, inconsistency, predictive classification rate or time requirements. The most commonly used method is usually the classification predictive classification rate, therefore, we use them for our experiments. In this paper, a modification of the Fuzzy C-Means (FCM) technique [34] is used to develop a fuzzy and unsupervised discretization algorithm. This proposal is based on Cauchy distribution to discretize numerical values. To assess the approach datasets from UCI repository, [20] are used. The results obtained after comparing with the classical discretization technique based on K-means (KM) algorithm [12] are satisfactory. The paper is organized as follow. In Section 2 a brief review of some work on discretization is carried out. Then, in Section 3 the discretization technique proposed is explained. Next, Section 4, the classifier used to assess the quality of discretization is exposed. Finally, in Section 5 and 6 the experiments, conclusion and future works are presented, respectively.

2. Background

Different discretization techniques can be found in literature due to the fact that there are not an universal technique to obtain categorical values that works properly with any data mining method. Thus, in [25], several discretization techniques are evaluated to use the most suitable for clinic data. The discretization performed by these authors is crisp and the techniques used are both supervised and unsupervised. In their conclusions, the authors affirm that supervised discretization is more particular and unsupervised is more general and it can be applied in more domains.

In [21], a fuzzy and unsupervised discretization is presented. In this work the discretization process is performed by means of clustering technique, selecting initial clustering centers by large density area and using density function as samples' weights to reduce effectively noise interference. Then, the compatibility of decision table in rough set theory is used as criteria to

adjust dynamically the parameters of the algorithm to achieve optimal discretization effect. Another clustering technique is exposed in [1]. In this case, the authors develop a clustering technique as a discretization technique to recognize solar images, extracting texture features of these images. In [23] a non-parametric discretization technique for continuous values with missing data is presented. This technique uses the statistical technique z-score with an index measure to impute the missing data values for numeric or continuous attributes. In [39] an iterative and novel scheme to dicretize is presented. This method dynamically discretizes the continuous random variables in intervals at each iteration. The interactive method is focused on estimating the likelihood of low-probability failure events instance of focusing on getting the overall shape of the distribution correct. The method is assessed using a dynamic Bayesian Networks.

The authors of [15] present a supervised and multivariate discretization algorithm called SMDNS based on rough sets, which is derived from the traditional algorithm naive scaler. This method simultaneously considers all attributes, that is, takes into account the interdependence among various attributes. The method iteratively merges adjacent intervals of continuous values according to a given criterion. A hybrid discretization method for naïve Bayesian classifiers is presented in [36]. The propose discretizer uses a nonparametric measure to assess the dependence level between a continuous attribute and the class.

In [37] other algorithm to perform fuzzy partition is developed. Specifically authors expose a two-step method to create membership function. In the first step the method divides domain of continuous attributes in intervals and then in the second step the different membership functions are created. For creating the membership functions they use four different measures, particularly partition width, standard deviation of examples, coverage rate of neighbor partitions and Entropy Based Fuzzy Partitioning. Another discretization technique that uses different measures to perform the discretization process is presented in [16]. In this case, the authors propose a weighted hybrid discretization technique based on entropy and contingency coefficient.

In [3], a method to discretize continuous attribute is proposed. This method performs the discretization during the learning phase of a decision tree. Specifically to perform the discretization the authors propose an extension of the method Ant Colony Decision Tree. In [19] a self-adaptive discretization method is proposed to discretize continuous values for association rule. The self-adaptive method proposed creates partitions which can give a high confidence to the calculated association rule while guaranteeing the relatively high support.

3. Discretization technique

This section describes the proposed discretization technique based on clustering analysis. This technique can be categorized according to the above categorization as a global, unsupervised and fuzzy discretization technique.

Clustering analysis consists of dividing a data into several clusters where data in same cluster have high similarity while data in different clusters are distinct each other. KM algorithm is one of the most popular clustering methods, [12]. This algorithm starts by guessing k cluster centers and then iterates the following steps until convergence is achieved. Each cluster is built by assigning each instance to the closest cluster center and each cluster center is replaced by the mean of the elements belonging to that cluster. Traditional clustering methods, such as KM, each instance is assigned to one cluster in an unequivocal way. As opposed to this, in fuzzy clustering an instance x may belong to different clusters at the same time, and the degree to which it belongs to the kth cluster is expressed in terms of a membership degree. Consequently, the boundary of single clusters and the transition between different clusters are usually soft rather than abrupt. An example of fuzzy clustering would be FCM. The k-means algorithm has been extended to the fuzzy c-means algorithm by Bezdek in the early eighties [2] and is one of the most widely used fuzzy clustering methods. The cluster analysis FCM aims to find the patterns in data by processing a range of clusters by the calculation of distances of registers to clusters centers through the FCM algorithm.

Consider the dataset X to a set of n instances $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^F$ and P to be a set of k clusters such that $P = \{c_0, c_1, \ldots, c_{k-1}\}$. Each instance x_j is composed of a set of numerical features x_{ij}. A value x_{ij} is the ith feature, with $i = 1, \ldots, p$, associated with the instance j. Each of the data points such as x_i may belong to one or more clusters depends on its degree of membership. Point x_i belongs to cluster c_j as long as its degree of membership to c_j is more than zero.

Clustering a dataset requires finding the center of each cluster and deciding to which cluster each point

belongs to. In FCM to find the center of a cluster, the sum of the distance between points in the cluster and its center is used as criterion. The criteria is represented by an objective function, J, which needs to be minimized with respect to P, a fuzzy c-partition of the dataset, and V, a set of c prototypes for cluster centers $V = \{v_1, v_2, \ldots, v_c\} \subset \mathbb{R}^F$. In general, fuzzy clustering techniques minimize an objective function that determines the centroid or prototypes of each cluster. In [7], FCM algorithm uses the objective function given by:

$$J_m(P, v) = \sum_{x \in X} \sum_{k=1}^{c} u_{xk}^m \cdot d_{xk}^2 \tag{1}$$

The formula incorporates the fuzzy membership function u_{xk} of the instance x to the group k:

$$u_{xk} = \frac{1}{1 + d_{xk}^2} \tag{2}$$

d_{xk} is the Mahalanobis distance [24] from the instance x to the cluster which prototype is v_k and an additional parameter m, as a weighted exponent for the fuzzy membership. Parameter m is the value that determines the degree to which partial members of cluster affect the clustering results. At the beginning of the process V is initialized with some prototype values that get updated during the process.

The function J is minimized by using the following equations for updating the membership degrees and V iteratively until $|v_i - v_{i-1}| < \epsilon$.

The μFCM algorithm is derived from FCM algorithm, and μFCM is explained in [34]. This algorithm is the approach proposed in this paper to be applied as discretization technique instead of classifier technique. Between others differences, the membership function of μFCM (u_{xk}) is deduced from the study of the objective function in [7], and these grades of membership are calculated as follow:

$$u_{xk} = \frac{\mu_{xk}^{3/2} \sqrt{g_k^{-1}}}{\sum_{j=1}^{c} \mu_{xj}^{3/2} \sqrt{g_j^{-1}}} \tag{3}$$

which express the relative deviation of each group k, where the parameters g_k and g_j are the determinants of the fuzzy covariance matrix [10] at k and j, respectively and the function μ is a type of Cauchy distribution, and it is calculated as:

$$\mu_{xk} = \frac{1}{1 + d_{xk}^2} \sqrt{g_k} \tag{4}$$

The fuzzy covariance matrix at cluster k is defined, in [10], as:

$$G_k = \left[\frac{\sum_{x \in X} u_{xk}^2 (x^\alpha - v_k^\alpha)(x^\beta - v_k^\beta)}{\sum_{x \in X} u_{xk}^2} \right], \tag{5}$$

where $x = (x^1, \ldots, x^F)$, $\mathbb{R}^F$ and $v_k = (v_k^1, \ldots, v_k^F)$, $\mathbb{R}^F$ is the prototype of cluster k. And $g_k = \det(G_k)$

The membership of all samples to all clusters defines a partition matrix as $U = [u_{xk}]$.

So that, the objective function (1) is rewritten here as:

$$J_{(k)} = \sum_{x \in X} \frac{1}{(1 + d_{xk}^2)^{3/2}} \sqrt{g_k} d_{xk}^2 \tag{6}$$

The μFCM algorithm computes interactively the clusters centers coordinates from a previous estimate of the partition matrix as:

$$v_k = \frac{\sum_{x \in X} u_{kx} \cdot x_k}{\sum_{x \in X} u_{kx}} \tag{7}$$

Algorithm 1 shows the μFCM process. It is an iterative process where one standard value is included in the computation, ϵ. The algorithm is composed of the following steps:

The μFCM algorithm is used as a discretization technique by applying it to each attribute separately. In this way, the values are classified in three clusters (good-regular-bad). From this performance, a "weight" is assigned to every value of the attribute selected, which corresponds with the μ-values. This process avoids the possibility of correlation, to avoid problems with the classifier.

Let's exemplify the difference between KM and μFCM. Figure 1 shows the two groups that the KM algorithm obtains. These groups are represented by points and diamonds. The algorithm places the point (0, 0) indistinctly a few times in one group and sometimes in the other group, the equidistant with the rest of the points indicates indifference before assigning it to one or the other. For its part, the algorithm μFCM performs a diffuse classification, which allows assigning a probability of belonging to each of the two groups.

Algorithm 1 Algorithm μFCM

1: Initialize the partition matrix U_0.

$$U_0 = [u_{kx}], \quad u_{kx} \in \{1, 0\}, \sum_{x \in X}^{c} u_{kx} > 0$$

2: At k-step: Calculate the centers vectors $v_k \in V_k$ with U_k as in (7):

$$v_k = \frac{\sum_{x \in X} u_{kx} \cdot x_k}{\sum_{x \in X} u_{kx}};$$

$$\mu_{xk} = \frac{1}{1 + d_{xk}^2 \sqrt{g_k^{-1}}}$$

3: Update U_{k+1}, with (3)

$$U_{k+1} = \left[\frac{\mu_{xk}^{3/2} \sqrt{g_k^{-1}}}{\sum_{j=1}^{c} \mu_{xj}^{3/2} \sqrt{g_j^{-1}}} \right]$$

4: If $\|U_{k+1} - U_k\| < \epsilon$ then STOP; otherwise return to step 2.

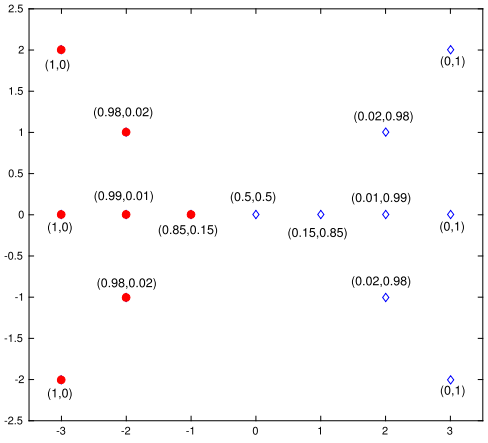


Fig. 1. Illustration for the sample of Butterfly dataset. The point and diamond differentiate the two groups classified with K-means algorithm, while the pairs in brackets indicate the probabilities of diffuse belonging of each point to each of the two groups in which the set has been segmented.

Table 1
μ-function and KM Classification for the sample of Butterfly dataset

x	μ-function		*KM*	
	μ_{xv_1}	μ_{xv_2}	*Cluster*1	*Cluster*2
$(-3, 2)$	0.0983	0.0035	1	0
$(-3, 0)$	0.3893	0.0038	1	0
$(-3, -2)$	0.0983	0.0035	1	0
$(-2, 1)$	0.4373	0.0068	1	0
$(-2, 0)$	0.9551	0.0071	1	0
$(-2, -1)$	0.4373	0.0068	1	0
$(-1, 0)$	0.1806	0.0154	1	0
$(0, 0)$	0.0428	0.0428	0	1
$(1, 0)$	0.0154	0.1806	0	1
$(2, -1)$	0.0068	0.4373	0	1
$(2, 0)$	0.0071	0.9551	0	1
$(2, 1)$	0.0068	0.4373	0	1
$(3, -2)$	0.0035	0.0983	0	1
$(3, 0)$	0.0038	0.3893	0	1
$(3, 2)$	0.0035	0.0983	0	1

This classification is depicted in Fig. 1 by probabilities of belonging shown in brackets. The algorithm μFCM obtains in the classification of the point $(0, 0)$ a probability of 0.5 of belonging to the group of the left and of 0.5 to the one of the right. Both algorithms (KM and μFCM) classify with absolute probabilities, 1 or 0, the points located on the abscissa -3 and 3. However, the points located on the abscissa -2 and 2 differ in the case of the classification by μFCM, because of the proximity to the centers detected by μFCM, v_1 $(-2.21, 0)$ and v_2 $(2.21, 0)$.

Table 1 shows the values obtain for μ-function for each of the centers v_1 $(-2.21, 0)$ and v_2 $(2.21, 0)$ of the Butterfly sample, after applying the algorithm μFCM. These values are in opposition to those obtained by the KM that are shown in the same table. These values represent the discretization perform for each algorithm.

In order to illustrate the process, the example of Iris Data has been considered. The attributes sepal length, sepal width, petal length and petal width are discretized independently, and the interpolation of μ-functions are represented in Figs 2, 4, 5 and 6 respectively.

The μ-function for each cluster created by the μFCM algorithm for the sepal length attribute is represented in Fig. 2. The graphic depicts three clusters represented by the prototypes in the maxims of the function. The μFCM algorithm uses the showed probabilities values of the μ-function in Fig. 2. In this way, for a 6.5 cm sepal length, the membership probability value

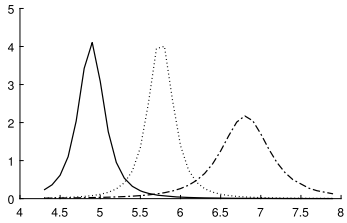


Fig. 2. A linear interpolation of the μ-function for the sepal length attribute in cm.

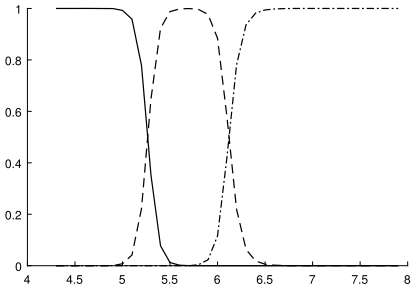


Fig. 3. The membership probability for the sepal length attribute.

is 98.17%, $u_{6.5,3} = 0.9817$, represented by the biggest prototype, with respect to the cluster represented by the central prototype is 1.69%, $u_{6.5,2} = 0.0169$; and 0% with respect to the farthest cluster, $u_{6.5,1} = 0$.

Usually, fuzzy clustering algorithms return the membership probabilities using graphics as Fig. 3. The values obtained through the μ-function of the μFCM algorithm are used and represented in Fig. 2.

In the following Figs 4, 5, 6, the μ-functions for the rest of attributes are represented. Each cluster is represented by a different line types and each prototype is identified by the maximum of the function.

Using the same notation employed to explain Fig. 2, Fig. 4 shows for a 2.4 cm sepal width the μ-function values obtained are $\mu_{2.4,1} = 2.7616$, $\mu_{2.4,2} = 0.0397$ and $\mu_{2.4,3} = 0.0385$.

In Fig. 4, it is shown for example that for a 4 cm petal length the following μ-function values are obtained: $\mu_{4,1} = 0.0017$, $\mu_{4,2} = 1.4510$ and $\mu_{4,3} = 0.0475$.

Finally, Fig. 6 shows for example that for a 1.6 cm petal width the following μ-function values are obtained: $\mu_{1.6,1} = 0.0037$, $\mu_{1.6,2} = 0.884$ and $\mu_{1.6,3} = 0.3480$.

Once the proposed μFCM discretization technique is detailed, the following section presents the classi-

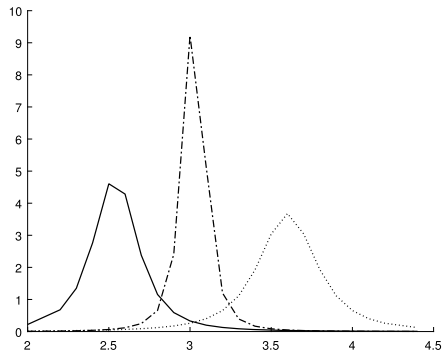


Fig. 4. A linear interpolation of the μ-function for the sepal width attribute in cm.

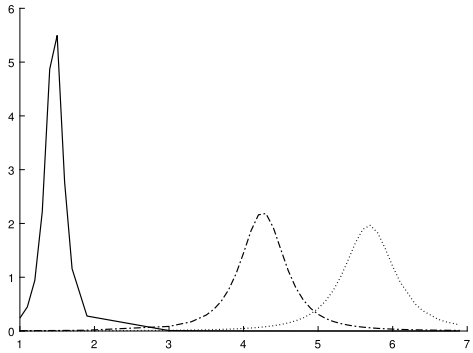


Fig. 5. A linear interpolation of the μ-function for the petal length attribute in cm.

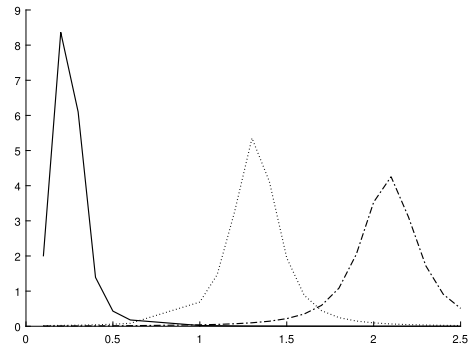


Fig. 6. A linear interpolation of the μ function for the petal width attribute cm.

fier used to evaluate the quality and suitability of this technique. In addition, this evaluator is also used in the comparison with the KM technique. The classifier used as evaluator is the Extreme Learning Machine.

4. Classifier

For Multilayer Perceptron (MLP), Extreme Learning Machine (ELM) provides a fast and efficient training [14]. Formalized by Huang [9,13], it is demonstrated that the ELM is an universal approximation for a wide range of random computational nodes. The MLP input weights are fixed to random values, so, the output weights can be easily obtained using the pseudo-inverse of the hidden neurons outputs matrix $\mathbf{H}$ for a given training set. Given a set of N input vectors, an MLP can approximate N cases with zero error, $\sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{t}_i\| = 0$, being $\mathbf{y}_i$ the output network for the input vector $\mathbf{x}_i$ with target vector $\mathbf{t}_i$. Thus, there exist β_j, $\mathbf{w}_j$ and b_j such that,

$$\mathbf{y}_i = \sum_{j=1}^{M} \beta_j f(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) = \mathbf{t}_i, \quad i = 1, \ldots, N. \tag{8}$$

where $\beta_j = [\beta_{j1}, \beta_{j2}, ..., \beta_{jm}]^T$ is the weight vector connecting the jth hidden node with the output nodes, $\mathbf{w}_j = [w_{j1}, w_{j2}, ..., w_{jn}]^T$ is the weight vector connecting the jth hidden node and the input nodes, and b_j is the bias of the jth hidden node.

The previous N equations can be expressed by:

$$\mathbf{HB} = \mathbf{T}, \tag{9}$$

where

$$\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_M, b_1, \ldots, b_M, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} f(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \ldots & f(\mathbf{w}_M \cdot \mathbf{x}_1 + b_M) \\ \vdots & \ldots & \vdots \\ f(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \ldots & f(\mathbf{w}_M \cdot \mathbf{x}_N + b_M) \end{bmatrix}_{N \times M} \tag{10}$$

$$\mathbf{B} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_M^T \end{bmatrix}_{M \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{11}$$

where $\mathbf{H} \in \mathbf{R}^{N \times M}$ is the hidden layer output matrix of the MLP, $\mathbf{B} \in \mathbf{R}^{M \times m}$ is the output weight matrix, and $\mathbf{T} \in \mathbf{R}^{N \times m}$ is the target matrix of the N training cases. The MLP training is given by the solution of the least square problem of (9). The optimal output weight layer is $\hat{\mathbf{B}} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudo-inverse [32]. ELM for training MLPs can be therefore summarized as shown in Algorithm 2.

Algorithm 2 Extreme Learning Machine (ELM)

Require: Given a training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in R^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \ldots, N\}$, an activation function f and an hidden neuron number M.

1: Assign arbitrary input weights $\mathbf{w}_j$ and biases b_j, $j = 1, \ldots, M$.
2: Compute the hidden layer output matrix $\mathbf{H}$ using (10).
3: Calculate the output weight matrix $\mathbf{B} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{B}$ and $\mathbf{T}$ are both defined in (11).

A problem that presents the ELM is to obtain the number of neurons for the MLP. This requires a pruning method for the ELM. Although there are several pruning methods [26–31], the most commonly used, to avoid the exhaustive search for the optimal value of M, is the ELM Optimally Pruned (OP-ELM) [29]. The OP-ELM sorts the hidden neurons (previously has been initialized to a very high initial number) according to their importance to solve the problem [33]. The pruning of neurons is done by choosing that combination of neurons that provides lower Leave-One-Out error [29]. For more detail, OP-ELM is summarized in Algorithm 3.

Each input characteristic provides a membership function. The union of all of them is trained with the OP-ELM (Fig. 7). This paper also discusses the use of a combination of MLPs (trained width OP-ELM). The use of multiple models may often improve the performance with respect to an individual model [11,38]. Such combination of networks are called committees. A combination of MLPs is presented where each network has been trained with a single input characteristic, because μFCM works individually with each input feature by generating a membership function for each of the partitions created. This allows to create a network committee formed by many networks as the input features have the database, and where each feature has been transformed into its membership function, thus forming the network input. Once trained each network, this provides us with an output that will be the input for a new network, newly trained with OP-ELM, that provides the classification accuracy (Fig. 8). It should be noted that once each network has been trained, the im-

Algorithm 3 Optimally Pruned-ELM (OP-ELM)

Require: Given a training set $\mathcal{D} = \{(\mathbf{x}_j, \mathbf{t}_j)|\mathbf{x}_j \in \mathbb{R}^n, \mathbf{t}_j \in \mathbb{R}^m, j = 1, \ldots, N\}$, a mix of activation functions (sigmoid, Gaussian and linear), and a large number of neurons M,

1: Randomly assign input weights $\{\mathbf{w}_i, b_i\}_{i=1}^{M}$.
2: Calculate the hidden layer output matrix $\mathbf{H}$ using $\mathbf{X}$ and input weights.
3: Ranking the hidden outputs using the MRSR algorithm, i.e., $\mathbf{H}$ is ranked, and set $\mathbf{H}^0$ as an empty matrix.
4: **for** $k = 1$ to N **do**
5: Add the kth node to the model $\rightarrow \mathbf{H}^k = [\mathbf{H}^{k-1}, \mathbf{h}_k]$, being $\mathbf{h}_k$ the kth column of $\mathbf{H}$.
6: Computes LOO error ($\epsilon_k^{\text{PRESS}}$) with $\mathbf{H}^k$.
7: **end for**
8: Select the network size (M^*) according to $\epsilon_{\text{PRESS}}^{M^*} < \epsilon_{\text{PRESS}}^{k}, \forall k \in (1, 2, \ldots, M)$.
9: Calculate the output weights matrix: $\mathbf{B}^* = (\mathbf{H}^*)^{\dagger}\mathbf{T}$

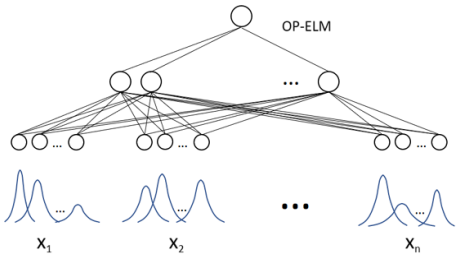


Fig. 7. Single classifier model. Each input is transformed with the membership function, thus forming the input of the network to be trained with the OP-ELM algorithm.

portance of that input for the new network is verified. In this way, networks that do not generate good performance are discarded, which may be due to the fact that this input characteristic is not relevant to learning the model. This transformation of each input characteristic to its membership function provided by the μFCM could be used to make a selection of relevant input characteristics because it is done individually for each input.

5. Experimental results

The proposed discretization technique is evaluated using the ELM explained in Section 4. Table 2 shows the number of input features and number of classes

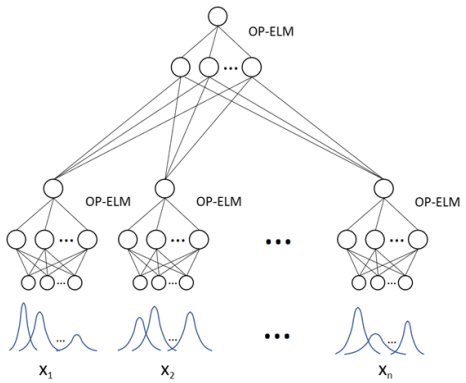


Fig. 8. Network committee used to train the model. Each input characteristic is trained by one network and its output is the input of the final network. All are trained with the OP-ELM algorithm.

Table 2
Dataset Description

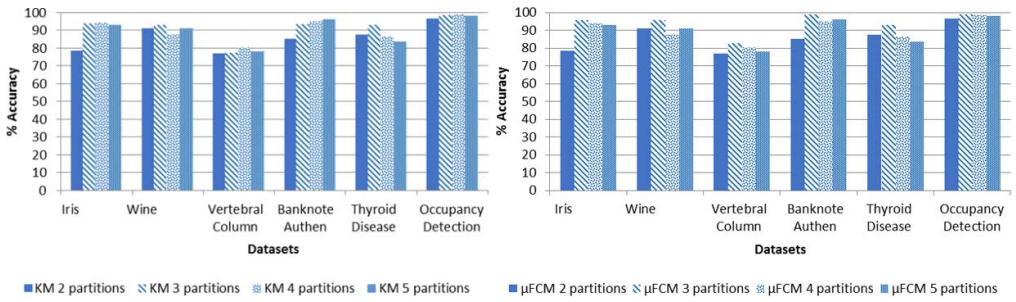
Name	Nu	N.Classes
Iris	4	3
Wine	13	3
Vertebral Column	6	2
Banknote Authentication	4	2
Thyroid Disease	5	3
Occupancy Detection	4	2

for each dataset, these datasets are obtained from UCI repository [20].

The proposed technique (μFCM) is compared with the classic technique K-means (KM). On the one hand, the goodness of the techniques is assessed by a Leave-One-Out Cross-Validation repeated 30 times. On the other hand, a series of tests have been carried out in order to validate the best number of partitions to divide each one of the numerical attributes of the datasets. In this way, in this study it is taken into account not only the accuracy obtained in classification but also the interpretability of the results obtained.

Figure 9 shows the results obtained by KM (Fig. 9(a)) and μFCM (Fig. 9(b)) techniques using different granularity in partitions. Specifically, a division of 2, 3, 4 and 5 partitions have been used for each attribute. In general, this figure shows how the division into 3 partitions achieves the best results.

The best results obtained with three partitions are shown in Table 3, where it can be seen the classification accuracy with its associated standard deviation.



(a) KM (b) μFCM

Fig. 9. Comparative for the KM and μFCM technique with different granularity in partitions.

Table 3
Experimental results with three partitions

Dataset	KM	μFCM
Iris	93.64 ± 0.72	95.62 ± 0.38
Wine	92.79 ± 1.17	95.51 ± 0.40
Vertebral Column	77.08 ± 0.41	82.60 ± 0.92
Banknote Authen.	84.98 ± 0.09	98.89 ± 0.15
Thyroid Disease	92.81 ± 0.67	93.01 ± 0.58
Occupancy Detection	96.43 ± 0.21	98.09 ± 0.09

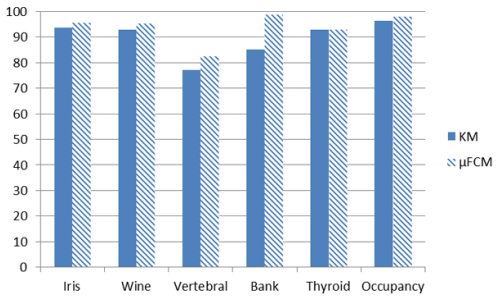


Fig. 10. Graphical comparison of the validation of the techniques KM and μFCM.

It can be clearly seen that μFCM gets better results. More graphically, the results are shown in Fig. 10.

To validate this assertion, a non parametric statistical test has been performed. Specifically, the Wilcoxons Signed Ranks Test is used [18]. This test compares two paired groups and can be used to test where the null hypothesis indicate that two populations have the same continuous distribution.

Table 4 shows the p-value and the negative and positive rank and the statistical Z that is based on negative rank. As it is shown in this table, the technique μFCM is compared statistically with KM technique, obtaining

Table 4
Statistical test result

μFCM-KM	Results
Negative ranks	0.0
Positives ranks	6.0
Ties	0.0
Z^a	-2.201
p-value	0.028

a p-value of 0.028, that means the null hypothesis (the result means of both technique are similar) is rejected with 97% of confidence level. According to the statistical Z, the technique with more satisfactory results is μFCM. In addition to the quantitative results, it must be also highlighted the qualitative part of them. Considering that these results are achieved with three partitions for numerical attributes, it can be indicated that the results obtained are satisfactory and interpretable, since for each attribute all the continuous values are reduced to 3 values.

In addition to the results obtained, a committee system to validate the results has been used. The use of a committee system has been tested with Iris and Vertebral datasets. Figures 11 and 12 show the classification of each individual network for each input features. It can be seen that features number 3 and 4 are the most relevant for Iris data, Fig. 13 shows the accuracy classification (in %) for iris data, where it has trained with the KM, μFCM and two different committees: where only the input feature number 2 has been removed, and another where the features 1 and 2 have been removed. It can be observed as improving the learning by eliminating the less relevant input features. Nevertheless, all features are similar for accuracy classification in the vertebral dataset (see Fig. 12), so, Fig. 14 shows the KM, μFCM and several different committees: with all

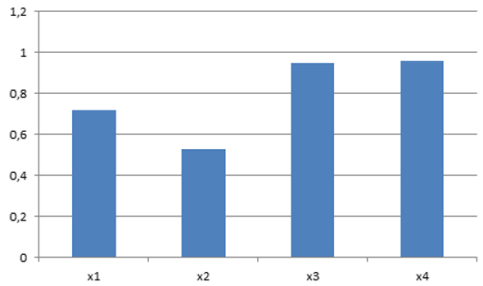


Fig. 11. Iris data. Detail of the learning of each input characteristic for the network committee. It can be seen that the best classification accuracy is for x3 and x4.

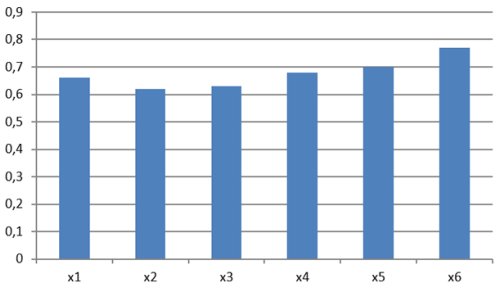


Fig. 12. Vertebral Column dataset. Detail of the learning (accuracy in %) of each input characteristic for the network committee.

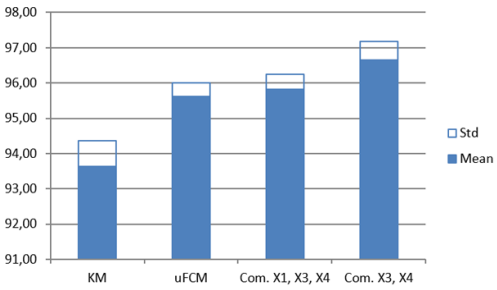


Fig. 13. Iris data. Classification accuracy (mean and standard deviation) with KM algorithms, μFCM and μFCM committee with all inputs features and removing each feature.

features and removing each feature separately. It can be observed that improves learning by eliminating the fourth input characteristic.

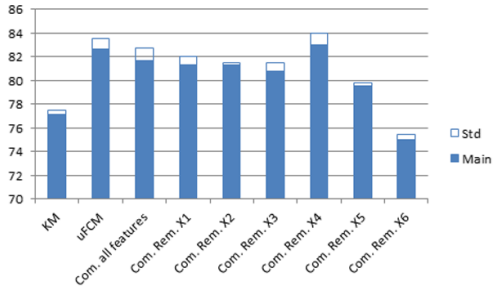


Fig. 14. Vertebral Column dataset. Classification accuracy (mean and standard deviation) with KM algorithms, μFCM and μFCM committee.

6. Conclusions and future work

In this paper, a discretization technique based on fuzzy clustering is proposed. This technique is based on the μFCM algorithm to perform a numerical discretization using Cauchy distribution. In order to assess the discretization technique a ELM classifier and a committee of ELM classifiers are applied. The classifiers are also applied to asses the discretization obtained for the KM technique. In addition, from a qualitative point of view, the best granularity has also been evaluated to carry out discretization. For this purpose, different granularities have been evaluated regarding the number of partitions for each numerical attribute. This evaluation has made it possible to verify that in general terms satisfactory results are obtained with 3 partitions in each attribute. This gives us a great interpretability of the results, since the technique allows to transform continuous values for each attribute into discrete values through 3 partitions. Therefore, from a qualitative point of view, the results are quite interesting. Besides, the results of the comparison between μFCM and KM techniques have been statistically validated, where the proposed technique (μFCM) obtaining the best results with 97% of confidence level.

As future work, the new objective functions of the discretization clustering technique will be implemented. Also, the option to discretize using probabilistic functions for the clustering technique instead of the membership function used in this study can be explored and analyzed.

Acknowledgement

This work is supported by the Spanish MINECO under grant TIN2016-78799-P (AEI/FEDER, UE).

References

[1] J.M. Banda and R.A. Angryk, On the effectiveness of fuzzy clustering as a data discretization technique for large-scale classification of solar images, in: *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*, IEEE, 2009, pp. 2019–2024. doi:10.1109/FUZZY.2009.5277273.

[2] J.C. Bezdek, R. Ehrlich and W. Full, FCM: The fuzzy c-means clustering algorithm, *Computers & Geosciences* **10**(2–3) (1984), 191–203. doi:10.1016/0098-3004(84)90020-7.

[3] U. Boryczka and J. Kozak, An adaptive discretization in the ACDT algorithm for continuous attributes, in: *Computational Collective Intelligence. Technologies and Applications*, 2011, pp. 475–484.

[4] R. Butterworth, D.A. Simovici, G.S. Santos and L. Ohno-Machado, A greedy algorithm for supervised discretization, *Journal of biomedical informatics* **37**(4) (2004), 285–292. doi:10.1016/j.jbi.2004.07.006.

[5] M.R. Chmielewski and J.W. Grzymala-Busse, Global discretization of continuous attributes as preprocessing for machine learning, *International journal of approximate reasoning* **15**(4) (1996), 319–331. doi:10.1016/S0888-613X(96)00074-6.

[6] U. Fayyad and K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: *International Joint Conf. Artificial Intelligence (IJCAI)*, 1993, pp. 1022–1029.

[7] A. Flores-Sintas, J.M. Cadenas and F. Martin, Partition validity and defuzzification, *Fuzzy Sets and Systems* **112**(3) (2000), 433–447. doi:10.1016/S0165-0114(98)00004-9.

[8] S. Garcia, J. Luengo, J.A. Sáez, V. Lopez and F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, *IEEE Transactions on Knowledge and Data Engineering* **25**(4) (2013), 734–750. doi:10.1109/TKDE.2012.35.

[9] G.B. Guang-Bin and L. Chen, Convex incremental extreme learning machine, *Neurocomputing* **70**(16) (2007), 3056–3062.

[10] D.E. Gustafson and W.C. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in: *1978 IEEE Conference on Decision and Control Including the 17th Symposium on Adaptive Processes*, 1978, pp. 761–766. doi:10.1109/CDC.1978.268028.

[11] L.K. Hansen and P. Salamon, Neural network ensembles, *IEEE transactions on pattern analysis and machine intelligence* **12**(10) (1990), 993–1001. doi:10.1109/34.58871.

[12] J.A. Hartigan, *Clustering Algorithms*, 99th edn, John Wiley & Sons, Inc., New York, NY, USA, 1975. ISBN 047135645X.

[13] G.B. Huang, D. Wang and Y. Lan, Extreme learning machines: A survey, *International Journal of Machine Learning and Cybernetics* **2**(2) (2011), 107–122. doi:10.1007/s13042-011-0019-y.

[14] G.B. Huang, Q.Y. Zhu and C.K. Siew, Extreme learning machine: Theory and applications, *Neurocomputing* **70**(1) (2006), 489–501. doi:10.1016/j.neucom.2005.12.126.

[15] F. Jiang and Y. Sui, A novel approach for discretization of continuous attributes in rough set theory, *Knowledge-Based Systems* **73** (2015), 324–334. doi:10.1016/j.knosys.2014.10.014.

[16] Y.-G. Jung, K.M. Kim and Y.M. Kwon, Using weighted hybrid discretization method to analyze climate changes, in: *Computer Applications for Graphics, Grid Computing, and Industrial Environment*, Springer, 2012, pp. 189–195. doi:10.1007/978-3-642-35600-1_28.

[17] R. Kerber, Chimerge: Discretization of numeric attributes, in: *Proceedings of the Tenth National Conference on Artificial Intelligence*, Aaai Press, 1992, pp. 123–128.

[18] W.H. Kruskal, Historical notes on the Wilcoxon unpaired two-sample test, *Journal of the American Statistical Association* **52**(279) (1957), 356–360. doi:10.1080/01621459.1957.10501395.

[19] D. Li, M. Zhang, S. Zhou and C. Zheng, A new approach of self-adaptive discretization to enhance the apriori quantitative association rule mining, in: *Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on*, IEEE, 2012, pp. 44–47. doi:10.1109/ISdea.2012.540.

[20] M. Lichman, UCI machine learning repository (http://archive.ics.uci.edu/ml). University of California, Irvine, School of Information and Computer Sciences, Irvine, CA, 2013.

[21] A. Liu, X. Li and J. Zhang, A soft partition discretization algorithm based on fuzzy clustering, in: *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, 2012, pp. 419–423. doi:10.1109/FSKD.2012.6234116.

[22] H. Liu, F. Hussain, C.L. Tan and M. Dash, Discretization: An enabling technique, *Data mining and knowledge discovery* **6**(4) (2002), 393–423. doi:10.1023/A:1016304305535.

[23] G. Madhu et al., A non-parametric discretization based imputation algorithm for continuous attributes with missing data values, *International Journal of Information Processing* **8**(1) (2014), 64–72.

[24] P.C. Mahalanobis, On the generalised distance in statistics, in: *Proceedings National Institute of Science, India*, Vol. 2, 1936, pp. 49–55, http://ir.isical.ac.in/dspace/handle/1/1268.

[25] D.M. Maslove, T. Podchiyska and H.J. Lowe, Discretization of continuous features in clinical datasets, *Journal of the American Medical Informatics Association* **20**(3) (2013), 544–553. doi:10.1136/amiajnl-2012-000929.

[26] F. Mateo and A. Lendasse, A variable selection approach based on the delta test for extreme learning machine models, in: *Proceedings of the European Symposium on Time Series Prediction*, 2008, pp. 57–66.

[27] Y. Miche, P. Bas, C. Jutten, O. Simula and A. Lendasse, A methodology for building regression models using extreme learning machine: OP-ELM., in: *ESANN*, 2008, pp. 247–252.

[28] Y. Miche and A. Lendasse, A faster model selection criterion for OP-ELM and OP-KNN: Hannan–Quinn criterion., in: *ESANN*, Vol. 9, 2009, pp. 177–182.

[29] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten and A. Lendasse, OP-ELM: Optimally pruned extreme learning machine, *IEEE Transactions on Neural Networks* **21**(1) (2010), 158–162. doi:10.1109/TNN.2009.2036259.

[30] Y. Miche, A. Sorjamaa and A. Lendasse, OP-ELM: Theory, experiments and a toolbox, in: *International Conference on Artificial Neural Networks*, Springer, 2008, pp. 145–154.

[31] H.J. Rong, Y.S. Ong, A.H. Tan and Z. Zhu, A fast pruned-extreme learning machine for classification problem, *Neurocomputing* **72**(1) (2008), 359–366. doi:10.1016/j.neucom.2008.01.005.

[32] D. Serre, *Matrices: Theory and Applications*, Springer, New York, 2002.

[33] T. Similä and J. Tikka, Multiresponse sparse regression with application to multidimensional scaling, in: *International Con-*

ference on Artificial Neural Networks, Springer, 2005, pp. 97–102.

[34] J. Soto, A. Flores-Sintas and J. Palarea-Albaladejo, Improving probabilities in a fuzzy clustering partition, *Fuzzy Sets and Systems* **159**(4) (2008), 406–421. doi:10.1016/j.fss.2007.08.016.

[35] H. Wang, P.S. Yu and J. Han, Data mining and knowledge discovery handbook, in: *Data Min. Knowl. Discov. Handb*, 2010, pp. 1269–1277.

[36] T.-T. Wong, A hybrid discretization method for naïve Bayesian classifiers, *Pattern Recognition* **45**(6) (2012), 2321–2325. doi:10.1016/j.patcog.2011.12.014.

[37] M. Zeinalkhani and M. Eftekhari, Fuzzy partitioning of continuous attributes through discretization methods to construct fuzzy decision tree classifiers, *Information Sciences* **278** (2014), 715–735. doi:10.1016/j.ins.2014.03.087.

[38] Z.-H. Zhou, J. Wu and W. Tang, Ensembling neural networks: Many could be better than all, *Artificial intelligence* **137**(1–2) (2002), 239–263.

[39] J. Zhu and M. Collette, A dynamic discretization method for reliability inference in dynamic Bayesian networks, *Reliability Engineering & System Safety* **138** (2015), 242–252. doi:10.1016/j.ress.2015.01.017.

2.4 Parallel implementation of fuzzy minimals clustering algorithm

Título	*Parallel Implementation of Fuzzy Minimals Clustering Algorithm*
Autores	Isabel Timón, Jesús Soto, Horacio Pérez-Sánchez y José M. Cecilia
Revista	Expert Systems With Applications
Año	2016
Estado	Publicado

Contribución de la doctoranda

Isabel María Timón Pérez declara, con el visto bueno del resto de autores, ser el principal autor y la principal contribuidora del artículo *Parallel Implementation of Fuzzy Minimals Clustering Algorithm*.

Expert Systems With Applications 48 (2016) 35–41

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Parallel implementation of fuzzy minimals clustering algorithm☆

Isabel Timón, Jesús Soto*, Horacio Pérez-Sánchez, José M. Cecilia

Bioinformatics and High Performance Computing Research Group (BIO-HPC), Computer Science Department, Universidad Católica San Antonio de Murcia (UCAM), Spain

ARTICLE INFO

Keywords:
Parallel fuzzy clustering
Fuzzy clustering
Fuzzy minimals

ABSTRACT

Clustering aims to classify different patterns into groups called *clusters*. Many algorithms for both hard and fuzzy clustering have been developed to deal with exploratory data analysis in many contexts such as image processing, pattern recognition, etc. However, we are witnessing the era of big data computing where computing resources are becoming the main bottleneck to deal with those large datasets. In this context, sequential algorithms need to be redesigned and even rethought to fully leverage the emergent massively parallel architectures. In this paper, we propose a parallel implementation of the fuzzy minimals clustering algorithm called *Parallel Fuzzy Minimal* (PFM). Our experimental results reveal linear speed-up of PFM when compared to the sequential counterpart version, keeping very good classification quality.

1. Introduction

We are witnessing the consequences of Web 2.0 where huge amounts of data are continuously generated. Sources of data such as commercial interactions, including financial transactions, search histories, and product information; public agencies that contribute with medical records, population databases; or scientists that routinely undertake large-scale simulations for weather prediction, drug discovery and so on, are only some examples of this landscape of data deluge. These progressively generated big datasets can be considered as valuable resources, since they can provide with key insights into human behavior, market trends, diseases, engineering safety, environmental change, etc. (Duranton, Black-Schaffer, De Bosschere, & Maebe, 2013; Manyika et al., 2011).

Clustering is a data-analysis technique that aims to organize a collection of patterns (usually represented as a vector of measurements, or a point in a multidimensional space) into groups (or clusters) based on a similarity metric (namely Euclidean, city-block, Mahalanobis, etc.) (Tan, Steinbach, & Kumar, 2006). As expected, patterns that belong to the same cluster are more similar to each other than they are to a pattern belonging to a different cluster (Jain, Murty, & Flynn, 1999). Clustering has been successfully applied to the analysis of datasets from several fields such as image processing, pattern recognition, analysis of microarray data in bioinformatics, etc, in order to provide valuable knowledge within these fields (Agrawal, Gehrke, Gunopulos, & Raghavan, 1998; Bezdek, 1981; de Hoon, Imoto, Nolan, & Miyano, 2004; Wu & Leahy, 1993).

Many data clustering algorithms have been proposed in the literature for many different scientific application (we refer the reader to Jain, 2010 for a review). A good data clustering algorithm should have the following characteristics: (1) *scalability* i.e. the ability to handle a growing amount of objects and attributes in a capable manner, (2) *adaptability* to determine clusters of different shape or size, (3) *self-driven* as it should require minimum knowledge of the problem domain (e.g., number of clusters, thresholds, termination condition parameters), (4) *stability* as it should remain stable in the presence of noise and outliers, and finally (5) *data-independency* as it should be insensitive to the way the objects are organized in the dataset (Han & Kamber, 2006).

Most of current clustering algorithms depend on iterative procedures in order to find local or global optimal solutions in high-dimensional datasets. The capability to find these solutions usually requires to perform many experiments with different algorithms and to study the influence of different dataset features. Hence, clustering algorithms have a high intrinsic time complexity. For example, the classic *k-means* method is NP-hard even when $k = 2$ (Tong & Kang, 2013). Therefore, the parallelization of clustering algorithms becomes mandatory in the era of big data.

The first parallelization proposals for a data-clustering algorithm were based on *k-means* (Dhillon & Modha, 2000; Nagesh, Goil, & Choudhary, 2000). However, *k-means* provides a hard-partition scheme; i.e. each data point belongs to exactly one cluster. Of particular interest to us are those clustering algorithms that provide multiple and non-dichotomous cluster memberships; i.e *fuzzy*

☆ *The authors declare that there is no conflict of interests regarding the publication of this article.*

* Corresponding author. Tel.: +34968278821.
E-mail addresses: imtimon@alu.ucam.edu (I. Timón), jsoto@ucam.edu (J. Soto), hperez@ucam.edu (H. Pérez-Sánchez), jmcecilia@ucam.edu (J.M. Cecilia).

http://dx.doi.org/10.1016/j.eswa.2015.11.011
0957-4174/

clustering. One of the most widely used fuzzy clustering methods is the fuzzy c-means (FCM) algorithm (Bezdek, Ehrlich, & Full, 1984). Some parallelization efforts have been done in the literature for FCM algorithm to deal with large datasets (Havens, Bezdek, Leckie, Hall, & Palaniswami, 2012; Kwok, Smith, Lozano, & Taniar, 2002; Modenesi, Costa, Evsukoff, & Ebecken, 2007; Rahimi, Zargham, Thakre, & Chhillar, 2004; Ravi, Suvarna, DSouza, & Reddy, 2012). However, the FCM's execution time grows exponentially with the problem size as it needs prior knowledge about the number of clusters to generate, and therefore, several executions should be done to find out the optimal number of clusters.

In this paper, we propose a parallel version of the Fuzzy Minimals (FM) clustering algorithm, which was proposed first by Flores-Sintas, Cadenas, and Martin (1998) and modified by Soto, Flores-Sintas, and Palarea-Albaladejo (2008). This algorithm does not need prior knowledge about the number of clusters and presents the advantage that the clusters do not need to be CWS (compact well-separated), being this feature a requirement for parallelization. Our performance evaluation focuses on speed-up and scalability factors, and it reveals that our approach obtains lineal speed-up while clustering quality remains stable compared to the sequential counterpart version. The rest of the paper is structure as follows: Section 2 shows a brief outline of the FM and *Parallel Fuzzy Minimals* (PFM) clustering algorithms. Section 3 describes the experimental results of PFM before we conclude the paper with the main conclusions obtained from our findings and some directions for future work.

2. Methods

2.1. The fuzzy minimals algorithm

The Fuzzy Minimals (FM) algorithm was initially proposed by Flores-Sintas et al. where authors demonstrate that FM algorithm fulfills the expected characteristics of a classification algorithm; i.e. scalability, adaptability, self-driven, stability and data-independent. In what follows, we reprise FM's description and we refer the reader to Flores-Sintas et al. (1998); Flores-Sintas, M. Cadenas, and Martin (2001); Soto et al. (2008) for insights in the definition and demonstration of FM algorithm.

In general, fuzzy clustering techniques like Fuzzy C-Means (FCM) algorithm (Bezdek et al., 1984) minimize an objective function that determines the prototypes of each cluster. Let X be a set of n data points,

$$X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^F.$$

where F is the dimension of the vector space. FM algorithm uses the objective function given by Eq. (1):

$$J_{(v)} = \sum_{x \in X} \frac{d_{xv}^2}{1 + r^2 d_{xv}^2}, \tag{1}$$

where d_{xv}^2 is the Euclidean norm that determines the distance between two points in the dataset. The factor r measures the *isotropy* in the dataset. The use of the Euclidean distance implies that we are assuming the homogeneity and isotropy of the feature space. Whenever the homogeneity and isotropy are broken, clusters are created in the features space. The factor r measures the disruption of the homogeneity and isotropy of the sample by a set of factors affecting the Euclidean distance in each group (Flores-Sintas et al., 2001). Eq. (2) shows the factor r calculation in a non-linear expression

$$\frac{\sqrt{|C^{-1}|}}{nr^F} \sum_{x \in X} \frac{1}{1 + r^2 d_{xm}^2} = 1, \tag{2}$$

where $|C^{-1}|$ is the determinant of the inverse of the covariance matrix. m is the mean of the sample X, d_{xm} is the Euclidean distance between x and m, and n is the number of elements of the sample.

In the FM algorithm, the objective function presented in Eq. 1 is reformulated as shown in Eq. (3)

$$J_{(v)} = \sum_{x \in X} \mu_{xv} \cdot d_{xv}^2, \tag{3}$$

where

$$\mu_{xv} = \frac{1}{1 + r^2 \cdot d_{xv}^2}, \tag{4}$$

Eq. (4) is the membership function that measures the degree of membership for a given element x to the cluster where v is the prototype. The FM algorithm is an iterative procedure that minimizes the objective function through Eq. (5), giving the prototypes that represents each cluster.

$$v = \frac{\sum_{x \in X} \mu_{xv}^2 \cdot x}{\sum_{x \in X} \mu_{xv}^2} \tag{5}$$

Algorithm 1 shows the FM algorithm. It is an iterative process where two standard values are included in the computation. ε_1 establishes the error degree committed in the minimum estimation; and ε_2 shows the difference between potential minimums.

Algorithm 1 Fuzzy Minimals algorithm, where n is the size of the dataset. V is the algorithm output that contains the prototypes found by the clustering process. F is the dimension of the vector space.

1: Choose ε_1 and ε_2 standard parameters.
2: Initialize $V = \{\} \subset \mathbb{R}^F$.
3: Estimate factor r.
4: **for** $k = 1; k < n; k = k + 1$ **do**
5: $v_{(0)} = x_k, t = 0, E_{(0)} = 1$
6: **while** $E_{(t)} \geq \varepsilon_1$ **do**
7: $t = t + 1$
8: $\mu_{xv} = \frac{1}{1+r^2 \cdot d_{xv}^2}$, using $v_{(t-1)}$
9: $v_{(t)} = \frac{\sum_{x \in X} \left(\mu_{xv}^{(t)}\right)^2 \cdot x}{\left(\mu_{xv}^{(t)}\right)^2}$
10: $E_{(t)} = \sum_{\alpha=1}^{F} \left(v_{(t)}^{\alpha} - v_{(t-1)}^{\alpha}\right)$
11: **end while**
12: **if** $\sum_{\alpha}^{F} (v^{\alpha} - w^{\alpha}) > \varepsilon_2, \forall w \in V$ **then**
13: $V \equiv V + \{v\}$.
14: **end if**
15: **end for**

As a fuzzy classification algorithm, the FM computation is similar than well-known FCM algorithm. However, FM algorithm minimizes a different objective function than FCM. The minimum values of the objective function are the prototypes that represent the clusters obtained by the classification process. FM algorithm does not need prior knowledge about the number of prototypes the user wants to identify in the dataset as it is the case of FCM algorithm. FM finds the number of prototypes according to the data-structure which is actually modeled by *factor r* in FM's objective function. Those prototypes are actually the output obtained from FM. Moreover, the clusters do not need to be CWS (compact well separated) in the FM algorithm which is also an important issue in many datasets.

2.2. The parallel fuzzy minimals algorithm

This section introduces our proposal, the *Parallel Fuzzy Minimal* (PFM) algorithm, which is designed to enhance the execution of the FM clustering algorithm previously explained. FM algorithm does not need to compare clusters to minimize the objective function like FCM does. PFM benefits from this property to divide the original dataset into different *data-partitions* where FM will be applied to. This partition of data does not lose information about global properties for

the classification, since each subset will be classified using an objective function that includes factor r, which has information about the overall data structure. Indeed, we consider the factor r as a global parameter that contains information about the whole dataset, and we use it in each of those data-partitions that will run in parallel, maintaining the main clustering characteristics of FM unaltered.

Algorithm 2 Parallel Fuzzy Minimals pseudocode where n is the total number of elements, p is number of processors, V is the set of prototypes, C is the set of cluster obtained by hierarchical clustering, X is the targeted dataset and r is the factor r that measures isotropy and homogeneity of the whole dataset.

```
Read X = {x_1, x_2, ..., x_n} ⊂ R^F
Initialize V = { } ⊂ R^F.
Initialize C = { } ⊂ R^F.
pid = 0
Estimate factor r
for k = 0; k < n; k = h + (n/p) do
    pid = pid +1
    Send (pid, &X[k], n/p, r)
end for
(All processors) Execute FM (X, n/p, r)
for pid = 0; pid < p; pid = pid + 1 do
    (Receive processor) V ≡ V + Receive(pid)
end for
(Master processor) C = Hierarchical clustering (V)
```

The algorithm 2 shows the PFM algorithm. The Master processor (master) prepares the execution and sends the data to each processor involved in the computation. First of all, the master reads the dataset to be classified (X), initializes some structures to store prototypes (V) and clusters of prototypes (C). Then, it computes the factor r, using the whole dataset (X). Next, the master divides the dataset equally among the processors so that each processor handles n/p data points (being n the total number of elements and p the number of processors involved in the computation). Once the different processors receive the information, they proceed with the FM clustering algorithm over the n/p data assigned to it and also with the r factor previously calculated by the master. Finally, the master gathers all prototypes into an unique group (V) before it proceeds with a hierarchical clustering to determine the final clustering result.

If the data-partition was performed based on a data order, then the classification algorithm would give us completely independent prototypes and thus the hierarchical clustering would not be needed as a final step. However, we do not assume this criteria and data-partitions are performed randomly. Therefore, the prototypes can be very close to each other and PFM needs to check whether these prototypes are actually representative or not. Hierarchical clustering groups data over a variety of scales by creating a cluster tree or dendrogram. The tree is not a single set of clusters, but rather a multilevel hierarchy, where clusters at one level are joined as clusters at the next level. This allows you to decide the level or scale of clustering that is most appropriate for your application.

3. Results and discussion

This section shows the experimental results obtained with the *Parallel Fuzzy Minimal* (PFM) clustering algorithm. We describe the datasets used for the evaluation before we show the experimental results. PFM is first validated by comparing its results with FCM on the well-known Anderson's Iris Data (Anderson, 1934). Then, the PFM's performance scalability and clustering quality are shown. Finally, we discuss the benefits of our proposal. We refer the reader to Flores-Sintas et al. (1998) for a comparison between FCM and FM.

Table 1
Benchmarks overview.

Codename	Number of points	Dimension	Size	Source
Iris data	150	$\mathbb{R}^4$	Small	(Anderson, 1934)
Ellipsoids	5.000	$\mathbb{R}^3$	Medium	Synthetically developed
USA hospitals	15.506	$\mathbb{R}^2$	Large	(NGA, 2015)

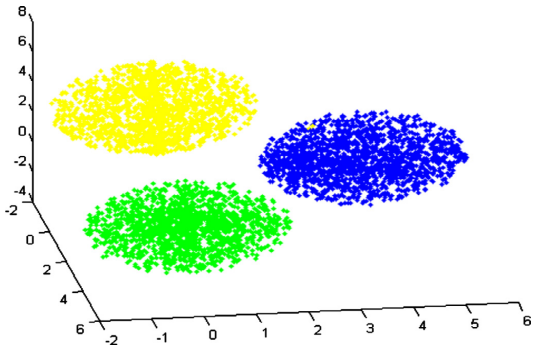


Fig. 1. Synthetically developed benchmark that stands for three well-separated ellipsoids.

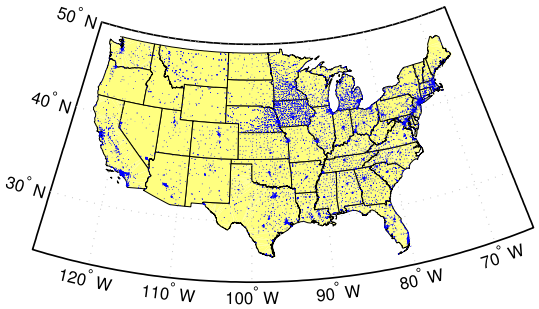


Fig. 2. 15.506 geographical coordinates of USA hospitals (source NGA).

3.1. Datasets

We test our algorithms using a set of benchmark instances that have different sizes and characteristics (see Table 1). Our first dataset is the well-known Andersons Iris Data (Anderson, 1934), which consists of the sepal length, sepal width, petal length and petal width for each of 150 irises. The first 50 plants, according to Andersons ordering, are Iris Setosa; the second 50 are Iris Versicolor and the last 50 are Iris Virginica. Iris Versicolor is a hybrid of Iris Setosa and Iris Virginica but it is much more similar to the latter. Consequently, Iris Setosa is easily identified. From the other side, it is very difficult to separate the other two groups.

Our second dataset is synthetically developed for testing the PFM's scalability. It consists of 5000 3-Dimensional points which represent three well-separated ellipsoids (see Fig. 1). It is selected to ensure a representative sample, i.e. these ellipsoids are close enough to each other in order to test the quality of our clustering.

Our last benchmark is a large-dataset that has up to 15.506 geographical coordinates of USA hospitals obtained from USA National Geospatial-Intelligence Agency (NGA) (NGA, 2015). It provides 2-dimensional points that are spread out among different states of USA (see Fig. 2). This benchmark has been disorganized randomly in order to avoid any correlation.

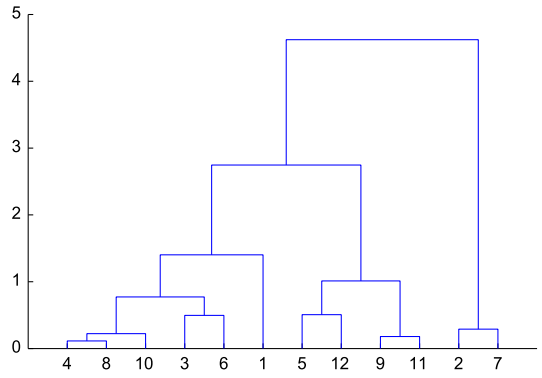


Fig. 3. Hierarchical clustering of Iris Data, showing a dendogram that divides the data into three main clusters.

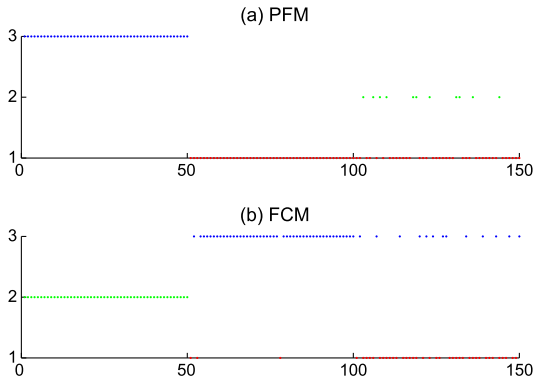


Fig. 4. Quality comparison between PFM And FCM on Iris dataset. In x-axis the first 50 plants are Iris Setosa; the second 50 are Iris Versicolor and the last 50 are Iris Virginica. The y-axis shows different clusters where the data points are included.

3.2. Experimental results

This Section analyzes Parallel Fuzzy Minimals (PFM) algorithm. We focus on the computational features of the Fuzzy Minimals (FM) algorithm and how it can be designed on parallel systems. To guarantee the correctness of our algorithms, a quality comparison between the results obtained by FCM, FM and PFM is also provided. First of all, we compare PFM to the well-known FCM, provided by Bezdek in Bezdek (1981) on the Anderson's Iris dataset. Then, we focus on performance evaluation of PFM algorithm using two different benchmarks previously described. The experiments are developed on a Windows-based laptop machine with 4 GB DDR3 memory and Intel Core I5 1,6 Mhz processor using Matlab 6.0 release 12.

3.3. FCM Vs. PFM

As previously mentioned, Anderson's Iris data is a small benchmark which is structured into three main groups, where one of them, Iris Setosa, it is easily identified but the others two are very difficult to separate. We divide the Iris dataset randomly into two data-partitions for setting up the PFM algorithm. We use only two data-partitions to make those partitions representative enough. Actually, PFM is designed to deal with large datasets where many data-partitions can be obtained without losing generality. Fig. 3 shows the dendogram generated by the PFM's final step; i.e. the hierarchical clustering. It shows three main clusters are found as expected.

Fig. 4 shows a quality comparison between FCM and PFM. The *x*-axis shows the Iris data according to Andersons ordering where the first 50 plants are Iris Setosa; the second 50 are Iris Versicolor and the last 50 are Iris Virginica. The y-axis shows the cluster that belongs each data. The ideal classification is given by equation $f(x_i) = m_i, x_i \in [1, 150], m_i = \{1, 2, 3\}$, then $f(x_i) = f(x_j) \forall i, j \in [1 + 50(k-1), 50k]$, with $k = \{1, 2, 3\}$. Fig. 4 also shows that both FM and FCM find three main clusters. Both of them clearly identify Iris Setosa data ant they have some troubles to identify the other two clusters. Those results can be improved after refinement in both algorithms but this is out of the paper's scope.

3.4. PFM Vs. FM

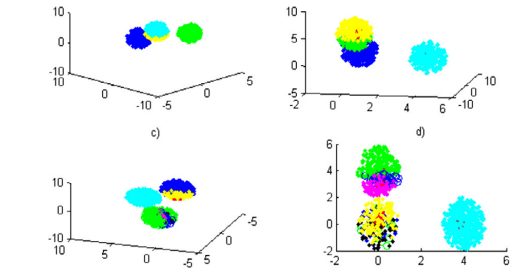
3.4.1. Ellipsoids dataset

Our experimental environment for this benchmark uses five different test cases. They are selected according to the number of data-partitions in which we divide the original dataset (see Fig. 1). First of all, we run the PFM algorithm with only one data-partition, which actually represents the full dataset; i.e. we apply FM clustering to the whole dataset. Henceforth, we proceed with the PFM clustering algorithm previously explained in Section 2.2. Four different test cases are executed with 2, 4, 5 and 8 data-partitions of the whole dataset respectively. We ensure that each of these data-partitions contains above 10% of the original dataset as we empirically demonstrate this is the minimum percentage of data, for this particular case, that each subset should have in order to be considered as representative.

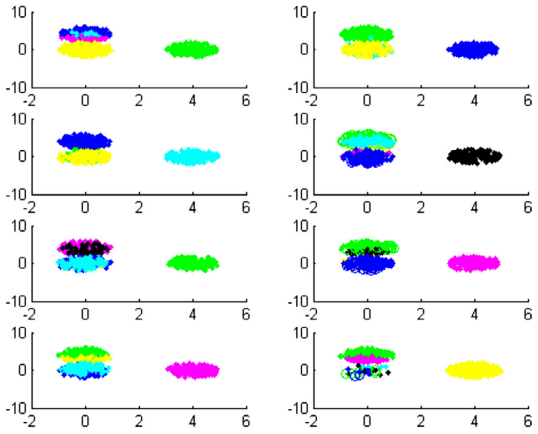
Table 2
Execution times (in seconds) for our Parallel Fuzzy Minimal Algorithm on different data-partition schemes. The number of processes are increased according to the number of subsets in which the original dataset has been divided.

Number of subsets (processes)	Execution time (seconds)	Total execution time (seconds)
1 subset	16.455,22	16.455,22
2 subset	4.438,90	8.206,04
	3.764,93	
4 subset	692,96	3.216,04
	760,61	
	907,39	
	855,08	
5 subset	466,55	2.778,98
	436,97	
	634,87	
	647,85	
	590,52	
8 subset	225,12	1.786,89
	177,94	
	213,95	
	279,42	
	193,60	
	291,01	
	213,02	
	190,63	

Table 2 shows the execution time in seconds for each of these test cases. The number of subsets, and thus the number of independent processes we execute, are shown on the left-hand side of this table. Then, we show the execution time for each independent process which is basically the execution time to run steps 2 and 3 of PFM algorithm. Finally, we summarize the total execution time, that also includes steps 1 and 4, consisting of both: compute the factor r for the full dataset (2, 20 s), gathering all prototypes into an unique group and proceed with a hierarchical clustering for this group to eventually determine the number of clusters (0, 57 s). We show a super lineal scalability along with the number of data-partitions (processes). The main computation of PFM is the calculation of the matrix inverse, so

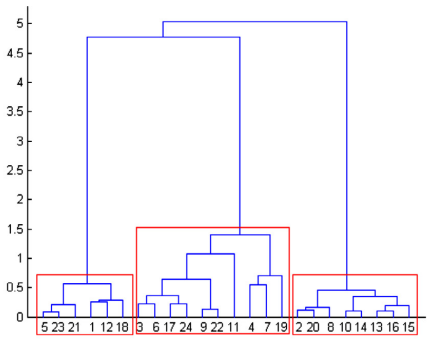


(a) Data points distribution for four data-partitions.

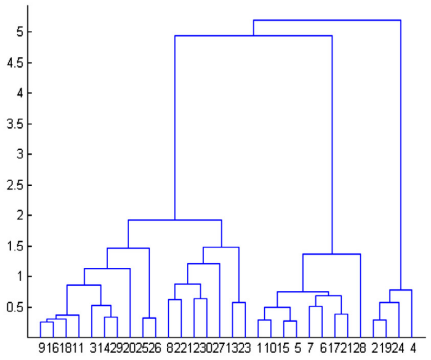


(b) Data points distribution for eight data-partitions.

Fig. 5. Data points distribution for 4 and 8 data-partitions test cases. Each color represents a different set of points that belong to the same prototype that has been found by PFM's.



(a) Hierarchical clustering for protoypes found by PFM on Ellipsoids dataset using four data-partitions.



(b) Hierarchical clustering for protoypes found by PFM on Ellipsoids dataset using eight data-partitions

Fig. 6. Hierarchical clustering for protoypes found by PFM on Ellipsoids dataset. The dendogram clearly shows three main clusters.

if the matrix dimensions are reduced, the computational cost of calculating the inverse of the matrix is reduced substantially. However, there is a bottleneck in the number of subsets we can divide original dataset. As previously explained, subsets should contain above 10% of the original dataset. The data structure is very important for this classification technique as Flores-Sintas stated in Flores-Sintas et al. (1998). If the original dataset is divided into many data-partitions, the intrinsic properties of the original dataset could be affected and therefore the factor r that measures the isotropy of the whole dataset could not be representative.

Fig. 5 a and b shows the classification for four and eight data-partitions respectively. They contain up to 4 (or 8) subfigures within them, showing the PFM's classification result for each data-partition based on prototypes selection (PFM's step 3 output). Data points in each subfigure represent the data points within each data-partition that belongs to each prototype found by PFM. The number of prototypes found by PFM in each data-partition is the number of different colors drawn in the chart. The Ellipsoids dataset contains up to three different clusters (see Fig. 1). However, at this stage we find many prototypes in each subset before performing the step 4 of PFM algorithm.

Fig. 6a and b shows the result of PFM's step four; i.e. the hierarchical clustering as applied to all prototypes found in the PFM's step 3. We use the Matlab dendogram plot of the hierarchical cluster tree to illustrate this. A dendogram consists of several U-shaped lines that connect data points in a hierarchical tree. The height of each U represents the distance between the two prototypes previously found. We use the distances among all prototypes found in the data-partition to determine the U. These figures clearly show three main clusters are found in both cases which is exactly the same than the initial dataset shown in Fig. 1.

3.4.2. USA hospitals dataset

Table 3 shows the execution time in seconds for our biggest benchmark; the USA hospitals dataset. On the left-hand side we show the number of data-partitions, and thus the number of processes. The execution time for each independent process and the total execution time is also shown. In this case, the computation of the factor r on the full dataset takes 18, 60 s and the hierarchical clustering takes 0, 67 s. The performance results for this benchmark also show good scalability along with the number of processors, obtaining up to 10x speed up factor for 15 data-partitions. In this case, we use 8 (PFM8) and 15 (PFM15) data-partitions as we empirically demonstrate they provide very good results. Indeed, this is a tuning process that can be automatized by means of autotuning techniques.

Fig. 7 shows the prototypes found by FM algorithm. Up to four clusters are found after hierarchical clustering; they are represented in different colors. Prototypes obtained with PFM8 and PFM15 are also represented in Figs. 8 and 9 respectively. As expected, the number of prototypes is higher as some of them are actually the same

Table 3
Execution times (in seconds) for our Parallel Fuzzy Minimal Algorithm on different data-partition schemes. The number of processes is increased according to the number of subsets in which the original dataset is divided.

Number of subsets (processes)	Execution time (seconds)	Total execution time (seconds)
1 subset	87.461,13	87.461,13
8 subset	1.824,80	19.687,04
	2.619,19	
	2.360,45	
	3.425,11	
	2.183,70	
	2.070,93	
	2.268,95	
	2.915,31	
15 subset	639,56	9.095
	656,79	
	740,30	
	816,33	
	679,47	
	669,71	
	779,61	
	604,17	
	959,73	
	696,19	
	524,15	
	669,39	
	659,12	
	690,40	
	678,01	

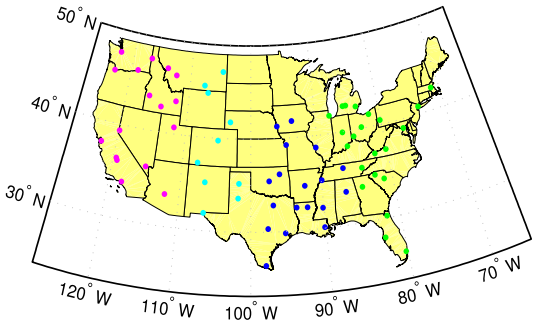


Fig. 7. FM's hierarchical clustering representation. Four clusters are found; each cluster is represented by a different color.

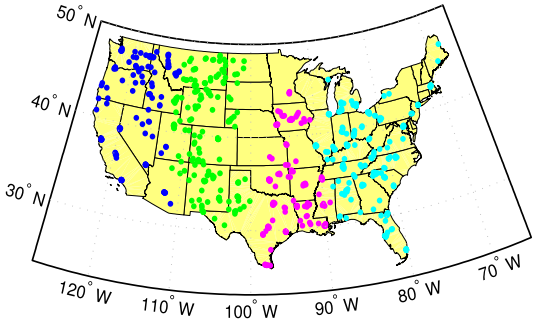


Fig. 8. PFM8's hierarchical clustering representation. Four clusters are found; each cluster is represented by a different color.

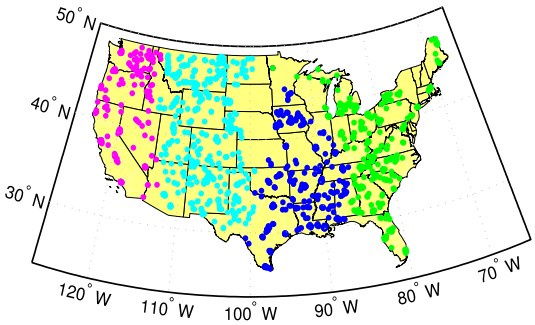


Fig. 9. PFM8's hierarchical clustering representation. Four clusters are found; each cluster is represented by a different color.

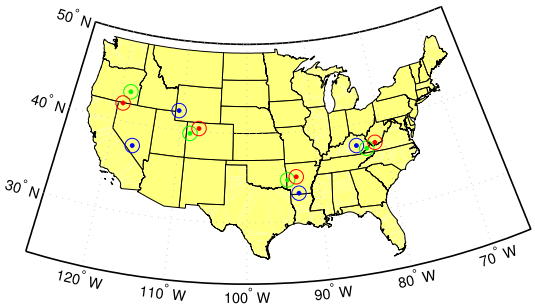


Fig. 10. Means for clustering in 4 group of total set of prototypes. Blue points are FM candidates, Green points are PFM8 candidates and red for PFM15.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

prototype. After hierarchical clustering PFM8 and PFM15 also identifies four clusters.

Finally, Fig. 10 shows the mean of all prototypes found within each cluster for each algorithm. Blue points represents FM candidates, green points PFM8 and red points PFM15. The prototypes distribution of FM, PFM8 and PFM 15 are very similar to each other. The main difference can be found on Hospitals at the west coast. In this part of the dataset, the number of correlations is very high compared to others. FM algorithm is highly influenced by correlations since the determinant of covariance matrix would be zero, and therefore there would not be matrix inverse. PFM, however, divides the dataset into data-partitions, and thus it is likely to have less correlations as they are randomly distributed among data-partitions. Nevertheless, we could not make sure correlations are uniformly distributed and thus the PFM's user have the opportunity to analyze the dataset and distribute the correlations among different data-partitions.

4. Conclusions

Big Data has the potential to transform development and accelerate social progress all over the world. There are several issues surrounding this new era, though, that should be addressed before taking a step further. Among them, we may highlight the lack of computational resources available to deal with this data deluge. Nowadays, the computational field is massively parallel as a consequence of the new trend in developing new microprocessors. This fact makes mandatory the redefinition of our algorithms to fully leverage the new massively parallel platforms. In this paper, we redefine a clustering technique called Fuzzy Minimals, to enhance the classification of

large datasets. Our experimental results reveal linear speed-up along with the number of parallel processes launched while the classification quality is still good enough with our initial assumptions. Moreover, we have noticed a bottleneck in the data division as the subsets created from the initial dataset should contain at least above 10% of the total data in the targeted dataset. Actually, we are working on establishing a general rule that defines the maximum number of subsets that a given data-partition could have in order to achieve peak performance. We are also working on executing our algorithm in a massively parallel environment such a supercomputer. To do so, we will migrate our Matlab implementation to the C programming language and MPI interface.

Acknowledgments

This research was supported by the Fundación Séneca (Agencia Regional de Ciencia y Tecnología de la Región de Murcia) under grant 18946/JLI/13 and by the Nils Coordinated Mobility under grant 012-ABEL-CM-2014A. We also thank Nvidia for the hardware donation under GPU research and Educational center. We want to thank anonymous reviewers for the valuable suggestions they have made to improve the overall quality of this paper. Finally, we warmly thank Antonio Flores-Sintas, who died while the submission process, for his contributions to this work.

References

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Record, 27*, 94–105.

Anderson, E. (1934). The irises in the gaspe peninsula. *Bulletin of the American Iris society, 59*, 2–5.

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers.

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). Fcm: the fuzzy c-means clustering algorithm. *Computers & Geosciences, 10*, 191–203.

Dhillon, I. S., & Modha, D. S. (2000). *A data-clustering algorithm on distributed memory multiprocessors*. Springer-Verlag.

Duranton, M., Black-Schaffer, D., De Bosschere, K., & Maebe, J. (2013). *The hipeac vision for advanced computing in horizon 2020*. HiPEAC High-Performance Embedded Architecture and Compilation.

Flores-Sintas, A., Cadenas, J., & Martin, F. (1998). A local geometrical properties application to fuzzy clustering. *Fuzzy Sets and Systems, 100*, 245–256.

Flores-Sintas, A., M. Cadenas, J., & Martin, F. (2001). Detecting homogeneous groups in clustering using the euclidean distance. *Fuzzy Sets and Systems, 120*, 213–225.

Han, J., & Kamber, M. (2006). *Data mining, southeast asia edition: concepts and techniques*. Morgan kaufmann.

Havens, T. C., Bezdek, J. C., Leckie, C., Hall, L. O., & Palaniswami, M. (2012). Fuzzy c-means algorithms for very large data. *IEEE Transactions on Fuzzy Systems, 20*, 1130–1146.

de Hoon, M. J., Imoto, S., Nolan, J., & Miyano, S. (2004). Open source clustering software. *Bioinformatics, 20*, 1453–1454.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters, 31*, 651–666.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR), 31*, 264–323.

Kwok, T., Smith, K., Lozano, S., & Taniar, D. (2002). Parallel fuzzy c-means clustering for large data sets. *Euro-Par 2002 parallel processing* (pp. 365–374). Springer Berlin Heidelberg.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). *Big data: the next frontier for innovation, competition, and productivity*. McKinsey Global Institute.

Modenesi, M. V., Costa, M. C., Evsukoff, A. G., & Ebecken, N. F. (2007). *Parallel fuzzy c-means cluster analysis*. Springer Berlin Heidelberg.

Nagesh, H. S., Goil, S., & Choudhary, A. (2000). A scalable parallel subspace clustering algorithm for massive datasets. In *Proceedings of the international conference on parallel processing, 2000.* (pp. 477–484). IEEE.

National Geospatial-Intelligence Agency (2015). https://www.nga.mil/Pages/Default.aspx (last accessed oct 27.10.16).

Rahimi, S., Zargham, M., Thakre, A., & Chhillar, D. (2004). A parallel fuzzy c-mean algorithm for image segmentation. In *Proceedings of the IEEE annual meeting of the fuzzy information, 2004. Processing NAFIPS'04.* (pp. 234–237). IEEE.

Ravi, A., Suvarna, A., DSouza, A., Reddy, G. R. M., et al. (2012). A parallel fuzzy c means algorithm for brain tumor segmentation on multiple mri images. In *Proceedings of international conference on advances in computing* (pp. 787–794). Springer India.

Soto, J., Flores-Sintas, A., & Palarea-Albaladejo, J. (2008). Improving probabilities in a fuzzy clustering partition. *Fuzzy Sets and Systems, 159*, 406–421.

Tan, P., Steinbach, M., & Kumar, V. (2006). *Introduction to data mining*. Addison Wesley.

Tong, H., & Kang, U. (2013). Big data clustering. In Charu C. Aggarwal, & Chandan K. Reddy (Eds.), *Data clustering: algorithms and applications*. CRC Press.

Wu, Z., & Leahy, R. (1993). An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*, 1101–1113.

2.5 High-throughput infrastructure for advanced ITS services: A case study on air pollution monitoring

Título	*High-Throughput Infrastructure for Advanced ITS Services: A Case Study on Air Pollution Monitoring*
Autores	José M. Cecilia, Isabel Timón, Jesús Soto, José Santa, Fernando Pereñíguez y Andrés Muñoz
Revista	IEEE Transactions on Intelligent Transportation Systems
Año	2018
Estado	Publicado

Contribución de la doctoranda

Isabel María Timón Pérez declara, con el visto bueno del resto de autores, ser el principal autor y la principal contribuidora del artículo *High-Throughput Infrastructure for Advanced ITS Services: A Case Study on Air Pollution Monitoring*.

2246 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 19, NO. 7, JULY 2018

High-Throughput Infrastructure for Advanced ITS Services: A Case Study on Air Pollution Monitoring

José M. Cecilia, Isabel Timón, Jesús Soto, José Santa, Fernando Pereñíguez, and Andrés Muñoz

***Abstract*— Novel cooperative intelligent transportation systems (ITS) serve as the basis for the provision of a number of services for drivers, occupants, and third parties. The vast amount of information to be collected, especially in vehicle-to-infrastructure (V2I) communication services, requires new algorithms and hardware platforms to cope with real-time requirements; however, this combination is not properly addressed in the literature. In this paper, we introduce a high-throughput hardware–software infrastructure to gather information from vehicles and efficiently process it to provide novel ITS services. We propose a parallelization approach of a fuzzy clustering technique on heterogeneous servers based on CPU and several GPUs, tailored to classification problems in V2I. The infrastructure is empirically tested to offer a geo-located pollution information service through the periodical collection of both vehicle's position and status data. We offer a real service that correctly identifies highly polluting traffic areas and drivers. The results indicate a good performance of the system under high loads, and our scalability analysis reveals a good operation in real-ambitious deployments thanks to the use of the both CPU and multiple GPUs, showing that our proposal can efficiently host cooperative services involving high processing in the ITS context.**

***Index Terms*— Pollution monitoring, fuzzy clustering, HPC, heterogeneous computing, V2I, intelligent transport systems.**

I. Introduction

ADVANCES in information and communication technologies have enabled the integration of novel services in road transportation. Intelligent Transportation Systems (ITS) pioneer contributions towards the application of information and communication technologies in vehicles to improve safety by means of sensors, actuators and embedded computers. In the last decade, advances in communication technologies have fostered cooperative systems. Cooperative ITS allow vehicles to communicate with other vehicles (V2V) and with the infrastructure (V2I), opening the development of a broad set of novel services [1]. The era of Internet of Things (IoT) has been recently assumed by the cooperative ITS community as a way to achieve a smarter exchange of information by connecting different entities such as cars, physical devices, roads or buildings with computing servers, sensors and actuators [2].

Within the cooperative ITS scenario, vehicles can generate valuable information to enable different services to improve the quality of traffic, driver's experience and even provide tools for environment protection. A clear example of the latter would be the provision of pollution parameters to third-party services and authorities to assure the air quality in cities. Indeed, this is a hot topic nowadays as indicated in a recent study made by top scientific advisers of the European Commission [3]. However, these ITS services require a communication and processing infrastructure capable to analyze the large amount of information generated by traffic entities. Therefore, cooperative ITS infrastructures are called to be redesigned and even be rethought in order to offer Big Data analytics in real-time.

Machine learning [4] has become essential for the predictive analysis of data deluge, but high performance computing (HPC) plays an equally important role, particularly when real-time response is crucial [5]. Machine learning methods can provide good solutions in a reasonable time frame, which is a key added value to common vehicle-to-infrastructure ITS services. Moreover, most of the machine learning techniques are inherently parallel by definition and therefore they are well-suited for parallelization on current HPC architectures [6]. HPC computing architectures are based on two different approaches: low-power processors for embedded and on board devices with limited computing power, and high throughput computing servers that rely on heterogeneous computing, mainly based on the CPU and accelerators like Graphics Processing Units (GPUs). Indeed, the intersection between big data algorithms and HPC is crucial when large and complex data sets need to be computed, stored and analyzed in a timely manner under highly scalable environments.

This paper presents a novel platform to efficiently gather information from vehicles and process it through a paralleled approach. It merges machine learning and parallel computing techniques to cover the ITS field and further support V2I services with recommendation and classification capabilities. As a proof of concept, in this paper we deploy the infrastructure to cover a geo-located air quality service that identifies polluting traffic areas and driving behaviors with a high carbon footprint. In addition, the generic nature of the solution would also allow

Manuscript received June 12, 2017; revised December 18, 2017; accepted March 10, 2018. Date of publication April 9, 2018; date of current version June 28, 2018. This work was supported by the Spanish Ministry of Economy and Competitiveness through the Project HETEROLISTIC (AEI/FEDER, UE) under Contract TIN2016-78799-P. The Associate Editor for this paper was C. Sommer. *(Corresponding author: José Santa.)*

J. M. Cecilia, I. Timón, J. Soto, and A. Muñoz are with Escuela Politécnica, Universidad Católica de Murcia, 30107 Murcia, Spain (e-mail: jmcecilia@ucam.edu).

J. Santa is with the Department of Information and Communications Engineering, University of Murcia, 30100 Murcia, Spain (e-mail: josesanta@um.es).

F. Pereñíguez is with the Department of Sciences and Informatics, University Centre of Defence, Spanish Air Force Academy, 91751 San Javier, Spain.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2018.2816741

the integration of other services requiring the processing of high volumes of data, such as traffic jam avoidance, dynamic route planning, parking and, in general, other crowd sensing schemes with the aim of providing smart mobility capabilities. Hence, the major contributions of this paper are:

1) A hardware-software infrastructure able to cope with the big data processing needs of high performance computing V2I services in real-time.
2) A machine learning method for fuzzy classification on heterogeneous computing architectures has been adapted to be deployed on multiple CPUs and GPUs by using OpenMP and CUDA at ITS back offices.
3) Several load balancing strategies are evaluated to pursue the performance when using high-demanding services in a cooperative ITS scenario.
4) Two pollution-related services have been implemented and evaluated for vehicle fuel-consumption monitoring and an air quality assurance, including a data gathering campaign to collect diagnosis data.
5) Finally, a performance evaluation analyses the scalability of the solution when increasing data flows, thus involving a high numbers of vehicles.

The rest of the paper is structured as follows. Section II provides a background about the main areas covered by this work. Related works are reviewed in Section III. Section IV introduces the general infrastructure to provide high-performance computational services in ITS, while Section V exemplifies its application to implement a traffic-related air pollution monitoring as a case study. Section VI describes the experimental methodology along with performance and quality evaluations. Finally, Section VII includes conclusions and future directions.

II. Background

A. Cooperative ITS

Services in cooperative ITS are usually categorized in the next groups [7]: safety, involving services intended to reduce accidents and safeguard vehicle occupants and pedestrians; traffic efficiency, dealing with the improvement of the road network capacity and reduce the travel time; and infotainment, mainly oriented to provide value-added comfort services, Internet access and multimedia.

A recent service set closely related to traffic efficiency is *green transport*, given the attention that pollution is receiving lately. For this reason, novel ITS services have started to consider the collection of pollution parameters to better inform authorities of high levels of COx and NOx gases. At the moment, air measurements are taken by weather stations installed at key points in cities, but future advances consider collecting pollution information from vehicles in real-time. Some contributions in this field bet on the integration of new sensors and special hardware to measure NO and CO gases, among other parameters, such as the works in [8] and [9]. On the contrary, another research line focuses on the usage of currently available features of cars to gather pollution parameters. This is the case of using an On-Board Diagnosis (OBD) interface to obtain such data, as can be seen in the works [10], [11]. In [12], this strategy is combined with the use of digital maps to geo-locate pollution issues, addressing the same objective than our case of study later discussed.

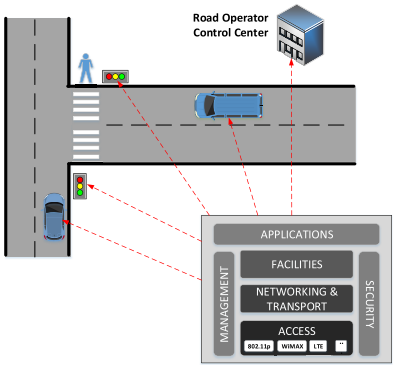


Fig. 1. ITS Station Reference Architecture (ITS-SRA).

In scenarios such as pollution monitoring, where large volumes of data are collected, there are performance issues that should be considered, involving efficient (vehicular) communications, new-age data processing algorithms adapted to deal with imprecise and incomplete data, and high performance computing platforms to cover real-time requirements.

B. ITS Communication Architecture

The development of ITS services has received a decisive impulse thanks to the definition of a common ITS communications architecture. This architecture is intended to be valid for a variety of communication scenarios (vehicle-based, roadside-based and Internet-based) through a diversity of access technologies (802.11p, infra-red, 2G/3G, satellite, etc.) and for a variety of application types. This common communication architecture is known as the *ITS Station Reference Architecture* (ITS-SRA) and is specified by ISO in [13] and by ETSI in [14]. The ITS-SRA establishes both the types of entities and an unified communications stack.

Regarding the entities participating in the implementation of vehicular services, the ITS-SRA identifies three main categories of *ITS Stations* (ITS-S): *vehicle ITS-S*, *roadside ITS-S* and *central ITS-S*. The last one identifies all the equipment located in the road operator's network infrastructure back-end and receives a particular attention in this paper. With respect to the communications stack, as depicted in Fig. 1, the ITS-SRA includes four horizontal layers surrounded by two vertical layers. On the top of the stack the *application layer* implements ITS application services like those related to avoiding collisions or achieving green transport. Middleware is included in the *facilities layer*, while networking modules are embedded in the *networking & transport layer*. Finally, the *access layer* brings together adaptation and medium access. Here it is worth mentioning 802.11p, an extension to the basic 802.11 that represents the facto solution for short-range vehicular communications. Additionally, internal processes and secure operations are supported by the *management* and

security layers, respectively. This stack is implemented partially or totally by ITS stations, depending on its role and requirements.

C. Processing Large Data-Sets for ITS Services

Emergent ITS-based applications are characterized by processing large data sets collected from different sources such as vehicles, citizens, sensors or even information generated by companies or individuals on the Internet. The analysis of these data within an administrative domain like a city has become increasingly relevant to identify new insights and offer novel solutions for upcoming problems [15]. Indeed, these large data sets must be usually processed with real-time requirements. Therefore, High Performance Computing (HPC) is becoming mandatory at the early stages of the infrastructure development to deal with such requirements.

The HPC market is witnessing the steady transition from homogeneous to heterogeneous computing systems [16], where nodes combine traditional multicore architectures (CPUs) and accelerators (mostly represented by GPU computing movement or Intel Xeon Phi cards). In heterogeneous computing systems, programmers play a fundamental role, as they might have to redesign applications to maximize performance with parallelism as a mandatory ingredient. In a few years, the GPGPU (General Purpose application on GPUs) field has grown very fast and evolve into one of the best ways of achieving high performance computing from novel heterogeneous platforms. Indeed, Compute Unified Device Architecture (CUDA) is a leading exemplar of GPGPU, offering a platform for GPUs that comprises both software and hardware.

Even with those great advances in the HPC fields, the amount of data to be processed grows exponentially each year in the big data era. Therefore, data analysis need to be performed with techniques that do not require large computational times. Actually, ITS-based applications do not require precision in many scenarios, but they may include imprecision by definition, since it depends on several factors such as communication and sensors reliability. Soft Computing methods are algorithms that include imprecision into the calculation on one or more levels [17]. The imprecision is targeted by these techniques either by decreasing the problem granularity or by "softening" the goal of optimization at some stage. Soft Computing uses natural processes as a role model and, at the same time, aims to formalization of these tasks like humans perform cognitive decisions in their daily activities. Moreover, soft computing algorithms are inherently parallel by its definition, and the combination of HPC and Soft Computing is mandatory to provide new applications for smart environments, and particularly, to successfully develop ITS services.

III. Related Work

Given the broad spectrum of potential services and technologies involved in a system like the one presented in this paper, related works must be considered from different viewpoints. Starting from the technological issues when collecting data from vehicles, we must take into account that this problem is not new. For instance, the work in [18] proposes a system to gather OBD information from vehicles by using 3G networks. This work also includes an intelligent system to identify potential mechanical problems in vehicles on the basis of the previous experience. In [19], the problem of vehicle monitoring is solved using IoT application protocols. In [20], pollution information is gathered from especial sensors installed in vehicles with Radio Frequency Identification (RFID) support. However, these works have in common that the computational resources required for the operation of these services under real settings are not considered. The system proposed in the current paper gathers status information from vehicles (including OBD data) but, as key differences with the works cited in this section, it includes a communication link based on recent advances in vehicular networks, a generic intelligent classification algorithm for ITS applications, and a parallelism strategy to overcome the computational problems of processing huge amounts of data.

Regarding proposals aligned with the aim of processing traffic data to provide ITS services, Attanasi *et al.* [21] tackle the performance problem of route guidance and traffic forecasting. To this end, a parallel version of the A* algorithm and a propagation model are used, and experiments involving several threads are performed obtaining significant gains with respect to the sequential executions. Xia *et al.* [22] propose a framework to offer real-time traffic parameters. To achieve this, authors propose a model based on DempsterShafer (D-S) and rough set theories to fuse the heterogeneous data gathered, and two types of parallelization are used: algorithm-centric and data-centric. While this work focuses on the efficient fusion of data from traffic sensors, our proposal only uses vehicle sensors without the need of external infrastructure. Tyagi *et al.* [23] introduce a statistical framework that uses the cumulative acoustic signal from a roadside microphone to classify the vehicular traffic density state. They use different classification algorithms, such as Support Vector Machine (SVM) and Bayes, but they focus on evaluating the cost and accuracy, and not on the overall system performance.

There are several works in the literature that combine the multi-agent paradigm with high performance computing techniques to offer real-time ITS services. Thus, Murueta *et al.* [24] propose a multi-agent framework that takes into account multiple sources of data to recommend the best routes. Agents implement a parallel version of the K shortest path algorithm outperforming the sequential execution. In [25] a multi-agent architecture is presented to offer cooperative routing services. However, data for the evaluation of the architecture is simulated, both for traffic sensors and for OBD data. In [26] it is introduced the use of a cloud environment as an alternative for achieving high computational performance in a multi-agent and multi-layer framework. Following the big data paradigm, authors use the Infrastructure as a Service (IaaS) approach and resources are provided on-demand according to the computational requirements of agents, although no performance evaluation is shown in the paper. While all these works rely on distributing the data and the algorithms among agents, our proposal gathers the

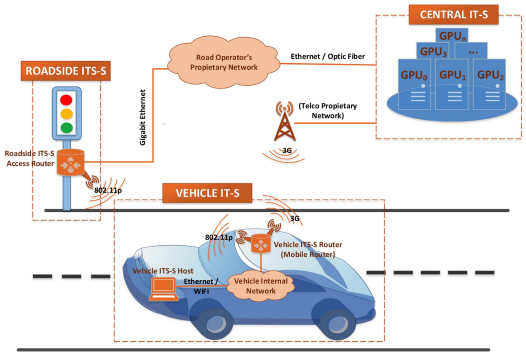


Fig. 2. Proposed architecture for high-performance ITS services.

data following a client-server model and applies parallelization techniques in the server, also obtaining real-time and efficient ITS services.

Especially focused on traffic pollution, the work in de Penning *et al.* [27] applies deep learning and symbolic reasoning to learn drivers' behavior and help them to reduce $C0_2$ emissions. To this end, their system gathers data from the vehicle's CAN-bus such as speed, gear, steering angle, etc. Ambiguity and errors in the data are dealt with a Bayesian inference model. The work in [28] extends the platform presented in [29] about vehicular sensing, but now considering a cloud and edge computing approach. One of the services evaluated is pollution monitoring and real prototypes for vehicular on-board units are presented and evaluated in detail. However, no computing performance nor parallelization techniques are taken into account. Ir our case we bet on a FC clustering approach, and we apply parallelization to overcome the computing of high volume of data in the back-end system. The advantages of parallelizing this kind of algorithms can be checked in [30], where the system is checked to work precisely while achieving a speedup of $\times 4$ approximately compared with the sequential equivalent. This justifies our approach towards the parallelization of the ITS data classification subsystem.

IV. A Parallel Infrastructure for General ITS Services

This section introduces the overall hardware-software infrastructure proposed in this work to deal with ITS services. First, the main hardware entities and the communication platform are described. Then, the GPU-based fuzzy clustering algorithm used is presented.

A. Overall Infrastructure

There are three main hardware entities in our proposal, as depicted in Fig. 2: *Vehicle ITS-S*, *roadside ITS-S* and *Central ITS-S*. The base communication architecture presented in [31] has been used, which is compliant with the standardized ISO/ETSI reference architecture specifications explained in section II-B. This stack has been augmented with additional IPv6-based protocols to support the mobility of nodes while maintaining Internet connectivity, and it is a perfect frame for infrastructure services.

The *Vehicle ITS-S* entity integrates the on-board networked nodes for accessing the whole network. The functionality in the vehicle is split into two nodes: *vehicle ITS-S host* and *vehicle ITS-S router* (also known as Mobile Router - MR). MR hides networking tasks to in-vehicle hosts and it is responsible for providing connectivity to in-vehicle hosts through a vehicle internal network. An unlimited number of hosts could connect to the vehicle internal network through a common WiFi access (e.g. IEEE 802.11a/b/g) or an Ethernet connection. To maintain external communication with roadside equipment and the road operator control center, we consider the use of 802.11p, to communicate with Roadside ITS stations located in the surroundings, and 3G as backup communication technology for those cases where the 802.11p connectivity is not available. Additionally, GPS enables the vehicle to be geo-located. The MR offers IPv6 mobility of the whole in-vehicle network, while the Vehicle ITS-S Host executes final applications that could access remote services. The *Roadside ITS-S* entity provides local wireless connectivity, acting as network attachment point for vehicles using short/medium-range communication technologies (e.g. 802.11p). As discussed in [31], when using 802.11p, it is obtained peaks of 5 Mbps of bandwidth, less than 10 ms of two-way delay and less than 15 % of packet losses. With 3G (using HSPA), these values decrease to 1 Mbps and 200 ms respectively, but similar packet losses are detected.

The *Central ITS-S* entity brings together all the computing resources available for processing our algorithms to offer novel ITS services. This entity collects data from vehicles to offer valuable services, such as our reference pollution service. It is based on a heterogeneous cluster with multiple CPUs and Nvidia GPUs that are CUDA compatible. It is noteworthy to remark that the *Central ITS-S* is heterogeneous in two different senses; first it includes different types of processing units, i.e. CPUs and multiple Nvidia GPUs; and second, these units may have different computing capabilities or improved instruction set architectures. Thus, our algorithms need to leverage such heterogeneous systems to get optimal performance.

B. The Fuzzy Clustering Algorithm

The software side of our infrastructure is based on a fuzzy clustering algorithm. This is a data-analysis technique aimed at organizing a collection of data-points in a multidimensional space into *clusters* (a.k.a groups). These clusters contain those data-points that are similar to each other depending on a particular metric [32]. There are different data clustering algorithms in the literature that have been applied in different scientific applications [33]. Of particular interest to us is the *Fuzzy Minimals* (FM) clustering algorithm [34] and its parallelization approach, called *Parallel Fuzzy Minimals* (PFM) [35]. FM is an unsupervised algorithm that does not require data-sets to be identified in advance. Besides, the FM algorithm allows clusters to overlap to each other, which is actually an important issue in many data-sets. Bearing this in mind, we now briefly

introduce the FM algorithm that was first described by us in [35].

The FM algorithm is a fixed-point iteration algorithm that minimizes an objective function given by Equations (1) and (2). Algorithm 1 shows the sequential baselines of the FM algorithm. Two input values are included in the computation. ε_1 establishes the error degree committed in the minimum estimation; and ε_2 shows the difference between potential minimums. Next, the r factor for the targeted data-set is calculated. Then, the prototypes are figured out by minimizing the Equation (1).

$$J_{(v)} = \sum_{x \in X} \mu_{xv} \cdot d_{xv}^2, \tag{1}$$

where

$$\mu_{xv} = \frac{1}{1 + r^2 \cdot d_{xv}^2}, \tag{2}$$

Algorithm 1 The Sequential Baselines of Parallel Fuzzy Minimals Algorithm

1: *LoadDataSet*()
2: Choose ε_1 and ε_2 standard parameters.
3: r = FactorRCalculation(dataset)
4: Prototypes = CalculatePrototypes(dataset, r)

The r factor measures the disruption of the homogeneity and isotropy of the sample by a set of factors affecting the Euclidean distance in each group [36]. Actually, clusters are created when the isotropy and/or homogeneity are broken. Equation (3) shows the r factor calculation in a non-linear expression

$$\frac{\sqrt{|C^{-1}|}}{n r^F} \sum_{x \in X} \frac{1}{1 + r^2 d_{xm}^2} = 1, \tag{3}$$

where $|C^{-1}|$ is the determinant of the inverse of the covariance matrix, m is the mean of the sample X, d_{xm} is the Euclidean distance between x and m, and n is the number of elements of the sample.

Algorithm 2 summarizes the prototype calculation procedure. FM looks for the number of prototypes using a data-structure which is actually modeled by r factor in FM's objective function (see Equations (1) and (2)). Equation (2) is the membership function that establish the degree of membership for a given element x to a particular cluster where v is the prototype. Indeed, these prototypes are the FM's output representing the minimum values of the objective function by the classification process.

C. Parallelization Approach

The parallelization of this fuzzy clustering algorithm relies on the fact that FM does not compare clusters among them to minimize the objective function like other fuzzy clustering techniques. Therefore, our algorithm can take advantage of this feature to divide the original data-set into different subsets, in which FM is performed independently. This subset does not lose information about global properties for the classification,

Algorithm 2 *CalculatePrototypes* Function of Fuzzy Minimals Algorithm, Where n Is the Size of the Data-Set. V Is the Algorithm Output That Contains the Prototypes Found by the Clustering Process. F is the Dimension of the Vector Space

1: Initialize $V = \{\ \} \subset \mathbb{R}^F$.
2: **for** $k = 1; k < n; k = k + 1$ **do**
3: $v_{(0)} = x_k$, $t = 0$, $E_{(0)} = 1$
4: **while** $E_{(t)} \geq \varepsilon_1$ **do**
5: $t = t + 1$
6: $\mu_{xv} = \frac{1}{1+r^2 \cdot d_{xv}^2}$, using $v_{(t-1)}$
7: $v_{(t)} = \frac{\sum_{x \in X} \left(\mu_{xv}^{(t)}\right)^2 \cdot x}{\left(\mu_{xv}^{(t)}\right)^2}$
8: $E_{(t)} = \sum_{\alpha=1}^{F} \left(v_{(t)}^{\alpha} - v_{(t-1)}^{\alpha}\right)$
9: **end while**
10: **if** $\sum_{\alpha}^{F} (v^{\alpha} - w^{\alpha}) > \varepsilon_2, \forall w \in V$ **then**
11: $V \equiv V + \{v\}$.
12: **end if**
13: **end for**

since each subset is classified using an objective function that includes r factor to provide information about the overall data-set. Indeed, the r factor is a global parameter that provides information about the whole data-set, therefore this factor is used in all subsets even though they are executed in parallel, keeping the main clustering characteristics of FM unaltered.

The r factor calculation does not require too much execution time compared to the prototype calculation as it will be justified in Section VI. Then, depending on the data-set size and the number of computational devices available on the Central ITS-S, our algorithm divides the data-set into a different number of subsets to enable prototype calculation on the GPUs. Each subset is therefore offloaded to a GPU, although not all GPUs may be used at once. If the data-set is not big enough, the prototype calculation could be developed on the CPU to avoid additional overheads.

With that scenario in mind the computation proceeds following a MapReduced strategy [37]. Firstly, the r factor is calculated using the whole dataset. Then, the algorithm decides the number of computational devices that are eventually involved in the computation. In case of multiple GPUs are required, several CPU threads are created using OpenMP technology to manage each GPU context. In a *homogeneous distribution*, the input data-set is equally distributed among the GPUs that are involved in the computation (map stage). Then, the FM algorithm proceeds on each subset in parallel to obtain a set of prototypes. Those prototypes are merged in the CPU to obtain the final set of prototypes (reduce stage).

The Central ITS-S may have different kinds of devices or even devices within the same family with different computing capabilities as previously explained. Therefore, the execution time of each independent task may differ, as it depends on the underlying GPU each subset runs on, which is actually unknown at compile-time. Given that the slowest GPU will determine the overall execution time, our mission is to make use of the idle time offered by the

most powerful GPUs. Then, we propose a *heterogeneous distribution* where each OpenMP thread calculates the subset size to be classified, whose size is given on the basis of the Dif parameter (see Equation (4)). Here, several runs of our algorithm are performed to empirically determine the performance differences between all computational devices in the Central ITS-S. This is only performed once, and this information is used for the following executions. Section VI provides more details about this experimentation with real evaluations. The main output of this stage are the following parameters: the threshold data-set size for offloading to a GPU, the threshold data-set size to scale to several GPUs, and the computational differences between GPUs. Importantly, at this stage, the algorithm is not trying to solve the problem in any meaningful sense, but these runs allow us to calculate the performance differences between GPUs. According to Equation (4), the slowest GPU will have $Dif = 1$; a GPU two times faster than the slowest GPU would have $Dif = 0.5$, and so on. Finally, the optimum number of threads is also experimentally obtained at this stage to increase the level of parallelism and to maximize processor occupancy.

$$Dif = \frac{Ex.time_{actualGPU}}{Ex.time_{slowestGPU}} \tag{4}$$

V. A Case Study: Traffic-Related Air Pollution Monitoring

The generic ITS infrastructure-based service described in Section IV is evaluated in real services related to the traffic pollution monitoring. Two services have been developed for this infrastructure and a testing campaign has been carried out to assess the operation of the whole system and, especially, the intelligent classification algorithm executed in the parallel computing architecture.

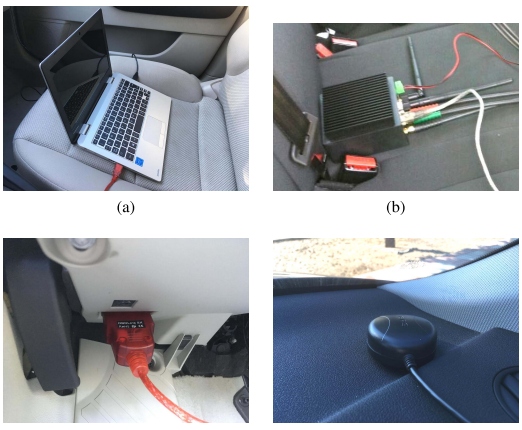
A. Traffic-Related Air Pollution Services

Our study evaluates two different traffic-related air pollution services:

(S1) *Fuel-consumption monitoring*. This service monitors the fuel-consumption for a single car. Once the vehicle arrives to its destination, our service provides information about the fuel-consumption level based on the vehicle's speed, RPM, engine temperature and fuel rate variables. Thus, it is an offline service where the deferred calculation can be accessed by the user.

(S2) *Air-pollution monitoring*. This is a real-time service in the area of green transport, with the aim of ensuring air quality in cities. Based on real-time crowdsourcing information from the fuel-consumption of cars within the same area, our service detects highly polluted areas in the city at the moment.

As a proof of concept, we have implemented both services to identify pollution areas in Murcia (Spain). Traffic-related air pollution is monitored within different locations using data collected from four vehicles. These cars send information, including GPS location (latitude and longitude) and fuel-consumption related variables previously described for service S1. Cars are classified by location first to identify targeted areas, and then by fuel-consumption level (as in service S1) to determine the pollution level of each of those areas.



(a) (b) (c) (d)

Fig. 3. Equipment used in the testbed.

Calculating the carbon footprint or, in general, the air pollution of vehicles implies the estimation of both: (1) the fuel flow, which is carried out using OBD-II data in our case; and (2) the emission factor per unit of fuel consumed. As used in [38], we consider that the emission factor is focused on the CO_2 gas, implying 2.64 Kg of CO_2 per one liter of diesel fuel. CO_2 comprises the most important part of the greenhouse gases emitted (around 77%) according to [38]. For the case of the regular gasoline, this ratio is 2.33 Kg of CO2 per 1 liter of fuel, although in our particular case we are using diesel cars.

B. Deployed Infrastructure

The main entities of the infrastructure described in Section IV are deployed in a testbed where the data collected from vehicles are used to detect pollution areas in Murcia (Spain). The deployment comprises the next set of nodes:

- A Toshiba Satellite CL10-C-102 laptop, depicted in Fig. 3a, with processor Intel Celeron N3050, 2 GB RAM, and Linux Ubuntu as operating system. This computer operates as Vehicle ITS Station Host.
- A Laguna LGN-20 from Commsignia, depicted in Fig. 3b, operating as Vehicle ITS Station Router. This unit provides mobile IPv6 connectivity to the laptop through a WiFi network, and it connects the vehicle network with the Central ITS Station through 3G during the tests.
- A computational back-end acting as Central ITS Station that is described in detail in the next part.

The laptop is connected with the car CAN bus through an OBD-II interface. The OBD scan tool used is OBDLink SX with USB interface, and it is connected with a proper connector available in the car, as showed in Fig. 3c. In the car used (Audi A3 1600 CC TDI) this connector is under the

steering wheel. An external USB GPS has been used to obtain the position of the vehicle, model Adopt SkyTraq Venus 8. It is a mouse model, and it is placed near the windscreen to improve coverage.

OBD data collected from the vehicle comprise a set of all supported parameter identifiers (PID) by the car. Among the most important parameters we can find the Mass Air Flow (MAF) and the Calculated Engine Load ($LOAD_{CALC}$), since they allow us to compute the Fuel Flow (FF), measured in liters per hour. As indicated in [39], the model used to compute the FF is as indicated in equation 5.

$$FF = \frac{CMAF \times 3600}{CAFR \times FD}$$
$$CMAF = MAF \times \frac{LOAD_{CALC}}{100} \quad (5)$$

$CAFR$ is the Corrected Air to Fuel Ratio, which is considered as an optimum (ideal) air to fuel ratio of 14.5g (14.7 g of air to 1 g of fuel), given that the OBD correction parameter to apply to this factor is not available in all cars. FD is the fuel density, which is 832 g/l.

All data are collected from the OBD-II interface and the GPS receiver by using the OBD GPS Logger software, and then they are sent over the IPv6 network by using one UDP datagram per data record gathered at a 1 Hz frequency. GPS information includes time, latitude, longitude and altitude, while the OBD parameters considered comprise 26 numerical values. This record is sent in comma-separated value (CSV) format.

C. Central ITS-S Description

The computational back-end (Central ITS-S) we have used to run the GPU-based fuzzy algorithm algorithm has four Intel Xeon X7550 processors running at 2 GHz and plugged into a quad-channel motherboard endowed with 128 Gigabytes of DDR3 memory. Moreover, two NVIDIA GPUs are connected through PCI-Express bus. A GPU NVIDIA Tesla Kepler K40c with 2880 CUDA cores (15 Streaming Multiprocessors and 192 Streaming Processors per Multiprocessor) running at boost clock of 0.88 GHz, giving a raw processing power of up to 5068 GFLOPS, and having up to 12 GB of GDDR5. A GPU NVDIA Fermi GeForce GTX 580 with 512 CUDA cores (16 Streaming Multiprocessors and 32 Streaming Processors per Multiprocessor) running at boost clock of 1.54 GHz, giving a raw processing power of up to 1581 GFLOPS, and having up to 1.5 GB of GDDR5.

In both platforms, gcc 4.8.2 with the -O3 flag was used for compilation on the CPU, and the CUDA toolkit version 8.0 was used for compilation on the GPU.

VI. Evaluation

This section shows the results obtained testing our parallel infrastructure by targeting the traffic-related air-pollution monitoring described above. The main objective of these experiments is two-fold. Firstly, we analyze the quality evaluation for the two different services described in Section V-A: fuel-consumption monitoring (S1) for one car, and air-pollution monitoring (S2) for several areas. Secondly, we discuss the scalability for the communication infrastructure. Finally, we evaluate the performance of the platform when processing the collected data and consider ambitious scenarios to analyse the expected scalability of the platform.

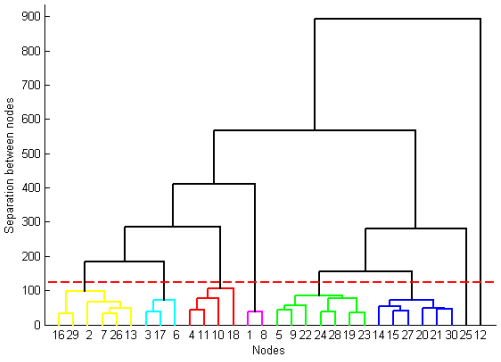


Fig. 4. Dendogram representing the final eight groups (nodes) out from the 68 initial groups obtained from the FM algorithm.

A. Classification Results

1) Service S1: Fuel-Consumption Monitoring: To verify the proper operation of this service, we analyze the results obtained with respect to fuel-consumption level from one vehicle. Firstly, the FM algorithm is applied to 843 samples containing data on car's speed, RPM, engine temperature and fuel rate. As a result, it is obtained 68 prototypes, i.e. 68 different groups. These groups are refined through a dendogram (see Fig. 4), resulting in eight final groups determined by the threshold represented as a red dash line (separation value of 120). This threshold has been found taking into account the number, size and distance among groups. Next, it is calculated the mean value for the fuel rate variable in each of the eight groups and they are sorted in ascending order, giving as a result the following mean values for each group:{0.4297, 0.4758, 0.8315, 0.9145, 1.3228, 2.0262, 3.0798, 4.7392}. We use then a color scale from light blue to red to identify each group, labeling the less fuel-consuming group as light blue and the most fuel-consuming one as red.

Once the groups for fuel-consuming patterns are identified, we can now add this information to any route of our vehicle. Fig. 5 shows a real example of a journey where each localization in the route is colored according to the classified fuel-consuming group. Note that less fuel is consumed at the starting and ending point (leftmost and rightmost parts of Fig. 5, respectively), whereas more fuel consumption is made in the central segment. In this segment, we identify two reductions in fuel consumption near longitude values -0.9 and -0.85, due to stops caused by traffic lights and crossroads.

A complementary view of this information is shown in Fig. 6. Here, each data sample in the X-axis is classified into one of the eight fuel-consumption groups along with the specific fuel rate in the Y-axis. The first 300 samples matches

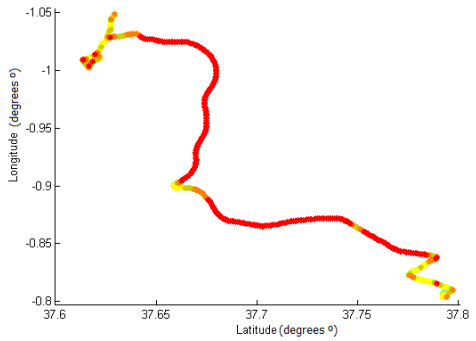


Fig. 5. Fuel consumption information during a real car journey, starting at longitude −1.01 and latitude 37.62, and ending at longitude −0.8 and latitude 37.79.

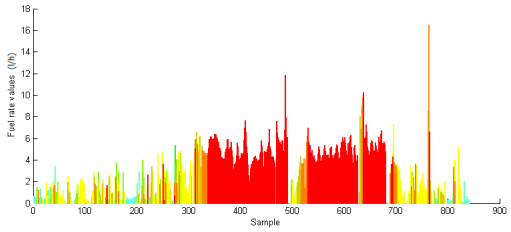


Fig. 6. Fuel rate values for each sample, classified in colors according to its fuel consumption group identified by the FM algorithm.

the starting segment in Fig. 5. Samples from 300 to 700 match the central segment, where more fuel consumption is made, and samples from 700 match the ending segment. It is worth mentioning the low fuel rate values (and fuel consumption levels) around samples 500 and 650, corresponding with the two fuel reductions in longitude values −0.9 and −0.85 observed in Fig. 5.

2) Service S2: Air-Pollution Monitoring: The correct operation of this service has been evaluated through the analysis of 4100 samples obtained from four vehicles in order to monitor traffic-related air-pollution in some specific areas. These samples contain the same data obtained from the vehicle as in service S1, namely vehicle's speed, RPM, engine temperature and fuel rate, along with GPS coordinates (latitude and longitude).

Firstly, the FM algorithm is applied to the streaming of 4100 samples to automatically determine the areas to be monitored according to the GPS data. Once an area is determined, a second execution of the FM algorithm is applied to the fuel-consumption related variables as in the first scenario, but in this case to the data of all vehicles within each different area. In this manner, different number of fuel-consumption levels are detected in each area and then they are converted to traffic-related air-pollution levels as explained in Section V, using the same color scale as in the first scenario. In particular, eight air-pollution levels were detected in Area A, six levels in Area B and seven levels in Area C.

Regarding Area A (see Fig. 7a), there are three lanes with similar air-polluting levels. There is an increase of this level near the roundabout (in the central segment of the area, between longitude values −1.1155 and −1.116), where a larger vehicle concentration is expected. On the other hand, in Area B (see Fig. 7b) the most interesting information is located at the bottom left corner, representing the area entrance and exit lanes. The entrance lane shows the higher air-pollution level in the area, due to slow incorporation and slow traffic flow. In contrast, the exit lane does not present this problem and no high fuel consumption is detected. Finally, in Area C (see Fig. 7c) we find one-direction lane with high pollution levels in the central segment (between latitude values 38.0035 and 38.0055), due to a crossroad causing a slow traffic flow and therefore a larger concentration of vehicles.

B. Scalability Issues in the Communication Infrastructure

Our system is able to combine several wireless technologies in order to maximize the performance in terms of quality of service and cost. 3G/4G networks have evolved in the last years and they are not considered as a potential bottleneck in V2I applications like the one supported by the presented platform. Currently, operators are able to serve hundreds of users in a limited space, by increasing the density of base stations. A more challenging scenario is presented by the incipient 802.11p technology, when used exclusively by many cars within the coverage of a sole base station. According to the experimental work in [40], ten nodes transmitting 200-byte data packets at a frequency of 50 Hz could collapse the medium and obtain packet delivery ratios of less than 20%. Bearing in mind our pollution monitoring scenario, with a data rate of 1 Hz, we could estimate that about 500 nodes could be supported under the same base station. This value should be adjusted by the extra size of our packets of 293 bytes (including our 26 numerical fields in CSV format), which could cause a slight reduction in performance, and the mobility conditions, which would imply more losses, as experienced in our previous works [31].

This rationale envisages a good potential value of vehicle density, which by far exceeds the real density that could be achieved in a real scenario considering experimental coverage ranges of 500 meters, assuming a mean vehicle length of 5 meters and the worst case where the 802.11p station is covering a road intersection. Hence, it could be stated that even when exclusively using the 802.11p vehicular communication technology, the network infrastructure is not considered an issue for the operation of the system presented here, supposing a proper deployment of base stations. Although the 802.11p medium were used by other applications at the same time, the low requirements of the pollution monitoring services presented would allow its correct operation.

C. Performance Evaluation Under High Loads

This section introduces the performance evaluation and scalability of our GPU-based fuzzy algorithm running on the heterogeneous CPU-GPU Central ITS-S previously described

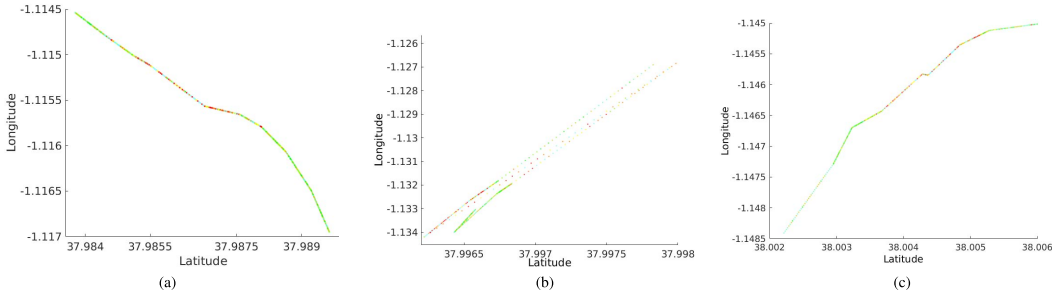


Fig. 7. Pollution levels detailed for the three areas.

in Section V-C. To deeply analyze the behavior of our implementation, we use synthetic data-sets that contain the same type of information previously described in Section V, but adding random numbers to either increase the number of rows or number of columns for the input data-set. In this manner, the performance behavior of our infrastructure in hypothetical scenarios can be analyzed.

Algorithm 1 shows that the main functions of the FM algorithm are $FactorRCalculation()$ and $CalculatePrototypes()$. There are up to three different parameters that affect performance of these functions, all of them related to the data to be classified.

- The *number of rows* that determines the number of inputs to be classified. For instance, in the fuel-consumption scenario, one car is submitting (or storing in a buffer to submit afterwards) information depending on recording of the OBD-II interface and the GPS receiver frequency (1 Hz in our case). In the air-pollution scenario, each row represents a car location, that is sent along with OBD-II information, and therefore there would be as many rows as cars submitting information to our infrastructure.
- The *number of columns* that determines the number of variables to be considered for classification. As previously explained in Section V, all data is collected from the OBD-II interface and the GPS receiver by using the OBD GPS Logger software. GPS information includes up to four different variables, namely time, latitude, longitude and altitude, while the OBD variables reach up to 26 numerical values including the Mass Air Flow (MAF) and the Calculated Engine Load ($LOAD_{CALC}$).
- Finally, the last performance parameter is the *number of prototypes* found by our GPU-based fuzzy algorithm. Our fuzzy algorithm performs a blind clustering search, not depending on the input information; indeed, it depends on the shape of data and on the input error values (ε_1 and ε_2, see Algorithm 1 and 2).

With that in mind, we now analyze these main performance factors to let the reader know about the infrastructure's scalability.

Fig. 8 shows the execution times of the $FactorRCalculation()$ and $CalculatePrototypes()$ functions for input data-sets ranging from 900 to 28000 data records from

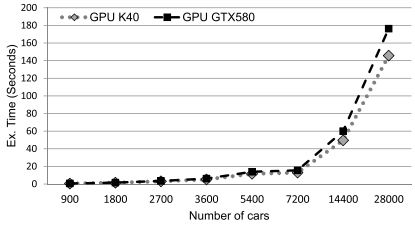


Fig. 8. Execution time in seconds of Fuzzy Minimals algorithm main functions ($FactorRCalculation()$ and $CalculatePrototypes()$) for different input data-sets, varying the number of Cars. We consider the processing time of our algorithm of up to 28000 cars sending information.

cars (i.e., rows). The $FactorRCalculation()$ is executed on the CPU and the $CalculatePrototypes()$ is executed on the GPU as it is data-parallel by its definition. In this case, the number of rows for the input data-set is increased to analyze the scalability. We leave the number of columns (i.e. the variables to be classified) set to two in this experiment. Performance results reveals that $CalculatePrototypes()$ is the main computational bottleneck, as its execution time increases exponentially along with the problem size. It is worth mentioning that $CalculatePrototypes()$ execution time does not depend only on the number of rows or columns like $FactorRCalculation()$, but also on the number of prototypes found in each data-set, which is unknown at runtime. The number of prototypes in the simulation shown in Fig. 8 for the different data-sets are 22, 45, 48, 54, 55, 80, 77 and 101.

Fig. 9 shows the execution times of both functions for different input data-sets, but now varying the number of variables considered in the classification algorithm (i.e., columns). We leave the number of rows set to 5400 in this experiment. In this case, performance results raise different conclusions than in the previous experiment. Increasing the number of variables to be classified affects more the $FactorRCalculation()$ in terms of performance. Let us remind that the r factor calculation includes the determinant of the inverse of the covariance matrix calculation, which is actually of size $number\_of\_columns \times number\_of\_columns$.

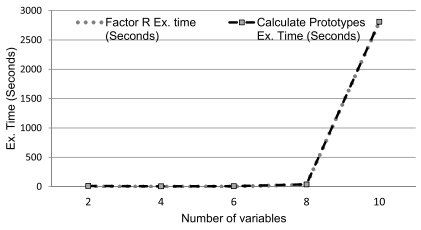


Fig. 9. Execution time in seconds of Fuzzy Minimals algorithm main functions (*FactorRCalculation*() and *CalculatePrototypes*()) for different input data-sets, varying the number of variables to be classified.

Up to six variables the *FactorRCalculation*() is still less time consuming than *CalculatePrototypes*(). However, from that point the *FactorRCalculation*() execution time grows drastically. Although the classification of more than six variables is not trivial at all, here we would recommend a preliminary analysis like Principal Component Analysis (PCA) that uses an orthogonal transformation to transform a set of possibly correlated variables into a set of values of uncorrelated variables called principal components.

The previous performance analysis reveals that our solution could further improve results by using extra nodes computing together, bearing in mind that our algorithm allows extra division of the data-set into more subsets to perform *CalculatePrototypes*() function in parallel. However, this distribution introduces several overheads that should be considered. With such problem division, if we are using several computational devices, at the installation stage it should be performed a load balancing strategy to get peak performance. Fig. 8 reports up to 20% speed-up factor between Tesla k40 and Geforce GTX580 for the *CalculatePrototypes*() function. The installation stage is only performed once and, for the set of benchmarks we are running, it takes around five minutes. This will provide knowledge about the platform in which we are running our experiments and, hence, obtain an indication for load balancing in the platform.

Finally, going into details with the most challenging scenario that we have evaluated, our infrastructure executes the PFM algorithm with 8 variables and 28,000 data rows in 145.57 seconds using only a Nvidia GPU (see Figures 8 and 9). Next, we determine the potential bottleneck of the back-end, where data from hundred or even millions of vehicles could be received. To conduct this evaluation, we will assume a challenging scenario where our air pollution monitoring services are deployed in Madrid, the most crowded city in Spain. According to the reports published by the Spanish Directorate-General of Traffic [41], in 2016 about 4.5 million of vehicles are registered in Madrid. In the worst case where all vehicles are on the road at the same time in the city and using our traffic-related pollution services, the back-end infrastructure will be receiving a traffic of 10.54 Gbps. This value takes into consideration the analysis performed in section VI-B where we have confirmed the capacity of the communication infrastructure to afford our traffic-related pollution services. Recorded variables from cars are 32-bit floats, thus at a 10.54 Gbps data rate, the back-end would receive 329,375,000 variables per second. Since each data row received from a car is composed of 8 variables, this gives us 41,171,875 8-variables data-sets per second. Assuming a system response threshold of 145.57 seconds for our previous worst scenario of 28,000 cars with a single GPU, our system would need 1471 GPUs to provide an 8-variables data-set classification in that time. For instance, the current Top-3 fastest supercomputer, Piz Daint include more than 5320 Nvidia Tesla P100 GPUs (see [42]). This means we are at the technology level to really deploy our infrastructure in a real scenario like Madrid where we would be able to provide air monitoring with a delay of less than three minutes in the worst scenario. This would enable an air pollution monitoring system for a city like Madrid with a refresh rate under 5 minutes.

VII. Conclusions and Future Work

Cooperative Intelligent Transportation Systems have been placed as an emergent field where many novel services will be provided to final users. These services will be based on the analysis of large data-sets generated by a great number of cars. In this paper we propose a high-throughput support infrastructure for advanced ITS services, combining both hardware and software techniques, to provide novel mechanisms that enable exascale data-analysis within this context. All infrastructure layers aimed at enabling a real ITS service based on high performance fuzzy clustering are discussed, including data collection from vehicles, ITS communication architecture to connect vehicles with the infrastructure, and heterogeneous central ITS-S based on CPU and multiple GPUs. An air-pollution monitoring service is tailored to this infrastructure showing that effective information can be provided to drivers and authorities in an effort to reduce the environmental impact.

Our results reveal that realistic services such as pollution monitoring can be developed and operated efficiently within this framework, offering a good performance of the system under high loads, thanks to the use of both CPU and multiple GPUs with appropriated configuration through load balancing techniques. Moreover, our scalability analysis estimates a realistic operation of the framework in ambitious deployments where data coming from millions of vehicles is collected and processed.

ITS services are at a relatively mature stage, however, hardware features included in upcoming heterogeneous processor generations may require different run-time strategies and algorithm redefinition to obtain peak performances in V2I applications. It is expected to get even higher time-response ratios on GPUs whenever new hardware features come along to boost performance into unprecedented gains where parallelism is called to play a decisive role. Indeed, other off-the-shelf big data frameworks like Hadoop or Spark will also offer a good scalability that are well-suited to be combined with our approach. Moreover, we may anticipate that the benefits of our approach would be also combined with other soft computing techniques like deep learning to make it extensive to other ITS services that require data-intensive analysis.

REFERENCES

[1] M. Annoni and B. Williams, *The History of Vehicular Networks*. Cham, Switzerland: Springer, 2015, pp. 3–21.

[2] O. Kaiwartya *et al.*, "Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects," *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

[3] J. Bujnicki *et al.*, "Closing the gap between light-duty vehicle real-world CO_2 emissions and laboratory testing," European Comission, Directorate-General for Res. Innov., Brussels, Belgium, Tech. Rep. 1/2016, Nov. 2016.

[4] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2016.

[5] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics," *J. Parallel Distrib. Comput.*, vol. 74, no. 7, pp. 2561–2573, 2014.

[6] J. M. Cecilia, J. M. García, A. Nisbet, M. Amos, and M. Ujaldón, "Enhancing data parallelism for ant Colony Optimization on GPUs," *J. Parallel Distrib. Comput.*, vol. 73, no. 1, pp. 42–51, 2013.

[7] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions*, document ETSI TR 102 638, European Telecommunications Standards Institute, Jun. 2009.

[8] J. May, D. Bosteels, and C. Favre, "An assessment of emissions from light-duty vehicles using PEMS and chassis dynamometer testing," *SAE Int. J. Eng.*, vol. 7, no. 3, pp. 1326–1335, 2014.

[9] M. Kousoulidou, "Use of portable emissions measurement system (PEMS) for the development and validation of passenger car emission factors, " *Atmospheric Environ.*, vol. 64, pp. 329–338, Jan. 2013.

[10] S. Zhang *et al.*, "Real-world fuel consumption and CO_2 (carbon dioxide) emissions by driving conditions for light-duty passenger vehicles in China, " *Energy*, vol. 69, pp. 247–257, May 2014.

[11] A. Alessandrini, F. Filippi, and F. Ortenzi, "Consumption calculation of vehicles using OBD data," in *Proc. 20th Int. Emission Inventory Conf.*, 2012, pp. 1–17.

[12] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "GreenGPS: A participatory sensing fuel-efficient maps application," in *Proc. ACM 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, 2010, pp. 151–164.

[13] *Intelligent Transport Systems—Communications Access for Land Mobiles (CALM)—Architecture*, Standards ISO 21217:2014, International Organization for Standardization, Apr. 2013.

[14] *Intelligent Transport Systems (ITS); Communications Architecture*, document ETSI EN 302 665, ETSI TC ITS, European Telecommunications Standards Institute, Sep. 2010.

[15] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[16] K. Roy, B. Jung, D. Peroulis, and A. Raghunathan, "Integrated systems in the more-than-Moore era: Designing low-cost energy-efficient systems using heterogeneous components," *IEEE Des. Test*, vol. 33, no. 3, pp. 56–65, Jun. 2016.

[17] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Commun. ACM*, vol. 37, no. 3, pp. 77–85, 1994.

[18] S.-H. Chen, J.-F. Wang, Y. Wei, J. Shang, and S.-Y. Kao, "The implementation of real-time on-line vehicle diagnostics and early fault estimation system," in *Proc. 5th Int. Conf. Genet. Evol. Comput.*, Aug. 2011, pp. 13–16.

[19] R. Dhall and V. K. Solanki, "An IoT based predictive connected car maintenance approach," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 4, no. 3, pp. 16–22, 2017.

[20] S. Manna, S. S. Bhunia, and N. Mukherjee, "Vehicular pollution monitoring using IoT," in *Proc. Int. Conf. Recent Adv. Innov. Eng. (ICRAIE)*, May 2014, pp. 1–5.

[21] A. Attanasi, E. Silvestri, P. Meschini, and G. Gentile, "Real world applications using parallel computing techniques in dynamic traffic assignment and shortest path search," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst.*, Sep. 2015, pp. 316–321.

[22] Y. Xia, X. Li, and Z. Shan, "Parallelized fusion on multisensor transportation data: A case study in CyberITS," *Int. J. Intell. Syst.*, vol. 28, no. 6, pp. 540–564, 2013.

[23] V. Tyagi, S. Kalyanaraman, and R. Krishnapuram, "Vehicular traffic density state estimation based on cumulative road acoustics," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1156–1166, Mar. 2012.

[24] P. O. P. Murueta, C. R. C. Pérez, and J. A. R. Uresti, "A parallelized algorithm for a real-time traffic recommendations architecture based in multi-agents," in *Proc. 13th Mexican Int. Conf. Artif. Intell.*, Nov. 2014, pp. 211–214.

[25] M. Fiosins, B. Friedrich, J. Görmer, D. Mattfeld, J. P. Müller, and H. Tchouankem, "A multiagent approach to modeling autonomic road transport support systems," in *Autonomic Road Transport Support Systems*, T. L. McCluskey, A. Kotsialos, J. P. Müller, F. Klügl, O. Rana, and R. Schumann, Eds. Cham, Switzerland: Springer, 2016, pp. 67–85.

[26] L. Heilig, R. R. Negenborn, and S. Voß, "Cloud-based intelligent transportation systems using model predictive control," in *Proc. Int. Conf. Comput. Logistics*, 2015, pp. 464–477.

[27] L. de Penning, A. S. D'Avila Garcez, L. C. Lamb, A. Stuiver, and J. J. C. Meyer, "Applying neural-symbolic cognitive agents in intelligent transport systems to reduce CO_2 emissions," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 55–62.

[28] R. Tse and G. Pau, "Deployment of vehicular edge clouds: Lessons and challenges," in *Proc. 8th Int. Conf. Digit. Image Process. (ICDIP)*, Aug. 2016, pp. 1–6.

[29] G. Pau and R. Tse, "Challenges and opportunities in immersive vehicular sensing: Lessons from urban deployments," *Signal Process., Image Commun.*, vol. 27, no. 8, pp. 900–908, 2012.

[30] M. Vadiveloo, R. Abdullah, M. Rajeswari, and A. A. Abu-Shareha, "Image segmentation with cyclic load balanced parallel fuzzy C—Means cluster analysis," in *Proc. IEEE Int. Conf. Imag. Syst. Techn. (IST)*, May 2011, pp. 124–129.

[31] P. J. Fernandez, J. Santa, F. Bernal, and A. F. Skarmeta, "Securing vehicular IPv6 communications," *IEEE Trans. Depend. Sec. Comput.*, vol. 13, no. 1, pp. 46–58, Jan. 2016.

[32] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Reading, MA, USA: Addison-Wesley, 2006.

[33] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[34] A. Flores-Sintas, J. Cadenas, and F. Martin, "A local geometrical properties application to fuzzy clustering," *Fuzzy Sets Syst.*, vol. 100, no. 1, pp. 245–256, 1998.

[35] I. Timón, J. Soto, H. Pérez-Sánchez, and J. M. Cecilia, "Parallel implementation of fuzzy minimals clustering algorithm," *Expert Syst. Appl.*, vol. 48, pp. 35–41, Apr. 2016.

[36] A. Flores-Sintas, J. M. Cadenas, and F. Martin, "Detecting homogeneous groups in clustering using the Euclidean distance," *Fuzzy Sets Syst.*, vol. 120, no. 2, pp. 213–225, 2001.

[37] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[38] H. Hilpert, L. Thoroe, and M. Schumann, "Real-time data collection for product carbon footprints in transportation processes based on OBD2 and smartphones," in *Proc. 44th Hawaii Int. Conf. Syst. Sci.*, Jan. 2011, pp. 1–10.

[39] V. Ribeiro, J. Rodrigues, and A. Aguiar, "Mining geographic data for fuel consumption estimation," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 124–129.

[40] K. Mori, O. Shagdar, S. Matsuura, M. Tsukada, T. Ernst, and K. Fujikawa, "Experimental study on channel congestion using IEEE 802.11p communication system," in *Proc. IPSJ Tech. Workshop Mobile Comput. Ubiquitous Commun.*, 2013, pp. 1–7.

[41] (2017). *Directorate-General of Traffic. Vehicle Fleet Annual Directory. 2016*. Accessed: Oct. 10, 2017. [Online]. Available: http://www.dgt.es/

[42] (2017). *Top 500 Supercomputer Site*. Accessed: Nov. 15, 2017. [Online]. Available: http://www.top500.org/

José M. Cecilia received the B.S. degree from University of Murcia in 2005, the M.S. degree from Cranfield University in 2007, and the Ph.D. degree from University of Murcia in 2011, all in computer science. He is currently an Associate Professor with Universidad Católica de Murcia. His research interests include heterogeneous architecture and bio-inspired algorithms.

Isabel Timón received the degree in mathematics from University of Murcia in 2011. She is currently pursuing the Ph.D. degree with Universidad Católica de Murcia with a focus on high performance fuzzy clustering techniques for big data.

Jesús Soto received the M.Sc. degree in mathematics from University of Murcia in 1993 and the Ph.D. degree in applied mathematics from University of Alicante in 2005. He is currently an Associate Professor with Universidad Católica de Murcia. His research interests include fuzzy clustering.

José Santa received the M.Sc. degree in computer engineering, the M.Sc. degree in advanced information and telematics technologies, and the Ph.D. degree in computer science from University of Murcia in 2004, 2008, and 2009, respectively. He is currently a Senior Research Fellow with University of Murcia. His research interests include ITS, mobile services, and networks.

Fernando Pereñíguez received the M.Sc. degree in computer engineering and the Ph.D. degree in computer science from University of Murcia, in 2007 and 2011, respectively. He is currently an Assistant Professor with University Centre of Defence, Spanish Air Force Academy. His research interests include security, privacy, and handoff management in mobile networks.

Andrés Muñoz received the Ph.D. degree in computer science from University of Murcia in 2011. He is currently a Senior Lecturer with Universidad Católica de Murcia. His main research interests include multi-agent systems, applications in intelligent systems, semantic Web technologies, and ambient intelligence.

Capítulo 3

Conclusiones y líneas de trabajo futuras

Esta tesis doctoral se desarrolla dentro de un marco innovador de trabajo para dar respuesta a la inminente necesidad de analizar grandes cantidades de datos en tiempo real. Para ello, se ha propuesto una investigación multidisciplinar que engloba diferentes disciplinas computacionales; estas son la algoritmia paralela, la computación de altas prestaciones y la ciencia de los datos.

La tesis se presenta en modalidad de compendio de publicaciones donde cada uno de los artículos científicos corresponde, en mayor o menor medida, a la consecución de los objetivos planteados. Asimismo, abre las puertas al desarrollo de nuevas investigaciones dentro de cada una de las áreas planteadas, así como en la intersección de las mismas.

En primer lugar, se realiza un amplio análisis de las técnicas de *machine learning* y en el artículo 2.2 se describen algunas de ellas que, posteriormente, fueron aplicadas a la predicción de los niveles de O_3, uno de los parámetros de contaminación atmosférica más perjudiciales.

Tal y como se establece en la publicación, entre las principales cuestiones que actualmente merecen atención en los países desarrollados se encuentra la concentración de la contaminación en las zonas urbanas que, entre otros

problemas, causa enfermedades crónicas. De hecho, el análisis de contaminantes como SO_2, NO_x, O_3, NH_3, CO y PM_{10} a través de técnicas de aprendizaje automático puede ayudar a predecir picos de contaminación así como a establecer umbrales y planes de acción para autoridades locales, conductores, industrias, etc.

Las técnicas estudiadas en este trabajo fueron *Random Forest*, *Decision Tree*, *Random Committee*, *Bagging* y KNN. Se concluyó que la técnica que obtenía el mejor ajuste en general es *Random Forest*, siendo esta afirmación validada por pruebas estadísticas. Los resultados indicaban un ajuste de R^2 entre el 80 y el 90 % en total y un error de predicción de O_3 inferior a 11 μ/m^3.

Ahondando en el campo del aprendizaje automático se plantea el trabajo 2.3, en el que se propone una técnica de discretización basada en la agrupación difusa. Esta técnica se basa en el algoritmo μFCM para realizar una discretización numérica utilizando la distribución Cauchy. Para evaluar la técnica de discretización se aplicó un clasificador ELM y un comité de clasificadores ELM. Estos clasificadores también se aplicaron para evaluar la discretización obtenida para la técnica de KM. Dicha evaluación permitió verificar que, en términos generales, se obtuvieron resultados satisfactorios con 3 particiones en cada atributo, lo que aportaba una gran interpretabilidad de los resultados, gracias a que la técnica permite transformar valores continuos para cada atributo en valores discretos a través de 3 particiones. Por lo tanto, desde un punto de vista cualitativo, los resultados fueron bastante interesantes. Además, los resultados de la comparación entre las técnicas de μFCM y KM se validaron estadísticamente, donde la técnica propuesta (μFCM) obtuvo los mejores resultados con un 97 % de nivel de confianza.

En la publicación 2.4 se lleva a cabo el diseño e implementación paralela de un algoritmo de *clustering* difuso (FM) eficiente y escalable. Redefinimos este algoritmo para mejorar la clasificación de grandes conjuntos de datos. Nuestros resultados experimentales revelaron una aceleración lineal en relación al número de procesos paralelos lanzados, mientras que la calidad de la clasificación seguía siendo suficientemente buena.

Una vez propuesto el algoritmo de *clustering* difuso, en el artículo 2.5 se

desarrolló una infraestructura de soporte de alto rendimiento para servicios ITS avanzados, combinando técnicas de *hardware* y *software*, para proporcionar mecanismos novedosos que permitieran el análisis de datos a gran escala dentro de este contexto. En este artículo se paraleliza el algoritmo FM en GPUs para satisfacer los requisitos temporales establecidos. Se adapta un servicio de vigilancia de la contaminación atmosférica a esta infraestructura como caso de éxito, demostrando que se puede proporcionar información eficaz a los conductores y a las autoridades en un esfuerzo por reducir el impacto medioambiental.

Nuestros resultados revelaron que servicios de tiempo real como la monitorización de la contaminación podían ser desarrollados y operados eficientemente dentro de este marco, ofreciendo un buen rendimiento del sistema bajo altas cargas, gracias al uso tanto de CPU como de múltiples GPUs. Además, el análisis de escalabilidad estimó un buen funcionamiento a tiempo real en servicios que recogen y procesan datos procedentes de millones de vehículos.

Ya fijado el punto de partida de la investigación en algoritmos de clasificación, se observa que el *Big Data* tiene el potencial de transformar el desarrollo y acelerar el progreso social en todo el mundo. Sin embargo, hay varias cuestiones en torno a esta nueva era que deben abordarse antes de dar un paso más allá. Entre ellas, cabe destacar la falta de recursos computacionales disponibles para hacer frente a este diluvio de datos. Hoy en día, el campo computacional es masivamente paralelo como consecuencia de la nueva tendencia en el desarrollo de nuevos microprocesadores. Este hecho hace obligatoria la redefinición de nuestros algoritmos para aprovechar al máximo las nuevas plataformas masivamente paralelas.

3.1 Líneas de trabajo futuras

En esta tesis doctoral se ha estudiado y diseñado un algoritmo de clasificación difusa y ha sido implementado en entornos *Big Data*, como es el área de las *smart cities*. Esta aplicación nos presenta diversos caminos para la conti-

nuidad del proyecto de investigación.

En primer lugar, se detecta que la gestión y procesamiento de datos en tiempo real supone la adaptación de los algoritmos que existen actualmente a nuevos entornos que lo permitan. Por tanto, una de las líneas de trabajo futuro debería centrarse en estudiar las propiedades intrínsecas de los algoritmos de clasificación para que permitan una implementación de estos en entornos masivamente paralelos, lo que suponga mejoras de la escalabilidad y aceleración en las iteraciones.

En lo relativo a la aplicación de dichos algoritmos, se identifican dos posibles áreas: el campo de las *smart cities* y el de la contaminación.

En el campo de las ciudades inteligentes cabe destacar que los servicios ITS se encuentran en una etapa relativamente madura; sin embargo, las características de *hardware* incluidas en las próximas generaciones heterogéneas de procesadores pueden requerir diferentes estrategias de tiempo de ejecución y redefinición de algoritmos para obtener el máximo rendimiento en las aplicaciones V2I. Se espera que los ratios de tiempo-respuesta de las GPU sean aún mayores cada vez que se incorporen nuevas funciones de *hardware* para aumentar el rendimiento y lograr avances sin precedentes, en los que el paralelismo debe desempeñar un papel decisivo. Una extensión en este área será la generación automática de recomendaciones a las autoridades locales, conductores y otros actores relacionados en la influencia de los factores de calidad del aire de acuerdo a las alertas planteadas.

En el área de la contaminación se resalta el estudio de nuevos parámetros como PM_{10} y SO_2, que también afectan seriamente la calidad del aire. Estos parámetros deben ser analizados y estudiados con el fin de crear modelos que ayuden a predecirlos. Una línea en la que trabajar sería la aplicación del PFM a los datos obtenidos en estaciones meteorológicas.

Y, finalmente, se plantea el estudio de los datos en series temporales. Es frecuente encontrar los datos obtenidos en estaciones meteorológicas en forma de series temporales, pues se realizan mediciones de diversos parámetros en determinados momentos, dando lugar a secuencias de datos ordenados cronológicamente a partir de las observaciones realizadas. El hecho

de procesar grandes cantidades de datos contenidos en series temporales conllevará que crezcan, exponencialmente, los tiempos de ejecución, lo que supondrá un reto a nivel de aceleración y de optimización del gasto computacional.

Capítulo 4

Publicaciones y calidad de las revistas

La tesis doctoral que se presenta está formada por un compendio de publicaciones científicas publicadas en revistas de alto impacto e indexadas por JCR. A continuación se muestran sus referencias bibliográficas completas.

Título	*Air-Pollution Prediction in Smart Cities through Machine Learning Methods: A Case of Study in Murcia, Spain*
Revista	Journal of Universal Computer Science
Año	2018
Estado	Publicado

Detalles de la revista *Journal of Universal Computer Science*
Editor-in-Chief: Dana Kaiser
ISSN: 0948-695x (versión impresa)
ISSN: 0948-6968 (versión electrónica)
Editorial: Graz Univ. Technology
Factor de impacto (2017): 1.066 (Journal Citation Reports)
Categoría: Computer Science, Theory & Methods
Ranking: Q3, posición: 58 de 103
Página Web: `http://www.jucs.org`

Título	*An unsupervised technique to discretize numerical values by fuzzy partitions*
Revista	Journal of Ambient Intelligence and Smart Environments
Año	2018
DOI	10.3233/AIS-180488
Estado	Publicado

Detalles de la revista
Journal of Ambient Intelligence and Smart Environments
Editor-in-Chief: Peter Sloot
ISSN: 1876-1364 (versión impresa)
ISSN: 1876-1372 (versión electrónica)
Editorial: IOS Press
Factor de impacto (2017): 0.878 (Journal Citation Reports)
Categoría: Computer Science, Artificial Intelligence
Ranking: Q4, posición: 105 de 132
Página Web: `https://content.iospress.com/journals/journal-of-ambient-intelligence-and-smart-environments/10/3`

Título	*Parallel Implementation of Fuzzy Minimals Clustering Algorithm*
Revista	Expert Systems With Applications
Año	2016
DOI	10.1016/j.eswa.2015.11.011
Estado	Publicado

Detalles de la revista *Expert Systems With Applications*
Editor-in-Chief: Dr. Binshan Lin
ISSN: 0957-4174 (versión impresa)
Editorial: Elsevier
Factor de impacto (2017): 3.768 (Journal Citation Reports)
Categoría: Computer Science, Artificial Intelligence
Ranking: Q1, posición: 20 de 132
Página Web: `www.elsevier.com/locate/eswa`

Título	*High-Throughput Infrastructure for Advanced ITS Services: A Case Study on Air Pollution Monitoring*
Revista	IEEE Transactions on Intelligent Transportation Systems
Año	2018
DOI	10.1109/TITS.2018.2816741
Estado	Publicado

Detalles de la revista *IEEE Transactions on Intelligent Transportation Systems*
Editor-in-Chief: Petros Ioannou
ISSN: 1524-9050 (versión impresa)
ISSN: 1558-0016 (versión electrónica)
Editorial: IEEE
Factor de impacto (2017): 4.051 (Journal Citation Reports)
Categoría: Transportation Science & Technology
Ranking: Q1, posición: 5 de 35
Página Web: `https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6979`

Bibliografía

[1] AGRAWAL, D. *et al.* Challenges and opportunities with big data: a white paper prepared for the computing community consortium committee of the computing research association.

[2] AGRAWAL, R., GEHRKE, J., GUNOPULOS, D. Y RAGHAVAN, P. *Automatic subspace clustering of high dimensional data for data mining applications*. ACM, 1998.

[3] ALEKSEEV, A. *et al.* Efficient data management tools for the heterogeneous big data warehouse. *Physics of Particles and Nuclei Letters 13*, 5 (2016), 689–692.

[4] ANDERSON, E. The irises of the Gaspé Peninsula. *Bulletin of the American Iris society 59* (1935), 2–5.

[5] ANDREOPOULOS, B., AN, A., WANG, X. Y SCHROEDER, M. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics 10*, 3 (2009), 297–314.

[6] ANDRITSOS, P., TSAPARAS, P., MILLER, R. Y SEVCIK, K. LIMBO: Scalable clustering of categorical data. In *International Conference on Extending Database Technology* (2004), Springer, pp. 123–146.

[7] ASANOVIC, K. *et al.* The landscape of parallel computing research: A view from Berkeley. Tech. rep., Technical Report UCB/EECS-2006-183, EECS Department, University of California, 2006.

[8] BEZDEK, J. Objective function clustering. In *Pattern recognition with fuzzy objective function algorithms*. Springer, 1981, pp. 43–93.

[9] BEZDEK, J., EHRLICH, R. Y FULL, W. FCM: The Fuzzy C-Means clustering algorithm. *Computers & Geosciences 10*, 2-3 (1984), 191–203.

[10] CATTELL, R. Scalable SQL and NoSQL data stores. *ACM Sigmod Record 39*, 4 (2011), 12–27.

[11] CHAUDHURI, S., DAYAL, U. Y NARASAYYA, V. An overview of business intelligence technology. *Communications of the ACM 54*, 8 (2011), 88–98.

[12] DE HOON, M., IMOTO, S., NOLAN, J., Y MIYANO, S. Open source clustering software. *Bioinformatics 20*, 9 (2004), 1453–1454.

[13] DHAS, J., VIGILA, S. Y STAR, C. Forecasting of stock market by combining machine learning and big data analytics. In *International Conference on Soft Computing Systems* (2018), Springer, pp. 385–395.

[14] DUDA, R., HART, P., Y STORK, D. *Pattern classification*. John Wiley & Sons, 2012.

[15] DUNN, J. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters.

[16] FLORES-SINTAS, A., CADENAS, J.M. Y MARTÍN, F. A local geometrical properties application to fuzzy clustering. *Fuzzy Sets and Systems 100*, 1-3 (1998), 245–256.

[17] GUHA, S., RASTOGI, R. Y SHIM, K. CURE: an efficient clustering algorithm for large databases. In *ACM Sigmod Record* (1998), no. 2, ACM.

[18] GUHA, S., RASTOGI, R. Y SHIM, K. ROCK: A robust clustering algorithm for categorical attributes. *Information systems 25*, 5 (2000), 345–366.

[19] HAN, J., PEI, J. Y KAMBER, M. *Data mining: concepts and techniques*. Elsevier, 2011.

[20] HAND, D. Principles of data mining. *Drug safety 30*, 7 (2007), 621–622.

[21] HAO, Y., USAMA, M., YANG, J., HOSSAIN, M. Y GHONEIM, A. Recurrent convolutional neural network based multimodal disease risk prediction. *Future Generation Computer Systems* (2018).

[22] HARTIGAN, J. *Clustering Algorithms*. John Wiley & Sons, Inc., 1975.

[23] HOCHBAUM, D. Y SHMOYS, D. A best possible heuristic for the k-center problem. *Mathematics of operations research 10*, 2 (1985), 180–184.

[24] HÖPPNER, F. *Fuzzy cluster analysis: methods for classification, data analysis and image recognition*. John Wiley & Sons, 1999.

[25] HOWARD, J. *et al.* Scale and performance in a distributed file system. *ACM Transactions on Computer Systems (TOCS) 6*, 1 (1988), 51–81.

[26] JAIN, A., MURTY, M. Y FLYNN, P. Data clustering: a review. *ACM computing surveys (CSUR) 31*, 3 (1999), 264–323.

[27] JAIN, A. Y DUBES, R. Algorithms for clustering data.

[28] KAUFMAN, L. Y ROUSSEEUW, P. *Clustering by means of medoids*. North-Holland, 1987.

[29] KAUFMAN, L. Y ROUSSEEUW, P. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[30] KHOTANZAD, A. Y BOUARFA, A. Image segmentation by a parallel, non-parametric histogram based clustering algorithm. *Pattern Recognition 23*, 9 (1990), 961–973.

[31] KIM, W. Parallel clustering algorithms: survey. *Parallel Algorithms, Spring* (2009).

[32] KUO, K., YU, H., RILEE, M., PAN, Y. Y ZHU, F. Exploiting transparent multimodal parallelization for high-performance big data analytics and machine learning. In *EGU General Assembly Conference Abstracts* (2018), vol. 20, p. 11376.

[33] LABRINIDIS, A. Y JAGADISH, H. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment 5*, 12 (2012), 2032–2033.

[34] MARTÍNEZ, R. *Metodologías basadas en minería de datos para el diseño y optimización de técnicas de clasificación automática*. PhD thesis, Universidad de Murcia, 2014.

[35] MATTSON, T., SANDERS, B. Y MASSINGILL, B. *Patterns for parallel programming*. Pearson Education, 2004.

[36] MOMENI, E., CARDIE, C. Y DIAKOPOULOS, N. A survey on assessment and ranking methodologies for user-generated content on the web. *ACM Computing Surveys (CSUR) 48*, 3 (2016), 41.

[37] PABÓN, O. *et al.* A comparison of NoSQL graph databases. In *2014 9th Computing Colombian Conference (9CCC)* (2014), IEEE, pp. 128–131.

[38] PEPPA, M., BELL, D., KOMAR, T. Y XIAO, W. Urban traffic flow analysis based on deep learning car detection from cctv image series. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 42*, 4 (2018).

[39] PODANI, J. Y SCHMERA, D. On dendrogram-based measures of functional diversity. *Oikos 115*, 1 (2006), 179–185.

[40] SOTO, J., FLORES-SINTAS, A. Y PALAREA-ALBALADEJO, J. Improving probabilities in a fuzzy clustering partition. *Fuzzy Sets and Systems 159*, 4 (2008), 406–421.

[41] TELLO-OQUENDO, L. *Design and Performance Analysis of Access Control Mechanisms for Massive Machine-to-Machine Communications in Wireless Cellular Networks*. PhD thesis, 2018.

[42] WHITE, S. Y SMYTH, P. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM International Conference on Data Mining* (2005), SIAM, pp. 274–285.

[43] WITTEN, I., FRANK, E., HALL, M. Y PAL, C. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[44] WU, Z. Y LEAHY, R. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence 15*, 11 (1993), 1101–1113.

[45] XU, R. Y WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on neural networks 16*, 3 (2005), 645–678.

[46] ZHANG, T., RAMAKRISHNAN, R. Y LIVNY, M. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery 1*, 2 (1997), 141–182.