

# Sistemas difusos para computación en Big Data

## Ecuaciones no resolubles con respecto a la derivada

Doble Grado en Ingeniería Informática y Matemáticas

---

Antonio Coín Castro

17 de Septiembre de 2020

Trabajo Fin de Grado

*E.T.S de Ingenierías Informática y de Telecomunicación*  
*Facultad de Ciencias*



**UNIVERSIDAD  
DE GRANADA**

Sistemas difusos para  
computación en Big Data

- Conjuntos y lógica difusa

- Fundamentos de Big Data

- Diseño e implementación de  
algoritmos

- Estudio comparativo

Ecuaciones no resolubles con  
respecto a la derivada

# SISTEMAS DIFUSOS PARA COMPUTACIÓN EN BIG DATA

---

El problema de aprendizaje de datos es un tema central en el aprendizaje automático. Una propuesta relevante en este sentido son los sistemas basados en reglas difusas, que permiten resolver problemas de forma aproximada pero efectiva.

Por otro lado, en la *era de la información* las cantidades de datos que se manejan son cada vez mayores, y surge el concepto de Big Data. ¿Están preparados los algoritmos existentes para tratar grandes cantidades de datos?

**Solución:** construir sistemas difusos **escalables**.

- Estudiar la teoría de conjuntos difusos, la lógica difusa y los sistemas difusos desde un punto de vista teórico.

- Estudiar la teoría de conjuntos difusos, la lógica difusa y los sistemas difusos desde un punto de vista teórico.
- Definir el concepto de Big Data, sus características y la infraestructura asociada.

- Estudiar la teoría de conjuntos difusos, la lógica difusa y los sistemas difusos desde un punto de vista teórico.
- Definir el concepto de Big Data, sus características y la infraestructura asociada.
- Diseñar, implementar y probar una serie de sistemas difusos para computación escalable.

# SISTEMAS DIFUSOS PARA COMPUTACIÓN EN BIG DATA

---

Conjuntos y lógica difusa



# Conjuntos difusos

## Definición (Conjunto difuso)

Un conjunto difuso  $A$  en  $X$  es el conjunto de pares ordenados

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

El conjunto  $A$  queda determinado por la función  $\mu_A : X \rightarrow [0, 1]$ , que asigna a cada elemento un **grado de pertenencia**.

Permiten modelar la imprecisión y la ambigüedad.

## Ejemplo

$A = \text{“una persona joven”}$ ,  $B = \text{“sobre 30 años de edad”}$ .

# Reglas difusas de tipo Si-Entonces

Consideramos  $A$  y  $B$  valores lingüísticos en dos universos  $X$  e  $Y$ , respectivamente. Estudiaremos reglas del tipo

Si  $x$  es  $A$  entonces  $y$  es  $B$ .

# Reglas difusas de tipo Si-Entonces

Consideramos  $A$  y  $B$  valores lingüísticos en dos universos  $X$  e  $Y$ , respectivamente. Estudiaremos reglas del tipo

Si  $x$  es  $A$  entonces  $y$  es  $B$ .

El razonamiento difuso se puede entender como un *modus ponens* generalizado.

$$\frac{\begin{array}{c} x \text{ es } A' \\ \text{si } x \text{ es } A \text{ entonces } y \text{ es } B \end{array}}{y \text{ es } B'}$$

Tenemos  $B' = A' \circ (A \rightarrow B)$ , donde  $\circ$  es un operador de composición.

# Sistemas de inferencia difusos

Un sistema de inferencia difuso recibe una entrada y produce una respuesta utilizando razonamiento difuso. Consta de cuatro componentes.

- Un **módulo de fuzzificación** con funciones de pertenencia para transformar la entrada en conjuntos difusos.
- Una **base de reglas** que contiene un conjunto de reglas difusas de tipo si-entonces.
- Un **mecanismo de razonamiento**, que realiza el procedimiento de inferencia.
- Un **mecanismo de defuzzificación** para producir una respuesta nítida (opcional).

**Mamdani.** Emplea reglas del tipo

si  $X_1$  es  $A_1$  y ... y  $X_n$  es  $A_n$   
entonces  $Y_1$  es  $B_1$  y ... y  $Y_m$  es  $B_m$ .

**Mamdani.** Emplea reglas del tipo

si  $X_1$  es  $A_1$  y ... y  $X_n$  es  $A_n$   
entonces  $Y_1$  es  $B_1$  y ... y  $Y_m$  es  $B_m$ .

**TSK.** Define reglas de la forma

si  $X_1$  es  $A_1$  y ... y  $X_n$  es  $A_n$   
entonces  $Y_1$  es  $f_1(X_1, \dots, X_n)$  y ... y  $Y_m$  es  $f_m(X_1, \dots, X_n)$ ,

donde cada  $f_i$  es una función nítida de la entrada.

# SISTEMAS DIFUSOS PARA COMPUTACIÓN EN BIG DATA

---

Fundamentos de Big Data

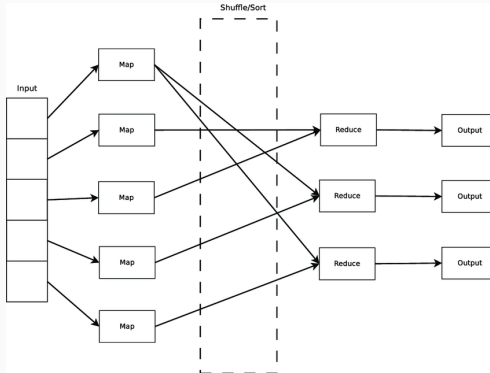
Utilizamos los datos para **resolver un problema**.

- Volumen
- Velocidad
- Variedad
- Veracidad
- Valor



# MapReduce y Apache Spark

Modelo de programación distribuida propuesto por Google en 2004.



**Figura 1:** Esquema de la arquitectura MapReduce.

# SISTEMAS DIFUSOS PARA COMPUTACIÓN EN BIG DATA

---

Diseño e implementación de algoritmos

# Algoritmo de Wang y Mendel

Propuesto por Wang y Mendel en 1992 para construcción de reglas a partir de datos.

1. Dividir el espacio de entrada en segmentos difusos.
2. Calcular la pertenencia de cada punto a todas las regiones, y quedarnos en cada caso con las de máxima pertenencia. Así formamos una regla difusa.
3. Calculamos un peso o *importancia* para cada regla: el producto de la pertenencia del punto a todas las regiones.
4. Simplificamos la base de datos eliminando duplicados y conflictos eligiendo siempre las reglas de mayor importancia.
5. Para la predicción, utilizamos el método de defuzzificación COA. Estamos resolviendo un problema de regresión.

# Implementación del algoritmo WM

**Etapas map.** Para cada punto calculamos la regla asociada y su importancia.

**Etapas reduce.** Agregamos todas las reglas, eliminando conflictos y duplicados.

```
val ruleBase = data.mapPartitions { case (x, y) =>
    val (ri, ro, degreeIn, degreeOut)
        = maxRegion(x, y, regionsIn, regionsOut)

    (ri, (ro, degreeIn * degreeOut))
}.reduceByKey { case (r1, r2) =>
    if (r1._2 > r2._2) r1 else r2
}.map { case (ri, (ro, _)) => (ri, ro) }
```

# Algoritmo subtractive clustering

Propuesto por S. Chiu en 1994 para encontrar número y valor inicial de centroides en clústering difuso.

**Idea:** cada punto es un posible centroide, y buscamos centroides suficientemente separados. Se asigna inicialmente a cada punto un potencial inversamente proporcional a la distancia a todos los demás.

# Algoritmo subtractive clustering

Propuesto por S. Chiu en 1994 para encontrar número y valor inicial de centroides en clústering difuso.

**Idea:** cada punto es un posible centroide, y buscamos centroides suficientemente separados. Se asigna inicialmente a cada punto un potencial inversamente proporcional a la distancia a todos los demás.

1. Se inicializa el potencial de cada punto y se elige aquel con mayor potencial como centroide.
2. Se recalculan los potenciales, que disminuyen de forma proporcional a la distancia al centroide elegido.
3. Se repite el proceso hasta que el potencial cae por debajo de un umbral.

# Versiones del algoritmo subtractive clustering

1. Versión **global**. Para inicializar el potencial se consideran todos los puntos, creando todas las posibles parejas.
2. Versión **local**. Se distribuyen los puntos en particiones y el algoritmo se aplica localmente. Después se concatenan los resultados.
3. Versión **híbrida**. Se aplica la versión local, se crean grupos de particiones y en ellos se aplica la versión global, utilizando los centroides como datos de entrada.

# Versiones del algoritmo subtractive clustering

1. Versión **global**. Para inicializar el potencial se consideran todos los puntos, creando todas las posibles parejas.
2. Versión **local**. Se distribuyen los puntos en particiones y el algoritmo se aplica localmente. Después se concatenan los resultados.
3. Versión **híbrida**. Se aplica la versión local, se crean grupos de particiones y en ellos se aplica la versión global, utilizando los centroides como datos de entrada.

Extensión a regresión: cada centroide es una regla difusa. Utilizamos defuzzificación COA.

Extensión a clasificación: asignar a cada punto la etiqueta del clúster con mayor pertenencia.



# Algoritmo Fuzzy C-Means

Propuesto por J. Dunn en 1973 y mejorado por J. Bezdek en 1981, para realizar clústering difuso.

El  $i$ -ésimo punto tendrá una pertenencia al  $j$ -ésimo clúster, digamos  $u_{ij}$ . Se pretende optimizar la función

$$J_m = \sum_x \sum_v u_{ij}^m \|x_i - v_j\|^2,$$

sujeta a las restricciones

$$u_{ij} > 0 \quad \forall i, j, \quad y \quad \sum_v u_{ij} = 1 \quad \forall i.$$

Se resuelve utilizando los multiplicadores de Lagrange para obtener unas reglas iterativas de actualización de la matriz de pertenencias y los centroides.

# Implementación del algoritmo FCM

$$u_{ij} = \frac{1}{\sum_{k=1}^c \left( \frac{\|x_i - v_j\|}{\|x_i - v_k\|} \right)^{\frac{2}{m-1}}}, \quad v_j = \frac{\sum_{i=1}^n u_{ij}^m x_i}{\sum_{i=1}^n u_{ij}^m}.$$

**Etapas map.** Se calculan para cada punto los valores necesarios para actualizar los centroides. No se mantiene en memoria la matriz de pertenencia.

**Etapas reduce.** Para cada centroide, se suman entre sí los valores obtenidos en la etapa anterior.

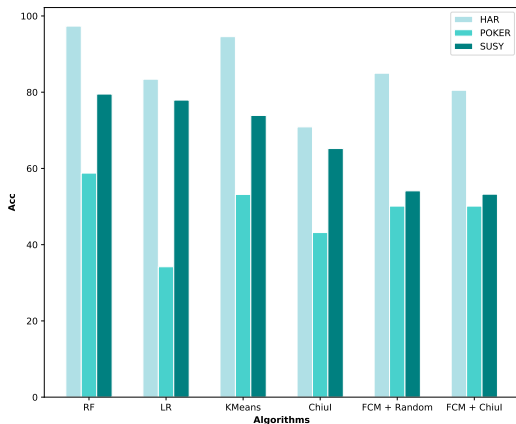
Extensión a clasificación: procedemos como en el algoritmo anterior. En este caso se asigna a cada clúster la etiqueta de la clase mayoritaria tras un conveniente  $\alpha$ -corte.

# SISTEMAS DIFUSOS PARA COMPUTACIÓN EN BIG DATA

---

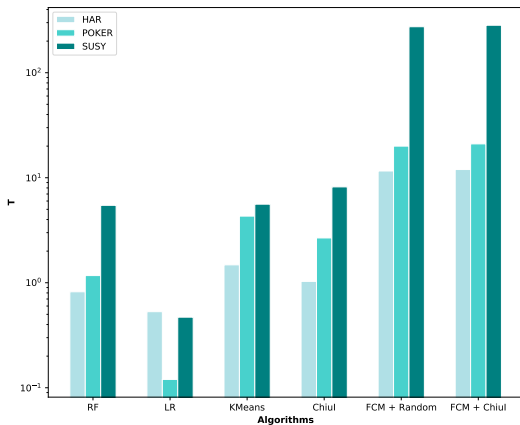
Estudio comparativo

# Comparación de precisión en clasificación



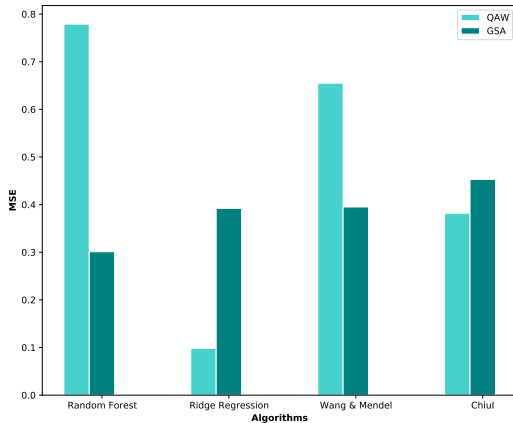
**Figura 2:** Comparación de la precisión obtenida en los tres conjuntos de datos elegidos para los algoritmos de clasificación.

# Comparación de tiempo en clasificación



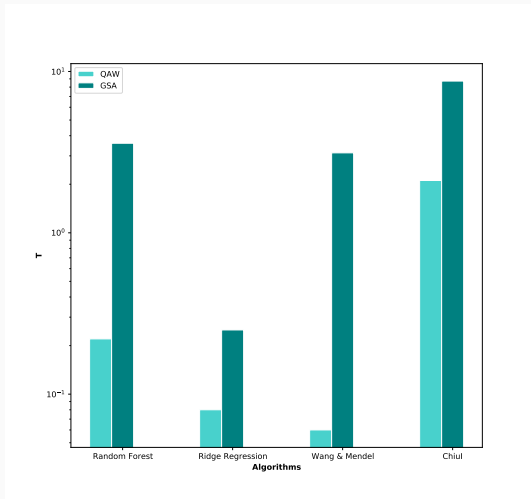
**Figura 3:** Comparación del tiempo de ejecución en los tres conjuntos de datos elegidos para los algoritmos de clasificación.

# Comparación de error en regresión



**Figura 4:** Comparación del Mean Squared Error en los dos conjuntos de datos elegidos para los algoritmos de regresión.

# Comparación de tiempo en regresión







**Figura 5:** Comparación del tiempo de ejecución en los dos conjuntos de datos elegidos para los algoritmos de regresión.

- Hemos realizado una aportación original desarrollando algoritmos escalables para aprendizaje de sistemas difusos, justificando su interés teórico y práctico.



- Hemos realizado una aportación original desarrollando algoritmos escalables para aprendizaje de sistemas difusos, justificando su interés teórico y práctico.
- Hemos comprobado que es necesario un diseño cuidadoso para adaptar los algoritmos al escenario Big Data. No basta una mera paralelización.

- Hemos realizado una aportación original desarrollando algoritmos escalables para aprendizaje de sistemas difusos, justificando su interés teórico y práctico.
- Hemos comprobado que es necesario un diseño cuidadoso para adaptar los algoritmos al escenario Big Data. No basta una mera paralelización.
- Nos hemos familiarizado con el modelo de referencia MapReduce y las herramientas del estado del arte en el Big Data, como Apache Spark y HDFS.

-  JANG, J. S. R., SUN, C. T., & MIZUTANI, E. *Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence*. (1997).
-  L.X. WANG, J. M. MENDEL. *Generating fuzzy rules by learning from examples*. (1992).
-  S. CHIU. *Fuzzy Model Identification Based on Cluster Estimation*. (1994).
-  JAMES C. BEZDEK ET AL. *Detection and Characterization of Cluster Substructure I. Linear Structure: Fuzzy c-Lines*. (1981).

## ECUACIONES NO RESOLUBLES CON RESPECTO A LA DERIVADA

---

Gracias por su atención