Aplicando la transformada integral de la probabilidad para reducir la complejidad de los árboles de decisión difusos multi-vía en problemas de clasificación Big Data

Mikel Elkano*†‡, Mikel Uriz*†, Humberto Bustince*†‡, Mikel Galar*†‡
*Departamento de Estadística, Informática y Matemáticas, Universidad Pública de Navarra, 31006 Pamplona, España
†GIARA, Navarrabiomed, Complejo Hospitalario de Navarra (CHN), Universidad Pública de Navarra (UPNA), IdiSNA Irunlarrea 3, 31008 Pamplona, España
‡Institute of Smart Cities, Universidad Pública de Navarra, 31006 Pamplona, España
Emails: {mikel.elkano, mikelxabier.uriz, bustince, mikel.galar}@unavarra.es

***Resumen*—Presentamos un nuevo método de particionamiento difuso distribuido para reducir la complejidad de los árboles de decisión difusos multi-vía en problemas de clasificación Big Data. El algoritmo propuesto construye un número fijo de conjuntos difusos para todas las variables y ajusta su forma y posición a la distribución real de los datos de entrenamiento. Para ello se aplica un proceso compuesto por dos fases: 1) transformación de la distribución original en una distribución uniforme estándar mediante la transformada integral de la probabilidad. Dado que generalmente la distribución original se desconoce, se calcula una aproximación de la función de distribución acumulada extrayendo los q-cuantiles del conjunto de entrenamiento; 2) construcción de una partición difusa de Ruspini en el espacio de características transformado empleando un número fijo de funciones de pertenencia triangulares uniformemente distribuidas. A pesar de realizar una transformación sobre los datos originales, la definición de los conjuntos difusos en el espacio original se puede recuperar aplicando la función cuantil. Los resultados experimentales revelan que el método de particionamiento propuesto permite mantener la precisión del árbol de decisión difuso multi-vía del estado del arte FMDT empleando hasta 6 millones menos de hojas.**

***Index Terms*—Árboles de Decisión Difusos; Transformada Integral de la Probabilidad; Función Cuantil; MapReduce; Apache Spark; Big Data**

I. Introducción

Los árboles de decisión (ADs) [1] son uno de los algoritmos de aprendizaje automático más populares, habiendo sido aplicados en una gran variedad de problemas como las finanzas [2], la clasificación de imágenes [3], la bioinformática [4], o la medicina [5]. La principal característica de los ADs es la habilidad para explicar el razonamiento de sus predicciones mediante grafos en forma de árboles. Cada nodo es una pregunta o test en un atributo determinado (por ejemplo, ¿$x > 0{,}5$?), cada rama representa la respuesta o el resultado del test, y los nodos terminales (hojas) contienen la decisión final. Generalmente los árboles se construyen aplicando una estrategia *top-down* llamada *particionamiento recursivo* [1], en donde los datos de entrada se dividen recursivamente en dos o más sub-espacios que aumentan la homogeneidad de las distribución de las clases.

La lógica difusa [6] ha mostrado ser una forma efectiva para mejorar el rendimiento de clasificación de los algoritmos de aprendizaje automático cuando se trabaja con datos imprecisos que generan incertidumbre [7], [8], incluyendo los árboles de decisión difusos (ADDs). Sin embargo, en entornos Big Data los excesivos requerimientos de tiempo y espacio de los ADDs afectan seriamente la escalabilidad de estos algoritmos. Segatori et al. propusieron una solución MapReduce que consiste en un nuevo método de particionamiento difuso y en un algoritmo de aprendizaje de ADD distribuido [9]. El discretizador genera una partición difusa fuerte y triangular para cada atributo continuo basándose en la entropía difusa, las cuales son empleadas posteriormente para construir el árbol. Los autores propusieron dos versiones de ADD que difieren en la estrategia de particionamiento recursivo: el ADD binario (o bi-vía) (FBDT) y el ADD multi-vía (FMDT). El ADD binario divide el espacio del atributo en dos sub-espacios (nodos hijo), mientras que el ADD multi-vía puede generar más de dos sub-espacios. A pesar de que la solución propuesta por Segatori et al. consigue buenos resultados en términos de precisión, normalmente los árboles construidos por este algoritmo están compuestos por miles e incluso millones de hojas, por lo que la complejidad del modelo es generalmente elevada.

En este trabajo presentamos un nuevo método de particionamiento difuso que reduce la complejidad de los árboles construidos por FMDT en términos de número de conjuntos difusos empleado por variable y número de hojas. El algoritmo propuesto aplica la *transformada integral de la probabilidad* [10] para ajustar un número fijo de conjuntos difusos a la distribución real del conjunto de entrenamiento. Esta transformación permite convertir las variables del conjunto de entrenamiento en variables aleatorias uniformes independientemente de su distribución original. Posteriormente, se construyen la particiones difusas de Ruspini [11] en el nuevo espacio transformado empleando funciones de pertenencia

triangulares uniformemente distribuidas. Los conjuntos difusos resultantes se utilizan posteriormente por el FMDT original para construir el árbol.

Para evaluar los beneficios de nuestra propuesta, hemos llevado a cabo un estudio empírico que consiste en 4 problemas de clasificación Big Data disponibles en los repositorios de UCI [12] y OpenML[1]. En este estudio comparamos la precisión y la complejidad del modelo de FMDT cuando se considera el método de particionamiento difuso original y el propuesto. Los resultados experimentales muestran una reducción significativa en la complejidad del modelo cuando se aplica nuestro método.

La estructura de este trabajo es la siguiente. La Sección II repasa los conceptos básicos de los ADDs y describe brevemente la solución distribuida de Segatori et al. para construir ADDs para Big Data. En la Sección III introducimos el método de particionamiento difuso propuesto. El marco y el estudio experimental se muestran en las Secciones IV y V. Finalmente, la Sección VI presenta las conclusiones de este trabajo.

II. Preliminares: Árboles de decisión difusos para Big Data

Los árboles de decisión (ADs) [1] son algoritmos de aprendizaje automático supervisado no-paramétricos empleados para tareas de clasificación y regresión. En este trabajo nos centramos en las tareas de clasificación, las cuales consisten en construir un modelo predictivo llamado *clasificador* que es capaz de clasificar ejemplos no etiquetados (desconocidos) en base a un conjunto de entrenamiento formado por ejemplos previamente etiquetados. Cada ejemplo $x = (x_1, \ldots, x_F)$ contenido en el conjunto de entrenamiento TR pertenece a una clase $y \in \mathbb{C} = \{C_1, ..., C_M\}$ (siendo M el número de clases del problema) y se caracteriza por un conjunto de F variables (también llamadas atributos o características), donde cada variable x_f puede tomar cualquier valor contenido en el conjunto $\mathcal{F}_f$. Por tanto, la construcción de un clasificador consiste en encontrar una función de decisión $h : \mathcal{F}_1 \times \ldots \times \mathcal{F}_F \rightarrow \mathbb{C}$ que maximice la precisión en la clasificación.

Un AD es un grafo dirigido acíclico donde cada nodo interno es un test sobre un atributo, cada rama representa el resultado del test, y cada nodo terminal (hoja) contiene la decisión final (etiqueta de la clase). Los ADs se construyen aplicando un *particionamiento recursivo* [1] del espacio de atributos. La selección del atributo considerado en el nodo de decisión está basada en métricas que miden la diferencia entre el nivel de homogeneidad de las clases contenidas en el nodo padre y en los nodos hijos. Para los atributos continuos se pueden aplicar soluciones basadas tanto en fuerza bruta como en estrategias de discretización. Las primeras comprueban todos los posibles puntos de corte en el conjunto de entrenamiento, mientras que las últimas dividen el dominio del atributo en un conjunto discreto de intervalos (también llamados *bins*). Dado que las soluciones por fuerza bruta pueden ser computacionalmente costosas, los ADs diseñados para Big Data suelen aplicar estrategias de discretización para acelerar la ejecución del algoritmo y reducir la complejidad del modelo.

Los árboles de decisión difusos (ADDs) [7] hacen uso de la lógica difusa [6] para manejar mejor la incertidumbre y crear fronteras de decisión suaves que mejoran la precisión de la clasificación. En este caso los atributos continuos están caracterizados por particiones difusas en lugar de un conjunto discreto de intervalos. Por consiguiente, un valor de entrada determinado puede pertenecer a uno o más conjuntos difusos con un grado de pertenencia determinado y activar múltiples ramas al mismo tiempo. Las particiones difusas permiten manejar transiciones progresivas entre los intervalos adyacentes y mejorar así la precisión de las predicciones cuando se manejan datos numéricos. A la hora de clasificar un nuevo ejemplo x, se calcula el grado de activación de cada hoja. Para ello se debe calcular el grado de activación de todos los nodos internos que pertenecen a la ruta de la hoja correspondiente de la siguiente forma. Dado el nodo actual CN que considera x_f como el atributo a dividir, se calcula el grado de activación $md^{CN}(x)$ de CN para x:

$$md^{CN}(x) = T\left(\mu^{CN}(x_f), md^{PN}(x)\right), \tag{1}$$

donde T es una T-norma, $\mu^{CN}(x_f)$ es el grado de pertenencia de x_f al conjunto difuso asociado con el nodo CN, y $md^{PN}(x)$ es el grado de activación del nodo padre PN para x. Posteriormente, se obtiene el grado de asociación $AD_m^{LN}(x)$ de la clase C_m en la hoja LN para x:

$$AD_m^{LN}(x) = md^{LN}(x) \cdot w_m^{LN}, \tag{2}$$

donde $md^{LN}(x)$ es el grado de activación de la hoja LN para x y w_m^{LN} es el peso de la clase C_m en LN. En la literatura se han propuesto diferentes definiciones para w_m^{LN} [13]. En este trabajo consideramos

$$w_m^{LN} = \frac{\sum_{x \in TR_{C_m}} md^{LN}(x)}{\sum_{x \in TR} md^{LN}(x)}, \tag{3}$$

donde TR_{C_m} es el conjunto de todos los ejemplos de entrenamiento que pertenecen a la clase C_m. Finalmente, se predice la clase de x siguiendo un determinado criterio, siendo los siguientes los más comunes:

- *Máxima activación*: se predice la clase que corresponde al máximo grado de asociación.
- *Voto ponderado*: para cada clase se calcula la suma de todos los grados de asociación correspondientes a la clase y se predice aquella que obtiene la mayor suma.

Los requerimientos de tiempo y memoria de los ADDs pueden causar serios problemas de escalabilidad cuando se tratan grandes conjuntos de datos. En este trabajo consideramos la solución distribuida propuesta por Segatori et al. en [9] para construir ADDs en Big Data, la cual consta de dos fases:

[1] https://www.openml.org/search?type=data

1. *Particionamiento difuso*. Se genera una partición difusa triangular fuerte para cada atributo continuo basándose en la entropía difusa. Para ello, el algoritmo selecciona la partición difusa candidata que minimiza la entropía difusa y divide el dominio del atributo en dos subconjuntos de forma recursiva hasta que se cumple una cierta condición de parada. A pesar de que las particiones difusas construidas por este método suelen ser precisas, el alto número de conjuntos difusos que habitualmente contienen aumenta la complejidad del modelo.
2. *Aprendizaje del ADD*. Se construye un ADD aplicando una de las dos estrategias de división consideradas por los autores: binaria (FBDT), que genera dos nodos hijos, o multi-vía (FMDT), que puede generar más de dos hijos. Ambos métodos emplean la ganancia de información difusa [14] para la selección del atributo. En este trabajo nos centramos en los árboles FMDT.

III. APLICANDO LA TRANSFORMADA INTEGRAL DE LA PROBABILIDAD PARA REDUCIR LA COMPLEJIDAD DE LOS ÁRBOLES DE DECISIÓN DIFUSOS MULTI-VÍA

En este trabajo proponemos un nuevo método de particionamiento difuso distribuido que reduce la complejidad de los ADDs generados por el algoritmo FMDT [9]. La solución propuesta reemplaza el método de particionamiento difuso original empleado por FMDT sin alterar el algoritmo de aprendizaje del ADD. Los objetivos de nuestra propuesta son los siguientes:

- Construir un número reducido de conjuntos difusos por atributo. El método original añade conjuntos difusos a la partición hasta que la ganancia de información difusa es menor que un cierto umbral, aumentando la complejidad del modelo. Nuestro método emplea un número fijo de conjuntos difusos para todos los atributos.
- Ajustar los conjuntos difusos a la distribución real de los atributos. La solución propuesta modifica tanto la forma como la posición de los conjuntos difusos para mejorar la capacidad de discriminación del modelo

Para cumplir estos objetivos proponemos un algoritmo que consiste en una fase de pre-procesamiento que resulta en un particionamiento difuso auto-adaptativo.

- Pre-procesamiento: las variables del conjunto de entrenamiento son transformadas en variables aleatorias uniformes aplicando la *transformada integral de la probabilidad* [10] descrita en el Teorema 1. Este teorema afirma que cualquier variable aleatoria continua puede ser convertida en una variable aleatoria uniforme estándar.

  Teorema 1. *Si X es una variable aleatoria continua con una función de distribución acumulada (FDA) $F_X(x)$ y si $Y = F_X(X)$, entonces Y es una variable aleatoria uniforme en el intervalo [0,1].*

  Demostración. Supongamos que $Y = g(X)$ es una función de X donde g es derivable y estrictamente creciente. Por tanto, su inversa g^{-1} existe únicamente. La FDA de Y se puede derivar usando

$$F_Y(y) = Prob\,(Y \leq y) = Prob\left(X \leq g^{-1}(y)\right) = F_X\left(g^{-1}(y)\right)$$

  y su densidad viene dada por

$$f_Y(y) = \frac{d}{dy}F_Y(y) = \frac{d}{dy}F_X(g^{-1}(y)) = f_X(g^{-1}(y)) \cdot \frac{d}{dy}g^{-1}(y).$$

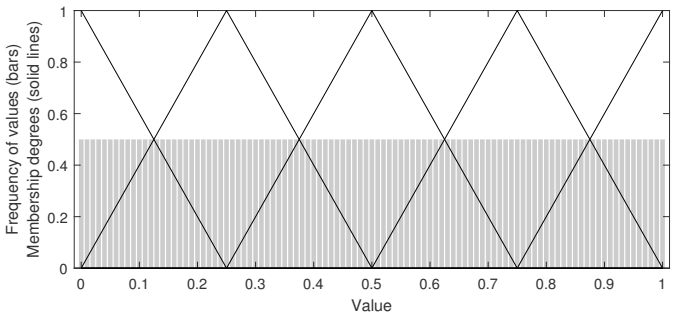
  Este procedimiento se llama la técnica de la FDA y permite que la distribución de Y se derive de la siguiente forma:

$$F_Y(y) = Prob\,(Y \leq y) = Prob\left(X \leq F_X^{-1}(y)\right) = F_X\left(F_X^{-1}(y)\right) = y$$

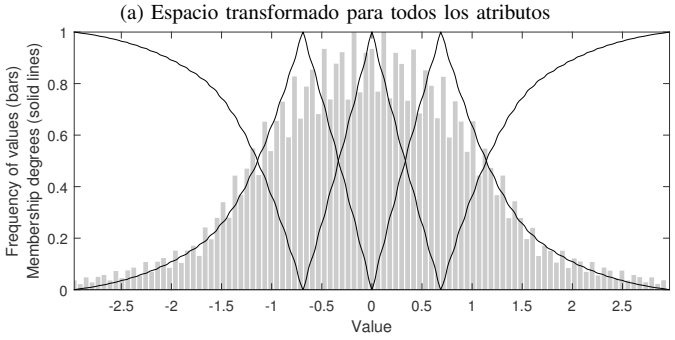
  ■

  Sin embargo, dado que la distribución original del conjunto de entrenamiento es desconocida, la FDA no puede ser calculada exactamente. Como solución proponemos calcular los q-cuantiles del conjunto de entrenamiento para obtener una FDA aproximada. Para ello, para cada variable, se ordenan todos los valores y se extrae cada cuantil. Si q es menor que el número de ejemplos en el conjunto de entrenamiento, el valor de la FDA para un valor dado se interpola linealmente en el intervalo [Q_{i-1}, Q_i], siendo Q_i el primer cuantil mayor que el valor. Si el valor es menor que el primer cuantil (Q_1) o mayor que el último cuantil (Q_{q-1}), el valor de la FDA será 0 o 1, respectivamente. De esta forma, las variables del nuevo conjunto de datos transformado seguirán una distribución aproximadamente uniforme independientemente de su distribución original. Por supuesto, la transformación sobre el conjunto de test se realiza interpolando la FDA con los cuantiles extraídos del conjunto de entrenamiento.
- Particionamiento: se construye una partición difusa fuerte de Ruspini [11] distribuyendo uniformemente un número fijo de funciones de pertenencia triangulares a lo largo del intervalo [0,1]. Cabe destacar que la definición de cada uno de los conjuntos difusos en el espacio original puede recuperarse aplicando la función cuantil [15]. En este caso, para cada punto que define la función de pertenencia triangular, el valor correspondiente se interpolaría calculando la inversa de la función lineal empleada para calcular la FDA entre los dos cuantiles más próximos. La Figura 1 muestra un ejemplo ilustrativo de cómo se distribuyen los conjuntos difusos en el espacio original y en el transformado del atributo *jet_1_eta* del conjunto de datos HIGGS. Las líneas sólidas y las barras representan las funciones de pertenencia y la distribución original de las variables, respectivamente.

Nótese que ambas fases (pre-procesamiento y particionamiento) están estrechamente relacionadas. Dado que, bajo nuestro punto de vista, una partición difusa de Ruspini con



(a) Espacio transformado para todos los atributos



(b) Espacio original del atributo *jet_1_eta*

Figura 1: Conjuntos difusos construidos para *jet_1_eta* en HIGGS.

funciones de pertenencia uniformemente distribuidas es una buena forma de modelar una distribución uniforme, nuestra hipótesis es que si es posible transformar la distribución de cualquier atributo en una distribución uniforme y al mismo tiempo poder realizar el proceso inverso, se obtiene una partición auto-adaptada para la distribución original del atributo. El hecho más interesante es que este resultado se obtiene sin necesidad de diseñar un nuevo método de particionamiento específico. El código fuente de nuestra propuesta está escrito en Scala 2.11[2] para Apache Spark 2.0.2 y está disponible en GitHub[3] bajo la licencia GPL.

IV. Marco experimental

En esta sección primero describimos los conjuntos de datos y las medidas de rendimiento empleadas para evaluar los métodos considerados en los experimentos (Sección IV-A). Posteriormente detallamos los parámetros y la configuración del entorno utilizados para la ejecución de los algoritmos (Sección IV-B).

IV-A. Conjuntos de datos y medidas de rendimiento

Para llevar a cabo el estudio experimental hemos considerado 4 problemas de clasificación Big Data disponibles en los repositorios de UCI [12] y OpenML[4]. La Tabla I muestra la descripción de los conjuntos de datos indicando el número de instancias (#Instancias), atributos (#Atributos) reales (R)/enteros(E)/categóricos(C)/totales(T), y clases (#Clases). Los nombres de BNG Australian (BNG) y HEPMASS (HEPM) han sido abreviados. En todos los experimentos se ha empleado un esquema de validación cruzada estratificada de 5 particiones.

Tabla I: Descripción de los conjuntos de datos.

Dataset	#Instancias	#Atributos				#Clases
		R	E	C	T	
BNG	1,000,000	8	6	0	14	2
HEPM	10,500,000	28	0	0	28	2
HIGGS	11,000,000	28	0	0	28	2
SUSY	5,000,000	18	0	0	18	2

El rendimiento en clasificación se ha medido empleando la denominada matriz de confusión (Tabla II), la cual almacena el número de ejemplos clasificados correcta e incorrectamente para cada clase. De esta matriz podemos obtener las siguientes

Tabla II: Matriz de confusión para un problema binario.

	Predicción positiva	Predicción negativa
Clase positiva	Verdadero positivo (TP)	Falso Negativo (FN)
Clase negativa	Falso Positivo (FP)	Verdadero Negativo (TN)

cuatro medidas:

- Ratio de verdaderos positivos: porcentaje de ejemplos positivos clasificados correctamente.
  $TP_{rate} = TP \ / \ (TP + FN)$.
- Ratio de verdaderos negativos: porcentaje de ejemplos negativos clasificados correctamente.
  $TN_{rate} = TN \ / \ (TN + FP)$.
- Ratio de falsos positivos: porcentaje de ejemplos negativos clasificados incorrectamente.
  $FP_{rate} = FP \ / \ (FP + TN)$.
- Ratio de falsos negativos: porcentaje de ejemplos positivos clasificados incorrectamente.
  $FN_{rate} = FN \ / \ (FN + TP)$.

Basados en estas métricas, el rendimiento de clasificación de cada método se ha medido empleando el porcentaje de precisión y el Área Bajo la Curva ROC (AUC) definidas como:

$$\%Precision = (TP+TN) \ / \ (TP+FN+FP+TN) \quad (4)$$

$$AUC = (1 + TP_{rate} + FP_{rate}) \ / \ 2 \quad (5)$$

IV-B. Parámetros y configuración del entorno

En cuanto a los parámetros utilizados para FMDT, hemos fijado los valores sugeridos por los autores en el artículo original [9]:

- Medida para calcular la impureza de los nodos: entropía difusa
- T-norma: producto
- Máximo número de bins para los atributos numéricos: 32

[2]http://www.scala-lang.org/
[3]https://github.com/melkano/uniform-fuzzy-partitioning
[4]https://www.openml.org/search?type=data

- Máxima profundidad del árbol: 5
- $\gamma = 0.1\,\%$; $\phi = 0.02 \cdot N$; $\lambda = 10^{-4} \cdot N$

Todos los métodos han sido ejecutados en un cluster de 8 nodos conectados a una Red de área Local Ethernet a 1Gb/s. La mitad de estos nodos están compuestos por 2 procesadores Intel Xeon E5-2620 a 2.4 GHz (3.2 GHz con Turbo Boost) con 12 núcleos virtuales en cada uno (de los cuales 6 son físicos). Tres de los nodos restantes están equipados con 2 procesadores Intel Xeon E5-2620 a 2.1 GHz con el mismo número de núcleos que los anteriores. El último nodo es el master, compuesto por un procesador Intel Xeon E5-2609 con 4 núcleos físicos a 2.4 GHz. Todos los nodos esclavos están equipados con 32GB de RAM, mientras que el maestro trabaja con 8GB de RAM. Con respecto al almacenamiento, todos los nodos emplean discos duros con velocidades de lectura/escritura de 128 MB/s. Todo el cluster funciona sobre CentOS 6.5, Apache Hadoop 2.6.0, y Apache Spark 2.0.2.

De acuerdo con los autores de FMDT, emplear más de 24 núcleos de CPU tiene un impacto negativo en el tiempo de ejecución de este método. Por consiguiente, hemos empleado 6 *executors* con 4 núcleos cada uno para ejecutar FMDT.

V. Estudio experimental

Para evaluar el rendimiento de nuestra propuesta hemos llevado a cabo un estudio empírico que cubre tres aspectos: rendimiento en la clasificación (Tabla III), complejidad del modelo (Tabla IV), y tiempo de ejecución (Tabla V). En todos los casos hemos considerado cuatro métodos: el FMDT original [9] y tres configuraciones diferentes del método propuesto que difieren en el número de conjuntos difusos (X) utilizados para los atributos numéricos (denotado como FMDT$_X$). Cabe señalar que el método FMDT original se quedó sin memoria mientras procesaba HEPMASS debido al gran número de hojas que se generaron a lo largo del entrenamiento, y por tanto no se muestran resultados para este método en HEPMASS.

Las Tablas III y IV revelan que el método de particionamiento difuso propuesto (FMDT$_X$) es capaz de mantener el rendimiento de clasificación de FMDT con modelos notablemente más simples. Las diferentes configuraciones de nuestra aproximación rinden de forma similar en términos de precisión y AUC (excepto en HIGGS), a pesar de que hay una tendencia positiva a favor del uso de más conjuntos difusos. Sin embargo, emplear más conjuntos difusos suele causar que el algoritmo de aprendizaje genere más hojas, aumentando la complejidad del modelo. A continuación analizamos los resultados obtenidos en cada uno de los conjuntos de datos:

- BNG: nuestro método mejora la precisión y el AUC de FMDT un 6 % y un 8 %, respectivamente. A pesar de que los árboles construidos por FMDT$_X$ son más profundos, éstos tienen entre 8 mil y 80 mil veces menos de hojas que FMDT.
- HEPM: el FMDT original construye demasiadas hojas como para poder afrontar este problema con el cluster descrito en la Sección IV-B, quedándose sin memoria durante la ejecución. Este hecho sugiere que nuestra aproximación es una solución candidata para evitar la explosión del número de hojas durante la inducción de los ADDs.
- HIGGS: el rendimiento de clasificación de FMDT$_5$ en este problema cae casi un 1 % con respecto al resto de métodos, revelando que 5 conjuntos difusos no son suficientes para capturar la complejidad de este problema. Sin embargo, el resto de configuraciones (FMDT$_7$ y FMDT$_9$) son capaces de mantener el rendimiento de clasificación de FMDT con 15 mil y 50 mil hojas, respectivamente, mientras que FMDT genera 6 millones de hojas. Además, el método de particionamiento difuso original genera casi el doble de conjuntos difusos que FMDT$_7$.
- SUSY: todas las configuraciones rinden de forma similar a FMDT en términos de capacidad de discriminación. Sin embargo, nuestro método construye árboles de 3 mil, 15 mil, y 50 mil hojas, mientras que FMDT genera 5 millones de hojas. En este caso, la diferencia en el número de conjuntos difusos empleados por cada método es aún mayor, ya que FMDT utiliza casi 23 conjuntos de datos por atributo.

Tabla III: Rendimiento de clasificación de cada método.

		%Precision		
Dataset	FMDT	FMDT$_5$	FMDT$_7$	FMDT$_9$
BNG	$80.23^{\pm 0,05}$	$86.79^{\pm 0,06}$	$86.93^{\pm 0,07}$	$86.97^{\pm 0,06}$
HEPM	-	$91.13^{\pm 0,02}$	$91.25^{\pm 0,02}$	$91.33^{\pm 0,02}$
HIGGS	$71.54^{\pm 0,02}$	$70.61^{\pm 0,02}$	$71.32^{\pm 0,03}$	$71.69^{\pm 0,03}$
SUSY	$79.29^{\pm 0,05}$	$79.15^{\pm 0,04}$	$79.49^{\pm 0,04}$	$79.66^{\pm 0,04}$
		AUC		
Dataset	FMDT	FMDT$_5$	FMDT$_7$	FMDT$_9$
BNG	$.7896^{\pm ,0004}$	$.8649^{\pm ,0006}$	$.8658^{\pm ,0007}$	$.8662^{\pm ,0007}$
HEPM	-	$.9113^{\pm ,0002}$	$.9125^{\pm ,0002}$	$.9133^{\pm ,0002}$
HIGGS	$.7143^{\pm ,0001}$	$.7033^{\pm ,0002}$	$.7114^{\pm ,0003}$	$.7155^{\pm ,0003}$
SUSY	$.7859^{\pm ,0004}$	$.7847^{\pm ,0004}$	$.7880^{\pm ,0004}$	$.7898^{\pm ,0004}$

La Tabla V muestra el tiempo requerido por cada método para las tres fases: particionamiento difuso, la inducción del ADD, y el tiempo total. En general, no hay diferencias significativas entre los diferentes métodos cuando se trata de la fase de particionamiento, a pesar de que el algoritmo propuesto es un 30 % más rápido que el método original en SUSY. Sin embargo, cuando se considera la inducción del ADD, la reducción en la complejidad del modelo cuando se emplea nuestro método resulta en tiempos de ejecución mucho más rápidos.

VI. Conclusiones

En este trabajo hemos presentado un nuevo método de particionamiento difuso distribuido que reduce la complejidad del modelo de los árboles de decisión difusos (ADDs) multivía en problemas de clasificación Big Data. El algoritmo

Tabla IV: Complejidad de cada modelo.

Dataset	FMDT	$FMDT_5$	$FMDT_7$	$FMDT_9$
Número de hojas				
BNG	83,044	1,211	4,807	9,492
HEPM	-	2,854	13,472	43,339
HIGGS	6,414,575	3,005	15,876	53,489
SUSY	5,225,134	2,977	14,989	49,038
Profundidad media				
BNG	3.02	4.67	5.00	4.35
HEPM	-	4.52	4.03	3.93
HIGGS	3.25	5.00	5.00	4.89
SUSY	3.68	5.00	5.00	4.76
Promedio de número de conjuntos difusos				
BNG	6.04	5.00	7.00	9.00
HEPM	-	5.00	7.00	9.00
HIGGS	13.01	5.00	7.00	9.00
SUSY	22.60	5.00	7.00	9.00

Tabla V: Tiempos de ejecución(s) de cada método.

Dataset	FMDT	$FMDT_5$	$FMDT_7$	$FMDT_9$
Particionamiento				
BNG	58	41	40	40
HEPM	-	295	292	294
HIGGS	252	273	274	276
SUSY	110	77	72	77
Aprendizaje				
BNG	25	23	22	24
HEPM	-	149	158	153
HIGGS	4,984	176	167	158
SUSY	1,282	76	75	77
Tiempo total				
BNG	84	65	63	65
HEPM	-	445	450	448
HIGGS	5,238	450	441	435
SUSY	1,392	154	148	155

propuesto consiste en transformar el conjunto de entrenamiento original de tal forma que todas las variables numéricas sigan una distribución uniforme estándar. Para ello se aplica la transformada integral de la probabilidad, la cual afirma que cualquier variable aleatoria continua puede ser convertida en una variable aleatoria uniforme en base a la función de distribución acumulada (FDA) original. Dado que la FDA es generalmente desconocida, proponemos aproximar esta función calculando los q-cuantiles del conjunto de entrenamiento e interpolando linealmente entre dichos cuantiles. Después de esta transformación, se crean las particiones difusas de Ruspini distribuyendo un número fijo de funciones de pertenencia triangulares de forma uniforme a lo largo del intervalo [0,1]. Para recuperar los puntos que definen los conjuntos difusos en el espacio original se aplica la función cuantil. El método de particionamiento propuesto es capaz de ajustar tanto la posición como la forma de los conjuntos difusos a la distribución real del conjunto de entrenamiento.

Para evaluar el rendimiento de nuestra propuesta, hemos realizado un estudio empírico que se centra en el algoritmo de inducción de ADDs multi-vía propuesto por Segatori et al. para Big Data (FMDT). Para ello, hemos reemplazado el método de particionamiento difuso original por el método propuesto, sin modificar la fase de inducción del ADD. Los resultados experimentales revelan que nuestro algoritmo genera árboles significativamente más simples que son capaces de mantener el rendimiento de clasificación del método original.

Agradecimientos

Este trabajo ha sido financiado parcialmente por el Ministerio de Ciencia, Innovación y Universidades de España con el proyecto TIN2016-77356-P.

Referencias

[1] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[2] M.-Y. Chen, "Predicting corporate financial distress based on integration of decision tree classification and logistic regression," *Expert Systems with Applications*, vol. 38, no. 9, pp. 11 261–11 272, 2011.

[3] C.-C. Yang, S. Prasher, P. Enright, C. Madramootoo, M. Burgess, P. Goel, and I. Callum, "Application of decision tree technology for image classification using remote sensing data," *Agricultural Systems*, vol. 76, no. 3, pp. 1101–1117, 2003.

[4] D. Che, Q. Liu, K. Rasheed, and X. Tao, "Decision tree and ensemble learning algorithms with their applications in bioinformatics," *Advances in Experimental Medicine and Biology*, vol. 696, pp. 191–199, 2011.

[5] J. Sanz, D. Paternain, M. Galar, J. Fernandez, D. Reyero, and T. Belzunegui, "A New Survival Status Prediction System for Severe Trauma Patients Based on a Multiple Classifier System," *Computer Methods and Programs in Biomedicine*, vol. 142, no. C, pp. 1–8, 2017.

[6] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338 – 353, 1965.

[7] C. Janikow, "Fuzzy decision trees: Issues and methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, no. 1, pp. 1–14, 1998.

[8] J. Sanz, H. Bustince, A. Fernández, and F. Herrera, "IIVFDT: Ignorance functions based interval-valued fuzzy decision tree with genetic tuning," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 20, no. SUPPL. 2, pp. 1–30, 2012.

[9] A. Segatori, F. Marcelloni, and W. Pedrycz, "On Distributed Fuzzy Decision Trees for Big Data," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 1, pp. 174–192, 2018.

[10] J. E. Angus, "The Probability Integral Transform and Related Results," *SIAM Review*, vol. 36, no. 4, pp. 652–654, 1994.

[11] E. H. Ruspini, "A new approach to clustering," *Information and Control*, vol. 15, no. 1, pp. 22–32, 1969.

[12] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[13] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining (Advanced Information Processing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2004.

[14] M. Zeinalkhani and M. Eftekhari, "Fuzzy partitioning of continuous attributes through discretization methods to construct fuzzy decision tree classifiers," *Information Sciences*, vol. 278, pp. 715–735, 2014.

[15] N. U. Nair, P. G. Sankaran, and N. Balakrishnan, *Quantile-Based Reliability Analysis*. New York, NY: Springer New York, 2013, ch. Quantile Functions, pp. 1–28.