Universidad Central “Marta Abreu” de Las Villas

Facultad Matemática, Física y Computación

Licenciatura en Ciencias de la Computación

TRABAJO DE DIPLOMA

Título:

“Algoritmos para el aprendizaje de funciones de pertenencia en un sistema borroso basado en instancias”.

Autor:

Christian Ariel Isac Palma

Tutores:

Dr. Carlos Morell Pérez

Dra. Yanet Rodríguez Sarabia

Villa Clara

2010

“Año 52 de la Revolución”

El conocimiento es la virtud y sólo si se sabe se puede divisar el bien.

Sócrates

A mis padres,

por su guía, su entrega y su ejemplo

a lo largo del camino recorrido.

Agradecimientos.

A Dayli por su cariño y comprensión.

A Olga Lidia y José Manuel por su apoyo en la obtención de este sueño.

A mis abuelos, tíos y primos.

A mis tutores por su ayuda y dedicación.

A mis profesores por su entrega.

A mis compañeros por su apoyo y cooperación.

A todos los que han puesto su granito de arena para la realización de este trabajo.

A todos mi admiración y respeto.

Gracias

Resumen.

En los algoritmos de aprendizaje basados en instancias la función de similitud definida juega un papel determinante en el buen desempeño de los mismos. En el caso de un sistema borroso basado en instancias las funciones de pertenencia definidas para la modelación borrosa de los datos de entrada, inciden considerablemente sobre la función de similitud definida. En este trabajo se presenta una alternativa para incrementar la eficiencia de un sistema borroso basado en instancias, mediante la obtención de particiones borrosas óptimas.

La partición borrosa óptima es determinada usando el algoritmo de Máximo Descenso. Este algoritmo es invocado para modificar los parámetros de las funciones de pertenencia en la dirección negativa del gradiente de la función de error. Adicionalmente, este procedimiento es combinado con el Método de Monte Carlo para evitar mínimos locales durante el proceso de búsqueda. Los resultados obtenidos con este método son competitivos comparados con otros métodos de aprendizaje automatizados recogidos en la literatura.

Abstract.

Instance based learning algorithms are sensitive to the definition of similarity function used. The performance of the resulting classifier largely depends on this component. Fuzzy instance based classifiers are also dependent of membership functions used to define fuzzy partitions for modeling continuous attributes. This work proposes an algorithm for estimating the best fuzzy partition for a given dataset in order to improve classification accuracy. The optimal fuzzy partition is determined using a steepest descent algorithm. This algorithm is invoked to modify membership function parameters in the negative gradient direction of the error function. Additionally, this procedure is combined with the Monte Carlo method to avoid local minimum during search process. Results obtained with this method are competitive compared to other state of the arts machine learning algorithms.

Índice.

Introducción.

La noción de similitud o distancia en un espacio de características que describen la observación es la base de muchos métodos de reconocimiento de patrones y de aprendizaje automatizado. En el reconocimiento de patrones los métodos de vecinos cercanos son ejemplos de métodos basados en similitud, en inteligencia artificial los métodos basados en instancias y de razonamiento basado en casos trabajan bajo el supuesto que problemas similares tienen soluciones similares (Morell et al., 2006, Rodríguez et al., 2008). Además el éxito de su aplicación en un dominio particular depende en gran medida del criterio de similitud a emplear, cuya formalización es una tarea difícil, dependiente del contexto, y para la cual es conveniente aplicar también aprendizaje automático. Es por ello que definir funciones o métricas de similitud es una de las tareas más importantes cuando se desarrollan métodos como los anteriores.

Las funciones de similitud (o el concepto dual de distancia) han sido muy estudiadas, pero haciendo énfasis en su relación con las tareas de clasificación. Las mismas pueden ser categorizadas en aquellas que manejan datos de entrada ordinales, ya sean continuos o discretos, nominales o simbólicos y heterogéneos (Rodríguez et al., 2008).

Esta investigación está centrada en aquellas funciones que manejan atributos continuos. Según (Rodríguez et al., 2009) algunas funciones de similitud difusas comparadas con funciones de similitud no difusas ofrecen resultados al menos comparables, tomando en cuenta el desempeño del método de los vecinos cercanos en problemas de clasificación.

El comportamiento de un sistema borroso basado en instancias depende de las funciones de pertenencia definidas para la modelación borrosa de los datos de entrada, y a su vez estas funciones de pertenencia inciden sobre la función de similitud definida. Por tanto si se obtiene una partición borrosa óptima se debe aumentar la eficiencia del método de los vecinos cercanos en problemas de clasificación. Es por ello que se plantea como objetivo general:

Objetivo General

Confeccionar algoritmos que estimen la partición borrosa óptima de un conjunto de datos de manera que se maximice la precisión de un sistema borroso basado en instancias.

El cumplimiento de tal objetivo dependerá de la satisfacción de los siguientes objetivos específicos.

Objetivos Específicos.

- Confeccionar un algoritmo que a partir de un conjunto de datos estime la partición borrosa óptima, mediante la modificación de los parámetros de las funciones de pertenencia usando el método del Gradiente de Máximo Descenso para minimizar el error del desempeño del método de los vecinos cercanos.

- Utilizar modificaciones apropiadas del algoritmo para acelerar la convergencia del método de Máximo Descenso y para evadir con éxito la existencia de mínimos locales.

- Validar los resultados que arrojen estos nuevos algoritmos mediante la experimentación con datos de prueba disponibles.

Luego de realizar el marco teórico se puede establecer la siguiente hipótesis de investigación.

Hipótesis:

T_1: Un sistema borroso basado en instancias logra una mayor precisión cuando las funciones de pertenencia se obtienen siguiendo el algoritmo propuesto.

La tesis está estructurada en introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. En el Capítulo I se relacionan algunas consideraciones de carácter teórico sobre el aprendizaje basado en instancia, la lógica borrosa y los métodos de optimización. En el Capítulo II se construye un modelo para la estimación de particiones borrosas a partir de particiones uniformes utilizando el método de Máximo Descenso, métodos Quasi-Newton y aprendizaje basado en instancias. En el Capítulo III se formula un modelo para obtener particiones borrosas usando el método de Monte Carlo y se establece una comparación con métodos clásicos de aprendizaje automatizado.

Capítulo I: Sistemas borrosos basados en instancias y la optimización.

El concepto cognitivo de similitud está muy relacionado con la forma de proceder de los humanos en la solución de problemas en muchos dominios de aplicación. La utilización de este concepto se incluye en el desarrollo de métodos para la clasificación, aproximación, asociación, entre otros. La hipótesis de que "problemas similares tienen soluciones similares"; suposición que es usualmente referida como la hipótesis del razonamiento basado en similitud, es común a todos ellos.

Los modelos creados a partir de estos enfoques tienen el mismo principio: crear a partir de un conjunto de objetos conocidos un conjunto de objetos de referencia e introducir una medida de similitud permitiendo relacionar una nueva solicitud con los objetos de referencia. La probabilidad *Pr(C/q)* de asignar la clase *C* a un vector *q*, dado un modelo de clasificación *M*, depende de los procedimientos usados en la construcción del modelo y de sus parámetros adaptativos o grados de libertad. Consecuentemente, en el razonamiento basado en similitud la elección del criterio de similitud (o distancia) influye decisivamente en que el algoritmo de aprendizaje seleccione una salida generalizada sobre otra, y por tanto en su desempeño.

1.1 Aprendizaje basado en instancias.

Los algoritmos de aprendizaje basado en instancias (IBL) se caracterizan por tres comportamientos: almacenan todos los ejemplos de entrenamiento, difieren su procesamiento hasta que se presente una solicitud, y las solicitudes se responden por la combinación de ejemplos de entrenamiento seleccionados mediante un esquema de similitud y combinados en una función de predicción. Recientemente a estos algoritmos se refieren como el término "inferencia basada en casos". Bajo esta clasificación incluye la regla de los *k* vecinos más cercanos, el razonamiento basado en casos (RBC) y el IBL.

Los IBL pertenecen a la familia de algoritmos perezosos. En la definición de la función de predicción se siguen variantes simples como el voto mayoritario o el voto mayoritario pesado por la similitud, y alternativas como la regresión polinomial localmente pesada.

En un sistema IBL usado para la clasificación, una instancia $c = \{f_1, f_2, ..., f_n, t_c\}$, consiste en n características que describen el problema y su respectiva clase t_c. El conjunto $T = \{t_1, t_2, ..., t_l\}$ denota todas las posibles clases en el dominio. La base de casos (CB) se define como una colección de casos precedentes. La similitud entre la descripción de dos objetos d y q es computada por la expresión (1.1) donde $sim_a(q_a, c_a)$ es la medida de similitud local para la característica a. En general, se asume que $sim_a(q_a, c_a) \in [0,1]$ para todas las características, entonces la similitud entre las descripciones de dos objetos $sim(d, q) \in [0,1]$ (Wilke and Bergmann, 1996).

$$sim(d, q) = \frac{1}{n} \sum_{a=1}^{n} sim_a(d_a, q_a) \tag{1.1}$$

donde la similitud local sim_a es definida usualmente como:

$$\mathrm{sim_a}(\mathrm{d_a}, \mathrm{q_a}) = \begin{cases} 1 - |\,\mathrm{d_a} - q_a\,| & para \quad atributos \quad numéricos \\ 1 \quad si \quad d_a = q_a & \\ 0 \quad si \quad d_a \neq q_a & \end{cases} \tag{1.2}$$

El algoritmo de los k vecinos más cercano (k-NN), es la base de muchos métodos de aprendizaje perezosos. Toma como entrada un caso q de consulta y predice la clase del mismo. Para ello, se obtienen los k casos más similares al caso q de la CB, y se predice la clase del caso de consulta considerando la clase de los k vecinos (Morell et al., 2006, Wilke and Bergmann, 1996). Por ejemplo, se puede usar el método del voto mayoritario pesado por la similitud sobre el conjunto de vecinos como método de predicción de la clase de salida. O sea, sea $p_{q,t}$ la probabilidad de que el caso q pertenezca a la clase t, definido como:

$$p_{q,t} = \frac{\sum_{r \in K} \delta_{r,t} * sim(q, r)^2}{\sum_{r \in K} sim(q, r)^2} \tag{1.3}$$

donde *K* es el conjunto de los vecinos y $\delta_{r,t}$ se define como:

$$\delta_{r,t} = \begin{Bmatrix} 1 : t_r = t \\ 0 : t_r \neq t \end{Bmatrix} \tag{1.4}$$

y $t_r \in T$ denota la clase del caso *r* (Wilke and Bergmann, 1996). Entonces la predicción del CBR es la clase con la probabilidad más alta calculada desde *K*. Si hay empates se rompen aleatoriamente.

El éxito de la aplicación de estos algoritmos en un dominio particular depende en buena medida del criterio de similitud a emplear, cuya formulación es una tarea difícil, dependiente del contexto, y para la cual sería conveniente aplicar aprendizaje automático.

1.2 Los sistemas híbridos.

En la actualidad la tendencia es a desarrollar sistemas híbridos, como una etapa superior en el desarrollo de los sistemas basados en casos (SBC), con la finalidad de obtener un producto que aproveche las ventajas de los enfoques que combina y minimice sus deficiencias. La lógica borrosa y los conjuntos borrosos que se han desarrollado como una semántica para representar, manipular y utilizar la imprecisión presente en aplicaciones del mundo real; se han aplicado exitosamente en el desarrollo de los sistemas híbridos debido a su simplicidad y similitud con el razonamiento humano.

En (Morell et al., 2004) se presenta una variante donde se combina el aprendizaje basado en instancias y la lógica borrosa.

1.3 Lógica borrosa.

La idea de los conjuntos borrosos viene de la siguiente observación: las clases de objetos en la vida diaria no tienen límites bien definidos. De allí que la fuente de imprecisión sea la ausencia de criterios definidos rigurosamente sobre la pertenencia a clases, en lugar de la presencia de variables aleatorias. La noción de conjuntos borrosos es completamente de naturaleza no estadística.

Sea *X* un universo de objetos y *x* un objeto de *X*. Un conjunto clásico *A*, *A* contenido en *X*, se define como una colección de elementos de *x* que pertenece a *X*, tal que cada *x* pertenezca o no pertenezca al conjunto *A*. Definiendo una función característica para cada elemento *x* de *X*, se puede representar un conjunto clásico *A* por un conjunto de pares ordenados *(x,0)* o *(x, 1)*, que indica $x \notin A$ o $x \in A$, respectivamente.

Un conjunto borroso expresa el grado para el cual un elemento pertenece a un conjunto. La función característica de un conjunto borroso toma valores entre 0 y 1, lo que denota el grado de pertenencia de un elemento en dicho conjunto.

Definición 1.1: Conjunto borroso y función de pertenencia, tomada de (Jang Jyh-Shing et al., 1997).

Sea *X* una colección de objetos denotados genéricamente por *x*, entonces un **conjunto borroso** *A* en *X* está definido como el conjunto de pares ordenados:

$$A = \{(x, A(x)) \mid x \in X\} \tag{1.5}$$

donde *A* es la función de pertenencia (FP) para el conjunto borroso *A*. La función de pertenencia asigna a cada elemento de *X* un grado de pertenencia entre 0 y 1.

Es claro que la definición de conjunto borroso es una simple extensión de la definición de un conjunto clásico donde la función característica puede tomar valores entre 0 y 1.

Usualmente se refiere a *X* como el universo de discurso, o simplemente como universo, y consiste en un orden (desorden) discreto de objetos o un espacio continuo.

Es obvio que la construcción de un conjunto borroso depende de dos cosas: la identificación del universo de discurso y la especificación de la función de pertenencia apropiada. La especificación de la función de pertenencia es subjetiva, lo que significa que su definición para el mismo concepto por diferentes personas puede variar considerablemente.

En la práctica, cuando el universo de discurso *X* es un espacio continuo, usualmente se particiona *X* en varios conjunto borrosos. Estos conjuntos borrosos tienen nombres como “largo”, “medio” o

"pequeño", llamados valores lingüísticos. Al igual que una variable puede asumir diversos valores, una variable lingüística puede asumir diferentes valores lingüísticos.

Definición 1.2: Variable lingüística, tomada de (Jang Jyh-Shing et al., 1997).

Una variable lingüística se caracteriza por un quíntuplo *(x, T(x), X, G, M)* en el cual *x* es el nombre de la variable; *T* es el conjunto de términos, o sea, el conjunto de sus valores o términos lingüísticos; *X* es el universo de discurso; *G* es una regla sintáctica que genera los términos en *T* y *M* es una regla semántica que asocia con cada valor lingüístico *A* su significado *M(A)*, donde *M(A)* denota un conjunto borroso en *X*.

Denotaremos por *X* el conjunto universo de una variable lingüística y *F* la familia de los conjuntos borrosos definidos sobre *X*. El *j*-ésimo conjunto borroso definido sobre *X* con *j = 1..N* se representa por $A_j \in F(X)$ al cual se le hace corresponder una función de pertenencia A_j*: X→[0,1]* que denotaremos de igual manera. No se hacen distinciones entre conjuntos borrosos y su función de pertenencia.

Además considérese que $P=\{A_1, A_2, \ldots, A_N\}$ es una partición borrosa de acuerdo a:

Definición 1.2: Partición borrosa, tomada de (Rodriguez Sarabia, 2007).

Una partición borrosa algebraica sobre *X* es una familia finita $\{A_j\}_{j\in I}$ en *F* que satisface las propiedades siguientes: $\forall x \in X, \sum A_j(x) = 1$ y $A_j \neq \Phi$, para todo $j \in I$.

Definición 1.3: Soporte, tomada de (Jang Jyh-Shing et al., 1997).

El soporte de un conjunto borroso *A* es el conjunto de todos los elementos *x* en *X* tal que *A(x) > 0*:

$$Soporte(A) = \{x \mid A(x) > 0\} \tag{1.6}$$

Definición 1.4: Núcleo o centro, tomada de (Jang Jyh-Shing et al., 1997).

El núcleo o centro de un conjunto borroso *A* es el conjunto de todos los elementos *x* en *X* tal que *A = 1*:

$$Núcleo(A) = \{x \mid A(x)=1\} \tag{1.7}$$

Definición 1.5: Normalidad, tomada de (Jang Jyh-Shing et al., 1997).

Un conjunto borroso *A* es normal si su núcleo no es vacío.

Definición 1.6: Punto de cruce, tomada de (Jang Jyh-Shing et al., 1997).

Un punto de cruce de un conjunto borroso *A* es un elemento $x \in X$ tal que $A(x)= 0.5$:

$$Punto\ de\ cruce(A) = \{x \mid A(x)= 0.5\} \tag{1.8}$$

Definición 1.7: Borroso único, tomada de (Jang Jyh-Shing et al., 1997).

Un conjunto borroso cuyo soporte tiene un único elemento *x* en *X* con grado de pertenencia 1 ($A(x)=1$) se denomina borroso único.

Definición 1.8: α-corte y α-corte fuerte, tomada de (Jang Jyh-Shing et al., 1997).

El α-corte o nivel α de un conjunto borroso *A* es un conjunto duro definido por:

$$A\alpha = \{x \mid A(x) \geq \alpha\} \tag{1.9}$$

El α-corte fuerte se define similarmente:

$$A\alpha = \{x \mid A(x) > \alpha\} \tag{1.10}$$

Definición 1.9: Convexidad, tomada de (Jang Jyh-Shing et al., 1997).

Un conjunto borroso *A* es convexo si y solo si, para cualquier $x_1, x_2 \in X$ y cualquier $\lambda \in [0,1]$,

$$A(\lambda x_1 + (1-\lambda)x_2) \geq min(A(x_1), A(x_2)) \tag{1.11}$$

Definición 1.10: Ancho de banda de conjuntos borrosos normales y convexos, tomada de (Jang Jyh-Shing et al., 1997).

Para un conjunto borroso normal y convexo A, el ancho de banda o amplitud se define como la distancia entre los dos únicos puntos de cruce:

$$Amplitud(A) = |x_2 - x_1|$$

donde $A(x_1) = A(x_2) = 0.5$.

Definición 1.11: Simetría, tomada de (Jang Jyh-Shing et al., 1997).

Un conjunto borroso A es simétrico si su función de pertenencia es simétrica respecto a un punto $x = c$, donde:

$A(c + x) = A(c - x)$ para todo $x \in X$.

Definición 1.12: Abierto a la izquierda, abierto a la derecha y cerrado, tomada de (Jang Jyh-Shing et al., 1997).

Un conjunto borroso A es abierto a la izquierda si $\lim_{x\to-\infty} A(x) = 1$ y $\lim_{x\to\infty} A(x) = 0$; abierto a la derecha si $\lim_{x\to-\infty} A(x) = 0$ y $\lim_{x\to\infty} A(x) = 1$; y cerrado si $\lim_{x\to-\infty} A(x) = \lim_{x\to\infty} A(x) = 0$.

1.3.1 Operaciones sobre conjuntos borrosos.

Definición 1.13: Subconjunto, tomada de (Jang Jyh-Shing et al., 1997).

El conjunto borroso A está contenido en el conjunto borroso B (es un subconjunto de B) o A es menor o igual que B, si y solo si, $A \leq B$ para todo x.

$$A \subseteq B \Leftrightarrow A(x) \leq B(x) \tag{1.12}$$

Definición 1.14: Unión (disyunción), tomada de (Jang Jyh-Shing et al., 1997).

La unión de dos conjuntos borrosos A y B es un conjunto borroso C ($C = A \cup B$), cuya función de pertenencia se define como:

$$C(x) = \max(A(x), B(x)) = A(x) \vee B(x) \tag{1.9}$$

Definición 1.15: Intersección (conjunción), tomada de (Jang Jyh-Shing et al., 1997).

La intersección de dos conjuntos borrosos A y B es un conjunto borroso C ($C = A \cap B$), cuya función de pertenencia se define como:

$$C(x) = \min(A(x), B(x)) = A(x) \wedge B(x) \tag{1.13}$$

Definición 1.16: Complemento (negación), tomada de (Jang Jyh-Shing et al., 1997).

El complemento de un conjunto borroso A es un conjunto borroso $\overline{A}$ ($\neg A$), cuya función de pertenencia se define como:

$$\overline{A}(x) = 1 - A(x) \tag{1.14}$$

Definición 1.17: Producto cartesiano y co-producto cartesiano, tomada de (Jang Jyh-Shing et al., 1997).

Sean A y B conjuntos borrosos en X e Y respectivamente. El producto cartesiano de A y B, denotado por AxB, es un conjunto borroso en el espacio producto XxY, con la función de pertenencia:

$$AxB(x, y) = min(A(x), B(y)) \tag{1.15}$$

Similarmente, el co-producto cartesiano $A + B$ es un conjunto borroso con la función de pertenencia:

$$A+B(x, y) = max(A(x), B(y)) \tag{1.16}$$

Para extender los operadores de unión e intersección se utilizan t-normas y t-conormas respectivamente.

Definición 1.23: *t*-norma, tomada de (Jang Jyh-Shing et al., 1997).

Sea un operador $T: [0,1] \times [0,1] \rightarrow [0,1]$, T es una norma triangular (*t*-norma) si satisface:

$$\begin{aligned} &T(x,y) \leq T(x,z) \text{ si } y \leq z && \text{(Monotonía)} \\ &T(x,y) = T(y,x) && \text{(Conmutatividad)} \\ &T(x\ T(y,z)) = T(T(x,y), z) && \text{(Asociatividad)} \\ &T(0,0) = 0,\ T(x,\ 1) = x && \text{(Acotamiento)} \end{aligned} \tag{1.17}$$

Las *t*-normas básicas son mínimo (M), producto (P), Lukasiewicsz (L):

$$\begin{aligned} &\text{Mínimo:} && T_M(a,b) = min(a,b) \\ &\text{Producto algebraico:} && T_P(a,b) = ab \\ &\text{Lukasiewicsz:} && T_L(a,b) = max(a+b-1,\ 0) \end{aligned} \tag{1.18}$$

Como operador dual de la *t*-norma se introduce la *t*-conorma.

Definición 1.24: *t*-conorma (*s*-norma), tomada de (Jang Jyh-Shing et al., 1997).

Sea un operador $S: [0,1] \times [0,1] \rightarrow [0,1]$, S es una conorma triangular (*t*-conorma) si satisface:

$$\begin{aligned} &S(x,y) \le S(x,z) \text{ si } y \le z && \text{(Monotonía)} \\ &S(x,y) = S(y,x) && \text{(Conmutatividad)} \\ &S(x, S(y,z)) = S(S(x,y), z) && \text{(Asociatividad)} \\ &S(1,1) = 1,\ S(x,0) = x && \text{(Acotamiento)} \end{aligned} \tag{1.19}$$

A continuación se muestran la *t*-conormas correspondientes a la *t*-normas definidas en (1.18):

$$\begin{aligned} &\text{Máximo:} && S_M(a,b) = max(a,b) \\ &\text{Suma algebraica:} && S_P(a,b) = a + b - ab \\ &\text{Lukasiewicsz:} && T_L(a,b)=a + b - max(a+b-1, 0) \end{aligned} \tag{1.20}$$

1.3.2 Funciones de pertenencia de una dimensión.

Las funciones de pertenencias se pueden representar explícitamente, analíticamente o gráficamente. A continuación se definen analíticamente varias funciones de pertenencia de las más usadas.

Definición 1.18: Función de pertenencia triangular, tomada de (Jang Jyh-Shing et al., 1997).

La función de pertenencia triangular es especificada por tres parámetros *{a, b, c}*, con $a < b < c$, como sigue:

$$Triangular(x,a,b,c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \tag{1.21}$$

Usando mínimo y máximo se tiene otra alternativa para especificar la fórmula precedente:

$$Triangular(x,a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right) \tag{1.22}$$

Otra forma de representar esta función es la siguiente:

$$Triangular(x,sl,c,sr) = \begin{cases} 1 - \dfrac{|x-c|}{sr} & si \quad x \in [c, c+sr] \\ 1 - \dfrac{|x-c|}{sl} & si \quad x \in [c-sl, c] \\ 0 & e.o.c. \end{cases} \tag{1.23}$$

donde *c* es el centro, *sl* y *sr* son el ancho hacia la izquierda y hacia la derecha respectivamente.

Definición 1.19: Función de pertenencia trapezoidal, tomada de (Jang Jyh-Shing et al., 1997).

La función de pertenencia trapezoidal es especificada por cuatro parámetros *{a, b, c, d}*, con $a < b < c < d$, como sigue:

$$Trapezoidal(x,a,b,c,d) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{c-x}, & c \le x \le d \\ 0, & d \le x \end{cases} \tag{1.24}$$

Usando mínimo y máximo se tiene otra alternativa para especificar la fórmula precedente:

$$Trapezoidal(x,a,b,c,d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right) \tag{1.25}$$

Otra forma de representar esta función es la siguiente:

$$Trapezoidal(x,sl,c,sr)=\begin{cases}1-\dfrac{|x-b|}{sl} & si \quad x\in[b-sl,b]\\ 1 & si \quad x\in(b,c)\\ 1-\dfrac{|x-c|}{sr} & si \quad x\in[c,c+sl]\\ 0 & e.o.c.\end{cases} \tag{1.26}$$

donde *b* y *c* son centros, *sl* y *sr* son el ancho hacia la izquierda y hacia la derecha respectivamente.

Definición 1.20: Función de pertenencia gaussiana, tomada de (Jang Jyh-Shing et al., 1997).

La función de pertenencia gaussiana es especificada por dos parámetros *{c, σ}*; *c* representa el centro y *σ* determina el ancho de la función:

$$Gaussiana(x,c,\sigma)=e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \tag{1.27}$$

Definición 1.21: Función de pertenencia campana generalizada, tomada de (Jang Jyh-Shing et al., 1997).

La función de pertenencia campana generalizada es especificada por tres parámetros *{a, b, c}*:

$$Bell(x,a,b,c)=\frac{1}{1+\left|\dfrac{x-c}{a}\right|^{2b}} \tag{1.28}$$

donde el parámetro *b* es usualmente positivo.

Definición 1.22: Función de pertenencia sigmoidal, tomada de (Jang Jyh-Shing et al., 1997).

La función de pertenencia sigmoidal está definida por:

$$Sigmoidal(x,a,c) = \frac{1}{1+e^{-a(x-c)}} \tag{1.29}$$

donde *a* controla la pendiente en el punto de cruce $x = c$.

Los métodos más comunes empleados en la construcción de funciones de pertenencia se basan con frecuencia en algunas de las siguientes estrategias:

- Evaluación subjetiva y construcción a partir de expertos, donde los especialistas en el dominio de aplicación simplemente dibujan o especifican diferentes curvas de pertenencia de las cuales eligen una tomando como base su propia experiencia.

- Frecuencias convertidas en probabilidades.

- Aprendizaje y adaptación, tomando como base la aplicación de las técnicas de aprendizaje automático.

La primera de ellas requiere que los expertos en el dominio de aplicación sean capaces de entender hasta cierto punto la teoría de la borrosidad, elemento que no siempre se puede lograr cuando se desarrolla una aplicación. Además es una estrategia que depende mucho del grado de expertisidad de los especialistas.

Debemos señalar que los diferentes enfoques utilizados con frecuencia están condicionados a la naturaleza de los datos con que se trabaja. Generalmente la segunda estrategia aparece asociada al tratamiento de datos simbólicos mientras que la tercera aparece asociada a datos numéricos.

Otros autores para el caso del tratamiento de datos simbólicos proponen la construcción de funciones de pertenencia en función del cálculo de las distancias entre los términos lingüísticos y basan la determinación de estas distancias en las frecuencias relativas de aparición de cada par de términos lingüísticos. Este último enfoque permite una suavidad mayor de los conjuntos borrosos asociados a las variables simbólicas, aspecto que se pierde en el resto de los enfoques para el tratamiento de este tipo de variables, y que constituye una de las bondades fundamentales que incorpora la lógica borrosa.

Algunos métodos que se reportan en la bibliografía podrían requerir de la participación de expertos al menos en la etapa inicial de la generación de las funciones de pertenencia. Un ejemplo de ello es el método basado en la interpolación que propone Joseph Chen. Como premisa para la aplicación de este método es necesario conocer la pertenencia para un conjunto finito de puntos, información que podría ser suministrada por un experto. A este método se le señala que aplicando métodos de interpolación basados en los mínimos cuadrados y los spline no siempre se logra construir buenas funciones de pertenencia (Pieñero Pérez, 2004).

Otros autores proponen métodos basados en la determinación de los centros de gravedad de los términos lingüísticos de una variable lingüística. En estos centros de gravedad la función de pertenencia alcanza el máximo valor. Para determinar el grado de pertenencia de un nuevo elemento se define una función que depende de la distancia entre el nuevo elemento y el centro de gravedad del conjunto borroso (Pieñero Pérez, 2004).

Hong (Hong and Lee, 1998) propone un método de aprendizaje para derivar automáticamente funciones de pertenencia desde un conjunto dado de casos. Este algoritmo genera en su primer paso un conjunto inicial de funciones que luego simplifica a partir de operaciones de unión y absorción entre ellas. Este algoritmo presenta como principales desventajas que solo permite generar funciones triangulares y que en la primera fase genera demasiadas funciones de pertenencia elevando significativamente la complejidad computacional del método.

En ocasiones las funciones de pertenencia que se quieren obtener son funciones con propiedades deseables para el análisis, por ejemplo continuas, derivables, etc. En estos casos generalmente es factible aplicar métodos numéricos clásicos o simples de análisis matemático que permitan determinar parámetros iniciales de las funciones de pertenencia y luego refinar éstos aplicando técnicas de optimización. Estas técnicas generalmente imponen como restricciones la necesidad de la derivabilidad de la funciones y además muchas de ellas sólo garantizan encontrar óptimos locales y no globales del espacio de búsqueda (Pieñero Pérez, 2004).

1.4 Medidas de similitud difusa.

Muchas medidas de similitud difusas han sido propuestas en la literatura (Rodríguez et al., 2009). El estudio de las mismas se divide en dos grupos: aquellas obtenidas a través de consideraciones métricas, tomando en cuenta la relación distancia-similitud y las obtenidas de la teoría de conjuntos.

Basados en la generalización de un modelo de distancia geométrico se obtienen la función de similitud de Manhattan a partir de la distancia de Manhattan definida como:

$$S(x,y) = 1 - \frac{1}{b}\sum_{i=1}^{b} | A_i(x) - A_i(y) | \tag{1.30}$$

Basado en la distancia de Canberra se obtiene la función de similitud definida como:

$$S(x,y) = 1 - \frac{\sum_{i=1}^{b} | A_i(x) - A_i(y) |}{\sum_{i=1}^{b} | A_i(x) + A_i(y) |} \tag{1.31}$$

Basados en la teoría de conjuntos se obtienen otras medidas de similitud, tal es el caso de la función de similitud difusa de Jaccard definida como:

$$S(x,y) = 1 - \frac{\sum_{i=1}^{b} T(A_i(x), A_i(y))}{\sum_{i=1}^{b} \perp (A_i(x), A_i(y))} \tag{1.32}$$

donde T es una t-norma y $\perp$ es la respectiva t-conorma.

1.5 Métodos de optimización.

Optimizar significa mejorar, perfeccionar; sin embargo, en el contexto científico la optimización es el proceso de tratar de encontrar la mejor solución posible para un determinado problema.

Los métodos de optimización numérica pueden ser clasificados en dos categorías principales:

- Métodos basados en derivadas.

- Métodos de búsqueda directa o libres de derivadas.

1.4.1 Métodos descendentes.

Los métodos descendentes permiten optimizar una función *E* definida en un espacio *n*-dimensional a través de un búsqueda iterativa eficientemente (Jang Jyh-Shing et al., 1997). Es decir, se hace una búsqueda punto a punto, donde el punto siguiente θ_{k+1} es determinado por un paso desde el punto actual θ_k en la dirección del vector *d*:

$$\theta_{k+1} = \theta_k + \lambda d \tag{1.33}$$

donde λ es un valor positivo que indica cuán grande es el salto de un punto a otro en la dirección *d*. Este nuevo punto debe satisfacer la siguiente inecuación:

$$E(\theta_{k+1}) < E(\theta_k) \tag{1.34}$$

La principal diferencia entre varios métodos descendentes es la forma en cómo determinan las sucesivas direcciones.

1.4.2 Métodos descendentes basados en el gradiente.

Cuando la dirección de descenso *d* es determinada por el gradiente de la función objetivo *E*, estos métodos de descenso son llamados métodos descendentes basados en el gradiente (Jang Jyh-Shing et al., 1997).

El gradiente de una función diferenciable $E: R^n \rightarrow R$ con respecto a θ es el vector de las primeras derivadas de *E* denotado por *g*:

$$g(x) = \left[\frac{\partial E(\theta)}{\partial \theta_1}, \frac{\partial E(\theta)}{\partial \theta_2}, ..., \frac{\partial E(\theta)}{\partial \theta_n} \right]^T \tag{1.35}$$

En estos métodos, generalmente, la dirección de descenso se obtiene desviando los gradientes a través de una multiplicación por una matriz definida positiva *G* para acelerar la velocidad de convergencia:

$$\theta_{k+1} = \theta_k + \lambda G g \tag{1.36}$$

1.4.3 Método del Gradiente de Máximo Descenso.

El método del Gradiente de Máximo Descenso es una de las técnicas más viejas de optimización para minimizar una función *E* en un espacio multidimensional, y uno de los métodos más usados a pesar de su lenta convergencia (Jang Jyh-Shing et al., 1997). En este método la dirección de descenso está determinada por:

$$d = - g \tag{1.37}$$

Luego la ecuación (1.33) se transforma en:

$$\theta_{k+1} = \theta_k - \lambda g \tag{1.38}$$

1.4.4 Método de Newton.

La idea fundamental en la que se basa este método es la siguiente: la función a minimizar $E(\theta)$ se aproxima en una vecindad del punto de iteración θ_k por una función cuadrática $q_k(\theta)$. La nueva aproximación θ_{k+1} de la solución θ^* se define como el punto en que $q_k(\theta)$ alcanza su valor mínimo. Para ello se define la función $q_k(\theta)$ como:

$$q_k(\theta) = E(\theta_k) + g(\theta_k)(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^t H(\theta_k)(\theta - \theta_k) \tag{1.39}$$

y θ_{k+1} como el punto donde se anula la derivada de $q_k(\theta)$, o sea:

$$\theta_{k+1} = \theta_k - H(\theta_k)^{-1} g(\theta_k) \tag{1.40}$$

donde *H* es la matriz hessiana, formada por las segundas derivadas parciales de la función *E*.

Esta expresión describe una iteración del método de Newton.

El método de Newton puede no converger, porque en un punto de iteración θ_k la matriz $H(\theta_k)$ sea singular. Pudiera también ocurrir que la dirección que utiliza el método no sea de descenso, o aún siéndolo, que un paso unitario no produzca un decrecimiento de la función objetivo. No obstante se han hecho modificaciones a este método para eliminar estas dificultades y conservar en lo posible sus buenas propiedades de convergencia local.

1.4.5 Método Quasi-Newton.

Una de las críticas fundamentales que se le hace al método de Newton y sus modificaciones es la dificultad del cálculo de las segundas derivadas y la inversión de la hessiana de la función *E* a minimizar, evaluada en el punto de iteración. La formulación general de estos métodos responde a la del método de Newton con búsqueda direccional, pero la inversa de la matriz hessiana $H(\theta_k)$ se sustituye por una aproximación B_k de dicha matriz, que se construye utilizando solamente información de las primeras derivadas de *E* y se actualiza en cada iteración mediante una fórmula de corrección. Idealmente, las aproximaciones convergen a la inversa del hessiano en el punto solución. La siguiente fórmula describe una iteración de este algoritmo.

$$\theta_{k+1} = \theta_k - \lambda B g \tag{1.41}$$

Inicialmente *B* puede ser cualquier matriz simétrica y definida positiva. A menudo se usa la matriz idéntica. Lo más importante en estos métodos es cómo se modifica la matriz *B* en cada iteración. Dos esquemas de modificación ampliamente usados son las fórmulas de modificación de Davidon-Fletcher-Powell (DFP) y las de Broyden-Fletcher-Goldfard-Shanno (BFGS) (Jang Jyh-Shing et al., 1997). Se conoce que BFGS es generalmente más tolerante a la inexactitud de la búsqueda lineal que DFP.

1.4.6 Métodos de búsqueda directa o libres de derivadas.

Los métodos de búsqueda directa usan solamente una función de error para guiar el proceso de búsqueda. Estos métodos no necesitan información de la derivada de la función objetivo para buscar un conjunto de parámetros que la minimicen (o maximicen). Se basan en repetir evaluaciones de la función objetivo y en la subsiguiente búsqueda de direcciones, después de cada evaluación siguiendo una determinada heurística que los guía. Generalmente son más lentos que los métodos basados en derivadas para problemas de optimización continua. Se caracterizan por una gran flexibilidad ya que la función objetivo puede ser tan compleja como se quiera. La mayoría de estos métodos son estocásticos, lo que significa que usan generadores de números aleatorios para determinar las subsecuentes direcciones de búsqueda. Esta naturaleza aleatoria le ofrece la probabilidad de encontrar una solución óptima. Entre los más populares podemos encontrar: el método de Monte Carlo, el método de búsqueda aleatoria, el recocido simulado, etc.

Muchos de ellos generalmente emplean más de una solución en cada iteración. La presencia de múltiples soluciones es explotada para encontrar mejores regiones de búsqueda. Estos métodos son llamados usualmente algoritmos basados en poblaciones. Dentro de esta gran familia de algoritmos los más usados han sido los algoritmos genéticos, la Optimización Basada en Enjambre de Partículas, etc.

El método de Monte Carlo es un método no determinístico o estadístico numérico usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud. El método se llamó así en referencia al Casino de Monte Carlo (Principado de Mónaco) por ser "la capital del juego de azar", al ser la ruleta un generador simple de números aleatorios. El nombre y el desarrollo sistemático de los métodos de Monte Carlo datan aproximadamente de 1944 y se mejoraron enormemente con el desarrollo de la computadora. El mismo se basa en buscar aleatoriamente cada vez valores menores de la función objetivo. Es decir se obtiene una estimación estadística del vector solución con una determinada distribución matemática, generalmente distribución uniforme en un determinado subconjunto de los números reales (Tan and Alexandrov, 2001).

Capítulo II: Estimación de parámetros de funciones de pertenencia a partir de particiones uniformes.

Muchas veces las métricas locales como la definida en la ecuación (1.2) no son satisfactorias. Por ejemplo, considere un sistema para el diagnóstico de enfermedades donde una característica típica es la temperatura corporal. Considere adicionalmente, un caso de consulta q, con valor $q_a = 38.2$ en el atributo temperatura corporal, y dos instancias x e y de la CB que contienen $x_a = 37.7$ y $y_a = 39$. Según la ecuación (1.2) x_a es más similar a q_a que y_a, mientras que un experto puede decirnos que 38.2 y 39 son más similares porque las dos temperaturas pueden ser consideradas como fiebre media (Morell et al., 2004).

2.1 Modelando variables numéricas como variables lingüísticas.

Usualmente las variables son numéricas (cuyo dominio es un subconjunto de los números reales) por lo que debemos modelarlas como variables lingüísticas. No haremos distinción entre variable y atributo.

Para trabajar con atributos numéricos A_i se asume que el dominio $dom(A_i)$ es equipado con una partición borrosa. De este modo, el dominio $dom(A_i)$ de los atributos numéricos es descrito por el significado de un conjunto de propiedades borrosas $F^1,...,F^{n_i}$, donde cada una puede ser vista como una atributo $A_i^j \in [0,1]$, tal que $A_i^j(x) = F^j(A_i(x))$. Cada atributo A_i^j es llamado atributo borroso (Rodríguez et al., 2009).

En el caso de atributos discretos, donde $dom(A_i) = \{V^1, V^2, ..., V^k\}$, se pueden reemplazar estos por atributos binarios $A_i^1, A_i^2, ..., A_i^k$. Cada A_i^j es identificado por la propiedad $A_i^j(x) = 1$ si $A_i(x) = V^j$ (Rodríguez et al., 2009).

Ejemplo:

A continuación se presenta una partición borrosa de los datos de la base de casos Iris. La misma contiene 3 clases cada una con 50 ejemplos, donde cada clase se refiere a un tipo de planta (Setosa, Versicolor y Virginica). Tiene cuatro atributos (largo y ancho del pétalo y el sépalo, en centímetros).

En la figura 2.1 se presenta gráficamente la partición borrosa para uno de sus atributos, modelado con tres términos lingüísticos: bajo, medio y alto.

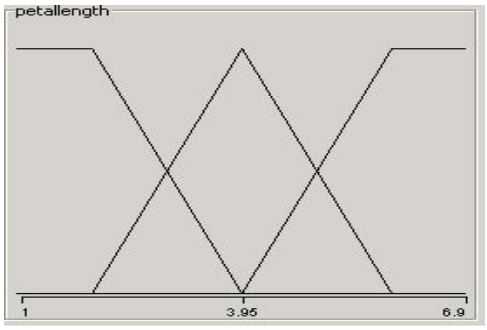

Figura 2.1. Partición borrosa de la base de casos Iris

En la tabla 2.1 se presentan los valores numéricos para dos atributos y la respectiva clase de tres casos y en la tabla 2.2 las valores correspondientes a la partición borrosa de los datos para los valores de la tabla 2.1.

Largo del sépalo	Ancho del sépalo	Clase
5.1	3.5	Iris-setosa
7.0	3.2	Iris-versicolor
6.3	3.3	Iris-virginica

Tabla 2.1. Valores de atributos de la base de casos Iris

Largo del sépalo = Bajo	Largo del sépalo = Medio	Largo del sépalo = Alto	Ancho del sépalo = Bajo	Ancho del sépalo = Medio	Ancho del sépalo = Alto	Clase
0.57	0.42	0.03	0	0.79	0.21	Iris setosa
0	0.5	0.5	0	1	0	Iris versicolor
0	0.96	0.04	0.01	0.98	0.01	Iris virginica

Tabla 2.2. Valores de la partición borrosa.

En (Rodríguez et al., 2009) se presenta un estudio donde se comparan varias medidas de similitud difusas, por lo que en este trabajo se toman dos de ellas por sus buenos resultados: la función de similitud de Manhattan y la función de similitud de Jaccard definidas en (1.30) y (1.32) respectivamente.

A continuación se presenta un experimento para medir el desempeño de estas funciones de similitud con distintas funciones de pertenencia. Se usan específicamente funciones de pertenencia gaussianas y triangulares.

Para realizar cada experimento se usó el WEKA (Witten and Frank, 1999). En cada uno se utilizan 15 bases de casos del repositorio de la Universidad de California (UCI). Estas bases de casos están caracterizadas por varios atributos numéricos predictivos y un atributo discreto de salida (Anexo 1). Cada prueba consiste en cinco corridas, cada una usando una de cinco particiones de los datos, seleccionada aleatoriamente de la base de casos (5-fold cross-validation). Es decir, cada corrida se basa en construir un conjunto de entrenamiento usando el 80% de los datos y evaluando el desempeño en el conjunto de pruebas (el 20% restante de los datos). Una simple corrida no da un estimado real del error. Por ello se repite cada prueba dos veces.

Para verificar si las diferencias observadas entre distintos algoritmos son estadísticamente significativas, se aplican distintas pruebas, específicamente pruebas no paramétricas, debido a que los resultados obtenidos en los experimentos no cumplen las condiciones requeridas para poder usar comparaciones paramétricas.

A continuación se resume el proceso a seguir para identificar si hay diferencias estadísticas. Primero aplicamos pruebas de comparación de múltiples poblaciones para probar la hipótesis nula de que todos los algoritmos de aprendizaje obtienen los mismos resultados en el promedio de los valores. Específicamente, se usa la prueba no paramétrica de Friedman. Si la prueba de Friedman rechaza la hipótesis nula, entonces se aplican otras pruebas. En este caso se aplica la prueba de Bonferroni-Dunn, la cual define que un método de aprendizaje tiene diferencias significativas del clasificador de mayor rango si el correspondiente promedio difiere, al menos, una distancia crítica. Para complementar el análisis estadístico, se usa el procedimiento de paso abajo de Holm porque se dice que la prueba de Bonferroni-Dunn es menos poderosa (Demsar, 2006, Rodríguez et al., 2009). La

prueba de Bonferroni-Dunn se aplica con un margen de error del 10% y la prueba de Holm con un margen del 5%. Se usa un valor alto de significación en la prueba de Bonferroni-Dunn porque ésta es más conservativa que la prueba de Holm (Orriols-Puig et al., 2008). En caso de que sea necesario establecer una comparación entre dos algoritmos, se utiliza la prueba de Wilcoxon (Demsar, 2006).

En la validación estadística se analizan las siguientes hipótesis:

H_0: No existen diferencias entre las funciones de similitud con distintas funciones de pertenencia.

H_1: Existen diferencias entre las funciones.

Para este experimento se utiliza el clasificador Ibk implementado en WEKA, con un *k* constante e igual a 3. Para obtener la partición borrosa para cada atributo se usa un algoritmo de discretización de intervalos de igual ancho, asegurando que dos funciones consecutivas se crucen en el límite del intervalo y que cada conjunto borroso tenga como soporte la mitad del ancho del intervalo, lográndose así **particiones borrosas uniformes**. Para todo el experimento se usan cinco términos lingüísticos.

La tabla 2.3 muestra los resultados experimentales para todas las bases de caso. En este experimento, el desempeño es medido como el por ciento de clasificación correcta de las instancias, y es obtenido para cada base de caso y cada algoritmo de aprendizaje.

Para verificar si las diferencias observadas son estadísticamente significativas se aplica la prueba no paramétrica de Friedman. Esta prueba acepta la hipótesis nula de que no hay diferencias significativas entre las distintas funciones de similitud con $p = 0.1162$.

Las funciones de pertenencia son descritas por parámetros, los que pueden ser optimizados con respecto a una medida global de error en un proceso de aprendizaje (Ichihashi, 1991, Nomura et al., 1992). Es por ello que se plantea que si se optimizan los parámetros de las funciones de pertenencia usadas en la obtención de la partición borrosa de los datos se puede obtener un mejor desempeño.

B.C.	EWFGM	EWFGJ	EWFTM	EWFTJ
1. Balance-scale	81.01	**83.61**	78.15	81.75
2. Wisconsin-breast-cancer	96.81	**97.14**	96.56	96.38
3. Pima_diabetes	69.88	70.19	**72.6**	**72.6**
4. Glass	73.04	58.5	**77.27**	62.29
5. Heart-statlog	**82.33**	79.85	81.33	81.44
6. Ionosphere	90.65	**90.97**	90.57	91.48
7. Iris	94.67	94.8	94.8	**95.2**
8. Liver-disorders	**67.13**	61.02	61.97	53.99
9. New-thyroid	**96.42**	91.3	95.52	91.48
10. Sonar	81.26	82.03	**86.98**	85.68
11. Tao	95.63	85.51	**96.17**	83.68
12. Vehicle	**70.9**	65.36	70.68	67.55
13. Vowel	96.01	92.84	**96.41**	95.11
14. Wisconsin Diag. breast-cancer	96.51	95.61	**96.54**	95.96
15. Wine	95.4	94.94	96.59	**96.98**
Rango	2.25	3.09	2.06	2.59
Posición	2	4	1	3

Tabla 2.3. Resultados obtenidos con el *k*-NN y distintas funciones de similitud difusas.

Consecuentemente, obtener altos grados de interpretabilidad y precisión es una tarea contradictoria en la modelación difusa, y en la práctica una de las dos prevalece sobre la otra. No obstante existe una nueva tendencia en la comunidad científica encaminada a lograr un balance entre la interpretabilidad y la precisión de un sistema difuso (Casillas et al.). El objetivo de este trabajo es lograr particiones borrosas con altos grados de interpretabilidad y precisión.

2.2 Modificación de los parámetros de las funciones de pertenencia usando el método del Gradiente de Máximo Descenso.

La idea que se maneja es obtener los parámetros de las funciones de pertenencia y luego aplicar el método del Gradiente de Máximo Descenso para así obtener los mejores parámetros que ajusten cada función de pertenencia. Sea θ un parámetro de una función de pertenencia y $E(\theta)$ alguna función de error para ser minimizada, el método del Gradiente de Máximo Descenso actualiza θ según (1.38).

Para ello se debe definir la función de error y las derivadas de cada uno de los parámetros a modificar con respecto a la misma. Se utiliza el error cuadrático definido como:

$$E = \sum_{q \in CB} \sum_{t \in T} (\delta_{q,t} - p_{q,t})^2 \tag{2.1}$$

Como la función de error *E* depende de $p_{q,t}$ definida en (1.3) y a la vez (1.3) depende de la similitud entre dos objetos definida en (1.1), (1.1) debe ser una función derivable en todo su dominio. Por ello se utiliza la función de similitud difusa de Jaccard con norma producto, computada en (2.2), donde A_i es la función de pertenencia al término lingüístico *i*.

$$sim(d,q) = \frac{\sum_{i=1}^{b} A_i(d) * A_i(q)}{\sum_{i=1}^{b} A_i(d) + A_i(q) - A_i(d) * A_i(q)} \tag{2.2}$$

2.2.1 Modificando los parámetros de las funciones de pertenencia gaussianas.

Se necesita que la función de pertenencia sea una función derivable por ello se utiliza la función de pertenencia gaussiana definida en (1.27). La figura 2.2 muestra gráficamente cómo queda para este caso una partición borrosa para un atributo de la base de casos Iris.

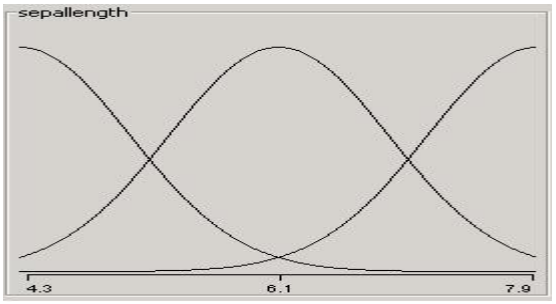

Figura 2.2: Partición borrosa de un atributo de la BC Iris.

Supóngase que se tienen *b* términos lingüísticos y *n* atributos, entonces se deben modificar *2nb* parámetros. Para un paso con el algoritmo del Gradiente de Máximo Descenso se necesitan las

derivadas de la función de error E definida en (2.1) con respecto a cada uno de los parámetros c_i y σ_i con $i = 1,2,...b$, esto es:

$$\frac{\partial E}{\partial c_i} = -2 \sum_{q \in CB} \sum_{t \in T} (\delta_{q,t} - p_{q,t}) \frac{\partial p_{q,t}}{\partial c_i} \tag{2.3}$$

$$\frac{\partial E}{\partial \sigma_i} = -2 \sum_{q \in CB} \sum_{t \in T} (\delta_{q,t} - p_{q,t}) \frac{\partial p_{q,t}}{\partial \sigma_i} \tag{2.4}$$

donde:

$$\frac{\partial p_{q,t}}{\partial c_i} = \left(\frac{2 \sum_{r \in K} (\delta_{r,t} - p_{q,t}) * sim(q,r) * \frac{\partial sim(q,r)}{\partial c_i}}{\sum_{r \in K} sim(q,r)^2} \right) \tag{2.5}$$

$$\frac{\partial p_{q,s}}{\partial c_i} = \left(\frac{2 \sum_{r \in K} (\delta_{r,t} - p_{q,t}) * sim(q,r) * \frac{\partial sim(q,r)}{\partial c_i}}{\sum_{r \in K} sim(q,r)^2} \right) \tag{2.6}$$

donde:

$$\frac{\partial sim(q,r)}{\partial c_i} = \frac{\frac{dA_i(q)}{\partial c_i} A_i(r) + A_i(q) \frac{dA_i(r)}{\partial c_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} - sim(q,r) \frac{\frac{dA_i(q)}{\partial c_i} + \frac{dA_i(r)}{\partial c_i} - \frac{dA_i(q)}{\partial c_i} A_i(r) - A_i(q) \frac{dA_i(r)}{\partial c_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} \tag{2.7}$$

y

$$\frac{\partial sim(q,r)}{\partial \sigma_i} = \frac{\frac{dA_i(q)}{\partial \sigma_i} A_i(r) + A_i(q) \frac{dA_i(r)}{\partial \sigma_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} - sim(q,r) \frac{\frac{dA_i(q)}{\partial \sigma_i} + \frac{dA_i(r)}{\partial \sigma_i} - \frac{dA_i(q)}{\partial \sigma_i} A_i(r) - A_i(q) \frac{dA_i(r)}{\partial \sigma_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} \tag{2.8}$$

donde:

$$\frac{dA_i(x)}{\partial c_i} = \frac{x - c_i}{\sigma_i^2} A_i(x) \tag{2.9}$$

$$\frac{dA_i(x)}{\partial \sigma_i} = \frac{(x - c_i)^2}{\sigma_i^3} A_i(x) \tag{2.10}$$

Para comparar el desempeño de este modelo se establece una comparación con el mismo clasificador pero utilizando particiones uniformes con igual función de similitud y de pertenencia. Para ello se analiza cómo se comporta el error cuadrático medio sobre el conjunto de prueba y el por ciento de clasificación correcta.

En la validación estadística se analizan las siguientes hipótesis:

H_0: No existen diferencias significativas entre ambas propuestas.

Las tablas 2.4 y 2.5 muestran los resultados experimentales.

B.C.	SDFGJ	EWFGJ
1. Balance-scale	80.64	**83.76**
2. Wisconsin-breast-cancer	**97**	96.93
3. Pima_diabetes	**71.1**	70.9
4. Glass	**67.29**	60.97
5. Heart-statlog	79.44	**79.81**
6. Ionosphere	**91.16**	90.74
7. Iris	93	**95**
8. Liver-disorders	**61.45**	59.57
9. New-thyroid	**94.42**	92.56
10. Sonar	**84.13**	82.21
11. Tao	**88.61**	84.06
12. Vehicle	**66.43**	66.25
13. Vowel	**92.98**	91.82
14. Wisconsin Diag. breast-cancer	95.78	**95.96**
15. Wine	**96.65**	94.94

Tabla 2.4. Por ciento de clasificación correcto.

B.C.	SDFGJ	EWFGJ
1. Balance-scale	0.3	**0.28**
2. Wisconsin-breast-cancer	**0.15**	**0.15**
3. Pima_diabetes	**0.46**	**0.46**
4. Glass	**0.29**	0.31
5. Heart-statlog	**0.39**	**0.39**
6. Ionosphere	**0.27**	**0.27**
7. Iris	0.18	**0.15**
8. Liver-disorders	**0.53**	0.54
9. New-thyroid	**0.17**	0.19
10. Sonar	**0.33**	0.34
11. Tao	**0.32**	0.38
12. Vehicle	**0.34**	**0.34**
13. Vowel	**0.11**	**0.11**
14. Wisconsin Diag. breast-cancer	**0.19**	**0.19**
15. Wine	**0.13**	0.16

Tabla 2.5. Error cuadrático medio.

La prueba de Wilconxon (Ver Anexo 2) muestra que no hay diferencias significativas entre los algoritmos SDFGJ y EWFGJ en cuanto al por ciento de clasificación correcto y al error cuadrático medio con valores de significación $p = 0.112$ y $p = 0.29$ respectivamente.

2.2.2 Modificando los parámetros de las funciones de pertenencia triangulares y trapezoidales.

Las funciones de pertenencia triangulares se encuentran entre las más usadas para representar particiones borrosas. Es evidente que estas funciones no son derivables en tres puntos de su dominio. Una heurística simple para tratar este problema es saltar la modificación de los parámetros en estos casos (Nomura et al., 1992). Si hay suficientes datos de entrenamiento, entonces el proceso de aprendizaje no se ve muy afectado por este método. De forma análoga se puede utilizar esta heurística para las funciones de pertenencia trapezoidales, las cuales no son derivables en cuatros puntos de su dominio.

Berenji y Khedkar aplicaron otra heurística, usando la media de las derivadas por la izquierda y por la derecha en los puntos que no son derivables en las funciones de pertenencia triangulares (Berenji and Khedkar, 1992). De forma análoga se puede usar esta heurística para las funciones trapezoidales.

Luego se propone modelar una variable lingüística utilizando funciones trapezoidales como las definidas en (1.26) para los términos de los extremos de la variable lingüística y funciones triangulares como las definidas en (1.23) para el resto de los términos lingüísticos (ver figura 2.1).

Las derivadas que se necesitan son las siguientes:

$$\frac{\partial E}{\partial c_i} = -2 \sum_{q \in CB} \sum_{t \in T} (\delta_{q,t} - p_{q,t}) \frac{\partial p_{q,t}}{\partial c_i} \tag{2.11}$$

$$\frac{\partial E}{\partial sl_i} = -2 \sum_{q \in CB} \sum_{t \in T} (\delta_{q,t} - p_{q,t}) \frac{\partial p_{q,t}}{\partial sl_i} \tag{2.12}$$

$$\frac{\partial E}{\partial sr_i} = -2 \sum_{q \in CB} \sum_{t \in T} (\delta_{q,t} - p_{q,t}) \frac{\partial p_{q,t}}{\partial sr_i} \tag{2.13}$$

donde:

$$\frac{\partial p_{q,t}}{\partial c_i} = \left(\frac{2 \sum_{r \in K} (\delta_{r,t} - p_{q,t}) * sim(q,r) * \frac{\partial sim(q,r)}{\partial c_i}}{\sum_{r \in K} sim(q,r)^2} \right) \tag{2.14}$$

$$\frac{\partial p_{q,t}}{\partial sl_i} = \left(\frac{2\sum_{r \in K} (\delta_{r,t} - p_{q,t}) * sim(q,r) * \frac{\partial sim(q,r)}{\partial sl_i}}{\sum_{r \in K} sim(q,r)^2} \right) \tag{2.15}$$

$$\frac{\partial p_{q,t}}{\partial sr_i} = \left(\frac{2\sum_{r \in K} (\delta_{r,t} - p_{q,t}) * sim(q,r) * \frac{\partial sim(q,r)}{\partial sr_i}}{\sum_{r \in K} sim(q,r)^2} \right) \tag{2.16}$$

donde:

$$\frac{\partial sim(q,r)}{\partial c_i} = \frac{\frac{dA_i(q)}{\partial c_i} A_i(r) + A_i(q) \frac{dA_i(r)}{\partial c_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} - sim(q,r) \frac{\frac{dA_i(q)}{\partial c_i} + \frac{dA_i(r)}{\partial c_i} - \frac{dA_i(q)}{\partial c_i} A_i(r) - A_i(q) \frac{dA_i(r)}{\partial c_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} \tag{2.17}$$

$$\frac{\partial sim(q,r)}{\partial sl_i} = \frac{\frac{dA_i(q)}{\partial sl_i} A_i(r) + A_i(q) \frac{dA_i(r)}{\partial sl_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} - sim(q,r) \frac{\frac{dA_i(q)}{\partial sl_i} + \frac{dA_i(r)}{\partial sl_i} - \frac{dA_i(q)}{\partial sl_i} A_i(r) - A_i(q) \frac{dA_i(r)}{\partial sl_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} \tag{2.18}$$

$$\frac{\partial sim(q,r)}{\partial sr_i} = \frac{\frac{dA_i(q)}{\partial sr_i} A_i(r) + A_i(q) \frac{dA_i(r)}{\partial sr_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} - sim(q,r) \frac{\frac{dA_i(q)}{\partial sr_i} + \frac{dA_i(r)}{\partial sr_i} - \frac{dA_i(q)}{\partial sr_i} A_i(r) - A_i(q) \frac{dA_i(r)}{\partial sr_i}}{\sum_{j=1}^{b} A_j(d) + A_j(q) - A_j(d) * A_j(q)} \tag{2.19}$$

Usando la heurística sugerida en (Nomura et al., 1992) se definen las derivadas de las funciones trapezoidales y triangulares.

Para las funciones triangulares:

$$\frac{dA_i(x)}{\partial c_i} = \begin{cases} -\frac{1}{sl_i} & si \quad x \in (c_i - sl_i, c_i) \\ \frac{1}{sr_i} & si \quad x \in (c_i, c_i + sr_i) \\ 0 & e.o.c. \end{cases} \tag{2.20}$$

$$\frac{dA_i(x)}{\partial sl_i} = \begin{cases} \frac{x - c_i}{(sl_i)^2} & si \quad x \in (c_i - sl_i, c_i) \\ 0 & e.o.c. \end{cases} \tag{2.21}$$

$$\frac{dA_i(x)}{\partial sr_i} = \begin{cases} \frac{x - c_i}{(sr_i)^2} & si \quad x \in (c_i, c_i + sr_i) \\ 0 & e.o.c. \end{cases} \tag{2.22}$$

Para las funciones trapezoidales:

$$\frac{dA_i(x)}{\partial sl_i} = \begin{cases} \frac{x - c_i}{(sl_i)^2} & si \quad x \in (b_i - sl_i, b_i) \\ 0 & e.o.c. \end{cases} \tag{2.23}$$

$$\frac{dA_i(x)}{\partial b_i} = \begin{cases} -\frac{1}{sl_i} & si \quad x \in (b_i - sl_i, b_i) \\ 0 & e.o.c. \end{cases} \tag{2.24}$$

$$\frac{dA_i(x)}{\partial c_i} = \begin{cases} \frac{1}{sr_i} & si \quad x \in (c_i, c_i + sr_i) \\ 0 & e.o.c. \end{cases} \tag{2.25}$$

$$\frac{dA_i(x)}{\partial sr_i} = \begin{cases} \frac{|x - c_i|}{(sr_i)^2} & si \quad x \in (c_i, c_i + sr_i) \\ 0 & e.o.c. \end{cases} \tag{2.26}$$

Para comparar el desempeño de este modelo se establece una comparación con las particiones uniformes usando esta función de similitud y función de pertenencia. Para ello se analiza cómo se comporta el error cuadrático medio sobre el conjunto de prueba y el por ciento de clasificación correcta.

En la validación estadística se analizan las siguientes hipótesis:

H_0: No existen diferencias significativas entre ambas propuestas.

Las tablas 2.6 y 2.7 muestran los resultados experimentales.

B.C.	SDFTJ	EWFTJ
balance-scale	**83.52**	82.48
wisconsin-breast-cancer	**96.12**	**96.12**
pima_diabetes	**73.11**	72.92
Glass	**68.88**	62.82
heart-statlog	**81.11**	**81.11**
ionosphere	91.02	**91.45**
iris	94.33	**95.67**
liver-disorders	55.8	**56.52**
new-thyroid	**93.26**	91.4
sonar	**85.08**	84.6
TAO_grid	**85.75**	83.95
vehicle	**68.26**	67.79
vowel	93.38	**94.04**
wdbc	95.61	**96.13**
wine	**97.2**	96.08

Tabla 2.6. Por ciento de clasificación correcto.

B.C.	SDFTJ	EWFTJ
balance-scale	**0.3**	0.31
wisconsin-breast-cancer	0.18	**0.17**
pima_diabetes	**0.45**	0.46
Glass	**0.28**	0.3
heart-statlog	**0.39**	**0.39**
ionosphere	0.27	**0.26**
iris	0.18	**0.16**
liver-disorders	0.56	**0.55**
new-thyroid	**0.17**	0.18
sonar	**0.33**	**0.33**
TAO_grid	**0.35**	0.39
vehicle	**0.34**	**0.34**
vowel	**0.09**	**0.09**
wdbc	**0.18**	**0.18**
wine	**0.13**	**0.13**

Tabla 2.7. Error cuadrático medio.

La prueba de Wilconxon (Ver Anexo 2) muestra que no hay diferencias significativas entre los algoritmos SDFTJ y EWFTJ en cuanto al por ciento de clasificación correcto y al error cuadrático medio con valores de significación $p = 0.279$ y $p = 0.582$ respectivamente.

2.2.3 Modificando los parámetros de las funciones de pertenencia usando la función de similitud difusa de Manhattan.

Se puede apreciar que el modelo propuesto no ofrece buenos resultados cuando se usa la función de similitud difusa de Jaccard. Además los resultados experimentales arrojan que el tiempo de entrenamiento es muy alto. Por ello se debe utilizar una función de similitud que mezcle cálculos sencillos con buenos resultados de clasificación como la de Manhattan, definida en (1.30).

Esta función no es derivable en todo su dominio pero por sus características, se puede aplicar la heurística sugerida en (Nomura et al., 1992), es decir, omitir la modificación de los parámetros donde no es derivable.

En vez de usar la función de similitud difusa de Jaccard se usa ésta, resultando un poco más sencillos los cálculos numéricos y así se obtiene un nuevo modelo matemático. Para ello se define su derivada con respecto a cualquier parámetro θ_i de una de las funciones de pertenencia utilizadas anteriormente.

$$\frac{\partial sim(q,r)}{\partial \theta_i} = -\text{sgn}(A_i(q) - A_i(r))(\frac{\partial A_i(q)}{\partial \theta_i} - \frac{\partial A_i(r)}{\partial \theta_i}) \tag{2.27}$$

El resto de las derivadas necesarias fueron computadas anteriormente.

Para comparar el desempeño de este modelo se establece una comparación con las particiones uniformes usando esta función de similitud. Para ello se analiza cómo se comporta el error cuadrático medio sobre el conjunto de prueba y el por ciento de clasificación correcta.

En la validación estadística se analizan las siguientes hipótesis:

H_0: No existen diferencias significativas entre ambas propuestas.

Las tablas de la 2.8 a la 2.11 muestran los resultados experimentales para las funciones de pertenencia gaussianas y triangulares.

B.C.	SDFGM	EWFGM
balance-scale	**82**	81.92
wisconsin-breast-cancer	**96.93**	**96.93**
pima_diabetes	**70.51**	69.07
Glass	**73.83**	71.72
heart-statlog	79.81	**81.48**
ionosphere	**90.46**	90.31
iris	**95**	94.33
liver-disorders	65.94	**66.52**
new-thyroid	**96.74**	96.28
sonar	**83.64**	81.97
TAO_grid	**96.61**	95.76
vehicle	**71.75**	71.63
vowel	94.49	**95.05**
wdbc	96.22	**96.57**
wine	**96.35**	94.66

Tabla 2.8. Por ciento de clasificación correcto para funciones de pertenencia gaussianas.

B.C.	SDFGM	EWFGM
balance-scale	**0.3**	**0.3**
wisconsin-breast-cancer	**0.16**	**0.16**
pima_diabetes	**0.46**	0.47
Glass	**0.26**	**0.26**
heart-statlog	**0.38**	**0.38**
ionosphere	**0.27**	0.28
iris	**0.16**	**0.16**
liver-disorders	**0.48**	**0.48**
new-thyroid	**0.13**	0.14
sonar	**0.34**	0.35
TAO_grid	**0.16**	0.17
vehicle	**0.31**	**0.31**
vowel	**0.09**	**0.09**
wdbc	0.18	**0.17**
wine	**0.12**	0.15

Tabla 2.9. Error cuadrático medio para funciones de pertenencia gaussianas.

La prueba de Wilconxon (Ver Anexo 2) muestra que no hay diferencias significativas entre los algoritmos SDFGM y EWFGM en cuanto al por ciento de clasificación correcto y al error cuadrático medio con valores de significación $p = 0.132$ y $p = 0.058$ respectivamente.

B.C	SDFTM	EWFTM
balance-scale	**81.04**	78.96
wisconsin-breast-cancer	**96.56**	**96.56**
pima_diabetes	**71.81**	70.9
Glass	**75.7**	75.45
heart-statlog	79.63	**80.56**
ionosphere	89.46	**90.74**
iris	94.67	**95.33**
liver-disorders	59.28	**59.86**
new-thyroid	**95.12**	**95.12**
sonar	85.08	**85.81**
TAO_grid	**96.35**	96.21
vehicle	70.03	**70.56**
vowel	95.2	**95.25**
wdbc	**95.87**	96.49
wine	**96.93**	96.35

Tabla 2.10. Por ciento de clasificación correcto para funciones de pertenencia triangulares.

B.C.	SDFTM	EWFTM
balance-scale	**0.3**	0.32
wisconsin-breast-cancer	**0.16**	0.17
pima_diabetes	**0.46**	**0.46**
Glass	**0.25**	**0.25**
heart-statlog	**0.39**	**0.39**
ionosphere	0.29	**0.27**
iris	0.18	**0.17**
liver-disorders	**0.52**	**0.52**
new-thyroid	**0.14**	0.15
sonar	0.33	**0.32**
TAO_grid	**0.16**	0.17
vehicle	**0.32**	**0.32**
vowel	**0.09**	**0.09**
wdbc	0.19	**0.18**
wine	**0.12**	**0.12**

Tabla 2.11. Error cuadrático medio para funciones de pertenencia triangulares.

La prueba de Wilconxon (Ver Anexo 2) muestra que no hay diferencias significativas entre los algoritmos SDFTM y EWFTM en cuanto al por ciento de clasificación correcto y al error cuadrático medio con valores de significación $p = 0.422$ y $p = 1$ respectivamente.

2.3 Presentación del algoritmo SDF.

Después de formuladas todas las derivadas la descripción del algoritmo es la siguiente:

1. Inicializar los parámetros.
2. Obtener la partición borrosa correspondiente a estos parámetros.
3. Computar el error E de acuerdo con (2.1).
4. Inicializar el factor de aprendizaje λ.

5. Mientras no se cumpla el criterio de parada

   a. $\forall i \ \theta_{i+1} = \theta_i - \frac{\partial E}{\partial \theta_i} \lambda$

   b. Obtener la nueva partición borrosa correspondiente a estos nuevos parámetros θ_{i+1}.

   c. Computar el nuevo error *E* de acuerdo con (2.1).

   d. Si el nuevo error es menor que el anterior y los nuevos parámetros satisfacen la definición de partición borrosa entonces actualizar los parámetros si no $\lambda = \frac{\lambda}{2}$.

La inicialización de los parámetros puede hacerse utilizando alguna técnica de construcción de funciones de pertenencia sencilla o puede hacerse de forma aleatoria. En este estudio se usa un algoritmo de discretización de intervalos de igual ancho, asegurando que dos funciones consecutivas se crucen en el límite del intervalo y que cada conjunto borroso tiene como soporte la mitad del ancho del intervalo (particiones uniformes).

La inicialización del factor de aprendizaje es un parámetro del algoritmo. Además en el trabajo se usa una heurística sugerida en (Jang Jyh-Shing et al., 1997). En la misma el factor de aprendizaje λ se obtiene de la siguiente manera:

1. $\lambda=1$
2. Mientras que λ sea menor que el mayor valor del gradiente
   a. $\lambda = \lambda * 10$
3. $\lambda = 1 / (\lambda * 10)$

La selección del criterio de parada debe asegurar la terminación del algoritmo. Los posibles criterios de parada para el algoritmo son:

- Un número fijo de pasos de aprendizaje.
- Un mínimo cambio en la función de error.

- Un mínimo cambio en los parámetros a modificar $\frac{\partial E}{\partial \theta_i} < \varepsilon$.

- Un mínimo factor de aprendizaje $\lambda < \varepsilon$

Es posible además combinar estos criterios.

En este trabajo se utiliza como criterio de parada un mínimo cambio en la función de error reforzado por un número determinado de iteraciones. Es decir, si en una cantidad de iteraciones dada, no se produce un mínimo cambio en la función de error entonces se da por terminado el proceso de modificación de los mismos.

2.4 Acelerando la convergencia del método de Máximo Descenso.

El método del Gradiente de Máximo Descenso se caracteriza por una lenta velocidad de convergencia (Jang Jyh-Shing et al., 1997). Para acelerar la convergencia del proceso de aprendizaje se usan métodos Quasi-Newton para modificar los parámetros, específicamente los métodos DFP y BFGS. A continuación se presenta un experimento donde se compara el desempeño de estos métodos con el método del Gradiente de Máximo Descenso para la función de similitud de Manhattan y función de pertenencia gaussiana.

En la validación estadística se analizan las siguientes hipótesis:

H_0: No existen diferencias entre los distintos algoritmos en cuanto al por ciento de clasificación correcta.

La tabla 2.12 muestra los resultados experimentales con todas las bases de caso.

B.C.	SDFGM	DFPFGM	BFGSFGM
balance-scale	**82**	79.36	**82**
wisconsin-breast-cancer	96.93	**97**	**97**
pima_diabetes	70.51	**71.23**	70.38
Glass	**73.83**	73.59	72.43
heart-statlog	79.81	80.56	**80.93**
ionosphere	90.46	**90.74**	90.17
iris	95	94.67	**95.67**
liver-disorders	**65.94**	64.78	64.64
new-thyroid	**96.74**	96.51	96.51
sonar	83.64	83.4	**83.87**
TAO_grid	**96.61**	96.42	96.34
vehicle	**71.75**	71.51	**71.75**
vowel	94.49	94.44	**94.7**
wdbc	**96.22**	96.04	96.13
wine	**96.35**	96.07	**96.35**
Rango	**1.7**	2.3	1.97
Posición	**1**	3	2

Tabla 2.12. Por ciento de clasificación correcto.

La prueba no paramétrica de Friedman acepta la hipótesis nula con $p = 0.22$, por lo que se puede concluir que no hay diferencias significativas entre los distintos algoritmos en cuanto al por ciento de clasificación correcto.

A continuación se establece una comparación entre los tiempos de procesamiento de estos algoritmos. La tabla 2.13 muestra los resultados experimentales.

B.C.	SDFGM	DFPFGM	BFGSFGM
balance-scale	21.78	**5.79**	7.16
wisconsin-breast-cancer	68.85	**46.86**	49.47
pima_diabetes	55.63	**31.43**	32.02
Glass	26.11	5.44	**5.05**
heart-statlog	17.54	10.94	**10.03**
ionosphere	**64.12**	88.21	87.34
iris	3.64	**0.57**	0.61
liver-disorders	11.1	5.38	**3.94**
new-thyroid	6.56	1.78	**1.39**
sonar	**54.67**	114.56	105.77
TAO_grid	67.73	15.91	**15.72**
vehicle	156.65	**150**	184.9
vowel	217.81	100.99	**85.81**
wdbc	**99.48**	150.18	157.9
wine	15.41	**4.75**	4.95
Rango	2.53	**1.73**	**1.73**
Posición	3	**Empate**	

Tabla 2.13. Tiempo de procesamiento.

La prueba no paramétrica de Friedman muestra que hay diferencias significativas entre los tiempos de procesamiento con $p = 0.04$. Como la prueba de Friedman detecta diferencias se aplica la prueba de Bonferroni-Dunn. La misma permite ilustrar gráficamente los resultados, que son visualizados en la figura 2.3. La distancia crítica es computada como:

$$CD = q_\alpha \times \sqrt{\frac{k(k+1)}{6N}} = 1.960 \times \sqrt{\frac{2 \times 3}{6 \times 15}} = 0.51$$

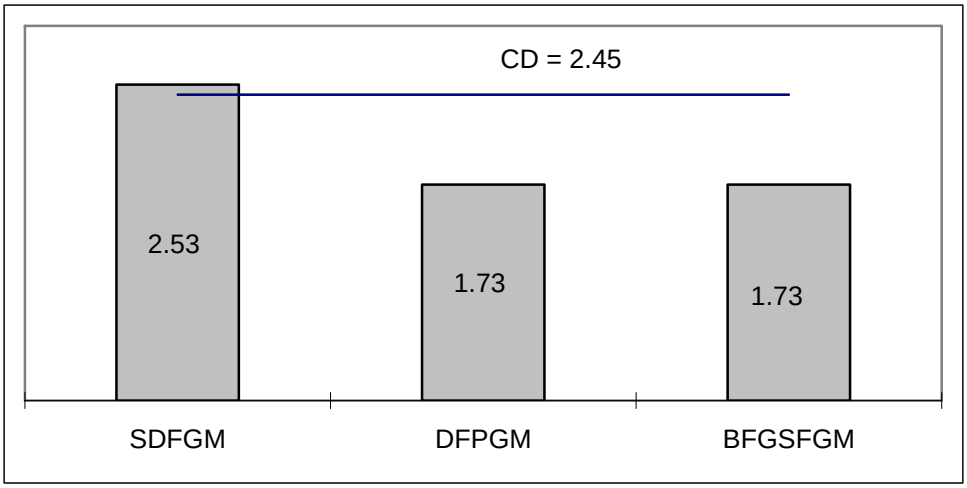

Figura 2.3. Comparación visual del por ciento de clasificación correcto.

En la figura 2.3 se aprecia que el rango del método de Máximo Descenso está por encima de la distancia crítica, por lo que hay diferencias significativas con respecto a los algoritmos de control, en este caso los métodos Quasi-Newton (DFP y BFGS).

2.5 Conclusiones parciales.

Se puede apreciar, a partir de los resultados de la comparación entre cada uno de los modelos propuestos, que solamente cuando se usa la función de similitud de Manhattan y funciones de pertenencia gaussinana, el modelo tiene un desempeño ligeramente superior a las particiones uniformes.

Además, si se utilizan métodos Quasi-Newton se puede acelerar la convergencia del algoritmo y por ende el proceso de aprendizaje se efectúa con más rapidez.

Capítulo III: Estimación de parámetros de funciones de pertenencia a partir de particiones aleatorias.

En el capítulo anterior la no disminución significativa del error y el por ciento de clasificación pueden estar condicionados a que se produce un sobreajuste en el conjunto de entrenamiento, o el punto inicial a partir del cual se realiza la optimización de los parámetros no es adecuado.

3.1 Control del sobreajuste.

Que se produzca un sobreajuste en el conjunto de entrenamiento significa que un menor error sobre el mismo no lleva a un mejor desempeño sobre un conjunto independiente de prueba. Para solucionar este problema se debe dividir el conjunto de entrenamiento en dos partes: uno para entrenar y otro como grupo de control. Luego en cada iteración del algoritmo se obtiene el error producido en el grupo de control y en caso que el mismo comience a aumentar se detiene el proceso de aprendizaje de los parámetros.

Los resultados experimentales arrojan que haciendo control del sobreajuste no se aumenta la eficiencia de los modelos propuestos.

3.2 Explorando el espacio de búsqueda.

A continuación se presenta un experimento donde se hace una inicialización aleatoria del punto inicial sobre 10 modelos independientes, tomándose los mejores valores y comparándose nuevamente con la obtención de particiones uniformes. Se usa la función de similitud de Manhattan y funciones de pertenencia Gaussianas, ya que es el modelo que mejor desempeño y estabilidad muestra. Para ello se analiza cómo se comporta el por ciento de clasificación correcto y el error cuadrático medio sobre el conjunto de prueba.

Las tablas 3.1 y 3.2 muestran los resultados experimentales.

B.C.	EWFGM	SDFGM
1. Balance-scale	81.01	**83.76**
2. Wisconsin-breast-cancer	96.81	**97.15**
3. Pima_diabetes	69.88	**72.98**
4. Glass	73.04	**76.19**
5. Heart-statlog	**82.33**	81.11
6. Ionosphere	90.65	**91.31**
7. Iris	94.67	**95.2**
8. Liver-disorders	**67.13**	66.55
9. New-thyroid	**96.42**	96.37
10. Sonar	81.26	**84.65**
11. Tao	95.63	**96.84**
12. Vehicle	70.9	**73.4**
13. Vowel	96.01	**96.72**
14. Wisconsin Diag. breast-cancer	96.51	**96.93**
15. Wine	95.4	**97.22**

Tabla 3.1. Por ciento de clasificación correcto.

La prueba de Wilconxon (Ver Anexo 3) muestra que hay diferencias significativas con valores de significación $p = 0.011$ en cuanto al por ciento de clasificación y $p = 0.007$ en cuanto al error cuadrático medio.

Ésto evidencia que considerando como punto inicial del modelo propuesto las particiones uniformes no siempre se obtiene una buena solución. Por tanto se puede constatar que el punto inicial de un método descendente influye significativamente en su desempeño, por ello se debe aplicar algún método de optimización que explore el espacio de búsqueda. Entre los diversos métodos existentes se elige el método de Monte Carlo, por su sencillez y probada eficacia.

B.C.	EWFGM	SDFGM
1. Balance-scale	0.3	**0.28**
2. Wisconsin-breast-cancer	**0.15**	**0.15**
3. Pima_diabetes	0.47	**0.45**
4. Glass	0.26	**0.25**
5. Heart-statlog	**0.37**	0.38
6. Ionosphere	0.27	**0.26**
7. Iris	**0.14**	**0.14**
8. Liver-disorders	**0.48**	**0.48**
9. New-thyroid	0.13	**0.12**
10. Sonar	0.34	**0.32**
11. Tao	0.17	**0.15**
12. Vehicle	0.32	**0.31**
13. Vowel	0.08	**0.07**
14. Wisconsin Diag. breast-cancer	**0.16**	**0.16**
15. Wine	0.14	**0.1**

Tabla 3.2. Error cuadrático medio.

3.2.1 Optimización de los parámetros de las funciones de pertenencia usando el método de Monte Carlo.

El siguiente esquema se emplea para describir nemotécnicamente la idea esencial del método de Monte Carlo, un algoritmo simple para resolver el problema de minimización a partir de la generación de *n* números aleatorios.

La descripción del algoritmo es la siguiente:

1. Inicializar los parámetros aleatoriamente, siguiendo una distribución uniforme en el intervalo de definición.

2. Obtener la partición borrosa correspondiente a estos parámetros.

3. Computar el error *E* de acuerdo con (2.1).

4. Durante un número determinado de iteraciones

    a. Generar aleatoriamente nuevos parámetros.

    b. Obtener la nueva partición borrosa correspondiente a estos nuevos parámetros.

    c. Computar el nuevo error *E* de acuerdo con (2.1).

    d. Si el nuevo error es menor que el anterior entonces actualizar los parámetros.

5. Aplicar a estos parámetros obtenidos el método SDF.

Para comparar el desempeño de este nuevo modelo se establece una comparación con las particiones uniformes usando función de similitud de Manhattan y función de pertenencia gaussiana. Para ello se analiza cómo se comporta el por ciento de clasificación correcta sobre el conjunto de prueba.

En la validación estadística se analizan las siguientes hipótesis:

H_0: No existen diferencias significativas entre ambas propuestas.

La tabla 3.3 muestra los resultados experimentales para funciones de pertenencia gaussianas y función de similitud difusa de Manhattan. El desempeño es medido como el por ciento de clasificación correcto de las instancias, y éste es obtenido para cada base de casos y cada algoritmo de aprendizaje.

La prueba de Wilconxon (Ver Anexo 2) muestra que hay diferencias significativas con valor de significación $p = 0.026$.

B.C.	EWFGM	MCFGM
Balance-scale	81.92	**82.4**
wisconsin-breast-cancer	96.93	**97**
pima_diabetes	69.07	**71.36**
Glass	71.72	**74.99**
heart-statlog	**81.48**	80
ionosphere	90.31	**90.74**
Iris	94.33	**94.33**
liver-disorders	**66.52**	66.23
new-thyroid	**96.28**	95.81
Sonar	81.97	**84.58**
TAO_grid	95.76	**96.64**
vehicle	71.63	**72.22**
vowel	95.05	**95.91**
Wdbc	96.57	**97.1**
Wine	94.66	**96.08**

Tabla 3.3. Por ciento de clasificación correcto.

3.3 Validando el modelo propuesto.

A continuación se establece una comparación entre el modelo propuesto y las particiones uniformes usando las funciones de similitud de Jaccard y de Manhattan y funciones de pertenencia triangulares y gaussianas. El objetivo de este experimento es comprobar si el nuevo modelo ofrece mejores resultados que el algoritmo de obtención de particiones uniformes.

B.C.	EWFGM	EWFGJ	EWFTM	EWFTJ	MCFGM
balance-scale	81.92	**83.76**	78.96	82.48	82.4
wisconsin-breast-cancer	96.93	96.93	96.56	96.12	**97**
pima_diabetes	69.07	70.9	70.9	**72.92**	71.36
Glass	71.72	60.97	**75.45**	62.82	74.99
heart-statlog	**81.48**	79.81	80.56	81.11	80
ionosphere	90.31	**90.74**	**90.74**	91.45	**90.74**
iris	94.33	95	95.33	**95.67**	94.33
liver-disorders	**66.52**	59.57	59.86	56.52	66.23
new-thyroid	**96.28**	92.56	95.12	91.4	95.81
sonar	81.97	82.21	**85.81**	84.6	84.58
TAO_grid	95.76	84.06	96.21	83.95	**96.64**
vehicle	71.63	66.25	70.56	67.79	**72.22**
vowel	95.05	91.82	95.25	94.04	**95.91**
wdbc	96.57	95.96	96.49	96.13	**97.1**
wine	94.66	94.94	**96.35**	96.08	96.08
Rango	3.13	3.87	2.63	3.17	**2.2**
Posición	3	5	2	4	**1**

Tabla 3.4. Por ciento de clasificación correcto.

Las dos últimas filas muestran el rango del promedio de cada clasificador y su posición en el ranking. La prueba de Friedman rechaza la hipótesis nula de que todos los sistemas se desempeñan de la misma forma con p = 0.046.

Como la prueba de Friedman detecta diferencias se aplica la prueba de Bonferroni-Dunn. La misma permite ilustrar gráficamente los resultados (Figura 3.1). La distancia crítica es computada como:

$$CD = q_\alpha \times \sqrt{\frac{k(k+1)}{6N}} = 2.241 \times \sqrt{\frac{5 \times 6}{6 \times 15}} = 1.29$$

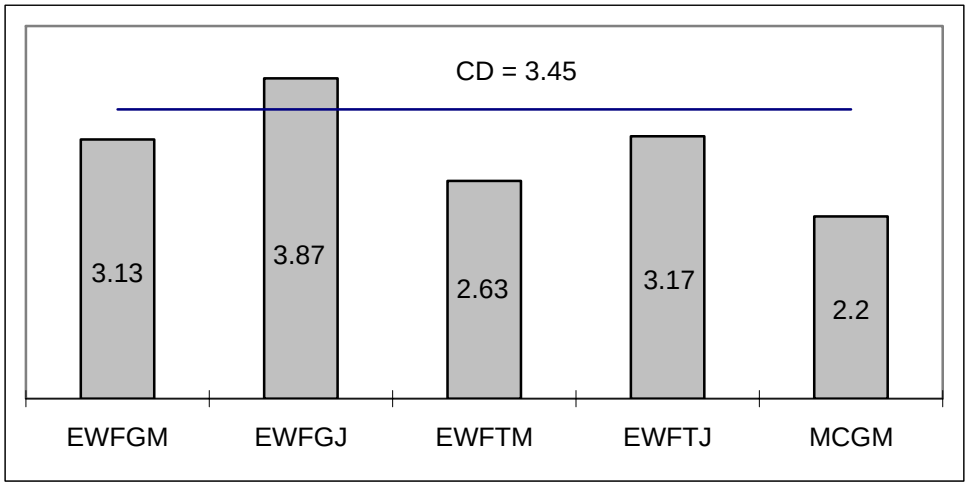

Figura 3.1. Comparación visual del por ciento de clasificación correcto.

En la figura 3.1 se puede apreciar que hay diferencias significativas entre el algoritmo de control (el modelo propuesto) y el algoritmo EWFGJ, ya que su rango está por encima de la distancia crítica. No obstante, en uno de los experimentos presentados con anterioridad quedó demostrado que entre el algoritmo de control y el algoritmo EWFGM la prueba de Wilconxon arroja diferencias significativas.

De forma concluyente la obtención de los parámetros de las funciones de pertenencia utilizando métodos de optimización aumenta el desempeño de un sistema borroso basado en instancias. Entonces se propone usar para este fin funciones de pertenencia gaussianas y la función de similitud difusa de Manhattan.

El modelo es comparado también con varias técnicas de aprendizaje automatizado, ampliamente conocidas, que representan gran variedad de paradigmas de aprendizaje. Específicamente se compara el método propuesto con C4.5, k-NN con función de distancia euclideana, Naive Bayes, Red Multicapas con propagación hacia atrás del error y regresión logística. Estos método se encuentran entre los más exitosos del aprendizaje automático (Orriols-Puig et al., 2008).

C4.5 implementado en WEKA bajo el nombre J48 es un árbol de decisión que mejora al algoritmo ID3 introduciendo métodos para manejar atributos continuos y valores perdidos. Naive Bayes es un clasificador probabilístico que estima los parámetros de un modelo Bayesiano. Todos estos métodos se ejecutan utilizando la configuración sugerida en WEKA.

La tabla 3.5 muestra los resultados experimentales para todas las bases de casos. En este experimento, el desempeño es medido como el por ciento de clasificación correcta de las instancias, y éste es obtenido para cada base de casos y cada algoritmo de aprendizaje.

B.C.	NaiveBayes	MLP	Ibk	J48	RL	MCFGM
balance-scale	90.4	**91.36**	86.16	78.08	91.04	82.4
wisconsin-breast-cancer	96.41	95.97	**97.07**	95.31	95.68	97
pima_diabetes	75.92	75.07	73.9	71.36	**77.41**	71.36
Glass	49.98	65.89	67.98	64.97	61.93	**74.99**
heart-statlog	**84.63**	80.19	78.15	77.41	84.07	80
ionosphere	82.76	90.45	86.47	89.18	88.19	**90.74**
iris	95.33	96	94.67	94.67	**96.33**	94.33
liver-disorders	55.22	**69.13**	62.32	**69.13**	67.39	66.23
new-thyroid	**97.21**	89.3	94.19	92.09	95.35	95.81
sonar	66.64	80.29	84.13	71.89	70.92	**84.58**
TAO_grid	80.93	86.04	96.29	95.79	83.55	**96.64**
vehicle	45.27	**81.5**	70.51	72.1	72.11	72.22
vowel	66.72	79.8	95.56	77.02	53.84	**95.91**
wdbc	93.42	96.48	96.66	93.32	96.39	**97.1**
wine	97.18	**98.05**	95.51	91.62	97.48	96.08
Rango	4.13	2.70	3.5	4.57	3.4	**2.70**
Posición	5	2	4	6	3	**1**

Tabla 3.5. Por ciento de clasificación correcto.

Las dos últimas filas muestran el rango del promedio de cada clasificador y su posición en el ranking. La prueba de Friedman rechaza la hipótesis nula de que todos los sistemas se desempeñan de la misma forma con $p = 0.033$. Por lo tanto se aplica la prueba de Bonferroni-Dunn para detectar cuál algoritmo de aprendizaje se desempeña equivalentemente con el algoritmo de mejor rango, en este caso el modelo propuesto en este trabajo. El desempeño de dos clasificadores es significativamente diferente si el correspondiente rango difiere al menos una distancia crítica, computada como:

$$CD = q_\alpha \times \sqrt{\frac{k(k+1)}{6N}} = 2.326 \times \sqrt{\frac{6 \times 7}{6 \times 15}} = 1.59$$

Cualquier algoritmo con el rango fuera de esta área es significativamente diferente del algoritmo de control. MCFGM se desempeña significativamente mejor que J48, pero ligeramente mejor que el resto. La figura 3.2 permite ilustrar gráficamente los resultados.

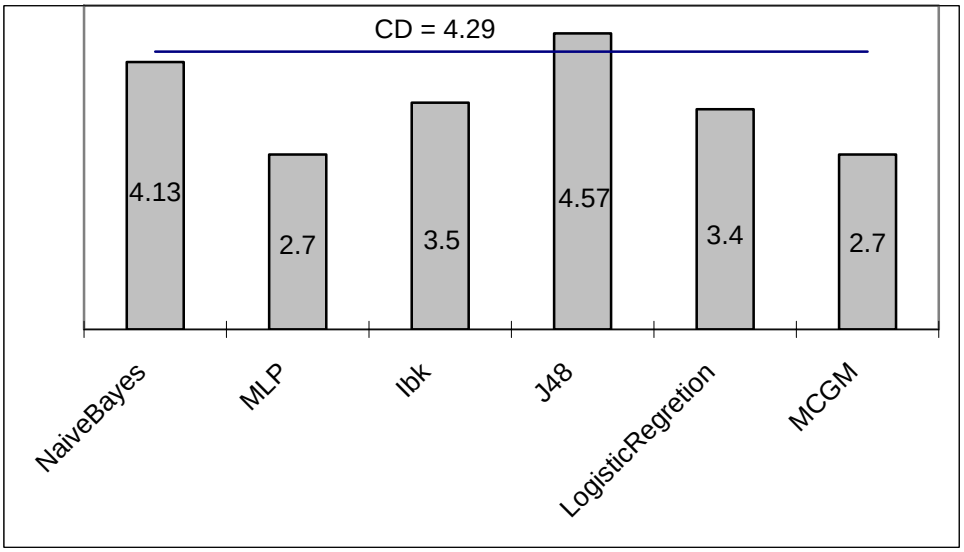

Tabla 3.2. Comparación visual del por ciento de clasificación correcto.

Para contrastar el resultado se aplica también el procedimiento de paso abajo de Holm (Ver Anexo 4), que es más poderoso que la prueba de Bonferroni-Dunn y no hay que hacer ninguna asunción de los datos. La prueba de Holm con $\alpha = 0.05$ detecta los mismos resultados.

3.4 Detalles de implementación del modelo propuesto.

El algoritmo fue implementado en WEKA (Witten and Frank, 1999). WEKA es un sistema multiplataforma y de amplio uso. Puede ser usado desde la perspectiva de usuario mediante las seis interfaces que brinda, a través de la línea de comando desde donde se pueden invocar cada uno de los algoritmos incluidos en la herramienta como programas individuales y mediante la creación de un programa Java que llame a las funciones que se desee.

Weka denomina clasificador a cualquier modelo con capacidad de predecir un valor nominal (clase discreta, clasificación) o un valor numérico (regresión). Los clasificadores son agrupados en diferentes paquetes de acuerdo a la técnica que empleen. Estos paquetes se encuentran dentro del paquete "weka.classifiers". Ellos son:

- bayes: algoritmos de redes bayesianas.
- functions: algoritmos de redes neuronales y regresión numérica.
- lazy: algoritmos perezosos (un aprendizaje que no es fuera de línea, se hace fuera del tiempo de ejecución).
- meta: algoritmos meta-clasificadores, es decir, que usan como entrada un clasificador base.
- misc: algoritmos de clasificación que no están incluidos en ninguna otra categoría.
- trees: los algoritmos de árboles de decisión.
- rules: algoritmos que implementan reglas de asociación.

Los métodos principales a tener en cuenta a la hora de implementar un nuevo clasificador para Weka son heredados de la clase abstracta *Classifier* y deben ser redefinidos de acuerdo al objetivo que persiga el algoritmo de aprendizaje a implementar. Esta clase es la más importante del paquete classifiers y constituye la superclase de todos los clasificadores existentes. Los métodos que deben ser redefinidos son:

- *buildClassifier:* se encarga de la construcción del modelo del clasificador tomando como parámetro las instancias de entrenamiento. Debe inicializar todas las variables que correspondan a las opciones específicas del esquema. Nunca debe modificar ningún valor de las instancias. Después de terminada la ejecución de este método el clasificador debe ser capaz de predecir la clase de cualquier instancia nueva.
- *classifyInstance:* permite clasificar una instancia concreta. Devuelve la clase en la que se ha clasificado o un valor perdido si no se consigue clasificar. En el caso de los clasificadores de predicción numérica la función devuelve el número predicho.

- *distributionForInstance:* cumple la misma función que el método anterior, sólo que el resultado de la clasificación se retorna de una forma diferente. Devuelve la distribución de una instancia en forma de vector. Si el clasificador no ha logrado clasificar la instancia dada devuelve un vector lleno de ceros. Si lo consigue y la clase es numérica devuelve un vector de un solo elemento que es el valor obtenido. En los demás casos devuelve un vector con el grado de pertenencia de la instancia dada a cada una de las clases.

El algoritmo propuesto es implementado como un metaclasificador. El mismo debe obtener los parámetros de las funciones de pertenencia óptimos a partir de la muestra de aprendizaje.El método *buildClassifier* es el encargado de realizar este proceso. Para ello se utiliza una validación cruzada dejando uno fuera (LOOCV del inglés *Leave One Out Cross Validation*). Es decir, el error cometido al clasificar cada una de las instancias de aprendizaje (sin tomarla en cuenta para hacer la predicción) sirve como indicador del futuro comportamiento del clasificador con los ejemplos de la muestra de control. En sucesivas iteraciones se modifican los parámetros de modo que se minimice este error.

Posteriormente se crea un clasificador IBk, que es el responsable de efectuar la clasificación de las instancias y se le pasa la partición borrosa obtenida. Cuando se realice una llamada al método *classifyInstance* del metaclasificador este le pasa el control al método *classifyInstance* del clasificador IBk construido.

Cada uno de los algoritmos de optimización (Máximo descenso, DFP, BFGS y Monte Carlo) es implementado como una clase que toma como parámetros una determinada función de error y un vector con los parámetros a optimizar. Esto permite la reutilización de estos métodos.

La función de error es definida en dependencia de la funciones de pertenencia y función de similitud a usar, ya que la misma es la encargada de obtener las derivadas del vector de parámetros. Por tanto el metaclasificador de acuerdo a sus parámetros codifica los parámetros de las funciones de pertenencia en un vector y construye la función de error. Primeramente se le pasa el control al método de Monte Carlo, este devuelve un vector con los mejores parámetros encontrados, posteriormente estos parámetros son pasados al método de Máximo descenso o a uno de los métodos Quasi-Newton implementados. Finalmente con los mejores parámetros se crea un clasificador IBk borroso y se da por terminado el proceso de aprendizaje.

3.3 Conclusiones parciales.

En este capítulo se propone un algoritmo de obtención de particiones borrosas bastante eficiente en cuanto al por ciento de clasificación de un sistema borroso basado en instancias, con respecto al algoritmo implementado en WEKA de obtención de particiones uniformes. Además es importante señalar que este algoritmo alcanza resultados comparables con algoritmos clásicos, superando ligeramente al *k*-NN con función de distancia euclideana.

Conclusiones.

En este trabajo se ha presentado un modelo para obtener particiones borrosas utilizando métodos de optimización y aprendizaje basado en instancias, cumpliéndose de esta forma el objetivo general planteado, ya que:

- Se formuló un algoritmo que obtiene los mejores parámetros de las funciones de pertenencia utilizando el método de Máximo Descenso.

- Se mejoró la velocidad de convergencia del mismo utilizando métodos Quasi-Newton en vez del método de Máximo Descenso.

- Se obtuvo un método de obtención de particiones borrosas utilizando en la fase inicial el método de Monte Carlo para generar una buena partición borrosa y posteriormente aplicarle uno de los métodos anteriores.

Los resultados experimentales muestran que el desempeño de un sistema borroso basado en instancias, así obtenido, mejora respecto a las particiones homogéneas. Cuando se emplean funciones de pertenencia gaussianas y la función de similitud difusa de Manhattan el algoritmo resulta competitivo con los mejores clasificadores del aprendizaje automático superando a aquellos exponentes seleccionados durante el diseño experimental con la excepción de la red neuronal multicapas entrenada con propagación hacia atrás con el cual obtiene resultados similares.

Recomendaciones.

Extender el presente estudio mediante el uso de métodos de optimización estocásticos y poblacionales. Este tipo de método no necesita conocimiento alguno de la derivada de la función de similitud o de la función de pertenencia por lo que se dispone de esa forma de mayor libertad de diseño del sistema.

Unido al uso de métodos de optimización libres de derivada se deberá utilizar funciones de similitud parametrizadas que generalizan la mayoría de las funciones conocidas.

Bibliografía

BERENJI, H. R. & KHEDKAR, P. (1992) Learning an Tuning Fuzzy Logic Controllers through Reinforcement. *IEEE Trans. Neural Networks,* 3**,** 724-740.

CASILLAS, J., CORDÓN, O., HERRERA, F. & MAGDALENA, L. Interpretability Improvements to Find the Balance Interpretability-Accuracy in Fuzzy Modeling: An Overview.

DEMSAR, J. (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research,* 7**,** 1-30.

HONG, T.-P. & LEE, C.-Y. (1998) Learning Fuzzy Knowledge from Training Examples. ACM.

ICHIHASHI, H. (1991) Iterative Fuzzy Modelling and Hierarchical Network. 49-52.

JANG JYH-SHING, R., CHUEN-TSAI, S. & EIJI, M. (1997) *Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence,* NJ.

MORELL, C., BELLO, R. & BALO, R. G. (2004) Improving k-NN by Using Fuzzy Similarity Functions. Springer.

MORELL, C., BELLO, R., BALO, R. G. & RODR GUEZ, Y. (2006) Learning Similarity Metrics from Case Solution Similarity. Springer.

NOMURA, H., HAYASHI, I. & N.WAKAMI (1992) A Learning Method of Fuzzy Inference Rules by Descent Method. *IEEE Int. Conf. on Fuzzy Systems 1992***,** 203-210.

ORRIOLS-PUIG, A., CASILLAS, J. & BERNADO-MANSILLA, E. (2008) Genetic-based machine learning systems are competitive for pattern recognition.

PIEÑERO PÉREZ, P. Y. (2004) Modelo para la clasificación automatizada a partir de aprendizaje automatizado aplicado al diagnóstico médico. Santa Clara, Universidad Central Marta Abreu de las Villas.

RODRIGUEZ SARABIA, Y. (2007) Generalización de la métrica basada en la diferencia de valores (VDM) para variables lingüísticas y su aplicación en sistemas basados en el conocimiento. Santa Clara, Universidad Central Marta Abreu de las Villas.

RODRÍGUEZ, Y., DE BAETS, B., GARCÍA, M. M., MORELL, C. & GRAU, R. (2008) A Correlation-Based Distance Function for Nearest Neighbor Classification.

RODRÍGUEZ, Y., DE BAETS, B. & MORELL, C. (2009) A goal-related distance function on fuzzy partitions and its use in nearest neighbours classification. *International Journal of Hybrid Intelligent Systems* 6**,** 1-9.

TAN, C. J. K. & ALEXANDROV, V. N. (2001) Relaxed Monte Carlo Linear Solver. Springer.

WILKE, W. & BERGMANN, R. (1996) Considering Decision Cost During Learning of Feature Weights. Springer.

WITTEN, I. H. & FRANK, E. (1999) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann.

Anexo 1.

Nombre de las Bases de casos	#instancias	#numéricos	#simbólicos	#clases
1. Balance-scale	625	4	0	3
2. Wisconsin-breast-cancer	683	9	0	2
3. Pima_diabetes	768	8	0	2
4. Glass	214	9	0	7
5. Heart-statlog	270	7	0	2
6. Ionosphere	351	33	0	2
7. Iris	150	4	0	3
8. Liver-disorders	345	6	0	2
9. New-thyroid	215	5	0	3
10. Sonar	208	60	0	2
11. Tao	888	9	0	2
12. Vehicle	846	18	0	4
13. Vowel	178	10	0	11
14. Wisconsin Diag. breast-cancer	569	30	0	2
15. Wine	178	13	0	3

Tabla 2.3. Descripción de las bases de casos.

Anexo 2.

Algoritmo	R^+	R^-	Valor p
EWFGJ vs. SDFGJ	32	88	0.112
EWFTJ vs. SDFTJ	30	61	0.279
EWFGM vs. SDFGM	28.5	76.5	0.132
EWFTM vs. SDFTM	57	34	0.422
EWFGM vs. MCFGM	17	88	0.026

Test de Wilconxon. Por ciento de clasificación correcto.

Algoritmo	R^+	R^-	Valor p
EWFGJ vs. SDFGJ	25.5	10.5	0.29
EWFTJ vs. SDFTJ	27	18	0.582
EWFGM vs. SDFGM	24.5	4.5	0.058
EWFTM vs. SDFTM	18	18	1

Test de Wilconxon. Error cuadrático medio.

Anexo 3.

Algoritmo	R^+	R^-	Valor p
EWFGM vs. SDFGM	15	105	0.011

Test de Wilconxon. Por ciento de clasificación correcto.

Algoritmo	R^+	R^-	Valor p
EWFGM vs. SDFGM	62.5	3.5	0.07

Test de Wilconxon. Error cuadrático medio.

Anexo 4.

Algoritmo	Z	Valor p	α/i
J48	2.73	0.006	0.01
Naive Bayes	2.1	0.03	0.0125
IBk	1.17	0.24	0.016
Logistic Regretion	1.02	0.3	0.025
MLP	6.5E-16	0.999	0.05

Test de Holm (Significación de 0.05).