Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación

Trabajo para optar por Título de

Licenciado en Ciencia de la Computación

Un método de aproximación de funciones basado en el enfoque de los prototipos más cercanos utilizando relaciones de similaridad

Autor

Marilyn Bello García

Tutores

Dra. María M. García Lorenzo

Dr. Rafael E. Bello Pérez

2012

Hago constar que el presente Trabajo para optar por Título de Licenciado en Ciencia de la Computación ha sido realizado en la facultad de Matemática, Física y Computación de la Universidad Central “Marta Abreu” de Las Villas (UCLV) como parte de la culminación de los estudios de Licenciatura en Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución para los fines que estime conveniente, tanto de forma total como parcial y que además no podrá ser presentado en eventos ni publicado sin la previa autorización de la UCLV.

_____

Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y que el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____	_____
Firma del Tutor	Firma del Tutor
Dra. María M. García Lorenzo	Dr. Rafael E. Bello Pérez

Dedicatoria

A mi mamá.

AGRADECIMIENTOS

Agradezco en primer lugar a mi mamá, por darme la vida, por cuidarme, por todo el amor, desvelo y dedicación que ha puesto en hacerme una mujer, por sacrificarse por mí una y mil veces, por celebrar todas esas pequeñas cosas que hago, por ser mi amiga, reír conmigo, llorar conmigo y perdonarme los defectos, por las veces que recogió mi caótico desorden en silencio, por ser quien le da rumbo, sentido y valor a mi familia, por ser una personas imprescindibles en mi vida, por soportarme y perdóname cuando tengo mis prontos, por apoyarme en todo momento, por sus palabras reconfortantes que me llegan justo cuando las necesito, por aceptarme como soy y ayudarme a corregir mis errores, por enseñarme que nada es fácil pero todo es posible, por siempre estar ahí a mi lado cuando la necesito, por ayudarme a lograr mis sueños y compartirlos conmigo, por creer en mí, por ser la tutora que siempre soñé, por ser una excelente profesora y sobre todas las cosas por ser mi mamá.

A mi abuela Adelfa, por ser esa persona que siempre está ahí para escucharme, para hacerme reír, para apoyarme cuando más lo necesito, por sus cariñosas y oportunas llamadas, por sus abrazos y sus besos, porque cuando tengo dificultades me anima con palabras que me hacen ver que “el sol saldrá mañana otra vez”, por sus brazos abiertos y su maravillosa sonrisa cuando me saluda.

A mi papá, por ser mi ejemplo a seguir, mi guía, mi orgullo, por ser esa persona que me hace reflexionar, por sus palabras tan concretas y directas cuando más las necesito, por enseñarme que para lograr las cosas en la vida hay que esforzarse y luchar por ellas, por ser parte de mi excelente claustro de profesores, por querer a mi mamá, a mi hermana y por quererme a mí a pesar de ser doña resabio.

A mi vecina Mary, porque siempre tiene tiempo para mí, por darme su apoyo incondicional en todo momento, por sus consejos y su energía positiva. Porque siempre está ahí escuchando mis cantaletas, por estar siempre muy pendiente de mi y sobre todo por hacerme sentir que siempre soy bienvenida en su casa como una hija más.

A mi familia, en especial a mi tía Dunia por cuidarme, quererme, consentirme y apoyarme desde que era una bebé y porque en los momentos más difíciles siempre me ayuda a ver el lado positivo de las cosas, a mi hermana Betty por ser mi hermana y estar

conmigo siempre, a mi abuela Teté por quererme y apoyarme, a mi abuelo Chury por hacerme reír cada vez que estoy con él, a mi hermano el cantante y deportista Jesús Rafael por ser la alegría de la casa, a mi tío Julio por su apoyo y por estar siempre pendiente de todas las cosas de la familia, a mi prima Yaima por ser como una hermana para mí y sobre todo a mi perrita Phoebe por ser la loquita más querida de la casa.

A todos mis compañeros de estudio y especialmente al grupo de los 8 (G-8): Laura, Enrique, Fernando, Karelia, Claudia, Millo y Jarvin, por ayudarme siempre que lo necesité, por ser mis compañeros de estudio en esta carrera tan difícil, por estar siempre presentes, por divertirnos y reírnos juntos en campismo, fiestas, festivales, juegos de futbol, caldosas y sobre toda las cosas por ser mis AMIGOS.

A mis vecinos Yurien, Cristina, Martica y Leo por ayudarme, apoyarme y estar a mi lado. Y sobre todo a Lobi por ser mi perro guardián en las caminatas al planetario.

A Isel, Gonzalo y Yaima por su gran apoyo y ayuda siempre que lo necesité.

A mis mejores amigas de la vocacional por estar siempre presentes.

A mi excelente profesor guía el Guille. Al excelente claustro de profesores que me impartió clases durante estos cinco años de la carrera, por ayudarme a convertirme en toda una profesional, en especial a los profesores Ramiro, Grau, Gladitas, Mateo, Zenaida, Isis, Tamara, Leticia, Deborah, Morell, Betty, Carlos García y Daniel.

A Dios, a la Virgen de la Caridad del Cobre y a mis ángeles de mi guarda por cuidar de mi familia y de mí en todo momento, por hacer realidad mis sueños y los de mi familia y sobre todo por darnos mucha salud y mucha felicidad.

A todos aquellos que de una manera u otra me brindaron su apoyo incondicional, creyeron en mí y sobre todo formaron parte junto conmigo durante estos cinco años de esta gran aventura estudiantil.

RESUMEN

En el trabajo se estudia el problema de la aproximación de funciones usando el paradigma de los prototipos más cercanos.

Se propone un método para construir prototipos usando relaciones de similaridad y un método de aproximación de funciones que utiliza del conjunto prototipos. Para cada clase de similaridad se construye un prototipo. Los resultados experimentales muestran que el método propuesto para construir prototipos logra una reducción significativa de la cantidad de instancias mientras la precisión se preserva.

El método aproximación de funciones que usa el conjunto prototipos y el constructor de prototipos se incorporaron a la plataforma WEKA. También se incorporó una adaptación del algoritmo de los k vecinos más cercanos para dar solución a problemas de aproximación de funciones usando relaciones de similaridad.

La incorporación de estos dos métodos de aproximación de funciones es de vital importancia para los investigadores del campo del Aprendizaje Automatizado al contar con nuevos métodos para dar solución a problemas de regresión.

El método de aproximación de funciones basado en prototipos más cercano que es propuesto en este trabajo se utilizó para resolver dos problemas reales de la Ingeniería Civil, específicamente en el pronóstico de la capacidad resistente en muros de mampostería y la capacidad resistente al deslizamiento longitudinal en las losas compuestas con lámina colaborante.

ABSTRACT

In this work the problem of the approximation of functions is studied using the paradigm of the nearest prototype.

It is proposed a method for building prototypes using relations of similarity and a method of approximation of function that makes use of the set of prototypes. A prototype is built for each class of similarity. The experimental results show that with the proposed method a relevant reduction of the quantity de instances is obtained while the precision is preserved.

The approximation method of function that makes use of the set of prototype and the prototype constructor are incorporated to the platform WEKA. Also an adaptation of the algorithm of the k nearest neighbors is incorporated for giving solution to the problem of approximation using relation of similarity.

The incorporation of these two methods of approximation of function has a high importance for the investigators of the field of the machine learning because this provides new methods for giving solutions to regression problems.

The method of approximation of function based in nearest prototypes proposed in this work was utilized to solve two real problems pertaining to the Civil Engineering, specifically in the prognostic of the resistant capacity in structural masonry and the resistant capacity to the glide longitudinal in the composite slabs with steel desk.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Herbert Simon define aprendizaje como cualquier cambio en un sistema que le permite a este resolver mejor una tarea la segunda vez u otra tarea similar.(Simon, 1983)

El Aprendizaje Automatizado o *Machine Learning* se define como un subcampo de la Inteligencia Artificial (IA) que se ocupa de aquellos programas capaces de aprender a partir de la experiencia.(Russell and Norvig, 1996)

El Aprendizaje Automatizado puede tener diferentes fuentes de conocimiento. Una puede ser el propio humano, que actúe como tutor para que la computadora aprenda. Otra puede ser un conjunto de ejemplos de problemas resueltos del dominio de aplicación.

El resultado del aprendizaje puede ser conocimiento explícito o implícito. Es explícito cuando producto del proceso de aprendizaje se obtiene conocimiento en alguna forma, por ejemplo reglas u operadores; ejemplo de este tipo de aprendizaje es el algoritmo ID3 y sus descendientes. Es implícito cuando no se obtiene conocimiento explícito desde el conjunto de ejemplos pero este sirve para que la computadora pueda resolver nuevos problemas o alcanzar mejores soluciones; también denominado aprendizaje perezoso o *lazy learning*; los algoritmos de aprendizaje basado en instancias son un ejemplo de este tipo de aprendizaje.

Los algoritmos de aprendizaje basado en instancias, presentan problemas de escalabilidad cuando el tamaño del conjunto de entrenamiento crece, porque su tiempo de clasificación es proporcional al tamaño del clasificador, es decir, a la cantidad de instancias de entrenamiento (García-Duran et al., 2010). El método k-Vecinos más Cercanos (k-NN) del inglés k-*Nearest Neighbour* (Cover and Hart, 1967) es un ejemplo de esto por su alto costo computacional cuando crece la cantidad de instancias.

Este problema se ha enfrentado mediante la modificación del conjunto de instancias en (Barandela, 2001) y (Barandela et al., 2002). De acuerdo a (Lozano et al., 2006), estos métodos se pueden clasificar en Edición o Reducción. El primero es un paso en el proceso de aprendizaje encargado de incrementar la precisión de las predicciones, cuando existe una cantidad sustancial de ruido en los datos de entrenamientos. El paso reducción es dirigido a determinar un conjunto pequeño de prototipos sin una significativa degradación en la precisión de la clasificación.

Una alternativa para resolver la limitante anterior es conocida como enfoque basado en prototipos (*Nearest Prototype*, NP) (Bezdek and Kuncheva, 2004). La idea es determinar el valor del rasgo de decisión de un nuevo objeto analizando su similaridad respecto a un conjunto de prototipos.

El propósito del enfoque NP es decrecer los costos de almacenamiento y procesamiento de las técnicas de aprendizaje basadas en instancias. De acuerdo a (Kim and Oommen, 2003), cuando hay una enorme cantidad de datos, una posible solución es reducir la cantidad de vectores ejemplos, mientras la eficacia en la solución de problemas basada en los datos reducidos se mantiene tan bien o casi tan bien, que cuando se usa el conjunto de datos original. Un pequeño conjunto de prototipos tiene la ventaja de un bajo costo computacional y requerimientos de memoria pequeños, mientras logra una eficacia similar, e incluso mejor que con todas las instancias. Para reducir la cantidad de instancias, hay dos estrategias: Selección y Remplazo (Jin et al., 2010). En el primer caso, una cantidad limitada de puntos del conjunto de datos original son conservados. En la segunda alternativa, el conjunto de datos originales se remplaza por un número de prototipos que no necesariamente coincide con alguna instancia original. Los algoritmos para encontrar prototipos también se pueden clasificar como determinísticos o no-determinísticos, dependiendo de si pueden o no controlar la cantidad de prototipos generados por los algoritmos (Kim and Oommen, 2003).

Entre las problemáticas del Aprendizaje Automatizado se pueden citar: los problemas de clasificación, asociación, formación de grupos o *clustering* y la selección de rasgos.

El problema de la selección de rasgos consiste en determinar con cual subconjunto de rasgos de todos los que permiten describir los objetos del dominio de aplicación se alcanza una mejor eficiencia o eficacia en el proceso de aprendizaje, según sea el objetivo que se persigue. La determinación del subconjunto de rasgos incide de diferentes formas en el aprendizaje, puede disminuir la información necesaria eliminando rasgos irrelevantes, o incrementar la precisión del conocimiento descubierto. La selección de rasgos puede dar como resultado un vector binario donde cada componente del vector representa cuales son los rasgos relevantes con 1 y cuales son irrelevantes con 0 o puede dar como resultado un conjunto de pesos $\mathbf{\{w_1, w_2,\ldots, w_{n-1}\}}$, que denota la importancia de cada rasgo (donde $\mathbf{w_i}$ toma un valor entre 0 y 1 para i=1,…, n-1).

La problemática de la clasificación consiste en a partir de un conjunto de aprendizaje conformado por ejemplos del área de aplicación que se trate encontrar un algoritmo que

permita clasificar un nuevo caso del dominio. Cada ejemplo se describe por un conjunto de rasgos predictores de distintos dominios y al menos un rasgo objetivo cuyo dominio representa una clase, categoría o grupo del problema en cuestión. De esta forma, el conjunto de aprendizaje se compone por subconjuntos representativos de las diferentes clases del problema y sobre la base de los ejemplos se aprende a que clase pertenece un nuevo ejemplo.

Sin embargo, no siempre el rasgo objetivo del problema es un rasgo discreto que representa a una categoría, pudiera presentarse el caso de que este atributo se representa por un valor de dominio continuo. En estos casos estamos en presencia de un problema de regresión o aproximación de funciones.

El enfoque de los prototipos más cercanos se ha utilizado tradicionalmente en problemas de clasificación, en los cuales el valor de decisión es un valor discreto que representa una clase. La aplicación del paradigma NP a los problemas de aproximación de funciones tiene sus particularidades, ya que al no contar con el concepto de clase, la construcción de los prototipos es diferente.

Actualmente aparecen las necesidades de predicción de una variable continua en ramas como: Ingeniería Civil, Meteorología, Bioinformática, Medicina, Economía, entre otras.

En la rama de la Ingeniería Civil una de las grandes problemáticas que ha tenido el hombre ha sido la necesidad de tener un lugar, un refugio donde protegerse de las inclemencias del tiempo. Para ello comenzó la búsqueda de materiales accesibles que fueran fáciles de utilizar y que proporcionaran la mayor comodidad. La mampostería es uno de los materiales de construcción con una gran diversidad de usos, en todas partes del mundo, tanto en el pasado como en el presente. Es un material que por su propia naturaleza, exhibe un comportamiento frágil ante los esfuerzos de tensión, caracterizado por una rápida degradación de su resistencia y su rigidez. Actualmente se cuenta con un volumen grande de información que proviene del conjunto de los programas experimentales internacionales realizados en el campo de la mampostería y constituye un problema no resuelto para los ingenieros civiles el pronóstico de la capacidad resistente de los muros de mampostería.

De la misma manera en la Ingeniería Civil las losas compuestas formadas por láminas de acero como encofrado colaborante constituyen soluciones muy eficientes en la construcción. Para los especialistas de esta rama es importante conocer los fallos que se producen en este tipo de estructuras.

El problema de la experimentación real clásica es que requiere de cuantiosos recursos. Se han elaborado herramientas computacionales sobre la base de métodos numéricos, capaces de simular de manera virtual experimentos reales. Estos avances han sido aprovechados en la actualidad debido a la factibilidad que tienen en el sentido de que el gasto económico es menor, sin embargo esta modelación virtual implica un alto costo computacional y tiene la desventaja de la laboriosidad necesaria en la elaboración de los modelos.

El **objetivo general** de esta investigación es:

Desarrollar un método de aproximación de funciones basado en el enfoque de los prototipos más cercanos utilizando relaciones de similaridad.

Para lograr este objetivo se plantean los siguientes **objetivos específicos**:

1. Desarrollar un método para construir prototipos usando relaciones de similaridad.
2. Desarrollar un método de aproximación de funciones que utilice el conjunto de prototipos.
3. Estudiar el efecto de la selección de rasgos en el método para construir prototipos y el aproximador de funciones.
4. Adicionar a la plataforma WEKA el método de aproximación de funciones y el constructor de prototipos.
5. Validar el método de aproximación de funciones propuesto en este trabajo comparándolo con otras técnicas de aproximación de funciones.
6. Aplicar el método propuesto en dos problemas reales del campo de la Ingeniería Civil.

Se formula la siguiente **hipótesis** de investigación:

La construcción de prototipos basado en relaciones de similaridad aminora el costo computacional del algoritmo de los k vecinos más cercanos en problemas de regresión y preserva la eficacia en la solución.

Las **tareas de investigación** trazadas son:

1. Definir el método para construir prototipo.

2. Proponer un método para aproximar funciones a partir de los prototipos construidos.
3. Analizar qué métodos de ponderación rasgos implementar para evaluar el efecto de la importancia de los rasgos en la construcción de prototipos basados en relaciones de similaridad.
4. Diseñar experimentos para comprobar la eficiencia y eficacia del método para aproximar funciones a partir de los prototipos construidos.
5. Evaluar la efectividad del método de aproximación de funciones propuesto en la solución de dos problemas reales de la Ingeniería Civil, específicamente en el pronóstico de la capacidad resistente en muros de mampostería y la capacidad resistente al deslizamiento longitudinal en las losas compuestas con lámina colaborante.

La **justificación de la investigación** es:

El método de aprendizaje basado en los prototipos más cercanos se ha aplicado satisfactoriamente para resolver problemas tanto de clasificación como aproximación de funciones; sin embargo, en su aplicación una problemática es como construir el conjunto de prototipos. La construcción del conjunto de prototipos sigue siendo de interés, pues determina tanto la eficiencia (a menor cantidad de prototipos más eficiencia en la solución de los problemas) como en la eficacia (la calidad de la solución obtenida depende de los valores objetivos de los prototipos). Esto es de especial interés en el caso de la aproximación de funciones, pues la mayoría de los métodos para la construcción de prototipos utilizan las clases.

La **tesis** está **estructurada** en tres capítulos:

En el primer capítulo se estudia el aprendizaje automatizado, se trata la aproximación de funciones como caso particular de la clasificación, describiéndose métodos para dar solución a esta problemática y se profundiza en el método de los prototipos más cercanos.

En el segundo capítulo se propone un método para la construcción de prototipos a partir de relaciones de similaridad construidas usando la medida calidad de la similaridad. El método propuesto incluye dos algoritmos, uno que permite realizar la aproximación a

partir de los prototipos construidos y otro para construir los prototipos. Se evalúa la eficiencia y eficacia del método propuesto.

En el tercer y último capítulo se describen el diseño e implementación del método de aproximación de funciones y su incorporación a WEKA. Finalmente se validan los resultados teóricos de la investigación en el área de la Ingeniería Civil, específicamente se da solución al pronóstico de la capacidad resistente en muros de mampostería y de la capacidad resistente al deslizamiento longitudinal en las losas compuestas con lámina colaborante.

CAPÍTULO 1 LA REGRESIÓN NUMÉRICA COMO UNA PROBLEMÁTICA DEL APRENDIZAJE AUTOMATIZADO.

En este capítulo se realiza una breve reseña sobre el aprendizaje automatizado, se caracteriza este atendiendo a varias categorías y son estudiados algunas de las problemática que se tratan en este campo. La aproximación de funciones se analiza como un caso particular de la clasificación y son descritos los métodos clásicos para dar solución, tanto a problemas de clasificación como de aproximación de funciones. Se parte de k-NN por su simple implementación, enfatizándose en su alta complejidad computacional cuando crece la cantidad de casos. Se estudia el método de los prototipos más cercanos como una variante para enfrentar esta limitante analizándose como trabajar éste en problemas de clasificación y regresión numérica.

1.1 Breve reseña sobre el Aprendizaje Automatizado.

El Aprendizaje Automatizado es una rama de la IA cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender, o sea crear programas que puedan, en un sentido similar a lo realizado por los humanos, aprender por sí mismos. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. En muchas ocasiones el campo de actuación del mismo se solapa con el de la Estadística, ya que las dos disciplinas se basan en el análisis de datos. Sin embargo, el Aprendizaje Automatizado se centra más en el estudio de la complejidad computacional de los problemas. Muchos problemas son de clase *NP-hard*, por lo que gran parte de la investigación realizada en esta rama se enfoca al diseño de soluciones factibles a esos problemas.

Desde que nacemos hasta que morimos los seres humanos adquirimos conocimientos, desarrollamos habilidades para analizar y evaluar a través de métodos y técnicas nuevos conocimientos, así como también por medio de la experiencia propia, pero a las computadoras hay que programarlas para que puedan “aprender”. También debemos tener en cuenta que no es sólo almacenar conocimiento, si no que es imprescindible saber aplicarlo teniendo en cuenta el cómo y el cuándo.

El Aprendizaje Automatizado tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

Algunos sistemas de este tipo intentan eliminar toda necesidad de intuición o conocimiento experto de los procesos de análisis de datos, mientras otros tratan de establecer un marco de colaboración entre el experto y la computadora. De todas formas, la intuición humana no puede ser reemplazada en su totalidad, ya que el diseñador del sistema ha de especificar la forma de representación de los datos y los métodos de manipulación y caracterización de los mismos.

1.1.1 Aprendizaje a partir de ejemplos.

El aprendizaje a partir de ejemplos en su formulación más simple es el proceso que opera sobre una colección de ejemplos, cada uno de los cuales está descrito mediante un conjunto de rasgos. Este proceso de aprendizaje tiene dos momentos principales: la generación de conocimiento o solución de un nuevo problema y la validación del resultado.

Para poder desarrollar estas dos etapas la colección de ejemplos se divide en dos conjuntos: el conjunto de entrenamiento y el conjunto de control. Con el primero se realiza el aprendizaje, mientras que con el segundo se evalúa la calidad de los resultados. La principal cualidad a medir en todo método de aprendizaje es su capacidad de generalización; es decir, la capacidad del método de poder resolver problemas no vistos antes. La capacidad de generalización se mide a partir de los resultados alcanzados con la muestra de control.

1.1.2 Clasificación del aprendizaje según los datos.

Según los datos de la muestra de entrenamiento, los cuales describen al problema en función de n rasgos de diferentes dominios, el aprendizaje se clasifica de cuatro formas diferentes y son:

Aprendizaje supervisado: El algoritmo produce una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema. Para este tipo se distinguen perfectamente rasgos predictores o rasgos de entrada y los rasgos objetivos o de salida, respondiendo estos a una etiqueta. Un ejemplo de este tipo de algoritmo es el

problema de clasificación, donde el sistema de aprendizaje trata de etiquetar (clasificar) una serie de vectores utilizando una entre varias categorías (clases). La base de conocimiento del sistema está formada por ejemplos etiquetados. Este tipo de aprendizaje puede llegar a ser muy útil en problemas de investigación biológica, biología computacional y Bioinformática.

Aprendizaje no supervisado: Todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos, o sea la base de conocimiento del sistema está formada por ejemplos no etiquetados. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas. Una forma de aprendizaje no supervisado es la agrupación o *clustering*.

Aprendizaje semisupervisado: En este tipo se combinan los dos modos anteriores para poder clasificar de manera adecuada. Se tiene en cuenta los datos marcados y los no marcados.

Aprendizaje por refuerzo: El algoritmo aprende observando el mundo que le rodea. Su información de entrada es el *feedback* o retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error.

1.1.3 Clasificación del aprendizaje según el tipo de conocimiento que se genera.

El problema del aprendizaje se ha enfrentado desde diferentes perspectivas: el enfoque simbólico, el conexionista y el evolutivo. El enfoque simbólico puede ser inductivo o perezoso. El enfoque conexionista está orientado al aprendizaje cuando se resuelven problemas usando redes neuronales artificiales. El enfoque evolutivo ocurre en los algoritmos genéticos.

El aprendizaje inductivo se usa para adquirir conocimiento (formulado en forma de descripciones intencionales) a partir de ejemplos. El objetivo de la inducción es formular afirmaciones que explican los hechos dados y se pueden aplicar a hechos no vistos con anterioridad. Estas afirmaciones pueden expresarse como patrones, y estos patrones se representan por vectores, ecuaciones, árboles, reglas o enunciados lógicos. Muchos problemas de inducción se pueden describir como sigue. Se parte de un conjunto de entrenamiento de ejemplos preclasificados, donde cada ejemplo (también llamado observación o caso) se describe por un vector de valores para rasgos o

atributos, y el objetivo es formar una descripción que pueda ser usada para clasificar ejemplos previamente no vistos con alta precisión (Bello and Osorio). El algoritmo ID3 desarrollado por (Quinlan, 1986), es un ejemplo de modelo de aprendizaje inductivo. ID3 sólo permite el tratamiento de datos simbólicos, sus extensiones C4.5 (Quinlan, 1993) y C5.0 (Quinlan, 2000) posibilitan el tratamiento de datos numéricos.

En el aprendizaje perezoso, al igual que en el aprendizaje inductivo, se aprende a partir de ejemplos, pero a diferencia de este:

i) Se usa descripción extensional, sin generar descripciones intencionales.
ii) El proceso de aprendizaje y el proceso de usar el conocimiento aprendido para resolver nuevos problemas no se separan.
iii) El paso de generalización se retrasa para la fase de solución de problemas.

Cuando se resuelve un nuevo problema P, la solución de un problema viejo se transfiere (quizás transformada) a P. Hay una generalización implícita entre el viejo y el nuevo problema (Bello and Osorio). Un método clásico de aprendizaje perezoso es el algoritmo de los k-Vecinos más Cercanos.(Cover and Hart, 1967)

Métodos de aprendizaje perezoso bien conocidos son: el aprendizaje basado en instancias, el razonamiento analógico, y el razonamiento basado en casos.

La esencia del aprendizaje basado en instancias es retornar como solución a un problema, la solución conocida a un problema similar. Los algoritmos de aprendizaje basado en instancias se basan en ejemplos modelos; cada concepto se representa por un conjunto de ejemplos, cada ejemplo puede ser una abstracción del concepto o una instancia individual del concepto. Una extensión del aprendizaje basado en instancias consiste en usar los k casos más parecidos en lugar del más cercano, lo cual es apropiado cuando los ejemplos se describen mediante datos mezclados, no sólo mediante rasgos con dominio real.

Los modelos de redes neuronales artificiales, o simplemente redes neuronales, se conocen por diversos nombres como modelos conexionistas o modelos de procesamiento distribuido paralelo. En lugar de ejecutar un programa secuencialmente como en una arquitectura Von Neumann, la red neuronal explora muchas hipótesis simultáneamente usando redes masivamente paralelas compuestas de muchos elementos de procesamiento conectados por enlaces con pesos. Los modelos de redes neuronales son especificados por la topología de la red (estructura y tipo de enlaces), las características de los nodos (modelo de la neurona) y las reglas de aprendizaje (método de ajustar los pesos). Perceptron Multicapa (MLP) del inglés *Multilayer Perceptron* es

un modelo de red neuronal artificial (Werbos, 1974, Rumelhart, 1986, Witten and Frank, 2005). El MLP es reconocido como una de las más potentes redes neuronales para solucionar un problema de clasificación supervisada a partir de ejemplos, no obstante, se le señala que se comporta como una caja negra y no facilita la interpretación de los resultados (Borgelt and Kruse, 2003). Este modelo de RNA se ha utilizado en múltiples aplicaciones a problemas de clasificación y de aproximación de funciones. (González, 1995, García and Bello, 2003, Areerachakul and Sanguansintukul, 2010)

Los algoritmos genéticos (AG) son modelos abstractos de la genética natural y los procesos evolutivos. El proceso principal de un AG se puede describir de la forma siguiente: al inicio se generan aleatoriamente soluciones o cromosomas para el problema que se resuelve. Luego se realizan iteraciones. Cada iteración consiste de varios pasos, en ellos se seleccionan buenas soluciones y se realizan cruces, ocasionalmente también mutaciones. Los algoritmos genéticos se pueden considerar métodos de búsqueda aleatoriamente guiados. También son presentados como un modelo de aprendizaje automatizado basado en la evolución natural; viendo el aprendizaje como una competencia entre una población de candidatos a solución de un problema.(Herrera and Lozano, 2003, García-Martínez, 2008)

1.2 Tipos de problemas del Aprendizaje Automatizado.

Se pueden citar entres las problemáticas del Aprendizaje Automatizado: los problemas de clasificación, asociación, formación de grupos y la selección de rasgos.

1.2.1 El problema de la selección de rasgos.

El problema de la selección de rasgos, dado por la necesidad de focalizar sobre la información más relevante para realizar las inferencias cuando se manejan grandes volúmenes de información, continúa siendo un problema de importancia (Xu and Setiono, 2003) (Yu and Liu, 2004a, Yu and Liu, 2004b, Yu and Liu, 2005). Esta situación aparece al tratar los datos de las empresas, y la información científica, donde existen muchos rasgos y muchos ejemplos; también se manifiesta en los procesos de búsqueda de información en la *World Wide Web* (WWW) donde no toda la información tiene la misma calidad.

La selección de rasgos consiste en seleccionar el subconjunto de rasgos relevantes del dominio de aplicación. Un rasgo relevante es un rasgo que cuando no se considera para describir el objeto a clasificar se deteriora el desempeño o la precisión del clasificador. Es útil para: reducir la dimensionalidad del problema, y con ello simplificar el clasificador, incrementar la velocidad de manipulación de los datos, y mejorar el desempeño del clasificador reduciendo la influencia de los ruidos.

El concepto de selección de atributos ha evolucionado con el tiempo. En una primera etapa el resultado de la operación de selección era binario, es decir, luego de la aplicación de los criterios que forman la operación de selección, un rasgo se considera o se ignora. La operación de selección S se puede modelar según la función de la expresión 1.1:

$$f_S(x_i) = \begin{cases} 1 & \text{si } x_i \text{ cumple los criterios de selección} \\ 0 & \text{en otro caso} \end{cases} \qquad (1.1)$$

Es decir, 1 si el atributo se considera relevante, y 0 si es irrelevante y por lo tanto, se ignorará.

Este enfoque se generalizó mediante la introducción del concepto de grado de relevancia, peso o importancia del atributo, el cual a cada atributo le asigna un valor en el intervalo [0,1], o sea como se muestra a continuación en la expresión 1.2:

$$f_S(x_i) = w_i, \qquad w_i \in [0,1] \qquad (1.2)$$

El proceso de selección de atributos involucra 4 pasos:

1. Generación de candidatos, lo cual incluye una estrategia de búsqueda.
2. Evaluación de candidatos, que requiere un criterio de evaluación.
3. Criterio de parada, que puede darse por la estrategia de búsqueda y puede ser el número de iteraciones realizadas, el número de atributos seleccionados, el que no se mejore el criterio de evaluación al añadir (quitar) otro atributo, que el error de clasificación está por debajo a un umbral, la no mejora en la calidad del algoritmo de aprendizaje puede usarse como criterio de parada, etc.
4. Validación de resultados, si se conocen previamente cuáles son los atributos relevantes, podemos comparar el resultado del algoritmo contra esos atributos conocidos. Como normalmente no se sabe, podemos comparar el error en la clasificación con y sin la selección de atributos.

Las técnicas de selección de rasgos pueden ser organizadas en tres categorías, dependiendo de cómo ellas combinen la búsqueda del subconjunto de rasgos con la construcción del modelo de clasificación y son: métodos de filtrado (*filter*), envoltura (*wrapper*) y sistemas integrados (*embedded*).(Saeys and Inza, 2007)

Las técnicas de filtrado calculan la relevancia de los rasgos sólo teniendo en cuenta las propiedades de los datos. En la mayoría de los casos se calcula el peso o relevancia para los rasgos, y los rasgos con más bajo peso son eliminados. Además, este subconjunto de rasgos se presenta como entrada al algoritmo de clasificación. Estás técnicas reducen dimensionalidad fácilmente, son simples y rápidas computacionalmente, y son independientes del algoritmo de clasificación. Como resultado, la selección de rasgos necesita ser ejecutada una sola vez, y entonces diferentes clasificadores pueden ser evaluados. Una desventaja común es que ignoran la interacción con el clasificador (la búsqueda en el espacio del subconjunto de rasgos está separada de la búsqueda en el espacio de la hipótesis).

Las técnicas de envoltura generan varios subconjuntos de rasgos y se evalúan mediante el entrenamiento y prueba del modelo de clasificación. Como consecuencia, la selección de rasgos está directamente relacionada con un modelo de clasificación específico. Sin embargo, como el espacio del conjunto de rasgos crece exponencialmente con el número de rasgos, se usan métodos heurísticos para guiar la búsqueda por un subconjunto óptimo. Estos métodos de búsqueda pueden ser divididos en dos clases: determinísticos y aleatorios. Las ventajas de estas técnicas incluyen la interacción entre la búsqueda de un subconjunto de rasgos y la selección del modelo, y la habilidad de tener en cuenta las dependencias entre rasgos. Sus desventajas son: el alto riesgo de sobreentrenamiento (*overfitting*) y la complejidad computacional, especialmente si construir el clasificador tiene un alto costo computacional.

En los sistemas integrados la selección de rasgos es parte del proceso de modelación, o usa parámetros del algoritmo de clasificación para seleccionar un conjunto de rasgos relevantes. Ejemplos de este tipo de selección de rasgos son los algoritmos de árboles de decisión (ID3, C4.5, CHAID, etc.), donde la selección de rasgos es construida implícitamente durante la construcción del árbol. Tanto el mecanismo de parada, como la poda del árbol son dos formas de selección de rasgos.

La generación de candidatos involucra una estrategia de búsqueda. Para esto se requiere especificar el punto inicial, que afecta la dirección de la búsqueda. Se puede empezar con un conjunto de atributos vacío y se van añadiendo otros (*forward selection*), se

puede empezar con todos los atributos y se van eliminando atributos (*backward elimination*) o se puede añadir y quitan atributos (bi-direccional). El punto inicial también puede ser un conjunto de atributos aleatorios. Dado un criterio de evaluación, se pueden utilizar diferentes estrategias de búsqueda. Por ejemplo, *Best-First*, aunque impone fuertes limitaciones de memoria, *Beam-Search*, para controlar la cantidad de memoria, *Hill Climbing*, Búsqueda Local, Búsqueda Tabú, ACO (*Ant Colony Optimization*), algoritmos genéticos, Optimización basada en Enjambres (*Swarm 0ptimization*), etc. En general, podemos dividir a las estrategias de generación de candidatos en: completas (el que sea completo no necesariamente implica que sea exhaustivo), heurísticas, y aleatorias.

En resumen, la selección de rasgos tiene dos usos principales: para reducir el número de rasgos en términos de los cuales se deben describir los objetos en modo eficiente y para encontrar los rasgos que inciden en el problema de manera determinante (o la importancia de los mismos).(Shulcloper and Arenas, 1999)

1.2.2 El problema de la clasificación.

La clasificación consiste en dado un universo de objetos (por ejemplo pacientes) se agrupa en un número dado de clases (enfermedades) de las cuales se tiene de cada una, una muestra de objetos (no todos) que se sabe pertenecen a ella (ya fueron diagnosticados) y el problema consiste en dado un nuevo objeto (un nuevo paciente) poder establecer sus relaciones con cada una de dichas clases (enfermedades). Ejemplo de este tipo de problemas, como ya hemos mencionado, lo constituye el diagnóstico diferencial de enfermedades; digamos por caso, el de bronquiolitis y asma en niños menores de cierta edad (cuatro años). En este caso se tiene una población bien definida: los niños menores de cuatro años; se tienen además sendos grupos de niños a los cuales se les ha practicado una serie de estudios que arrojó como conclusión el padecimiento de una u otra enfermedad. El problema es, dado un niño que llega de pronto a un cuerpo de emergencia hospitalaria con un cuadro de sibilancia, determinar si es una bronquiolitis o un asma, para poder aplicar el tratamiento correcto en el momento indicado.(Shulcloper and Arenas, 1999)

El objetivo de la clasificación es determinar la clase de un nuevo problema a partir de un conjunto de aprendizaje conformado por ejemplos del área específica, donde cada ejemplo se describe por un conjunto de rasgos predictores de distinto dominio (discreto o continuo) y un rasgo objetivo discreto o rasgo clase, de manera que el conjunto de

aprendizaje se compone por subconjuntos representativos de las diferentes clases y a partir de ellos se aprende a que clase pertenecen nuevos ejemplos.

No existe un modelo de clasificador mejor que otro de manera general; para cada problema nuevo es necesario determinar con cuál se pueden obtener mejores resultados, y es por esto que han surgido varias medidas para evaluar la calidad de la clasificación y comparar los modelos empleados para un problema determinado. Las medidas más conocidas para evaluar la clasificación están basadas en la matriz de confusión que se obtiene cuando se prueba el clasificador en un conjunto de datos que no intervienen en el entrenamiento. A continuación en la Figura 1.1 se muestra la matriz de confusión de un problema de dos clases:

		Clasificado como	
		Si	No
Clase real	Si	Verdadero Positivo (VP)	Falso Negativo (FN)
	No	Falso Positvo (FP)	Verdadero Negativo (VN)

Matriz de Confusión

$N = VP + VN + FP + FN$

- Tasa de acierto $s = \frac{VP+VN}{N}$
- Tasa de error $\varepsilon = 1 - s$

Figura 1.1 Matriz de confusión de un problema de dos clases

Donde, VP y VN son los elementos bien clasificados de la clase positiva y negativa respectivamente. FP y FN son los elementos negativos y positivos mal clasificados respectivamente.

Como forma de comparar clasificadores se propone la curva ROC (Fawcett, 2005). Se usa el área bajo esta curva, denominada AUC (*Area Under the Curve*) como un indicador de la calidad del clasificador. En tanto dicha área esté más cercana a 1, el comportamiento del clasificador está más cercano al clasificador perfecto (aquel que lograría 100% de VP con un 0% de FP).

Pero estas medidas no son suficientes, usándolas así simplemente, para evaluar un clasificador. La forma de dividir los datos en conjunto de entrenamiento y prueba es también muy importante. Existen varias técnicas para esto, la más vieja es el método R (*resubstitution*) el cual se basa en entrenar y probar el clasificador con la misma base de datos. Este método puede traer como consecuencia un sobre-aprendizaje del clasificador, o sea que el clasificador más que generalizar el conocimiento de los datos,

aprenda estos "de memoria". Otro método usado es el método H (*Hold-out*), que divide la base, en un % para entrenamiento y otro % para prueba o muestra de control. Una versión de éste es el *Data Shuffle* que realiza n veces el método H y promedia los resultados. Validación cruzada k veces (*K-fold Cross-validation*) es uno de los más usados, este método se basa en dividir la base en k particiones y realizar k procesos de entrenamientos y pruebas, donde el proceso i se basa en tomar la partición i para prueba y el resto para entrenamiento. Sea I la cantidad de instancias de la base de datos, si k=I entonces el método se denomina *Leave-One-Out*. El método *Boostrap* se basa en la generación de n conjuntos de cardinalidad I desde el conjunto de datos original, con reemplazo. Actualmente se usa también en vez de dos conjuntos de datos, tres: uno para entrenamiento, uno para prueba y un tercero para validación.(Kuncheva, 2005, Kohavi, 1995, Efron and Gong, 2008)

Métodos para dar solución al problema de clasificación:

Diferentes métodos se han empleado para dar solución al problema de clasificación entre los cuales aparecen: k-Vecinos más Cercanos, Redes Bayesianas, árboles de decisión, redes neuronales artificiales, entre otros.

Redes Bayesianas: Una red bayesiana es un modelo gráfico probabilístico que representa un conjunto de variables y sus dependencias probabilísticas. Son grafos acíclicos dirigidos, donde cada nodo representa un atributo y de cada variable en el espacio se especifican dos informaciones la estructura de dependencias condicionales y las distribuciones de probabilidades correspondientes. Estas redes pueden usarse para inferir un valor objetivo dado los valores observados de otras variables, y recíprocamente, inferir el valor probable de una variable, a partir de la evidencia de otras y/o del valor objetivo.(Mitchell, 1997, Witten and Frank, 2005)

Árboles de decisión: Un árbol de decisión es un grafo acíclico donde cada nodo especifica una prueba de algún rasgo y cada arco que sale del nodo corresponde a alguno de los valores posibles del rasgo que representa ese nodo. Estos árboles clasifican casos no conocidos comenzando por la raíz del árbol, probando el rasgo especificado en el nodo y en dependencia del valor de ese rasgo se mueve al próximo nodo. Este proceso es repetido entonces hasta alcanzar un nodo hoja. El algoritmo clásico para construir árboles de decisión es el ID3. El algoritmo ID3 descubre reglas de clasificación mediante la construcción de árboles de decisión. ID3 aprende construyendo un árbol de decisión (*decision tree*, DT) *top-down*, es un método para

aproximar funciones de valores discretos. Un DT clasifica las instancias ordenándolas *top-down* de la raíz a las hojas. Cada nodo interior del árbol especifica una prueba de algún atributo y las hojas son las clases en las cuales se clasifican los ejemplos, cada rama descendiente de un nodo interior corresponde a un valor posible del atributo probado en ese nodo. Este algoritmo no trabajaba para atributos continuos y el propio autor sugiere una variante, el C4.5 que usa puntos de corte e introduce varias medidas para evitar el *overffitting*, en particular los criterios de parada de la división y de poda del árbol. (Quinlan, 1986, Quinlan, 1993, Mitchell, 1997)

Redes Neuronales Artificiales: Las redes neuronales son modelos matemáticos que permiten obtener las relaciones funcionales subyacentes entre los datos involucrados en problemas de clasificación, reconocimiento de patrones, regresiones, etc. (Hammer and Villmann, 2003, Mitchell, 1997). Una red neuronal puede ser caracterizada por el modelo de la neurona, el esquema de conexión que presentan sus neuronas, o sea su topología, y el algoritmo de aprendizaje empleado para adaptar su función de cómputo a las necesidades del problema particular. Existe una amplia variedad de modelos de neuronas, cada uno se corresponde con un tipo determinado de función de activación de la neurona. La topología es la forma específica de conexión (arquitectura) y la cantidad de neuronas conectadas (el número de parámetros libres) que describen una red. Se ha producido una amplia variedad y clasificación de topologías de redes neuronales, la mayoría de ellas se encuentran ubicadas en uno de dos grandes grupos: las redes multicapa de alimentación hacia adelante (*Feed-Forward Neuronal Networks*, FFN) y las redes recurrentes (*Recurrent Neuronal Networks*, RNN). (Bonet, 2008)

Las redes MLP constituyen un ejemplo genérico de las redes FFN. El poder de representación de las redes MLP está relacionado con la existencia de al menos una capa de neuronas ocultas, las cuales transforman la entrada, de manera no lineal, en una representación interna, de ahí que se hable de redes multicapas; donde la primera capa representa neuronas sensoras o de entrada, que se corresponden con los rasgos predictores del problema, una o dos capas ocultas donde la cantidad de neuronas de la segunda capa oculta es la tercera parte de las neuronas de la primera capa oculta y una capa de salida donde pueden existir tantas neuronas como clases del problema existan. (Lippmann, 1988) señala el poder de representación arbitrariamente compleja que puede llegar a presentar un MLP de 4 capas. La red MLP usa un algoritmo de entrenamiento de gran potencialidad, conocido como *Backpropagation*. Este algoritmo aplica la

técnica gradiente descendente para la minimización del error de funcionamiento de la red.

Algoritmos basados en casos: La esencia de los llamados algoritmos basados en casos está en que su entrenamiento es simplemente el almacenamiento de los casos, no necesitan crear reglas, ni árboles, ni ajustar parámetros. Necesitan de la definición de una medida de distancia para comparar cada nueva instancia con las de la base de conocimientos, de forma que la instancia más cercana se usa para asignar la clase. A menudo se usa más de una instancia cercana y la clase mayoritaria es la asignada al nuevo caso, éste se denomina k-Vecinos más Cercanos. Este método se considera retardado y supervisado (pues su fase de entrenamiento se hace en un tiempo diferente al de la fase de prueba) y su argumento principal es la distancia entre instancias. Este algoritmo se ha convertido en uno de los métodos de clasificación supervisada más usados. (Aha and Kibler, 1991, Cover and Hart, 1967, Dasarathy, 1991)

1.2.3 La predicción numérica o regresión.

La definición del problema de regresión es parecida al de la clasificación, tendremos variables predictoras o atributos y un rasgo objetivo que en este caso es numérico o continuo. El objetivo de la regresión es predecir el valor numérico para una variable a partir de los valores conocidos de otras.

Todas las técnicas de validación en clasificación *Hold-out*, *K-fold Cross-validation*, *Leave-One-Out*, *Bootstraping* son válidas para predicción numérica. La diferencia está en que ahora debemos medir el error de otra forma. Se puede medir el error cometido al aproximar un conjunto de valores $\{v_1,\ldots,v_n\}$ por su estimación $\{\hat{v}_1,\ldots,\hat{v}_n\}$ de la forma en que se muestra en la Figura 1.2:

Error Cuadrático Medio (ECM) $ECM = \frac{\sum_{i=1}^{n}(v_i-\hat{v}_i)^2}{n}$	ECM estandarizado (ECME) $ECME = \sqrt{\frac{\sum_{i=1}^{n}(v_i-\hat{v}_i)^2}{n}}$
Error Medio Absoluto (EMA) $EMA = \frac{\sum_{i=1}^{n}\lvert v_i-\hat{v}_i\rvert}{n}$	Error Absoluto Relativo (EAR) $EAR = \frac{\sum_{i=1}^{n}\lvert v_i-\hat{v}_i\rvert}{\sum_{i=1}^{n}\lvert v_i-\bar{v}\rvert}$
Coeficiente de correlación $r_{v\hat{v}} = \frac{\sum_{i=1}^{n}(v_i-\bar{v})(\hat{v}_i-\bar{\hat{v}})}{(n-1)\sigma_v\sigma_{\hat{v}}}$	

Figura 1.2 Medición del error cometido al aproximar un conjunto de valores.

Métodos para dar solución al problema de regresión:

Diferentes métodos se han empleado para dar solución al problema de regresión entre los cuales aparecen métodos estadísticos como: regresión lineal, árboles de regresión, etc.; métodos basados en modelos de redes neuronales como MLP, LVQ, etc. y métodos basados en instancias como k-NN.

Regresión lineal: En estadística la regresión lineal o ajuste lineal es un método matemático que modela la relación entre una variable dependiente **Y**, las variables independientes $\mathbf{X_i}$ y un término aleatorio **ε**. Este modelo puede ser expresado como se muestra en la expresión (1.3):

$$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon \qquad (1.3)$$

Donde $\mathbf{Y_t}$ es la variable dependiente u objetivo; $\mathbf{X_1, X_2,\ldots, X_p}$ son las variables independientes o predictoras; $\mathbf{\beta_0, \beta_1, \beta_2,\ldots, \beta_p}$ son parámetros que miden la influencia que las variables independientes tienen sobre la variable dependiente, donde $\mathbf{\beta_0}$ es la intersección o término "constante", las $\mathbf{\beta_i}$ con **(i>0)** son los parámetros respectivos a cada variable independiente, y **p** es el número de parámetros independientes a tener en cuenta en la regresión. (Torres-Reyna, 2009)

Árboles de regresión: es un árbol de decisión cuyas hojas predicen una cantidad numérica, ese valor numérico se calcula como la media del valor para la variable clase de todos los ejemplos que han llegado a esa hoja durante el proceso de construcción del árbol. La evaluación de un nuevo ejemplo es idéntico a los árboles de decisión, durante el proceso de predicción es posible utilizar un suavizado de los valores del ejemplo a tratar, con el fin de salvar las posibles discontinuidades presentes en los datos. El criterio de selección de una variable en la construcción del árbol está basado en una reducción del error esperado: reducción de la desviación o varianza en la variable objetivo. Al final el árbol se poda para evitar el sobreajuste. El algoritmo clásico de árboles de regresión fue desarrollado por Breiman. (Breiman et al., 1984)

Métodos basados en redes neuronales: los modelos de RNA (Ripley, 1996, Bishop, 1995, Fausett, 1994) también han sido utilizados para dar solución a problemas de regresión. Generalmente se trata de topologías dirigidas hacia delante, donde la capa de entrada se corresponde con las neuronas sensoras o de entrada que están en correspondencia con los rasgos de entrada del problema y la última capa representa la capa de salida. El aprendizaje de los pesos se adapta en función del error cometido, y

por tanto es suficiente con medir de forma adecuada el error. Entre los modelos más trabajados en tal sentido se encuentran:

RNA del tipo multi-capa (MLP) (Rynkiewicz, 2012) y algoritmo de aprendizaje *Back-Propagation* (BP): generalmente se habla de redes de 3 capas, donde la segunda capa es la capa oculta y la capa de salida tiene una única neurona, el algoritmo de aprendizaje BP del tipo supervisado duro se basa en minimizar el gradiente descendente del error y aparecen diferentes heurísticas para optimizar el empleo de BP. El cálculo de la variación del peso en a la capa de salida se basa en una regla delta generalizada mientras que en las capas ocultas se aplica regla de Hebb. Entre las heurísticas propuestas se analizan variantes híbridas para proponer la mejor topología, obtener el conjunto inicial de pesos y ajustar el coeficiente de aprendizaje (García and Bello, 2003). Se reportan ejemplos de su aplicación en problemas de predicciones financieras (Skabar and Cloete, 2001).

Linear Vector Quantization (LVQ): es un popular modelo de RNA para dar solución a problemas de multiclasificación basados en los prototipos más cercanos. Ampliamente utilizados por su claro proceso de aprendizaje y fácil implementación. Se basa en el principio de formación de mapas topológicos para establecer características comunes entre las informaciones (vectores) de entrada a la red. Una red LVQ posee dos capas: la de entrada, con **m** neuronas se corresponde con la cantidad de rasgos de los datos de entrada y la de salida, con **c** neuronas. Cada neurona de la capa de entrada está conectada con todas las de la capa de salida y cada una de la capa de salida está conectada con sus vecinas. Las conexiones laterales son inhibitorias, además, los pesos de las conexiones entre la entrada y la salida dependerán de estas conexiones. El aprendizaje se basa en el principio de cuantificación de vectores consistente en representar un gran conjunto de datos con un número finito de prototipos, donde cada uno representa a todos los elementos que se encuentran más cerca de él, en términos de distancia Euclidiana (Kohonen, 1990). Sin embargo, existen extensiones de LVQ para problemas de regresión donde a diferencia del algoritmo de LVQ para clasificación, el algoritmo aprende los mejores prototipos de valores, obteniéndose una partición efectiva del espacio de rasgos similar a como lo haría *K-Means*.(GRBOVIC and VUCETIC, 2009)

Métodos basados en instancias como k-NN también se han utilizado en problemas de aproximación de funciones.(GYORFI, 1981, Khelifi et al., 2011, Navot et al., 2005)

1.3 k-NN como método para dar solución a problemas de clasificación y regresión.

La esencia del aprendizaje basado en instancias es retornar como solución a un problema, la solución conocida a un problema similar. El k-NN constituye un algoritmo clásico de esta forma de solución a problemas y ha sido empleado en problemas de clasificación y regresión. El método básicamente consiste en comparar la nueva instancia a clasificar con los datos o casos existentes del problema en cuestión, recuperando los k casos más cercanos, lo cual depende del parecido entre los atributos del nuevo caso con los casos de la muestra de aprendizaje o entrenamiento. Como resultado del mismo se devuelve la clase mayoritaria de aquellos k casos más cercanos a él.

Este método consta de 3 pasos fundamentales, precisamente el 3er paso marca la diferencia en predecir un rasgo objetivo discreto o continuo. Sea el conjunto de **n** casos o ejemplos $\mathbf{E_i}$, cada $\mathbf{E_i}$ se describe por el vector de **m** rasgos predictores $\mathbf{(a_1\ (E_i),\ a_2\ (E_i),\ldots,\ a_m\ (E_i))}$ y un rasgo objetivo, donde $\mathbf{a_j\ (E_i)}$ denota el valor del rasgo **j** en el ejemplo $\mathbf{E_i}$. Sea **P**, definido como $\mathbf{(a_1\ (P),\ldots,\ a_{m-1}\ (P))}$ el objeto a clasificar si el rasgo objetivo es discreto o a predecir si el rasgo objetivo es continuo entonces:

En el caso de predecir un rasgo objetivo discreto:

P1. Calcular la distancia entre **P** y cada ejemplo a partir de la expresión (1.4).

$$d(P,Ei)=\sqrt{\sum_{j=1}^{m} w_j * \left(\partial(a_j(P),a_j(Ei))\right)^2}$$

$$\partial(x,y)=\begin{cases}|x-y| & \text{si } a_j \text{ es un rasgo continuo}\\ 0 & \text{si } a_j \text{ es un rasgo discreto y } x=y\\ 1 & \text{si } a_j \text{ es un rasgo discreto y } x\neq y\end{cases} \qquad (1.4)$$

P2. Seleccionar las **K** instancias más similares a **P**.

P3. Para cada clase en $\mathbf{C=\{C_1, C2,\ldots, C_t\}}$ calcular la medida $\mathbf{\sigma(P,C_t,K)}$ definida por la expresión (1.5):

$$\sigma(P, C_t, K) = \frac{\sum_{x \in K} l(C_t, x^*)/(1 + d(P, x))}{\sum_{x \in K} 1/(1 + d(P, x))} \quad (1.5)$$

$$l(C_t, x^*) = \begin{cases} 1 & \text{si } C_t = x^* \\ 0 & \text{en otro caso} \end{cases}$$

Donde: **x*** es la clase a la que pertenece **x**.

P3.1 Asignar a **P** la clase para el cual se alcanza mayor $\boldsymbol{\sigma(P, C_t, K)}$.

En el caso de predecir un rasgo objetivo continuo:

Se define similarmente al problema de clasificación, lo que en lugar de conocer la clase a la que pertenece cada ejemplo se conoce el valor continuo $\mathbf{Y_x}$ asociado a cada ejemplo. Los pasos P1 y P2 se definen igual que el problema de clasificación, mientras que los pasos P3 y P3.1 se sustituyen por el paso P3*.

P3*. Está definido por la expresión (1.6):

$$y = \begin{cases} \dfrac{\sum_{x \in K} Y_x}{|K|} & \text{si los K objetos cuenta igual} \\ \dfrac{\sum_{x \in K} w_x Y_x}{\sum_{x \in K} w_x} & \text{si se tiene en cuenta un voto ponderado} \end{cases} \quad (1.6)$$

Donde: **y** es el valor calculado para la función en el problema **P** e $\mathbf{Y_x}$ es el valor objetivo en el ejemplo **x** que es uno de los k casos más cercanos (similares).

Medidas para determinar las instancias similares al problema:

En el paso P1 de ambos algoritmos se pueden utilizar tanto distancias como funciones de semejanza. Obviamente, si se calculan distancias se seleccionaran los **K** ejemplos de menor distancia al problema, mientras que si se usan funciones de semejanza se seleccionaran los **K** ejemplos más similares.

a. Distancia Euclídea:

$$E(x, y) = \sqrt{\sum_{a=1}^{m} (x_a - y_a)^2}$$

Donde **x** e **y** son dos vectores de atributos que representan ejemplos, m es la cantidad de atributos usados para describir cada ejemplo y $\mathbf{x_a}$ e $\mathbf{y}_a$ son los valores del atributo a-ésimo en los ejemplos **x** e **y** respectivamente.

b. Distancia de Chebychev:

$$Ch(x, y) = \max|x_a - y_a| \quad a=1,..., m$$

c. Distancia de Manhattan:

$$M(x, y) = \sum_{a=1}^{m} |x_a - y_a|$$

d. Heterogeneous Euclidean Overlap Metric (HEOM):

$$HEOM(x, y) = \sqrt{\sum_{a=1}^{m} d_h (x_a - y_a)^2}$$

$$d_h(x_a, y_a) = \begin{cases} 1, & \text{si } x_a \text{ o } y_a \text{ son desconocidos} \\ \partial(x_a, y_a) & \text{si } a \text{ es simbólico} \\ E_N(x_a, y_a) = \dfrac{|x_a - y_a|}{\max_a - \min_a} & \text{para el resto} \end{cases}$$

$$\partial(x_a, y_a) = \begin{cases} 0, & \text{si } x_a = y_a \\ 1, & \text{si } x_a \neq y_a \end{cases}$$

Ventajas del método k-NN:

Entre las ventajas del método se tiene que es de rápido aprendizaje y su capacidad de solución a partir de pocos ejemplos.

Limitantes del método k-NN:

Sin embargo, el k-NN tiene limitantes tales como:

- Costo computacional que se incrementa con las dimensiones del conjunto de entrenamiento dado por el incremento del tiempo de respuesta, siendo la complejidad temporal O (nd), tal que O (d) es la complejidad de la distancia usada y O (n) lo que se requiere para explorar todo el conjunto de ejemplos y la complejidad espacial estará dada por la necesidad de almacenar los n ejemplos descritos por m rasgos.
- Deterioro de la eficacia en presencia de ruido, o con clases muy solapadas.
- El no tratar eficientemente la relevancia.

1.4 *Nearest Prototype* como una alternativa para disminuir el costo computacional de k-NN.

La calidad de los resultados del aprendizaje estará dada en gran medida por la calidad del conjunto de entrenamiento. Este es el caso del método k-NN, si el conjunto de instancias no es representativo del problema que se investiga, posee ruido, o tiene objetos innecesarios o superfluos, el clasificador se verá afectado. Además, desde el punto de vista de la eficiencia del proceso de aprendizaje resulta de interés disminuir la cantidad de objetos a usar.

Sobre esta base y como vías de eliminar las limitantes del k-NN se han propuesto distintas heurísticas entre las que se encuentran la modificación del conjunto de entrenamiento o edición del mismo. Aparece así la necesidad de seleccionar o construir prototipos. En el primer caso se trata de obtener un subconjunto de los objetos del conjunto de entrenamiento y para el segundo crear nuevos objetos que serán utilizados como matriz de entrenamiento o modificar los objetos originales.

Una de las técnicas de aprendizaje utilizadas tanto en problemas de clasificación como aproximación de funciones es la basada en los prototipos más cercanos, del inglés *nearest prototype* (Bezdek and Kuncheva, 2004). La idea es determinar la clase de un objeto desconocido analizando su similaridad a un conjunto de prototipos, seleccionados o generados a partir de un conjunto de entrenamiento dado con clases conocidas. Los prototipos representan las características típicas de los objetos de una categoría en lugar de condiciones necesarias o suficientes; los prototipos pueden ser

abstracciones de las mismas instancias previamente observadas, o pueden ser los ejemplos directamente observados (Serra and Cucchiara, 2009). Para reducir la cantidad de prototipos, muchos métodos de selección y construcción se han propuestos (Jin et al., 2010).

Entre los métodos de selección de prototipos se pueden encontrar:

Selección por condensación: La condensación pretende buscar conjuntos minimales (o al menos pequeños), conservar los objetos representativos y aumentar eficiencia, sin sacrificar demasiado la eficacia. El método *Condensed Nearest Neighbors* (CNN) es el primer método de selección de objetos reportado en la literatura. Fue desarrollado por Hart en 1966. Es un método altamente dependiente del orden de presentación de los objetos. Su propio creador plantea que si el riesgo de Bayes es alto, entonces la lista resultante contendrá esencialmente todos los puntos presentes en el conjunto original y no se llevará a cabo ninguna reducción importante en el tamaño del conjunto. (Hart, 1968, Dasarathy, 1991)

El método *Reduced Nearest Neighbor* (RNN), creado por Gates (Gates, 1972), trata de eliminar los objetos innecesarios en la lista resultante del método CNN de Hart. Una desventaja de este método es que es dependiente del orden. Según su propio autor en muchos casos el tiempo extra de búsqueda de objetos para eliminar no se justifica con el nivel de compactación extra alcanzado.

Selección por edición: Los métodos de edición basada en el error tienen como objetivo eliminar los objetos "ruidosos" o atípicos, disminuir el riesgo de Bayes y aumentar la eficacia. El *Edited Nearest Neighbor* (ENN) es el primer método de edición reportado en la literatura basado en el error de clasificación. Fue creado por Wilson en 1972 (Wilson, 1972) y consiste en la eliminación de todos los objetos que son mal clasificados por el k-NN. Este proceso se realiza por lotes, por lo que los objetos son marcados primero, y después se eliminan simultáneamente todos los marcados.

En 1980 surgió el método *Multiedit*, el cual fue desarrollado por Devijver y Kittler (Devijver and Kittler, 1980, Devijver and Kittler, 1982). Consiste en la partición aleatoria de la matriz de entrenamiento en componentes, aplicando después a cada partición un ENN que utiliza un clasificador 1-NN entrenado con la partición siguiente. Después de cada iteración, se unen las particiones resultantes y se repite el proceso hasta que no existan cambios en iteraciones sucesivas. Según (Bezdek and Kuncheva,

2004) este método trabaja bien en grandes conjuntos de datos. Sin embargo, para conjuntos de datos pequeños, o con clases entremezcladas, puede eliminar todos los objetos de una clase. Si las clases están perfectamente separadas, prácticamente no elimina objetos.

También se conocen híbridos de métodos de edición con métodos de condensación. (Ferri, 1998)

Selección por búsqueda combinatoria: búsqueda aleatoria (Skalak, 1994, Kuncheva and Bezdek, 1998), búsqueda basada en algoritmos genéticos (Kuncheva and Bezdek, 1998, Skalak, 1994, Chang and Lippmann, 1991, Aha and Kibler, 1991, Kuncheva, 1995, Kuncheva, 1997) y la búsqueda tabú (Ceveron and Ferri, 2001, Bezdek and Kuncheva, 2004).

Entre los métodos de construcción de prototipos se encuentran:

Métodos pre-supervisados: Los métodos pre-supervisados usan los datos de entrenamiento y las etiquetas clases para localizar los prototipos. Entre ellos podemos encontrar:

LVQ (*Learning vector quantization*): La familia LVQ comprende un amplio espectro competitivo de esquemas de aprendizaje. Unos de los diseños básicos que puede ser usado para la generación de prototipos es el algoritmo LVQ1 (Kohonen, 1995). Un conjunto inicial de prototipos etiquetados es elegido de manera que el número de prototipos **n** sea mayor o igual que la cantidad de clases **c**. Entonces se seleccionan aleatoriamente **n** elementos desde **X** (conjunto de entrenamiento) para ser los prototipos iniciales, así cada clase es representada por al menos un prototipos. LVQ1 necesita 3 parámetros adicionales que deben especificarse por el usuario: la razón de aprendizaje $\boldsymbol{\alpha_k}$ es un valor entre 0 y 1; una constante $\boldsymbol{\eta}$ es un valor entre 0 y 1 y un número de iteraciones **T**. LVQ1 termina cuando no hay error en la clasificación en un paso completo a través de **X** o cuando el número de iteraciones especificado se alcanza.(Bezdek and Kuncheva, 2004, Kohonen, 1995)

DSM (*Decision surface mapping*): Es una variante de LVQ donde Geva y Sitte's (Geva and Sitte, 1991) aseguran una mejor clasificación en las fronteras del conjunto de entrenamiento que la que LVQ hace.(Bezdek and Kuncheva, 2004)

LVQTC (*LVQ with traning counters*): Este algoritmo opera similar a LVQ1 pero el ajuste de los objetos va a depender de dos factores: la distancia existente del objeto a un

determinado elemento de la matriz de entrenamiento y el historial del objeto.(Odorico, 1997, Bezdek and Kuncheva, 2004)

Métodos post-supervisados: Los métodos post-supervisados primero encuentran los prototipos sin tener en cuenta las etiquetas clases de los datos de entrenamiento y entonces asignan una etiqueta clase a cada prototipo (reetiquetado). Entre ellos podemos encontrar:

Vector Quantization (VQ): Ha sido usado para encontrar prototipos por muchos años. El algoritmo VQ comienza con una selección aleatoria desde **X** (conjunto de entrenamiento) para construir el conjunto inicial de prototipos no etiquetados según una cantidad definida por el usuario y termina cuando se alcanza el número de iteraciones **T** definidas por el usuario o cuando $| \mathbf{v_{k+1}} - \mathbf{v_k} | \leq \varepsilon$. (Xie et al., 1993, Bezdek and Kuncheva, 2004)

Generalized Learning Vector Quantization - Fuzzy GLVQ-F: Es un método no supervisado para encontrar prototipos el cual es similar VQ pero difiere en el paso actualización de prototipos.(Karayiannis and Bezdek, 1996, Bezdek and Kuncheva, 2004)

Clustering and relabeling: Una forma de construir el conjunto de prototipos consiste en agrupar y tomar como conjunto de prototipos los centroides de los *clusters*.(Bezdek and Kuncheva, 2004, Bezdek et al., 1998, Bezdek et al., 1999)

En la Figura 1.3 se muestra una visión general de métodos para generar prototipos.(Bezdek and Kuncheva, 2004)

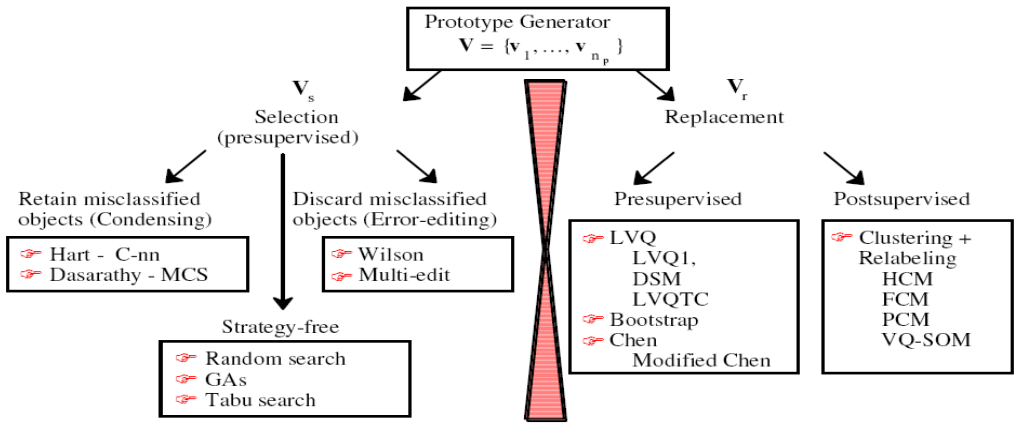


Figura 1.3 Una visión general de métodos para generar prototipos.

Desde otra perspectiva, los algoritmos para encontrar prototipos pueden ser clasificados como determinísticos o no-determinísticos, dependiendo de si pueden o no controlar la cantidad de prototipos generados por los algoritmos (Kim and Oommen, 2003).

De acuerdo a (Lozano et al., 2006), los métodos para construir los conjuntos de prototipos se pueden clasificar en Edición o Reducción. Las técnicas de Edición, aunque también producen una reducción de la matriz de aprendizaje, tienen como objetivo fundamental el obtener una muestra de entrenamiento de mejor calidad para una mejor precisión en la clasificación (Devijver and Kittler, 1980, Cortijo and Blanca, 1997). Por su parte, las técnicas de Reducción persiguen como objetivo disminuir el tamaño de la matriz de aprendizaje. Se trata de disminuir el trabajo computacional y, a veces, se está dispuesto a pagar con un poco menos de precisión en la clasificación, pero con más eficiencia computacional.(Wilson and Martínez, 1998, Wilson and Martínez, 2000, Lozano and Sánchez, 2003, Koggalage and Halgamuge, 2004)

Un componente importante en el aprendizaje NP es una función de similaridad que calcula la similaridad entre las entradas y los prototipos. Usualmente, se selecciona la distancia Euclidiana. Si el nuevo caso **x** y los prototipos son representados por vectores de valores reales, la similaridad casi siempre se basa en alguna función de distancia entre **x** y los prototipos. Sin embargo, en el caso de datos mezclados este enfoque no es aplicable. Para superar este problema se han propuesto varios algoritmos que sustituyen la métrica Euclidiana estándar por una versión adaptiva, existiendo la necesidad de otro tipo de función para comparar **x** y los prototipos. Por eso, la selección de una función correcta para calcular la similaridad entre las entradas y los prototipos es un tema importante en los enfoques basados en prototipos. (Hammer et al., 2005)

El enfoque de los prototipos más cercanos se ha usado tradicionalmente en problemas de clasificación. El caso de los problemas de regresión se estudia en (GRBOVIC and VUCETIC, 2009).

Clasificación basada en prototipos más cercanos (*nearest prototype classifier)*:

Dado un conjunto de **p** prototipos **V**, cada prototipo es un vector real que pertenece a una de **c** clases. A un nuevo objeto no etiquetado el clasificador le asigna la clase del prototipo más cercano, la cercanía se define en términos de una distancia en **R.** El problema es encontrar un buen conjunto mínimo de prototipos que permita alcanzar el error mínimo en la clasificación. Buenos prototipos tienen dos propiedades deseables: cardinalidad mínima y máxima precisión de clasificación.(Bezdek and Kuncheva, 2004)

Si el conjunto de prototipos **V** está formado por todos los objetos del universo **X**, en este caso el método se llama *nearest neighbor rule* (1-NN); o **V** contiene un prototipo por cada clase que es la media de los ejemplos para esa clase en **X**, en este caso el método se denomina *minimum distance classifier* (1-NP). Ambos son casos especiales de *nearest prototype classifier*.(Bezdek and Kuncheva, 2004)

La característica común a los *nearest prototype classifier* es que ellos calculan una similaridad entre un objeto desconocido **x** y cada prototipo **Vi (s(x,Vi))**. La regla de clasificación estándar *nearest prototype* (1-NP) asigna a un objeto x no etiquetado la clase del prototipo mas similar.(Bezdek and Kuncheva, 2004)

Problemas para aplicar este enfoque a la aproximación de funciones:

Aplicar el método 1-NN es equivalente al método k-NN para aproximación de funciones, de modo que nos interesa estudiar la variante 1-NP, para lo cual es necesario construir prototipos, los métodos de construcción de prototipos en problemas de clasificación se basa en construir prototipos a partir de los objetos de cada clase. Pero en los problemas de aproximación de funciones no hay clases, de modo que el enfoque usado en clasificación para construir prototipo no es aplicable en aproximación de funciones, entonces surge el problema de: ¿Cómo construir los prototipos?

Aplicar el enfoque del prototipo más cercano a la aproximación de funciones:

A partir de un conjunto de objetos, construir clases de similaridad, para cada clase de similaridad construir un prototipo. Para calcular el valor del rasgo de decisión de un nuevo objeto se busca los prototipos más cercanos (similares) y se promedia los valores de decisión de esos prototipos.

1.5 Conclusiones parciales del capítulo.

El Aprendizaje Automático tiene una amplia gama de aplicaciones, entre las que se encuentran los problemas de la clasificación. La regresión es un caso particular de la clasificación, se tienen variables predictoras y un rasgo objetivo que en este caso es continuo.

El método k-NN es un método de rápido aprendizaje que se utiliza tanto para resolver problemas de clasificación como de regresión y se caracteriza por su capacidad de solución a partir de pocos ejemplos, sin embargo su costo computacional se incrementa

con las dimensiones del conjunto de entrenamiento. Este problema se ha enfrentado mediante la modificación del conjunto de instancias.

Una alternativa para aminorar el costo computacional es el enfoque basado en prototipos. La idea es determinar el valor del rasgo de decisión de un nuevo objeto analizando su similaridad respecto a un conjunto de prototipos, seleccionados o generados a partir de un conjunto inicial de instancias. Existen diversos métodos para construir prototipos pero se trabajará aquellos dirigidos a la reducción del conjunto de entrenamiento.

Un componente importante en el aprendizaje basado en los prototipos más cercanos es una función de similaridad que calcula la similaridad entre las entradas y los prototipos. El problema es encontrar un buen conjunto mínimo de prototipos que permita alcanzar el error mínimo en la clasificación.

Aplicar el enfoque del prototipo más cercano a la aproximación de funciones presupone cambios a los métodos existentes dirigidos a problemas de clasificación.

CAPÍTULO 2 DESARROLLO Y VALIDACIÓN DEL MÉTODO DE APROXIMACIÓN DE FUNCIONES BASADO EN EL ENFOQUE DE LOS PROTOTIPOS MÁS CERCANOS UTILIZANDO RELACIONES DE SIMILARIDAD.

En este capítulo se estudia el problema de aproximación de funciones usando el enfoque de *nearest prototype*. Se propone un método para la construcción de prototipos a partir de relaciones de similaridad construidas usando la medida calidad de la similaridad, construyéndose un prototipo por cada clase de similaridad. El método propuesto incluye dos algoritmos, uno que permite realizar la aproximación a partir de los prototipos construidos, llamado Algoritmo NP-FP (*Nearest Prototype - Function Approximation*), presentado en la sección 2.1, y otro para construir los prototipos, llamado Algoritmo NP-BASIR (*Nearest Prototype - Based On Similarity Relations*), que se presenta en la sección 2.2. Adicionalmente se estudia un método de aproximación de funciones k-NN utilizando relaciones de similaridad. Por último, se evalúa la eficiencia y eficacia del método de aproximación de funciones propuesto usando conjuntos de datos internacionales y se realiza un análisis estadístico de los resultados.

2.1 Método de aproximación de funciones basado en el enfoque de los prototipos más cercanos utilizando relaciones de similaridad.

Sea $\mathbf{X = \{X_i: i=1,\ldots, n\}}$ un conjunto de datos, cada instancia $\mathbf{X_i}$ es un par $\mathbf{(x_i,y_i)}$, donde $\mathbf{x_i}$ es vector de entrada (cuyas dimensiones corresponden a **m** rasgos predictores, o sea, $\mathbf{k=1,2,\ldots, m}$) y $\mathbf{y_i \in \Re}$ (rasgo objetivo), y usualmente el vector $\mathbf{x_i \in \Re^n}$; pero, el método que se propone permite trabajar con datos mezclados.

A partir del conjunto X se construye un conjunto de $\mathbf{n_p}$ prototipos ($\mathbf{n_p}$ es significativamente inferior que **n**), o sea, $\mathbf{P= \{P_j: j=1,2,\ldots, n_p\},}$ usando el método propuesto en la sección 2.2. El aproximador de funciones usa este conjunto de prototipos para inferir el valor de salida $\mathbf{y_{new}}$ para un nuevo problema $\mathbf{X_{new}(x_{new},y_{new})}$.

Algoritmo NP-FP (*Nearest Prototype - Function Approximation*):

P1: Calcular la similaridad entre el problema $\mathbf{x_{new}}$ y cada prototipo usando la expresión (2.2).

P2: Encontrar el conjunto de prototipos más similares al problema $\mathbf{x_{new}}$, denotado por **NP ($\mathbf{x_{new}}$)**.

P3: Calcular el valor de salida, denotado por $\mathbf{y_{new}}$, usando la expresión (2.1).

La expresión (2.1) se usa para calcular el valor de salida $\mathbf{y_{new}}$ de un nuevo problema:

$$y_{new} = \frac{\sum_{y_j \in NP(x_{new})} y_j}{|NP(x_{new})|} \qquad (2.1)$$

La expresión (2.2) se usa para calcular la similaridad entre el problema y cada prototipo $\mathbf{P_j}$ (denominada similaridad global):

$$F(x_{new}, P_j) = \sum_{k=1}^{m} w_k * sim_k(x_{new}[k], P_j[k]) \qquad (2.2)$$

Los pesos $\mathbf{w_k}$, generalmente están normalizados tal que $\sum_{k=1}^{m} w_k = 1$, se usan para fortalecer o debilitar la relevancia de cada dimensión. La función $sim_k(x_{new}[k], P_j[k])$ mide el grado de similaridad entre los vectores $\mathbf{x_{new}}$ y $\mathbf{P_j}$ (denominada similaridad local). El empleo de funciones de similaridad pesadas, métricas y no métricas han sido estudiadas por diferentes autores, como (Hammer et al., 2005, Lozano et al., 2006). El empleo de medidas de similaridad local para cada rasgo permite trabajar con datos mezclados.

2.2 Método para construir prototipos utilizando relaciones de similaridad construidas a partir de la medida calidad de la similaridad.

El método de construir prototipos que se propone se puede clasificar como de remplazo, reducción, y no-determinístico.

A partir del conjunto de instancias **X**, se construye el conjunto de prototipos **P** en la forma siguiente. Se emplean las relaciones de similaridad **R1** y **R2** entre dos instancias **X1** y **X2** definidas por las expresiones (2.3) y (2.4) respectivamente:

$$\mathbf{X_1\ R1\ X_2 \text{ si y solo si } F(x_1, x_2) \geq e_1} \qquad (2.3)$$

$$\mathbf{X1\ R2\ X2 \text{ si y solo si } sim_k(y1,y2) \geq e2} \qquad (2.4)$$

Donde $\mathbf{X_1}$, $\mathbf{X_2} \in \mathbf{X}$, $\mathbf{e_1}$ y $\mathbf{e_2}$ son umbrales, y $\mathbf{sim_k}$ es la medida de similaridad local entre los valores de salida de las instancias $\mathbf{X_1}$ y $\mathbf{X_2}$.

Algoritmo NP-BASIR (*Nearest Prototype - Based On Similarity Relations*):

Este procedimiento iterativo genera un conjunto de prototipos **P**. Para cada instancia en **X** no considerada antes, se construye su clase de similaridad usando las relaciones de similaridad definidas en (2.3) y (2.4), y se genera un prototipo a partir de las instancias en la clase de similaridad. Para controlar cuales instancias han sido consideradas por el procedimiento, se usa una estructura de datos llamada **Used []**, en la cual **Used [i]** tiene un valor de 1 si la instancia **i** ha sido usada por el algoritmo NP-BASIR, ó 0 en otro caso.

P1: Inicializar el contador de instancias:

Used[i] ← 0, Para **i** =1… n

P ← ϕ

P2: Comienza procesamiento de la instancia $\mathbf{X_i}$:

Si en **X** hay instancia no usada entonces

i ← índice de la primera instancia no usada en **X**

Sino

i ← 0

Si **i**=0 entonces

Finalizar el proceso de construcción de prototipos

Sino

Used [i] ← 1.

P3: Construir la clase de similaridad de la instancia $\mathbf{X_i}$ de acuerdo a **R1** y **R2**:

Construir $\mathbf{C_{R1R2}\ [X_i]}$, donde $\mathbf{C_{R1R2}\ [X_i]}$ denota la clase de similaridad de la instancia $\mathbf{X_i}$ de acuerdo a la relación de similaridad **R** (compuesta a partir de **R1** y **R2**):

$$\mathbf{X_1 R\ X_2} \text{ si y solo si } \mathbf{F(x_1, x_2) \geq e_1} \text{ y } \mathbf{sim_k(y_1, y_2) \geq e_2}$$

P4: Generación de un nuevo prototipo $\mathbf{P_j}$:

Para k=1,2,…, m+1

$\mathbf{P_j[k]} \leftarrow \mathbf{f\ (V_i)}$, donde $\mathbf{V_i}$ es el conjunto de valores en el rasgo **k** de las instancias en $\mathbf{C_{R1R2}\ [X_i]}$

P5: Agregar el nuevo prototipo:

$\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{P_j}$

P6: Ir a **P2**

En el paso **P4**, **f** denota un operador de agregación, por ejemplo: si los valores en $\mathbf{V_i}$ son reales se debe usar el promedio, si son discretos, la moda. El propósito es construir un prototipo o centroide para un conjunto de objetos similares. Aquí se usó un enfoque similar para construir el centroide que el empleado en otros métodos tales como (Wu and Kumar, 2008) y (Mitra, 2010).

En el desarrollo de esta investigación se utilizó el método estudiado en (Filiberto and Bello, 2010b), (Filiberto and Bello, 2010a) y (Filiberto and Bello, 2011) para construir las relaciones de similaridad **R1** y **R2**. El objetivo es encontrar las relaciones **R1**y **R2** que maximizan la medida calidad de la similaridad definida por (2.5):

$$\theta(X) = \left\{ \frac{\sum_{i=1}^{n} \varphi(X_i)}{n} \right\} \qquad (2.5)$$

Donde $\boldsymbol{\varphi(X_i)}$ se calcula para cada instancia $\mathbf{X_i}$ de **X**, e indica la cantidad de objetos que están en la intersección de los conjuntos $\mathbf{N1(X_i)}$ y $\mathbf{N2(X_i)}$, los cuales respectivamente denotan los objetos que son similares a $\mathbf{X_i}$ según los rasgos predictores $\mathbf{x_i}$ y los que son similares a $\mathbf{X_i}$ según el rasgo objetivo $\mathbf{y_i}$; para construir **N1** se utiliza la función de similaridad **F** definida por (2.2) y un umbral $\mathbf{e_1}$, mientras que para construir **N2** se emplea una medida de similaridad local y un umbral $\mathbf{e_2}$.

Usando la suma pesada definida por (2.2), y dadas las funciones de comparación para cada rasgo, el problema se reduce a encontrar el conjunto de pesos $\mathbf{W = \{w_1, w_2, ..., w_m\}}$, para construir la función de semejanza **F**.

El problema de encontrar el conjunto de pesos **W** asociados con los m rasgos predictores se resuelve mediante una búsqueda heurística. Para realizar el proceso de búsqueda se consideraron las metaheurísticas Optimización Basada en Partículas (*Particle Swarm Optimization*, PSO) (Kennedy and Eberhart, 1995), en el cual se utiliza como función heurística la medida calidad de la similaridad y UMDA (Algoritmo de Distribución Marginal Univariado), este último método pertenece a la clase de algoritmos con Estimación de Distribuciones (*EDA*, siglas en inglés) (Mühlenbein, 1998) y (Mühlenbein et al., 1999), en ellos la nueva población es generada a partir de la distribución de probabilidad estimada del conjunto seleccionado. Estas metaheurísticas

fueron elegidas considerando que han mostrado ser eficientes en los estudios realizados (Filiberto, 2012).

Ambas son metaheurísticas poblacionales y de mejora. Esto significa que se parte de una población de individuos (que representan soluciones potenciales), y mediante un proceso iterativo estos se van modificando buscando aumentar su calidad de acuerdo a la función de evaluación heurística. En este caso los individuos de la población son vectores de dimensión m y cada uno representa un conjunto de pesos. En la sección 2.4 se muestran los parámetros utilizados para implementar ambas metaheurísticas.

2.3 Método de aproximación de funciones k-NN utilizando relaciones de similaridad.

A los efectos de comparar el desempeño del método propuesto para dar solución a problemas de aproximación de funciones con el método de los vecinos más cercanos k-NN, se propone utilizar la misma función de similaridad definida por (2.2), y utilizar el conjunto de pesos que maximizan la medida calidad de la similaridad, para encontrar las instancias similares.

Sea $\mathbf{X = \{X_i: i=1,\ldots, n\}}$ un conjunto de datos, cada instancia $\mathbf{X_i}$ es un par $\mathbf{(x_i,y_i)}$, donde $\mathbf{x_i}$ es vector de entrada (cuyas dimensiones corresponden a **m** rasgos predictores discretos o continuo) y $\mathbf{y_i \in \Re}$ (rasgo objetivo) entonces: se quiere predecir $\mathbf{y_{new}}$ para un nuevo problema $\mathbf{X_{new}(x_{new},y_{new})}$.

Algoritmo KNN-BASIR (*K Nearest Neighbour - Based On Similarity Relations*):

P1: Calcular la similaridad entre el problema $\mathbf{x_{new}}$ y cada ejemplo a partir de la expresión (2.2).

P2. Seleccionar las **K** instancias más similares a $\mathbf{x_{new}}$.

P3: Calcular el valor de salida $\mathbf{y_{new}}$, usando la expresión (2.6).

La expresión (2.6) se usa para calcular el valor de salida $\mathbf{y_{new}}$ de un nuevo problema, donde $\mathbf{Y_k}$ es el valor objetivo en el ejemplo **K** más similar:

$$y_{new} = \frac{\sum_{y_k \in K} Y_k}{|K|} \qquad (2.6)$$

Al usar la expresión (2.2) para realizar la recuperación de los objetos vecinos a uno dado, se usa la regla del vecino más similar en lugar del vecino más cercano, la regla del vecino más similar es una extensión de la regla del vecino más cercano.(Ruiz, 2010). Los pesos usados en (2.2) se calculan usando el método basado en la medida calidad de la similaridad descrito antes.

2.4 Resultados experimentales. Análisis estadístico.

Resulta necesario comparar la eficiencia y eficacia del método de aproximación de funciones propuesto, para lo cual se diseñan experimentos que incluyen la comparación de este con otros aproximadores, y en particular la comparación con el KNN-BASIR descrito anteriormente.

Para este estudio se emplearon conjuntos de datos reconocidos internacionalmente, descritos en Tabla 2.1. Estos conjuntos de datos son provenientes del depósito de datos para aprendizaje automatizado disponibles en el sitio ftp de la Universidad de Irvine, California[1].

Base de casos	Cantidad de casos	Cantidad de rasgos	Dominio de los rasgos predictores	Dominio del rasgo objetivo
baskball	96	5	numeric	numeric
bodyfat	252	15	numeric	numeric
detroit	13	14	numeric	numeric
diabetes_numeric	43	3	numeric	numeric
elusage	55	3	1 nominal y 2 numeric	numeric
pollution	60	16	numeric	numeric
pwlinear	200	11	numeric	numeric
pyrim	74	28	numeric	numeric
schlvote	37	6	1 nominal y 5 numeric	numeric
sleep	51	8	numeric	numeric
veteran	137	8	4 nominal y 4 numeric	numeric
vineyard	52	4	numeric	numeric
ipi	92	20	9 nominal y 11 numeric	numeric
cpu_act	1000	22	numeric	numeric
fried	1000	11	numeric	numeric
mv	2000	11	2 nominal y 9 numeric	numeric

Tabla 2.1: Descripción de los conjuntos de datos.

En los experimentos, en la expresión (2.2) se usa la medida de similaridad local definida por (2.7), la cual puede ser usada tanto para rasgos de dominio continuo como discreto, donde $\mathbf{D_k}$ denota el dominio del rasgo **k**:

[1] http://www.ics.uci.edu/mlearn/MLRepository.html

$$\mathrm{sim}_k\left(x[k], y[k]\right) = \begin{cases} 1 - \dfrac{\left|x[k] - y[k]\right|}{\mathrm{Max}(D_k) - \mathrm{Min}(D_k)} & \text{si k es continuo} \\ 1 & \text{si k es discreto y } x[k] = y[k] \\ 0 & \text{si k es discreto y } x[k] \neq y[k] \end{cases} \qquad (2.7)$$

El desempeño del método para la aproximación de funciones propuesto en este trabajo ***NpBasirRegression*** (algoritmo NP-FP más algoritmo NP-BASIR) se compara con el método ***KnnBasirRegression*** (algoritmo KNN-BASIR). Se usan valores de **K** =1 y **K** =3. Se comparan además los efectos de los umbrales $\mathbf{e_1}$ y $\mathbf{e_2}$, los cuales determinan las instancias similares respecto a los rasgos predictores y objetivo respectivamente, utilizándose 0.85 y 0.95 como valores propuestos.

Se emplea el coeficiente de reducción Re (·) (Kim and Oommen, 2003), definido por la expresión (2.8) para evaluar la reducción de la cantidad de instancias obtenida por NP-BASIR, donde: **n** representa la cantidad de instancias y $\mathbf{n_p}$ la cantidad de prototipos generados:

$$\mathrm{Re}(.) = \frac{n - n_p}{n} * 100 \qquad (2.8)$$

En la experimentación se utilizó la metaheurística PSO y UMDA para encontrar el conjunto de pesos para los rasgos predictores.

Los parámetros utilizados en la experimentación para la metaheurística PSO fueron: cantidad de partículas 40, se recomienda que debe estar entre 10 y 40 (Parsopoulos and Vrahatis, 2002, Hu and Eberhart, 2002) y la cantidad de ciclos 100, se recomienda que el número de generaciones debe estar entre 100 y 200 (Parsopoulos and Vrahatis, 2002).

En esta experimentación se utilizaron para la metaheurística UMDA valores para los parámetros como: número de individuos 40, cantidad de generaciones o número de evaluaciones 100 y porciento de selección por truncamiento 0.5.

Se utilizaron 6 juegos de parámetros para realizar el estudio comparativo entre el método propuesto ***NpBasirRegression*** y ***KnnBasirRegression,*** ellos son:

1) $\mathbf{e_1}$=$\mathbf{e_2}$=0.85, **K** =1;
2) $\mathbf{e_1}$=$\mathbf{e_2}$=0.95, **K** =1;
3) $\mathbf{e_1}$=$\mathbf{e_2}$=0.85, **K**=1 para ***NpBasirRegression*** y **K**=3 para ***KnnBasirRegression***;

4) $\mathbf{e_1}$=$\mathbf{e_2}$=0.95, **K**=1 para ***NpBasirRegression*** y **K**=3 para ***KnnBasirRegression***;
5) $\mathbf{e_1}$=$\mathbf{e_2}$=0.85, **K** =3;
6) $\mathbf{e_1}$=$\mathbf{e_2}$=0.95, **K** =3.

Para realizar la experimentación se utiliza validación cruzada (Demsar, 2006) con 10 particiones. Para medir el error cometido al aproximar se utilizó el promedio del error medio absoluto, o sea, el promedio de las diferencias entre el valor deseado y el producido por el método (PMD) .(Weber and Foix, 2007)

En el procesamiento estadístico de todos los resultados experimentales se utilizó la herramienta estadística SPSS (Pardo and Ruiz, 2002, Grau, 1994). Para el análisis estadístico de todos los resultados se realizaron pruebas no paramétricas debido a que la significación resultante luego de aplicar el test de Kolmogorov-Smirnov para medir el posible ajuste de cada una de las muestras a la distribución normal fue menor que 0.05 lo que significa que se debe rechazar la hipótesis de normalidad. Precisamente el test de Friedman y Wilcoxon son pruebas no paramétricas que permite estudiar la existencia o no de diferencias significativas entre las muestras.

Experimento #1: Comparar ***NpBasirRegression*** con ***KnnBasirRegression*** utilizando las 16 bases de casos descritas anteriormente y los 6 juegos de parámetros y como forma para obtener los pesos la metaheurística PSO. En la tabla 2.2 y 2.4 se muestran los resultados alcanzados del error medio absoluto cometido al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression*** respectivamente. En la tabla 2.3 aparecen reflejados los resultados de la cantidad máxima y mínima de prototipos construidos y sus respectivos coeficientes de reducción. Obsérvese que para bases de casos grandes (de tamaño mayor que 1000) sólo se utilizaron los juegos de parámetros (1) (3) (5) ya que con $\mathbf{e_1}$=0.85 y $\mathbf{e_2}$=0.85 hay una mayor reducción del conjunto de objetos, ver estos resultados de la reducción en la tabla 2.3.

Base de casos	**Resultados con cada juego de parámetros**					
	1	2	3	4	5	6
baskball	0.08	0.09	0.08	0.09	0.08	0.07
bodyfat	3.48	1.77	3.15	2.89	3.05	1.83
detroit	30.42	28.35	27.19	31.31	36.45	31.66
diabetes_numeric	0.70	0.68	0.55	0.68	0.49	0.48
elusage	9.91	11.73	9.26	11.73	8.62	9.20
pollution	42.92	42.74	40.87	39.66	36.41	33.90

pwlinear	2.63	2.48	2.34	1.95	2.57	1.87
pyrim	0.07	0.06	0.06	0.06	0.06	0.06
schlvote	92.04	90.56	92.04	92.04	91.91	89.28
sleep	2.58	2.20	2.32	2.20	2.87	2.56
veteran	133.91	111.27	120.55	112.35	122.79	111.43
vineyard	2.22	1.98	2.08	1.95	2.04	1.80
ipi	7.56	7.14	7.26	5.19	7.22	5.84
cpu_act	4.11	-	4.50	-	6.69	-
fried	3.09	-	3.09	-	3.19	-
mv	3.19	-	3.50	-	3.36	-

Tabla 2.2 Resultados al aproximar con ***NpBasirRegression***.

Base de casos (cantidad de casos)	**Cantidad de prototipos Máxima/Mínima**						**Coeficiente de reducción**					
	1	2	3	4	5	6	1	2	3	4	5	6
baskball (96)	22/16	74/66	23/17	69/64	21/16	73/69	77.1/ 83.3	23.0/ 31.3	76.0/ 82.3	28.1/ 33.3	78.1/ 83.3	24.0/ 28.1
bodyfat (252)	18/11	137/ 120	17/12	120/ 114	16/12	120/ 115	92.9/ 95.6	45.6/ 52.4	93.3/ 95.2	52.4/ 54.8	93.7/ 95.2	52.4/ 54.4
detroit (13)	10/7	12/11	9/7	12/11	8/7	12/11	23.1/ 46.2	7.69/ 15.4	30.8/ 46.2	7.69/ 15.4	38.5/ 46.2	7.69/ 15.4
diabetes_numeric (43)	14/11	35/31	14/11	35/31	14/11	35/31	67.4/ 74.4	18.6/ 27.9	67.4/ 74.4	18.6/ 27.9	67.4/ 74.4	18.6/ 27.9
elusage (55)	11/8	38/34	13/11	38/34	11/8	35/31	80.0/ 85.5	30.9/ 38.2	76.4/ 80.0	30.9/ 38.2	80.0/ 85.5	36.4/ 43.6
pollution (60)	18/15	54/52	19/15	54/52	18/16	54/52	70.0/ 75.0	10.0/ 13.3	68.3/ 75.0	10.0/ 13.3	70.0/ 73.3	10/ 13.3
pwlinear (200)	70/63	164/ 160	79/71	170/ 167	80/73	162/ 158	65.0/ 68.5	18.0/ 20.0	60.5/ 64.5	15.0/ 16.5	60.0/ 63.5	19.0/ 21.0
pyrim (74)	17/12	45/39	15/11	45/42	16/11	51/43	77.2/ 83.8	39.2/ 47.3	79.7/ 85.1	39.2/ 43.2	78.4/ 85.1	31.1/ 41.9
schlvote (37)	8/6	27/20	9/6	20/17	9/6	27/22	78.4/ 83.8	27.0/ 45.9	75.7/ 83.8	45.9/ 54.1	75.7/ 83.8	27.0/ 40.5
sleep (51)	14/12	37/33	15/11	37/33	13/12	37/33	72.5/ 76.5	27.5/ 35.3	70.6/ 78.4	27.5/ 35.3	74.5/ 76.5	27.5/ 35.3
veteran (137)	19/14	66/57	19/13	60/55	24/22	58/51	86.1/ 89.8	51.8/ 58.4	86.1/ 90.5	56.2/ 59.9	82.5/ 83.9	57.7/ 68.8
vineyard (52)	14/11	28/25	14/11	28/25	14/11	42/33	73.1/ 78.8	46.2/ 51.9	73.1/ 78.8	46.2/ 51.9	73.1/ 78.8	19.2/ 36.5
ipi (92)	21/18	45/40	19/17	41/35	19/16	42/38	77.2/ 80.4	50.1/ 56.5	79.3/ 81.5	55.4/ 62.0	79.3/ 82.6	54.3/ 58.7
cpu_act (1000)	18/14	-	17/13	-	16/12	-	98.2/ 98.6	-	98.3/ 98.7	-	98.4/ 98.8	-
fried (1000)	210/ 193	-	210/ 193	-	220/ 199	-	79.0/ 80.7	-	79.0/ 80.7	-	78.0/ 80.1	-
mv (2000)	54/49	-	71/67	-	61/56	-	97.3/ 97.6	-	96.5/ 96.7	-	97.0/ 97.2	-

Tabla 2.3 Resultados de la reducción del conjunto de objetos al aproximar con ***NpBasirRegression***.

Base de casos	Resultados con cada juego de parámetros					
	1	2	3	4	5	6
baskball	0.10	0.09	0.08	0.08	0.08	0.07
bodyfat	2.09	1.63	1.55	2.38	1.54	1.80
detroit	34.55	28.35	34.80	37.99	39.52	31.66
diabetes_numeric	0.77	0.69	0.52	0.49	0.56	0.48
elusage	10.38	17.13	8.43	18.02	8.63	8.63
pollution	45.71	43.05	38.49	37.57	41.72	34.92
pwlinear	2.69	2.51	2.05	1.75	2.81	1.89
pyrim	0.07	0.07	0.06	0.06	0.06	0.06
schlvote	107.73	50.29	72.09	65.10	78.89	50.02
sleep	2.70	2.30	2.04	2.56	2.46	2.56
veteran	92.66	103.29	90.08	104.60	94.41	103.14
vineyard	2.44	2.25	2.17	1.70	2.09	1.83
ipi	5.88	5.66	5.51	5.40	5.80	4.87
cpu_act	2.72	-	2.33	-	2.35	-
fried	3.13	-	3.91	-	3.19	-
mv	1.73	-	1.39	-	1.24	-

Tabla 2.4 Resultados al aproximar con ***KnnBasirRegression***.

Análisis estadístico de los resultados del experimento #1: La aplicación del test de Friedman aportó que existen diferencias significativas ya que la significación es menor que 0.05. A continuación se muestran estos resultados en la Figura 2.1:

Estadísticos de contraste[a]

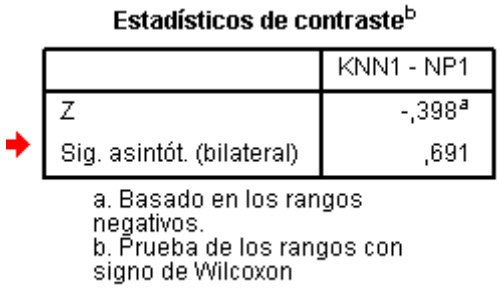
N	13
Chi-cuadrado	41,721
gl	11
Sig. asintót.	,000

a. Prueba de Friedman

Figura 2.1 Resultados del test de Friedman.

Los resultados anteriores indicaron la necesidad de realizar el test de Wilcoxon para muestras pareadas. Se eligieron para estas pruebas los siguientes pares:

1. ***NpBasirRegression*** versus ***KnnBasirRegression*** utilizando el juego de parámetros 1: la aplicación del test de Wilcoxon aportó que no existen diferencias significativas entre las muestras ya que su valor de significación fue de 0.691, o sea mucho mayor que 0.05. A continuación se muestran estos resultados en la Figura 2.2:



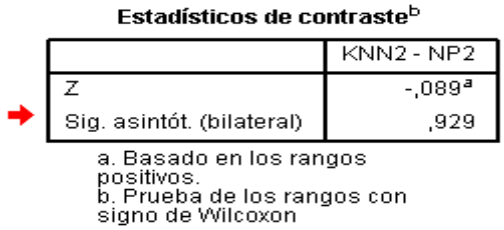
Estadísticos de contraste[b]

	KNN1 - NP1
Z	-,398[a]
Sig. asintót. (bilateral)	,691

a. Basado en los rangos negativos.
b. Prueba de los rangos con signo de Wilcoxon

Figura 2.2 Resultados del test de Wilcoxon.

2. ***NpBasirRegression*** versus ***KnnBasirRegression*** utilizando el juego de parámetros 2: la aplicación del test de Wilcoxon aportó que no existen diferencias significativas entre las muestras ya que su valor de significación fue de 0.929, o sea mucho mayor que 0.05. A continuación se muestran estos resultados en la Figura 2.3:



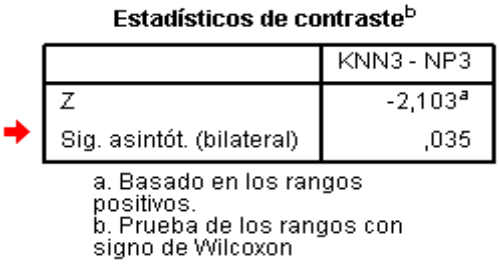
Estadísticos de contraste[b]

	KNN2 - NP2
Z	-,089[a]
Sig. asintót. (bilateral)	,929

a. Basado en los rangos positivos.
b. Prueba de los rangos con signo de Wilcoxon

Figura 2.3 Resultados del test de Wilcoxon.

3. ***NpBasirRegression*** versus ***KnnBasirRegression*** utilizando el juego de parámetros 3: la aplicación del test de Wilcoxon aportó que si existen diferencias significativas entre las muestras ya que su valor de significación fue de 0.035, o sea menor 0.05 y reflejó que para este juego de parámetros ***KnnBasirRegression*** se comporta mucho mejor que ***NpBasirRegression***, esto es un resultado lógico debido a que con este juego de parámetros se favorece a ***KnnBasirRegression.*** A continuación se muestran estos resultados en la Figura 2.4:



Estadísticos de contraste[b]

	KNN3 - NP3
Z	-2,103[a]
Sig. asintót. (bilateral)	,035

a. Basado en los rangos positivos.
b. Prueba de los rangos con signo de Wilcoxon

Figura 2.4 Resultados del test de Wilcoxon.

4. ***NpBasirRegression*** versus ***KnnBasirRegression*** utilizando el juego de parámetros 4: la aplicación del test de Wilcoxon aportó que no existen diferencias significativas entre estas muestras ya que su valor de significación fue de 0.433, o sea mayor que 0.05. Además se refleja que aunque para este juego de parámetros también se está favoreciendo a ***KnnBasirRegression***, en este caso no existen diferencias entre ambos métodos. A continuación se muestran estos resultados en la Figura 2.5:

Estadísticos de contraste[b]

	KNN4 - NP4
Z	-,784[a]
Sig. asintót. (bilateral)	,433

a. Basado en los rangos positivos.
b. Prueba de los rangos con signo de Wilcoxon

Figura 2.5 Resultados del test de Wilcoxon.

5. ***NpBasirRegression*** versus ***KnnBasirRegression*** utilizando el juego de parámetros 5: la aplicación del test de Wilcoxon aportó que no existen diferencias significativas entre estas muestras ya que su valor de significación fue de 0.388, o sea mayor que 0.05. A continuación se muestran estos resultados en la Figura 2.6:

Estadísticos de contraste[b]

	KNN5 - NP5
Z	-,863[a]
Sig. asintót. (bilateral)	,388

a. Basado en los rangos positivos.
b. Prueba de los rangos con signo de Wilcoxon

Figura 2.6 Resultados del test de Wilcoxon.

6. ***NpBasirRegression*** versus ***KnnBasirRegression*** utilizando el juego de parámetros 6: la aplicación del test de Wilcoxon aportó que no existen diferencias significativas entre estas muestras ya que su valor de significación

fue de 0.233, o sea mayor que 0.05. A continuación se muestran estos resultados en la Figura 2.7:

Estadísticos de contraste[b]

	KNN6 - NP6
Z	-1,192[a]
Sig. asintót. (bilateral)	,233

a. Basado en los rangos positivos.
b. Prueba de los rangos con signo de Wilcoxon

Figura 2.7 Resultados del test de Wilcoxon.

Experimento #2: De forma similar al experimento 1 se comparan ***NpBasirRegression*** con ***KnnBasirRegression***, pero ahora se usa para obtener los pesos la metaheurística UMDA. En la tabla 2.5 y 2.7 se muestran los resultados alcanzados del error medio absoluto cometido al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression*** respectivamente. En la tabla 2.6 aparecen reflejados los resultados de la cantidad máxima y mínima de prototipos construidos y sus respectivos coeficientes de reducción. Obsérvese también que para bases de casos grandes (de tamaño mayor que 1000) sólo se utilizaron los juegos de parámetros (1) (3) (5) ya que con $\mathbf{e_1}$=0.85 y $\mathbf{e_2}$=0.85 hay una mayor reducción del conjunto de objetos, ver estos resultados de la reducción en la tabla 2.6.

Base de casos	**Resultados con cada juego de parámetros**					
	1	2	3	4	5	6
baskball	0.09	0.09	0.08	0.08	0.08	0.70
detroit	28.29	30.89	25.60	29.73	37.82	33.80
diabetes_numeric	0.60	0.68	0.53	0.60	0.61	0.52
elusage	10.01	11.57	9.37	11.57	7.92	10.56
pollution	43.40	42.88	42.68	41.35	41.59	38.07
pwlinear	2.31	2.33	2.07	2.48	1.83	2.16
pyrim	0.08	0.06	0.07	0.06	0.07	0.06
schlvote	76.13	76.13	90.69	76.13	91.48	95.51
sleep	2.65	2.18	2.51	1.95	2.65	2.64
veteran	100.4	101.56	94.19	105.95	99.91	94.49
vineyard	2.39	2.23	2.11	2.20	2.17	1.94
ipi	8.38	5.29	7.61	5.16	8.08	5.64
cpu_act	3.94	-	4.11	-	6.55	-
fried	3.27	-	2.45	-	2.63	-
mv	7.71	-	5.12	-	4.70	-

Tabla 2.5 Resultados al aproximar con ***NpBasirRegression***.

Base de casos (cantidad de casos)	Cantidad de prototipos Máxima/Mínima						Coeficiente de reducción					
	1	2	3	4	5	6	1	2	3	4	5	6
baskball (96)	28/23	87/86	24/19	87/85	22/16	87/85	70.8/ 76.0	9.38/ 10.4	75.0/ 80.2	9.38/ 11.5	77.1/ 83.3	9.38/ 11.5
detroit (13)	9/7	12/11	9/7	12/11	9/7	12/11	30.8/ 46.2	7.69/ 15.4	30.8/ 46.2	7.69/ 15.4	30.8/ 46.2	7.69/ 15.4
diabetes_numeric (43)	18/14	34/31	14/11	39/38	13/12	39/37	58.1/ 67.4	20.9/ 27.9	67.4/ 74.4	9.30/ 11.6	69.8/ 72.1	9.30/ 14.0
elusage (55)	11/8	48/47	18/16	49/48	19/16	49/47	80.0/ 85.5	12.7/ 14.5	67.3/ 70.9	10.9/ 12.7	65.5/ 70.9	10.9/ 14.5
pollution (60)	12/8	49/47	9/7	54/54	11/9	51/48	80.0/ 86.7	18.3/ 21.7	85.0/ 88.3	10.0/ 10.0	81.7/ 85.0	15.0/ 20.0
pwlinear (200)	145/ 142	180/ 180	121/ 113	180/ 180	131/ 121	180/ 179	27.5/ 29.0	10.0/ 10.0	39.5/ 43.5	10.0/ 10.0	34.5/ 39.5	10.0/ 10.5
pyrim (74)	11/6	34/28	11/6	36/34	12/9	34/31	85.1/ 91.9	54.1/ 62.2	85.1/ 91.9	51.6/ 54.1	83.8/ 87.8	54.1/ 58.1
schlvote (37)	34/33	34/33	25/21	34/33	32/29	34/33	8.11/ 10.8	8.11/ 10.8	32.4/ 43.2	8.11/ 10.8	13.5/ 21.6	8.11/ 10.8
sleep (51)	21/19	41/38	21/18	42/39	22/21	43/40	58.8/ 62.7	19.6/ 25.5	58.8/ 64.7	17.6/ 23.5	56.9/ 58.8	15.7/ 21.6
veteran (137)	107/ 101	124/ 123	117/ 110	124/ 123	115/ 108	124/ 123	21.9/ 26.3	9.49/ 10.2	14.6/ 19.7	9.49/ 10.2	16.1/ 21.2	9.49/ 10.2
vineyard (52)	15/13	47/46	14/11	38/33	16/12	47/45	71.2/ 75.0	9.62/ 11.5	73.1/ 78.8	26.9/ 36.5	69.2/ 76.9	9.62/
ipi (92)	18/15	48/44	17/14	48/45	18/16	45/43	80.4/ 83.7	47.8/ 52.2	81.5/ 84.8	47.8/ 51.1	80.4/ 82.6	51.1/ 53.3
cpu_act (1000)	14/10	-	15/12	-	15/11	-	98.6/ 99.0	-	98.5/ 98.8	-	98.5/ 98.9	-
fried (1000)	108/ 97	-	62/55	-	288/ 273	-	89.2/ 90.3	-	93.5/ 94.5	-	71.2/ 72.7	-
mv (2000)	14/12	-	31/27	-	18/16	-	99.3/ 99.4	-	98.5/ 98.7	-	99.1/ 99.2	-

Tabla 2.6 Resultados de la reducción del conjunto de objetos al aproximar con ***NpBasirRegression***.

Base de casos	Resultados con cada juego de parámetros					
	1	2	3	4	5	6
baskball	0.10	0.09	0.08	0.08	0.09	0.70
detroit	28.35	30.89	32.15	37.60	34.80	33.80
diabetes_numeric	0.64	0.69	0.54	0.52	0.67	0.52
elusage	10.38	16.04	8.11	17.11	8.19	18.02
pollution	49.62	44.22	42.54	38.77	43.69	38.12
pwlinear	2.36	2.33	1.66	2.23	1.90	2.16
pyrim	0.08	0.06	0.06	0.06	0.06	0.06
schlvote	53.52	72.98	53.10	63.27	53.43	70.86
sleep	2.65	2.18	2.48	2.72	2.63	2.81
veteran	102.58	101.56	101.57	104.40	100.33	94.49
vineyard	2.41	2.23	2.18	1.92	2.07	1.95
ipi	5.25	5.25	4.88	5.80	4.38	5.51
cpu_act	2.64	-	2.33	-	2.26	-
fried	3.41	-	2.22	-	2.55	-
mv	5.12	-	1.46	-	1.84	-

Tabla 2.7 Resultados al aproximar con ***KnnBasirRegression***.

Análisis estadístico de los resultados del experimento #2: La aplicación del test de Friedman aportó que no existen diferencias significativas ya que la significación es mayor que 0.05. A continuación se muestran estos resultados en la Figura 2.8:

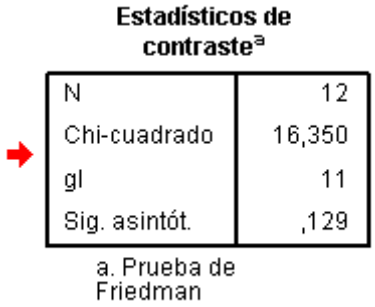


Figura 2.8 Resultados del test de Friedman.

Experimento # 3: Comparar ***NpBasirRegression*** con otros aproximadores de funciones que aparecen en WEKA. Se utilizaron las 16 bases de casos anteriormente descritas y los resultados alcanzados del error medio absoluto cometido al aproximar con ***NpBasirRegression*** dados en: la Tabla 2.2 columna 1 y 5 (es decir con valores para el parámetro **k** de 1 y 3 respectivamente y con $\mathbf{e_1}$ y $\mathbf{e_2}$ igual 0.85 utilizando la metaheurística PSO) y la Tabla 2.5 columna 1 y 5 (es decir con valores para el parámetro **k** de 1 y 3 respectivamente y con $\mathbf{e_1}$ y $\mathbf{e_2}$ igual 0.85 utilizando la metaheurística UMDA). Sólo se utilizaron para la comparación estos resultados debido a que con $\mathbf{e_1}$=0.85 y $\mathbf{e_2}$=0.85 hay una mayor reducción del conjunto de objetos. En la tabla 2.8 se muestran los resultados alcanzados al aproximar con *Multilayer Perceptron, Linear Regression y Regression Tree (REPTree).*

Base de casos	**Resultados**		
	Multilayer Perceptron	Linear Regression	REPTree
baskball	0.08	0.07	0.08
bodyfat	0.60	0.52	0.78
detroit	33.06	44.92	55.91
diabetes_numeric	0.55	0.50	0.57
elusage	10.76	8.71	10.92
pollution	47.39	32.44	50.16
pwlinear	1.68	1.80	1.65
pyrim	0.08	0.07	0.10
schlvote	111.53	78.38	77.14
sleep	3.16	2.99	2.76
veteran	164.24	92.31	104.49
vineyard	2.09	2.30	2.45

ipi	9.57	10.80	5.41
cpu_act	2.46	6.05	2.75
fried	1.20	2.08	2.30
mv	0.23	3.40	0.52

Tabla 2.8 Resultados al aproximar con *Multilayer Perceptron, Linear Regression* y *REPTree*.

Análisis estadístico de los resultados del experimento #3: La aplicación del test de Friedman aportó que no existen diferencias significativas ya que la significación es mucho mayor que 0.05. A continuación se muestran estos resultados en la Figura 2.9:

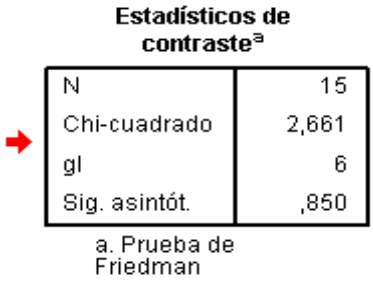


Figura 2.9 Resultados del test de Friedman.

2.5 Conclusiones parciales del capítulo.

Se mostró a través del estudio experimental con bases de datos internacionales el buen desempeño del método ***NpBasirRegression***, lo cual se resume en:

1. El método de construcción de prototipos basado en relaciones de similaridad que maximizan la medida calidad de la similaridad permite obtener conjuntos de prototipos con una disminución significativa de la cantidad de instancias.
2. No existen diferencias significativas entre la eficacia en la aproximación de funciones lograda si se utilizan los prototipos respecto a la que se alcanza si se usan todas las instancias cuando se aplica ***KnnBasirRegression***.
3. No existen diferencias significativas entre la eficacia de la aproximación de funciones obtenida si se usa el conjunto de prototipos respecto a la que se alcanza con otros aproximadores como *Multilayer Perceptron, Linear Regression y Regression Tree.*

CAPÍTULO 3 DESARROLLO DEL MÉTODO *NpBasirRegression* Y SU INCORPORACIÓN A LA PLATAFORMA WEKA. SOLUCIÓN A DOS PROBLEMAS REALES DE LA INGENIERÍA CIVIL.

En este capítulo se describen las clases fundamentales para el diseño e implementación del método de aproximación de funciones ***NpBasirRegression***, así como su incorporación a WEKA. Se realiza una breve descripción en los epígrafes 3.1.1 y 3.1.2 de las dos metaheurísticas utilizadas: PSO y UMDA. Finalmente se validan los resultados teóricos de la investigación en el área de la Ingeniería Civil, específicamente se da solución al pronóstico de la capacidad resistente en muros de mampostería y de la capacidad resistente al deslizamiento longitudinal en las losas compuestas con lámina colaborante.

3.1 Diseño e implementación del método de aproximación de funciones *NpBasirRegression*.

Para el diseño e implementación del método de aproximación de funciones ***NpBasirRegression*** descrito en el capítulo 2 basado en el enfoque de los prototipos más cercanos utilizando relaciones de similaridad se implementaron las siguientes clases:

Prototype: Clase representativa del objeto prototipo, donde un prototipo se representa por un arreglo de dimensión igual a la cantidad de atributos de la base de casos y cuyos valores pueden ser nominales o numéricos.

NpBasir: Clase para la construcción de conjunto de prototipos, esta clase tiene dos métodos principales: ***buildsimilarityclass(x)*** que se ocupa de construir la clase de similaridad del objeto **x** si no ha sido usado y ***constructorsetprototype()*** método para la construcción de prototipos basado en relaciones de similaridad descrito en el capítulo 2 epígrafe 2.2.

NpFp: Clase que tiene como método principal ***approximationcase(case$_{new}$)*** descrito en el capítulo 2 epígrafe 2.1 este método es el encargado de predecir el valor numérico de un nuevo caso a partir de un conjunto de prototipos.

El diagrama de clases fundamentales para implementar ***NpBasirRegression*** aparece en la Figura 3.1.

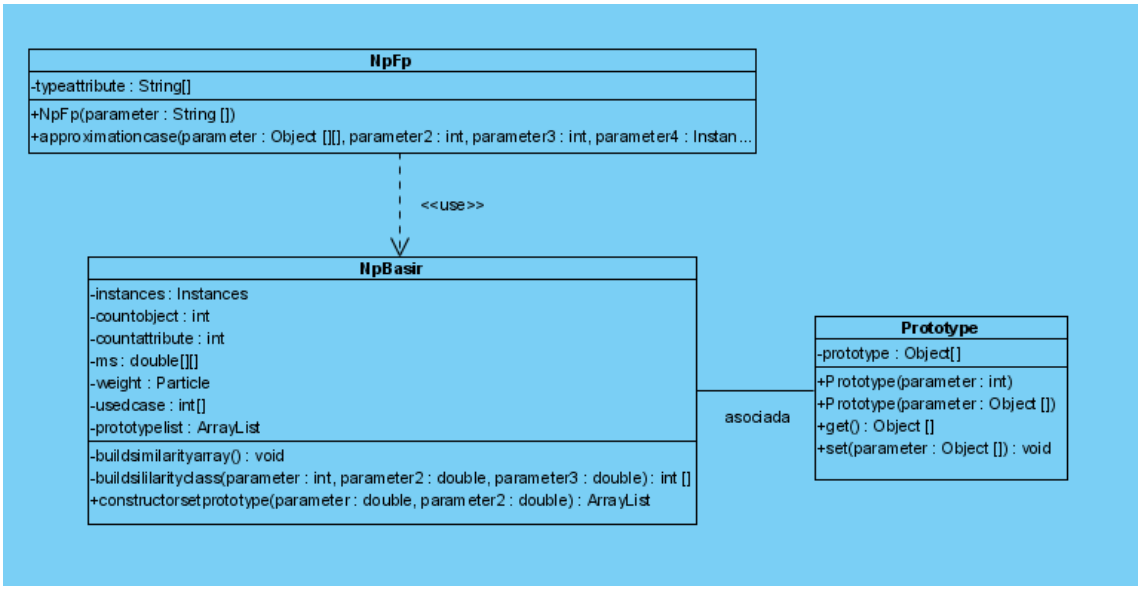


Figura 3.1 Diagrama de clases fundamentales para implementar ***NpBasirRegression***.

3.1.1 Implementación de un método de ponderación de rasgos basado en la metaheurística PSO.

Optimización basada en nubes de partículas (*Particle Swarm Optimization*, PSO) (Kennedy and Eberhart, 1995) es un conjunto de técnicas inspiradas en el comportamiento de las bandadas de aves o bancos de peces. Cada partícula tiene una medida de calidad, así como una posición y velocidad en el espacio de la búsqueda, donde la posición determina el contenido de la posible solución. Cada partícula conoce la posición de sus vecinos, interactúa con ellos, "aprende" y ajusta su posición y velocidad parcialmente atraída a su mejor posición hasta el momento y parcialmente atraída a la mejor posición de la vecindad. Las soluciones o partículas son guiadas por la de mejor solución encontrada hasta el momento convirtiéndose en el líder. Dicho de otra forma, la bandada vuela por el espacio de búsqueda de un problema n-dimensional y evalúa las posiciones que alcanza cada partícula, según la función a optimizar, llevando un registro de los mejores puntos que se alcanzan.

Los componentes básicos del método son:

$\mathbf{X_i}$: Partícula (vector unidimensional).

$\mathbf{\{X_1, X_2,\ldots\}}$: Bandada (conjunto de partículas).

$\mathbf{\{V_1, V_2,\ldots\}}$: Velocidades (vector m dimensional asociado a cada partícula que indica su movimiento).

$\mathbf{\{Xpbest_1, Xpbest_2,\ldots\}}$: Mejores puntos del espacio localizados por cada partícula.

Xgbest: Mejor punto localizado por la bandada.

Para el diseño e implementación del método de ponderación de rasgos basado en la metaheurística PSO se implementaron las siguientes clases:

Particle: Clase representativa del objeto partícula donde una partícula es un arreglo unidimensional de tamaño igual a la cantidad de rasgos predictores de la base de casos. ***Pso***: Clase principal, tiene dos métodos fundamentales ***evaluateparticle(particle)*** y ***algorithmPso()***, el primero de los cuales se ocupa de evaluar la partícula y devolver su calidad y el segundo es el método principal de la clase ***Pso*** que implementa esta metaheurística y cuyo pseudocódigo se describe a continuación:

Algoritmo PSO:

Paso 1: Inicializar aleatoriamente

Band /*arreglo bandada*/

Veloc /*arreglo velocidades*/

Xpbest← los mejores valores entre **Band** y **Veloc**

Xgbest ←con el mejor valor de **Xpbest**.

Paso 2: Repetir hasta una cantidad de ciclos dados

Para cada elemento **i** en **Band**

Actualizar con **Regla de actualización (1) $\mathbf{Veloc_i}$**

Actualizar con **Regla de actualización (2) $\mathbf{Band_i}$**

Fin de ciclo

Para cada elemento **i** en **Band**

Evaluar $\mathbf{Band_i}$ y comparar con $\mathbf{Xpbest_i}$

Actualizar $\mathbf{Xpbest_i}$ con el mejor valor

Actualizar **Xgbest** con el mejor valor en **Xpbest**

Paso 3: Devolver **Xgbest**.

Regla de actualización (1):

$newVeloc_i = cr*(w\ (k)*Veloc_i+c1*rand\ ()*(Xpbest_i-Band_i)+c2*rand\ ()*(Xgbest-Band_i))$

Regla de actualización (2):

$newBand_i = Band_i+newVeloc_i$

Donde:

El parámetro **w** es peso de la inercia, **c1** la razón de aprendizaje cognitivo, **c2** la razón de aprendizaje social y **cr** coeficiente de constricción, el cual es una mejora al algoritmo PSO con el objetivo de que converja al óptimo de la función con más frecuencia.

c1*$rand_1$ ()+c2*$rand_2$ ()≤4, donde $rand_1$ () y $rand_2$ () son números aleatorios entre 0 y 1 y **c1**=**c2**=2.05 (Parsopoulos and Vrahatis, 2002, Filiberto, 2012, Bratton and Kennedy, 2007).

w (k)= Wmax- (Wmax-Wmin)/Ncmax*k, donde **Wmax**=1.4, **Wmin**=0.4 y **k** es el ciclo (Parsopoulos and Vrahatis, 2002, Filiberto, 2012, Parsopoulos and Plagianakos, 2001).

La expresión (3.1) se usa para calcular el coeficiente de constricción:

$$cr = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad \varphi = c1 + c2 \qquad (3.1)$$

El diagrama de clases fundamentales para la implementación de la metaheurística PSO aparece en el Anexo 1.

3.1.2 UMDA como otra metaheurística para ponderar rasgos.

Con el objetivo de utilizar UMDA (Algoritmo de Distribución Marginal Univariado) como otra metaheurística para ponderar rasgos, se utiliza la biblioteca EDALib.jar desarrollada por (Caballero, 2007, Madera, 2009) en la Universidad de Camagüey.

UMDA es uno de los miembros de la familia de Algoritmos de Estimación de Distribución de probabilidad conocidos como EDA. Los EDAs son algoritmos evolutivos basados en poblaciones al igual que los algoritmos genéticos (AG) (Herrera and Lozano, 2003, García-Martínez, 2008), en los que la transición de una generación a otra ha sido modificada. A diferencia de los AG en los EDAs no hay operadores de cruce ni de mutación. En su lugar, la nueva población de individuos se muestrea a partir de una distribución de probabilidad, que se estima a partir de una base de datos formada por individuos de generaciones anteriores. Mientras que en los AG las interrelaciones entre las distintas variables involucradas en la representación de los individuos no son, en general, consideradas, en los EDAs las interrelaciones se expresan de forma implícita en la distribución de probabilidad conjunta asociada a los individuos seleccionados en

cada iteración. (Larrañaga and Puerta, 2003, Larrañaga and Lozano, 2001) (Caballero, 2007) (Madera, 2009)

A continuación se muestra en la Figura 3.2 un pseudocódigo del Algoritmo de Estimación de Distribuciones (EDA).

Algoritmo 1 Algoritmo de estimación de distribuciones (EDA).

```
1   D_0 ← Generar la población inicial (m individuos)
2   Evaluar la población D_0
3   l = 1
4   Repeat until condición de parada
5     D_{l-1}^{Se} ← Seleccionar s ≤ m individuos de D_{l-1}
6     Estimar un modelo probabilístico M a partir de
      D_{l-1}^{Se}
7     D_{l-1}^{m} ← Muestrear m individuos a partir de M
8     Evaluar D_{l-1}^{m}
9     D_l ← Seleccionar m individuos de D_{l-1} ∪ D_{l-1}^{m}
10    l = l + 1
11  end
```

Figura 3.2 Algoritmo de estimación de distribuciones (EDA).

Al igual que en la mayoría de los algoritmos generacionales, se parte de una población inicial con **m** individuos, generada (en la mayoría de los casos) aleatoriamente. En el segundo paso, un número **s** menor que **m** de individuos se seleccionan (normalmente aquellos con los mejores valores en cuanto a la función de evaluación) como base de datos para la estimación de modelo. A continuación se induce el modelo probabilístico n-dimensional que mejor refleja las interdependencias entre las **s** variables. A partir del modelo inducido se genera una población auxiliar mediante muestreo. Por último, la nueva población $\mathbf{D_l}$ se obtiene a partir de la población anterior $\mathbf{D_{l-1}}$ y de la población auxiliar. Normalmente, esta selección se realiza de forma elitista. (Larrañaga and Puerta, 2003, Larrañaga and Lozano, 2001, Madera, 2009)

Existen multitud de EDAs en función del modelo usado para estimar en cada generación la distribución de probabilidad conjunta a partir de los individuos seleccionados, entre ellos están los modelos univariados como UMDA. En UMDA, en cada generación la distribución de probabilidad conjunta, $\mathbf{p_l(x)}$, que sirve para estimar el comportamiento de los individuos seleccionados, se factoriza como producto de las distribuciones univariadas independientes (Larrañaga and Puerta, 2003, Larrañaga and Lozano, 2001, Caballero, 2007), esto se muestra en la expresión 3.2:

$$p_l(x) = p\left(x \middle| D_{l-1}^{Se}\right) = \prod_{i=1}^{n} p_l(x_i) \qquad (3.2)$$

Donde **x** es un individuo, **n** es la cantidad de variables o atributos del individuo, **l** es el número de la iteración o generación del proceso y $\mathbf{D_{l-1}^{Se}}$ son los individuos seleccionados en la generación **l-1**.

Y cada distribución univariada se estima a partir de las frecuencias marginales como se muestra en la expresión 3.3:

$$p_l(x) = \frac{\sum_{j=1}^{N} \delta_j \left(X_i = x_i | D_{l-1}^{Se}\right)}{N} \tag{3.3}$$

$$\delta_j\left(X_i = x_i \middle| D_{l-1}^{Se}\right) = \begin{cases} 1 \text{ si en el } j-\text{ésimo caso de } D_{l-1}^{Se}, X_i = x_i \\ 0 \text{ en otro caso} \end{cases}$$

A continuación se muestra en la Figura 3.3 el pseudocódigo del algoritmo UMDA:

$D_0 \leftarrow$ Generar M individuos (la poblacion inicial) al azar

Repeat for $l = 1, 2, \ldots$ hasta que se verifique el criterio de parada

$D_{l-1}^{Se} \leftarrow$ Seleccionar $N \leq M$ individuos de D_{l-1} de acorde al metodo de seleccion

$p_l(x) = p(x|D_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i) = \prod_{i=1}^{n} \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i | D_{l-1}^{Se})}{N} \leftarrow$ Estimar la distribucion de probabilidad conjunta

$D_l \leftarrow$ Muestrar M individuos (la nueva poblacion) de $p_l(x)$

Figura 3.3 Algoritmo UMDA

La diferencia principal entre las variantes de UMDA es respecto al dominio de definición discreto o continuo de la función objetivo. Para el dominio continuo que es el de interés en este trabajo los parámetros que se estiman son la media y la varianza de cada variable de una distribución normal univariada, esto a partir de la población seleccionada.(Caballero, 2007)

3.2 Incorporación de *NpBasirRegression* a la plataforma WEKA.

El proyecto WEKA[2] es un entorno de software para la experimentación y prueba de diferentes algoritmos de Aprendizaje Automatizado que se desarrolló en la Universidad de Waikato, Nueva Zelanda (Witten and Frank, 2005). Esta plataforma libre implementa, en el lenguaje de programación Java, una gran variedad de métodos

[2] Herramienta de código abierto escrita en Java. Disponible bajo licencia pública GNU en http://www.cs.waikato.ac.nz/ml/weka/

estadísticos y de Inteligencia Artificial, así como técnicas para comparar sus resultados. Ello posibilita, que ante un problema dado que requiere aprendizaje automatizado, cualquier investigador pueda experimentar "en lote" con varios de estos métodos y determinar con cuales se obtienen los mejores resultados.

WEKA proporciona varios algoritmos de aprendizaje automatizado que pueden utilizarse fácilmente; por ejemplo: se puede preprocesar el conjunto de datos, introducirlos en un esquema de aprendizaje y analizar el resultado y su eficiencia. Además posee un sistema de interfaces gráficas de usuario que permiten la exploración de los datos (***Explorer***) y la experimentación (***Experimenter***) con los diversos algoritmos implementados.

3.2.1 Incorporación de un nuevo aproximador de funciones a la plataforma WEKA.

Por ser de código abierto, WEKA es además extensible ya que tiene una estructura de paquetes bien organizada y facilita a otros programadores la tarea de agregar modelos y algoritmos, o modificar los ya existentes de forma ordenada. Los paquetes principales son los siguientes (Guevara, 2007):

associations: contiene las clases que implementan los algoritmos de asociación.

attributeSelection: contiene las clases que implementan técnicas de selección de atributos.

classifiers: agrupa todas las clases que implementan algoritmos de clasificación y éstas a su vez se organizan en subpaquetes de acuerdo al tipo de clasificador.

clusterers: contiene las clases que implementan algoritmos de agrupamiento.

core: paquete central que contiene las clases controladoras del sistema. Es usado en la mayoría de las clases existentes. Las clases principales de este paquete son: ***Attribute***, ***Instance***, e ***Instances***.

datagenerators: paquete que contiene clases útiles en la generación de conjuntos de datos atendiendo al tipo de algoritmo que será usado.

estimators: contiene las clases que realizan estimaciones (generalmente probabilísticas) sobre los datos.

experiment: agrupa las clases controladoras que permiten la realización de experimentos con varias bases y diferentes algoritmos.

filters: está constituido por las clases que implementan algoritmos de preprocesamiento.

gui: contiene todas las clases que implementan la interfaz visual con el usuario.

Para añadir un nuevo aproximador en WEKA se debe conocer los métodos imprescindibles que deben ser implementados, así como las clases y estructuras que WEKA posee para el trabajo con los mismos. Además es necesario seguir cierta metodología descrita en (Guevara, 2007, Bonet, 2008, Gonzáles and Araujo, 2006) para la incorporación de un nuevo clasificador, con el objetivo de lograr una total compatibilidad con la herramienta y su óptimo funcionamiento.

WEKA tiene implementada muchas facilidades de magnitudes de salidas y evaluación como la validación cruzada. A la hora de implementar un nuevo aproximador o clasificador es importante tener en cuenta la clase abstracta ***Classifier***, que es la que se utiliza para dar todas estas facilidades. Esta clase, que es la más importante en el paquete ***classifiers*** y constituye una superclase de todos los clasificadores existentes, y tiene definido los principales métodos que debe tener un aproximador o clasificador, algunos de ellos deben ser redefinidos de acuerdo al objetivo que persiga el algoritmo de aprendizaje a implementar. En regresión hay que redefinir 2 métodos que son los que a continuación describimos:

buildClassifier(): se encarga de la construcción del modelo del aproximador 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 y en el caso del aproximador de predecir el valor continuo para cualquier instancia nueva (Ver Anexo 3).

classifyInstance(): permite clasificar o predecir el valor continuo de una instancia concreta (Ver Anexo 3). En el caso de la clasificación devuelve la clase en la que se ha clasificado o “desconocido” si no se consigue clasificar. Este algoritmo está diseñado de manera general también para regresión, de forma que en este caso devuelve un valor continuo.

Una vez implementados los métodos esenciales para el funcionamiento del aproximador, es necesaria la implementación de otros métodos auxiliares para facilitar el trabajo con la interfaz del usuario. En particular, se deben modificar los métodos ***listOptions(), setOptions()*** y ***getOptions()*** para que sea posible establecer los parámetros del entrenamiento por parte del usuario (Ver Anexo 3).

Debido a las redefiniciones anteriores se crea el paquete ***weka.need.regression*** que contiene todas las clases implementadas para ***NpBasirRegression*** y todos los métodos de ponderación de rasgos que fueron descritos en el epígrafe 3.1.

En el Anexo 2 aparece el diagrama de clases fundamentales del paquete ***weka.need.regression***.

Para adicionar un nuevo método de ponderación de rasgos para ser usado en ***NpBasirRegression*** hay que incorporarlo en el paquete ***weka.need.regression*** y redefinir los métodos: ***buildClassifier(), listOptions(), setOptions()*** y ***getOptions()*** de la clase ***NpBasirRegression***.

3.2.2 ¿Cómo utilizar el nuevo aproximador de funciones *NpBasirRegression* en la plataforma WEKA?

A continuación ofrecemos una breve guía de cómo utilizar el aproximador implementado para la experimentación dentro de la plataforma WEKA.

El primer paso luego de ejecutar WEKA y seleccionar la opción de ***Explorer*** (ver Figura 3.4) en el Menú ***Applications*** es seleccionar la base de casos que se va a utilizar. Para ello se hace uso del botón ***Open file*** que se encuentra en la pestaña ***Preprocess***. Al seleccionar una base de casos se visualiza información sobre la misma relativa al nombre de la relación, número de instancias, cantidad de atributos, además de algunas estadísticas (ver ejemplo en Figura 3.5).

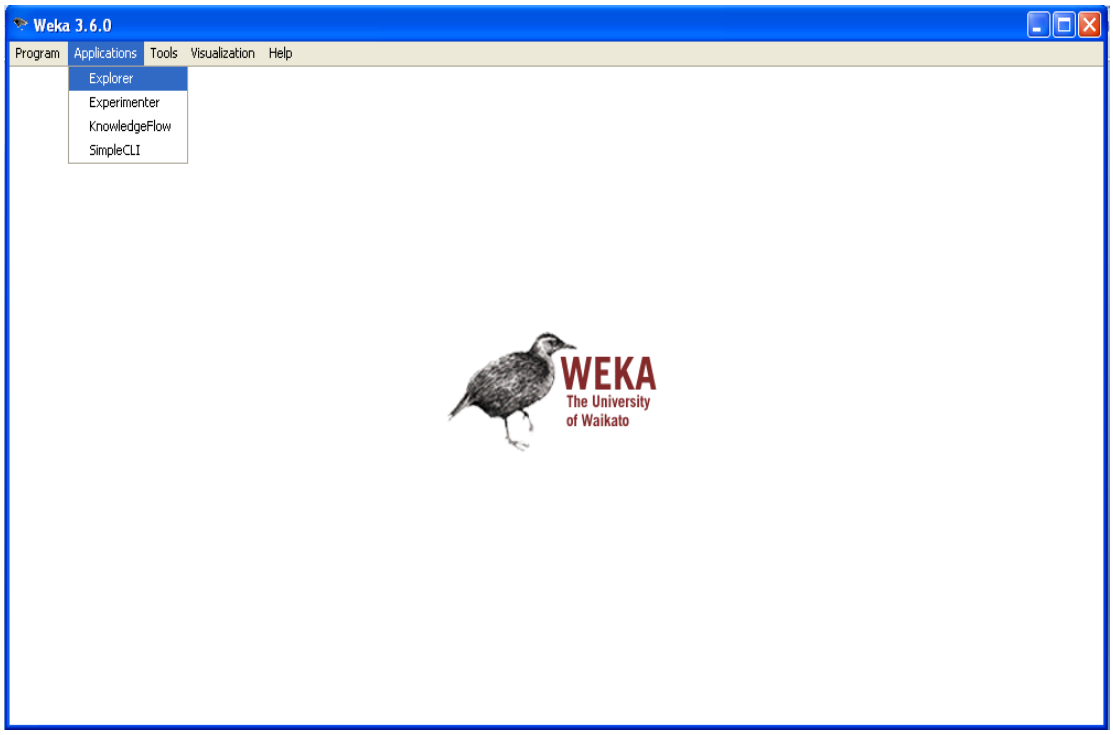


Figura 3.4 Vista inicial de la plataforma WEKA.

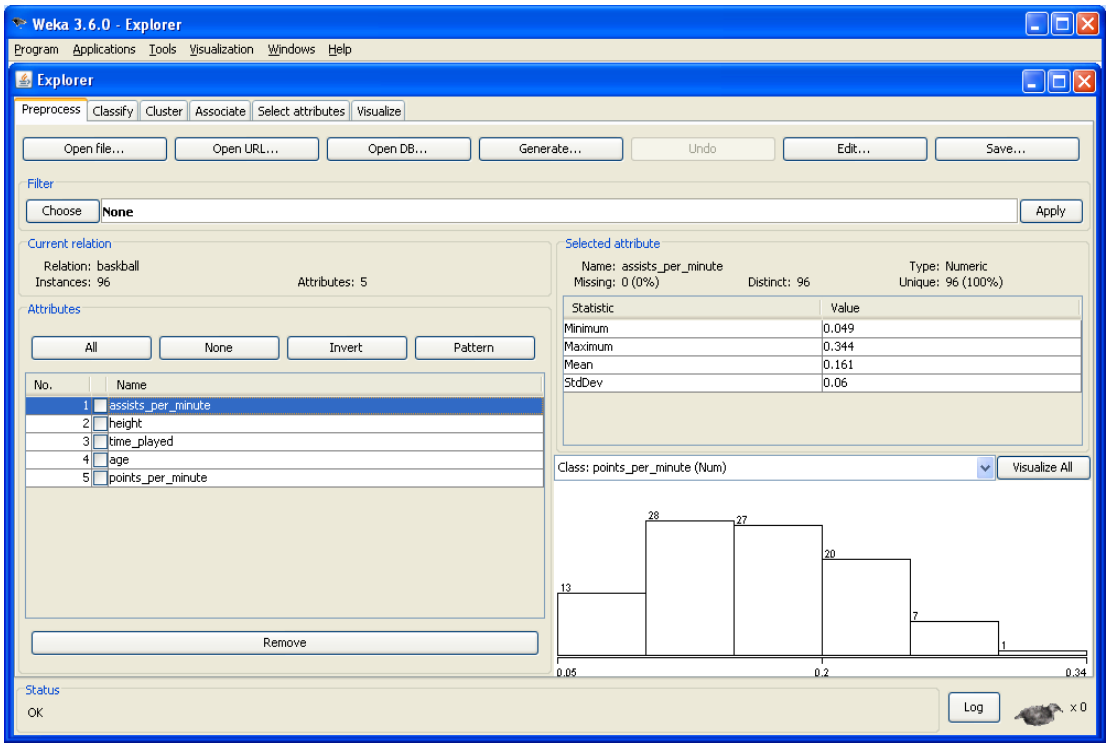


Figura 3.5 Carga de la base de casos a utilizar por el modelo.

Una vez abierta la base de casos que se va a utilizar, se procede a seleccionar el modelo de clasificación o aproximación que se quiere utilizar, para ello presionar ***Classify***. El botón ***Choose*** despliega un árbol que contiene todas las técnicas de clasificación

implementadas en la plataforma. El método propuesto ***NpBasirRegression*** se puede encontrar dentro de ***functions*** (ver Figura 3.6).

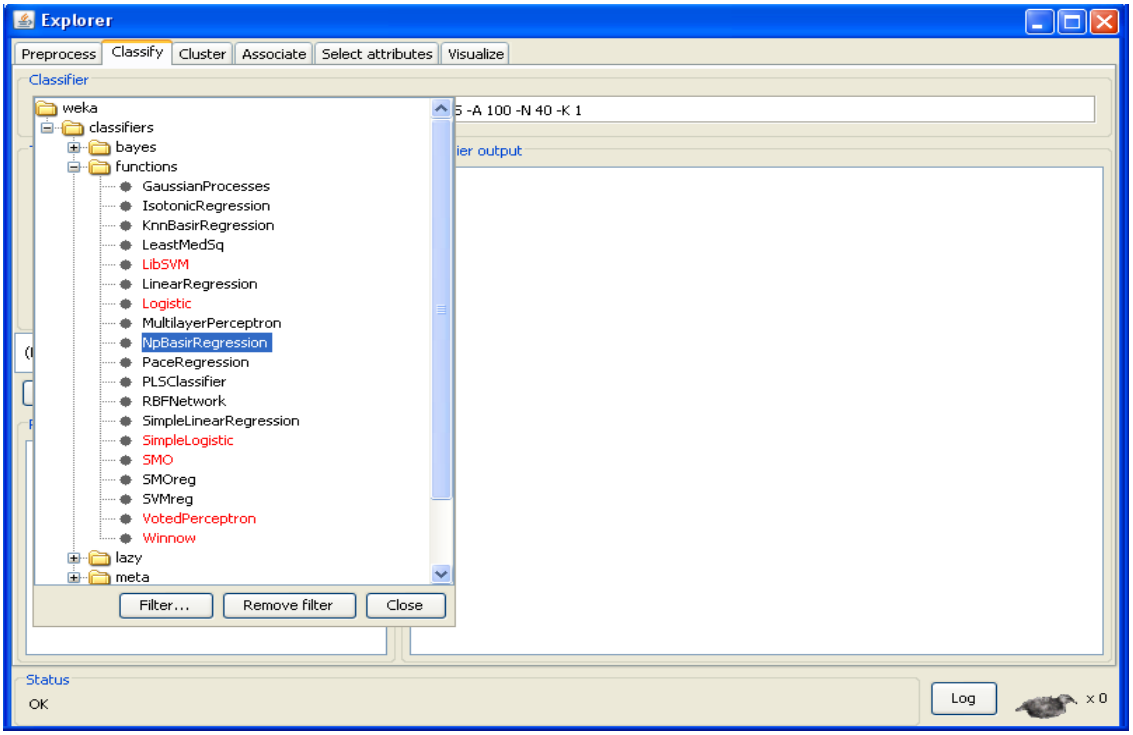


Figura 3.6 Árbol para la selección del modelo a utilizar.

Luego de seleccionar el aproximador, se deben ajustar sus parámetros para realizar el proceso de validación (ver Figura 3.7).

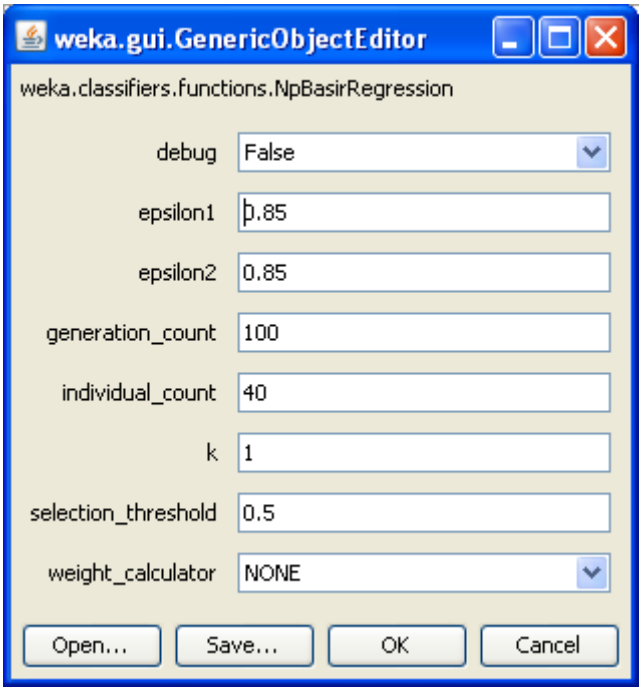


Figura 3.7 Ventana para la edición de los parámetros de entrenamiento del modelo.

Los parámetros que se pueden ajustar son los siguientes:

1. ***debug***: parámetro booleano. Si se marca como ***True*** el aproximador imprime más información durante el proceso de validación. Asume por defecto ***False***.
2. ***epsilon1***: se corresponde con el umbral para determinar las instancias similares respecto a los rasgos predictores, y que denotamos por $\mathbf{e_1}$ en la relación de similaridad (2.3). Se recomiendan valores entre 0.85 y 0.95. A mayor valor $\mathbf{e_1}$ mayor exactitud en la precisión pero menor reducción del conjunto de objetos. Asume por defecto 0.85.
3. ***epsilon2***: se corresponde con el umbral para determinar las instancias similares respecto al rasgo objetivo y que denotamos por $\mathbf{e_2}$ en la relación de similaridad (2.4). A mayor valor mayor exactitud en la precisión pero menor reducción del conjunto de objetos. Asume por defecto 0.85.
4. ***generation_count***: se refiere a la cantidad de ciclos o generaciones. A mayor valor más oportunidad de encontrar el óptimo pero mayor costo computacional. Asume por defecto 100.

5. ***individual_count***: cantidad de partículas o tamaño de la bandada para PSO y cantidad de individuos para UMDA. A mayor valor más oportunidad de encontrar el óptimo pero mayor costo computacional. Asume por defecto 40.
6. ***k***: recupera los k prototipos más similares. Se recomienda sean valores impares 1, 3,5. Asume por defecto 1.
7. ***selection_threshold:*** se refiere al porciento de selección por truncamiento. Asume por defecto 0.5.
8. ***weigth_calculator:*** método para ponderar los pesos de los rasgos. Se incorporaron 2 variantes: calcular los pesos utilizando la metaheurística Optimización Basada en Partículas (***PSO)*** y el Algoritmo de Distribución Marginal Univariado (***UMDA)***. No se puede afirmar que una variante sea mejor que la otra, esto depende de las características de la base de caso. Se asume por defecto (***NONE***), esto significa que se le da igual peso a todos los rasgos predictores es decir 1/cantidad de rasgos predictores.

A continuación se especifica qué tipo de validación se desea utilizar: con el mismo conjunto de entrenamiento, con otra base de casos del mismo problema, realizando una validación cruzada de k particiones o utilizando un porciento determinado de la base de entrenamiento. Finalmente para comenzar el proceso de validación se utiliza el botón ***Start***.

Una vez terminado el proceso de validación, el WEKA ofrece algunas estadísticas en la vista de salida del aproximador, entre las que se encuentran: el error medio absoluto, el coeficiente de correlación, la raíz del error cuadrático medio, entre otras (ver Figura 3.8). Obsérvese que existen variaciones entre los resultados mostrados para un aproximador si lo comparamos con un clasificador de rasgo objetivo discreto. Si los resultados de estas estadísticas son buenos entonces el modelo debe ser capaz de generalizar y predecir el valor continuo de nuevos casos.

```
Classifier output

=== Run information ===

Scheme:       weka.classifiers.functions.NpBasirRegression -C 1 -E 0.85 -P 0.85 -A 100 -A 40 -K 1
Relation:     baskball
Instances:    96
Attributes:   5
              assists_per_minute
              height
              time_played
              age
              points_per_minute
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

weka.classifiers.functions.NpBasirRegression@b8a16c

Time taken to build model: 4.83 seconds

=== Cross-validation ===
=== Summary ===
Correlation coefficient                  0.4884
Mean absolute error                      0.0855
Root mean squared error                  0.1096
Relative absolute error                100.1626 %
Root relative squared error            100.7649 %
Total Number of Instances               96
```

Figura 3.8 Fragmento de la salida del aproximador después de una validación cruzada de 10 particiones.

3.3 Aplicaciones de *NpBasirRegression* en la solución de problemas reales.

En el capítulo 2 se mostró, a través del estudio experimental con bases de datos internacionales, el buen desempeño que tiene el método propuesto en esta investigación. En este epígrafe se solucionan dos problemas reales relacionados con la rama de la Ingeniería Civil.

El procedimiento seguido para resolver los dos problemas reales relacionados con la rama de la Ingeniería Civil haciendo uso del método ***NpBasirRegression*** (algoritmo NP-FP más algoritmo NP-BASIR) fue el siguiente:

1. Definir la medida de similaridad local para los rasgos.
2. Establecer los umbrales de similaridad deseados.
3. Calcular la importancia de los rasgos (utilizando cualquiera de los métodos de ponderación de rasgos descritos en el epígrafe 3.1).
4. Construir el conjunto de prototipos (utilizando NP-BASIR).

5. Aplicar el método de aproximación de funciones NP-FP utilizando los prototipos construidos.

3.3.1 Aplicación en la Ingeniería Civil: Pronóstico de la capacidad resistente en Mampostería.

La mampostería es uno de los materiales de construcción con una gran diversidad de usos, en todas partes del mundo, tanto en el pasado como en el presente. Existen varios vestigios de poblados prehistóricos construidos con piedras asentadas con barro desde las Islas Aran, en Irlanda, hasta Catal Hüyük, en Anatolia. Unos 10000 años después, se utilizó este mismo sistema constructivo por los incas en Ollantaytambo, cerca del Cusco, donde todavía quedan construcciones importantes, con muros de piedra natural asentada con mortero de barro y techos de rollizos de madera cubiertos con una gruesa capa de paja. Estas son evidencias encontradas de lo que pudiera ser el comienzo del uso de la mampostería en la construcción.

Actualmente, materiales como el concreto y el acero estructural tienen un empleo activo en las construcciones modernas, pero la mampostería no deja de tener un papel importante. Su utilización se ha concentrado fundamentalmente en la construcción de viviendas unifamiliares y edificios multifamiliares de alta y mediana altura, en la mayoría de los países latinoamericanos y en otros del mundo.

La mampostería es un material que por su propia naturaleza, exhibe un comportamiento frágil ante los esfuerzos de tensión, caracterizado por una rápida degradación de su resistencia y su rigidez. Afortunadamente, este comportamiento es susceptible de mejorarse significativamente mediante la colocación de acero de refuerzo.

Es posible diferenciar tres tipos de muros constructivos clasificados a partir de la cuantía y de la disposición del acero: 1) mampostería confinada, 2) mampostería reforzada interiormente y 3) mampostería simple o no reforzada en muros.

La mampostería confinada es una de las más utilizadas en los países latinoamericanos, principalmente en Argentina, Chile, Colombia, Costa Rica, Ecuador, El Salvador, Guatemala, Honduras, México, Nicaragua y Perú; y también, aunque en menor medida, en algunos países de Asia y Europa, tales como Grecia, Italia, Eslovenia, China e Indonesia.

Tiene gran importancia la mampostería correctamente reforzada y construida. Debido a su buen comportamiento ante solicitaciones sísmicas, se puede utilizar en todo rango de riesgo sísmico, con gran desempeño, cuestión que ha impulsado, a partir de la década de los años cincuenta del siglo pasado, a extensos programas de investigación, principalmente utilizando el carácter experimental, con la meta de determinar tipologías, configuraciones y procedimientos para su análisis y diseño racionales, cobrando una mayor importancia tanto en países latinoamericanos como europeos, asiáticos y en Japón a partir de la década de los 80.

En la década de 1960 y a finales de los 70, la ingeniería de la mampostería evolucionó dando pasos importantes, cuando se comenzaron programas experimentales en México, antecediéndoles investigaciones efectuadas en otros países de mundo. Es en este país, donde se desarrollan programas de investigación proyectados para evaluar el comportamiento de los materiales y modalidades de refuerzos tradicionales y algunas invenciones.

Actualmente se cuenta con un volumen grande de información que proviene del conjunto de los programas experimentales internacionales realizados en el campo de la mampostería, donde la revolución digital ha facilitado la captura de la información y su almacenamiento. El problema de la experimentación real clásica es que requiere de cuantiosos recursos, por otro lado la modelación virtual implica un alto costo computacional y tiene la desventaja de la laboriosidad necesaria en la elaboración de los modelos. Existen diversos métodos de cálculo, que aún distan de ofrecer resultados ajustados a la experimentación. Uno de los métodos clásicos que mejores resultados muestra es el presentado en (Riahi, 2007).

Problema:

Calcular el cortante de agrietamiento de muros de mampostería, para esto se tiene una base de casos Agrietamiento (Figura 3.9) de 281 casos, conformada a partir de las investigaciones experimentales realizadas a muros de mampostería. Cada caso se compone de 34 rasgos predictores de ellos 12 nominales y el resto numérico y 1 rasgo objetivo continuo que es el cortante de agrietamiento experimental (V_{agrexp}).

Atributos	Tipo de dato
Institución	nominal
Escala	nominal
Espesor del recubrimiento de mortero por una cara (trec)	real
Relación de aspecto (hl)	real
Área del muro (lxt)	real
Modalidad	nominal
Refuerzo adicional (refadic)	nominal
Tipo de piezas (piezas)	nominal
Material de las piezas (material)	nominal
Relación entre el área neta y área bruta de las piezas de mampostería (anab)	real
Área de las piezas de mampostería (axc)	entero
Media de la resistencia a compresión de las piezas sobre el área bruta (fp)	real
Media a la resistencia a compresión de la pilas (fm)	real
Media de la resistencia a compresión diagonal de los muretes (vm)	real
Módulo de elasticidad de la mampostería (em)	real
Módulo de cortante de la mampostería (gm)	entero
Modalidad del ensayo (modensaye)	nominal
Razón entre la dimensión de los castillos paralela al muro y el ancho de los castillos (hcbc)	real
Acero longitudinal en castillos perimetrales o de confinamiento (barracast)	nominal
Esfuerzo de fluencia del acero de refuerzo vertical en castillos y dalas (fy)	real
Cuantía del acero de refuerzo en castillos (p)	entero
Estribos en castillos perimetrales o de confinamiento (estribocast)	nominal
Esfuerzo de fluencia del acero de refuerzo transversal en castillos y dalas (fyt)	entero
Distribución o separación de estribos en castillos perimetrales (distribucionest)	nominal
Refuerzo Vertical (refv)	nominal
Esfuerzo de fluencia de acero vertical en el muro (fyv)	real
Porcentaje del refuerzo vertical (pv)	real
Refuerzo Horizontal (refh)	nominal
Esfuerzo de fluencia del refuerzo horizontal o malla de alambre soldado en el muro (fyh)	entero
Porcentaje del refuerzo horizontal (ph)	real
Resistencia del concreto en castillos y dalas (fc)	real
Resistencia a compresión de los cubos (fb)	real
Resistencia a compresión de los cubos (fb2)	real
Esfuerzo vertical sobre el área total (cita)	real
Cantidad de instancias: 281	

Figura 3.9 Descripción del conjunto de casos de mampostería (Agrietamiento).

Solución:

Para realizar el cálculo del valor del cortante de agrietamiento se utilizó el método propuesto en este trabajo ***NpBasirRegression***. Se utilizó como medida de similaridad local la definida en la expresión (3.4) y se utilizó 0.85 como valor para los umbrales de similaridad $\mathbf{e_1}$ y $\mathbf{e_2}$ para los rasgos predictores y rasgo objetivo respectivamente. Se usan valores para **K** de 1 y 3. Para la metaheurística PSO se utilizaron valores para los parámetros tales como: cantidad de ciclos 100 y cantidad de partículas 40. En el caso de la metaheurística UMDA se utilizaron valores para los parámetros: cantidad de generaciones 100, cantidad de individuos 40 y porciento de selección por truncamiento

0.5. Se utilizaron también los métodos ***KnnBasirRegression***, *Multilayer Perceptron*, *Linear Regression* y *Regression Tree (REPTree)* con el objetivo de comparar los resultados alcanzados por ***NpBasirRegression*** con los resultados alcanzados por estos otros aproximadores de funciones.

En la expresión (3.4) se muestra la medida de similaridad local, la cual puede ser usada tanto para rasgos de dominio continuo como discreto, donde $\mathbf{D_k}$ denota el dominio del rasgo **k**.

$$\mathrm{sim}_k\left(x[k], y[k]\right) = \begin{cases} 1 - \dfrac{\left|x[k] - y[k]\right|}{\mathrm{Max}(D_k) - \mathrm{Min}(D_k)} & \text{si k es continuo} \\ 1 & \text{si k es discreto y } x[k] = y[k] \\ 0 & \text{si k es discreto y } x[k] \neq y[k] \end{cases} \tag{3.4}$$

Resultados:

En la Tabla 3.1 y en la Figura 3.10 se muestra la reducción del conjunto de datos Agrietamiento al aproximar con ***NpBasirRegression*** utilizando UMDA y PSO como métodos de ponderación de rasgos y valores para **K** de 1 y 3.

Conjunto de Datos	Método de Ponderación de Rasgos	K	Cantidad de Prototipos Máxima/Mínima	Coeficiente de Reducción Máxima/Mínima
Agrietamiento	UMDA	1	69/48	75.4/82.9
		3	67/57	76.2/79.7
	PSO	1	66/55	76.5/80.4
		3	66/58	76.5/79.4

Tabla 3.1 Reducción del conjunto de casos Agrietamiento al aproximar ***NpBasirRegression.***

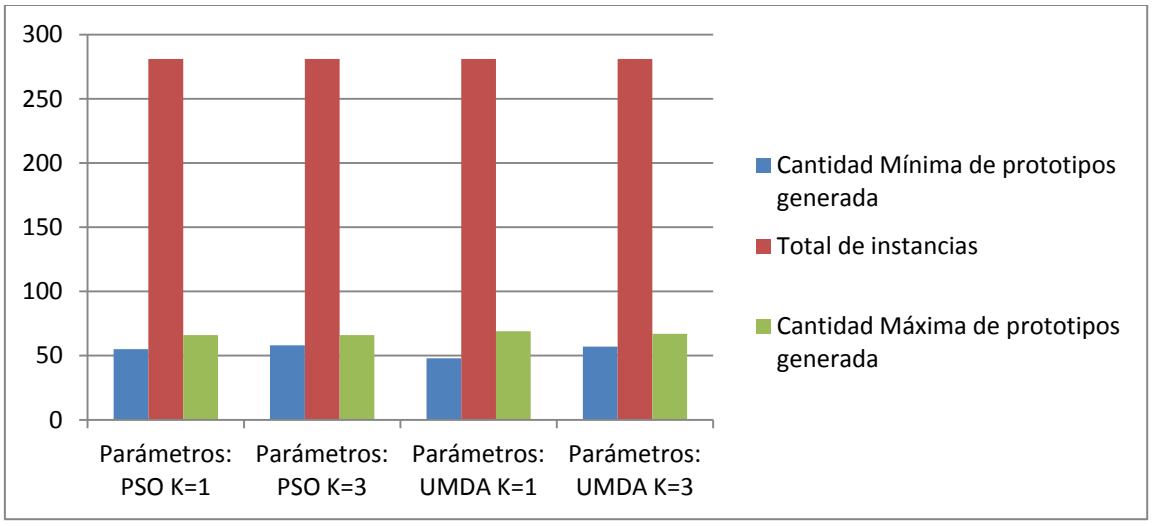


Figura 3.10 Reducción del conjunto de ejemplos Agrietamiento al aproximar ***NpBasirRegression.***

En la Tabla 3.2 y Figura 3.11 se muestran los resultados alcanzados al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression*** utilizando UMDA y PSO como métodos de ponderación de rasgos y valores para **K** de 1 y 3.

Conjunto de Datos	Método de Ponderación de Rasgos	Métodos	K	Error Medio Absoluto
Agrietamiento	UMDA	NpBasirRegression	1	3.81
		NpBasirRegression	3	3.99
		KnnBasirRegression	1	3.45
		KnnBasirRegression	3	3.42
	PSO	NpBasirRegression	1	3.96
		NpBasirRegression	3	4.81
		KnnBasirRegression	1	3.89
		KnnBasirRegression	3	4.01

Tabla 3.2 Resultados al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression***.

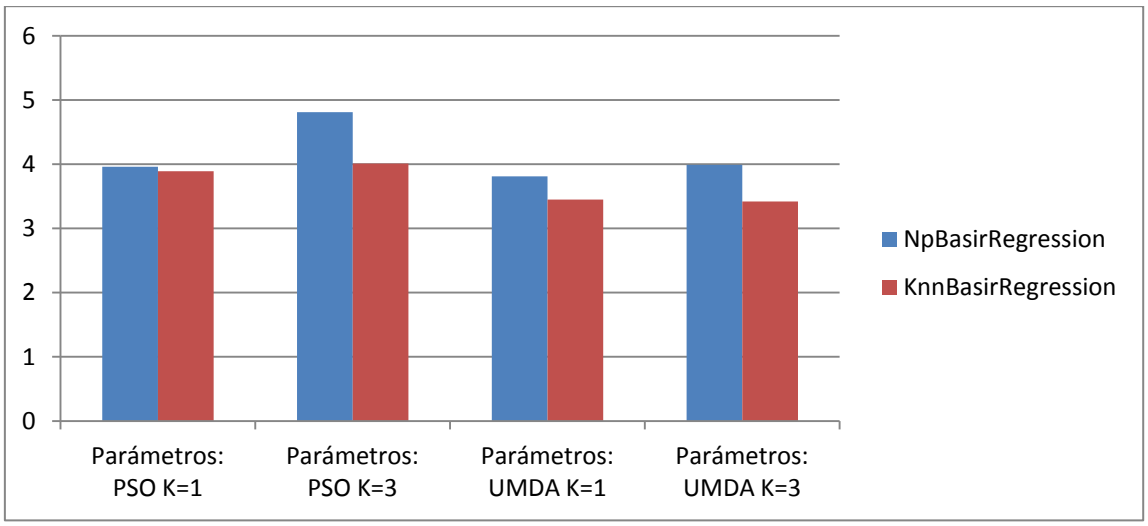


Figura 3.11 Resultados al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression***.

En la Tabla 3.3 se muestran los resultados al aproximar con *Multilayer Perceptron*, *Linear Regression* y *REPTree*.

Conjunto de Datos	Métodos	Error Medio Absoluto
Agrietamiento	Multilayer Perceptron	5.45
	Linear Regression	4.74
	REPTree	4.40

Tabla 3.3 Resultados al aproximar con *Multilayer Perceptron*, *Linear Regression* y *REPTree*.

En la Figura 3.12 se muestra los resultados alcanzados al aproximar con ***NpBasirRegression*** utilizando UMDA como método de ponderación de rasgos con **K** igual 1 (con este juego de parámetros se obtiene el mejor resultado al aproximar con ***NpBasirRegression,*** lo cual nos permite corroborar la calidad de los prototipos construidos) y los resultados alcanzados al aproximar con *Multilayer Perceptron*, *Linear Regression* y *REPTree.*

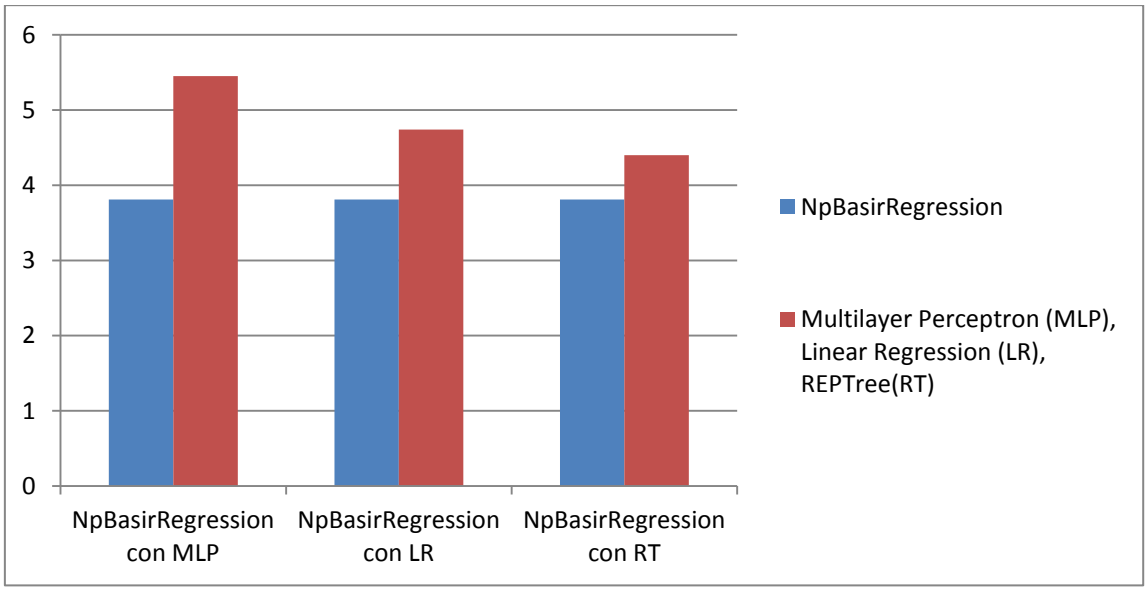


Figura 3.12 Resultados al aproximar con ***NpBasirRegression***, *Multilayer Perceptron*, *Linear Regression* y *REPTree.*

Análisis de los resultados:

Se logra obtener una reducción del 83 % del conjunto de datos Agrietamiento. A pesar de esta gran reducción del conjunto de instancias los resultados al aproximar con ***NpBasirRegression*** (especialmente cuando se utiliza UMDA como método de ponderación de rasgo y **K** igual 1) son similares a los resultados obtenidos por ***KnnBasirRegression***. Además con ***NpBasirRegression*** se logran resultados con mejor

eficacia con respecto al error con respecto a otros métodos de regresión numérica tales como *Multilayer Perceptron, Linear Regression* y *REPTree.*

3.3.2 Aplicación en la Ingeniería Civil: Pronóstico de la capacidad resistente al deslizamiento longitudinal en las losas compuestas con lámina colaborante (TU).

En la Ingeniería Civil las losas compuestas formadas por láminas de acero como encofrado colaborante constituyen soluciones muy eficientes en la construcción. Para los especialistas de esta rama es importante conocer los fallos que se producen en este tipo de estructuras. Normalmente para dar solución a estos problemas se emplean métodos numéricos, y la solución que se obtiene es aproximada. Por este motivo, los modelos numéricos tienen que someterse a un proceso de calibración matemática y física. La calibración matemática garantiza una aproximación adecuada y estabilidad de la solución obtenida. Por su parte, la calibración física garantiza que el modelo implementado tenga correspondencia con el fenómeno físico que se estudia. Es por esta razón, que el proceso de calibración física necesita disponer de ensayos previos del fenómeno estudiado.

Los métodos experimentales, a pesar de su relativa fiabilidad, tienen como principal inconveniente el elevado costo que representan, pues es necesario previamente construir una lámina colaborante o porción de ella con su geometría exacta, incluido su sistema de embuticiones, para poder llevar a cabo los estudios experimentales. El construir una versión piloto de una lámina colaborante o porción de ella, por lo general es de un costo sumamente elevado; mucho más que el de los propios ensayos, que ya por sí solos son bastante caros y requieren de equipamiento especializado para su ejecución. (López, 2010)

Problema:

Por el alto costo que representa la realización de estos ensayos tanto a escala normal y reducida, es que se opta por la utilización de técnicas de inteligencia artificial para predecir el fallo estructural en losas compuestas. Para esto, se tiene la base de casos "TU" de 429 casos, la cual fue conformada a partir de los resultados de las investigaciones reportadas en (López, 2010). La descripción de cada caso en "TU" se

puede encontrar en la Figura 3.13. Cada caso tiene 13 rasgos predictores todos numéricos y el atributo objetivo es el fallo que se produce.

Atributos	**Tipo de dato**
Resistencia del hormigón del bloque (Ruhgon)	real
Ángulo de perfilado (alfa)	real
Espesor de la lamina metálica colaborante (delta)	real
Ancho de la greca de la lámina (b)	real
Altura o peralto de la lámina (h)	real
Longitud de las embuticiones (L)	real
Profundidad de la embutición (d)	real
Ancho de las embuticiones (a)	real
Paso entre embuticiones (P)	real
Área	real
Seno del valor del rasgo alfa (sen_alfa)	real
Seno del valor del rasgo beta (sen_beta)	real
Seno del valor del rasgo gamma (sen_gamma)	real
Cantidad de ejemplos: 429	

Figura 3.13 Descripción de los casos de la base de deslizamiento longitudinal en las losas compuestas con lámina colaborante (TU).

Solución:

Para dar solución a esta problemática se utilizó ***NpBasirRegression.*** Se utilizaron como métodos de ponderación de rasgo UMDA y PSO, para ambos métodos los valores para los parámetros fueron: cantidad de generaciones 100, cantidad de partículas o individuos 40 y en el caso de UMDA un porciento de selección por truncamiento de 0.5. Se utilizó la medida de similaridad local definida anteriormente en la expresión (3.4), como valor para los umbrales de similaridad $\mathbf{e_1}$ y $\mathbf{e_2}$ 0.85 y valores para **K** de 1 y 3. Se utilizaron también los métodos ***KnnBasirRegression***, *Multilayer Perceptron*, *Linear Regression* y *REPTree* con el objetivo de comparar los resultados alcanzados por ***NpBasirRegression*** con los resultados alcanzados por estos otros aproximadores de funciones.

Resultados:

En la Tabla 3.4 y en la Figura 3.14 se muestra la reducción del conjunto de entrenamiento "TU" al aproximar con ***NpBasirRegression*** utilizando UMDA y PSO como métodos de ponderación de rasgos y valores para **K** de 1 y 3.

Conjunto de Datos	Método de Ponderación de Rasgos	K	Cantidad de Prototipos Máxima/Mínima	Coeficiente de Reducción Máxima/Mínima
TU	UMDA	1	119/99	72.3/76.9
		3	111/97	74.1/77.4
	PSO	1	101/79	76.5/81.6
		3	100/77	76.7/82.1

Tabla 3.4 Reducción del conjunto de entrenamiento TU al aproximar ***NpBasirRegression.***

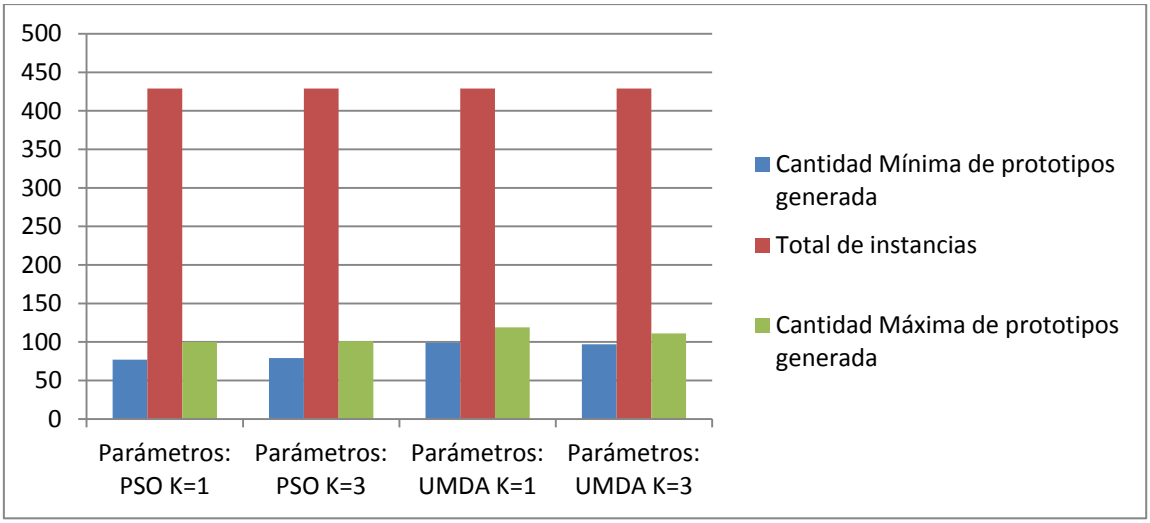


Figura 3.14 Reducción del conjunto de entrenamiento TU al aproximar ***NpBasirRegression.***

En la Tabla 3.5 y Figura 3.15 se muestran los resultados alcanzados al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression*** utilizando UMDA y PSO como métodos de ponderación de rasgos y valores para **K** de 1 y 3.

Conjunto de Datos	Método de Ponderación de Rasgos	Métodos	K	Error Medio Absoluto
TU	UMDA	NpBasirRegression	1	17.46
		NpBasirRegression	3	22.66
		KnnBasirRegression	1	17.01
		KnnBasirRegression	3	15.83
	PSO	NpBasirRegression	1	24.01
		NpBasirRegression	3	29.82
		KnnBasirRegression	1	22.01
		KnnBasirRegression	3	20.28

Tabla 3.5 Resultados al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression***.

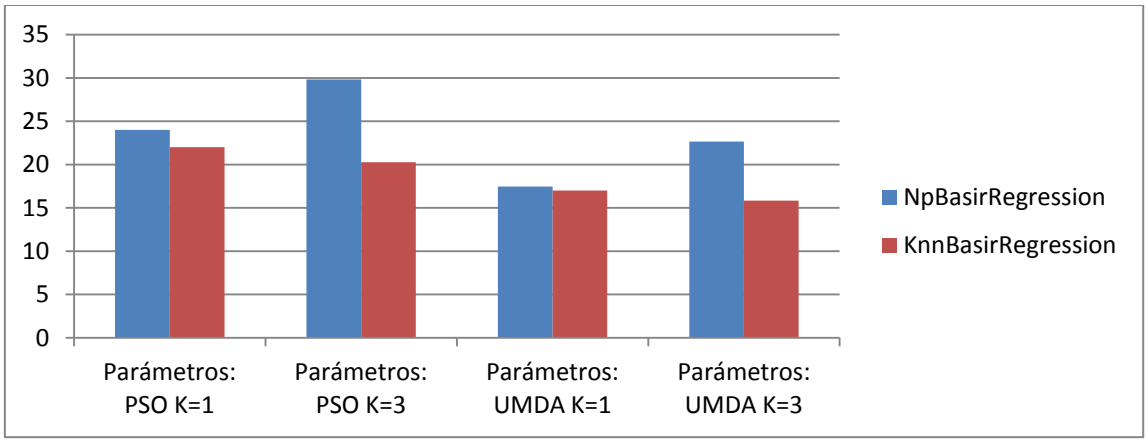


Figura 3.15 Resultados al aproximar con ***NpBasirRegression*** y ***KnnBasirRegression***.

En la Tabla 3.3 se muestran los resultados al aproximar con *Multilayer Perceptron*, *Linear Regression* y *REPTree*.

Conjunto de Datos	Métodos	Error Medio Absoluto
TU	Multilayer Perceptron	14.39
	Linear Regression	25.15
	REPTree	17.49

Tabla 3.3 Resultados al aproximar con *Multilayer Perceptron*, *Linear Regression* y *REPTree*.

En la Figura 3.16 se muestra los resultados alcanzados al aproximar con ***NpBasirRegression*** utilizando UMDA como método de ponderación de rasgos con **K** igual 1 (con este juego de parámetros se obtiene el mejor resultado al aproximar con ***NpBasirRegression***) y los resultados alcanzados al aproximar con *Multilayer Perceptron*, *Linear Regression* y *REPTree.*

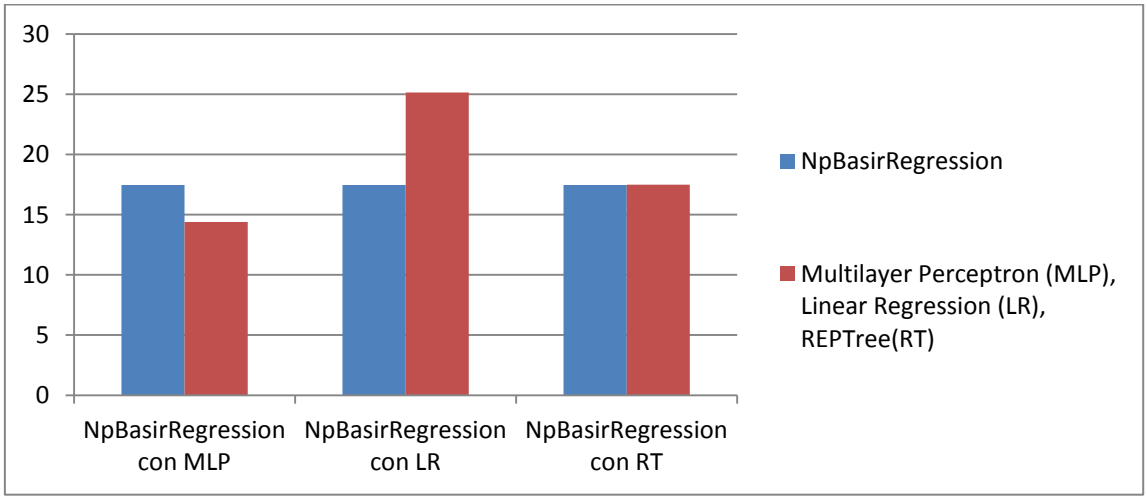


Figura 3.12 Resultados al aproximar con ***NpBasirRegression***, *Multilayer Perceptron*, *Linear Regression* y *REPTree.*

Análisis de los resultados:

Se logra obtener una reducción del 82 % del conjunto de datos "TU". Al aproximar con ***NpBasirRegression*** (especialmente cuando se utiliza UMDA como método de ponderación de rasgo y **K** igual 1) se obtienen resultados similares a los obtenidos por *REPTree* y ***KnnBasirRegression***, sólo en dos casos se aprecian algunas diferencias que pudieran ser despreciables. Además con ***NpBasirRegression*** se logran resultados más eficaces que los obtenidos por *Linear Regression.*

3.4 Conclusiones parciales del capítulo.

Se incorpora a WEKA un constructor de prototipos y el aproximador de funciones que utiliza el constructor de prototipos, se adicionan dos métodos de ponderación de rasgos basados en metaheurísticas poblacionales y se modifican los métodos que favorecen la visualización de los resultados referidos a la cantidad de prototipos.

Se solucionaron problemas reales relacionados con la rama de la Ingeniería Civil, específicamente el cálculo de agrietamiento en muros de mampostería y el fallo estructural en losas compuestas. Los resultados alcanzados por ***NpBasirRegression*** en la solución de estas dos problemáticas confirmaron los obtenidos en los estudios experimentales desarrollados en el capítulo 2, o sea:

- Se logra una reducción significativa de más del 80%, tanto para el conjunto de casos Agrietamiento como para el conjunto "TU".
- ***NpBasirRegression*** tiene un desempeño similar a ***KnnBasirRegression*** en relación a la eficacia. Sin embargo, ***NpBasirRegression*** supera en eficacia a *Multilayer Perceptron*, *Linear Regression* y *REPTree* (los cuales trabajan con todo el conjunto de entrenamiento) cuando se trabajó con Agrietamiento y en el caso de "TU" se alcanzan resultados similares a *REPTree* y supera a *Linear Regression.*

CONCLUSIONES

Se desarrolló un método de construcción de prototipos basado en relaciones de similaridad que maximizan la medida calidad de la similaridad, el cual permite obtener conjuntos de prototipos con una disminución significativa de la cantidad de ejemplos del conjunto de entrenamiento.

Se diseñó e implementó un nuevo método de aproximación de funciones basado en los prototipos más cercanos utilizando relaciones de similaridad.

Se incorpora a WEKA el constructor de prototipo y el aproximador de funciones que utiliza el constructor de prototipos, ***NpBasirRegression***. Se adicionan dos métodos de ponderación de rasgos basados en metaheurísticas poblacionales (PSO y UMDA) y la medida calidad de la similaridad. Además se adiciona a WEKA una adaptación del algoritmo de los k vecinos más cercanos para dar solución a problemas de aproximación de funciones ***KnnBasirRegression*** basado en relaciones de similaridad.

Se compararon los resultados alcanzados por ***NpBasirRegression*** con los alcanzados por otros aproximadores de funciones como ***KnnBasirRegression***, *Multilayer Perceptron, Linear Regression* y *Regression Tree*; a partir de estudio experimental con 16 conjuntos de datos reconocidos internacionalmente y 6 juegos de parámetros concluyéndose que:

- No existen diferencias significativas entre la eficacia en la aproximación de funciones lograda utilizando ***NpBasirRegression*** (que utiliza sólo el conjunto de prototipos) respecto a la que se alcanza con todos los ejemplos cuando se aplica ***KnnBasirRegression***.
- No existen diferencias significativas entre la eficacia de la aproximación de funciones obtenida usando ***NpBasirRegression*** respecto a la que se alcanza usando otros aproximadores como *Multilayer Perceptron*, *Linear Regression* y *Regression Tree*.

Se solucionaron problemas reales relacionados con la rama de la Ingeniería Civil, específicamente en el pronóstico de la capacidad resistente en muros de mampostería y la capacidad resistente al deslizamiento longitudinal en las losas compuestas con lámina

colaborante. En la solución de ambas problemáticas se utilizó ***NpBasirRegression***. En ambos casos se logra una reducción significativa de los conjuntos de casos y se alcanza un desempeño similar al obtenido por otros métodos de aproximación de funciones que trabajan con todo el conjunto de ejemplos, siendo en ocasiones más eficaz el método implementado.

RECOMENDACIONES

Los resultados alcanzados, así como la experimentación realizada permitió formular nuevas tareas de investigación, las cuales quedan formuladas como recomendaciones del presente trabajo y son:

- ✓ Implementar otros métodos de construcción de prototipos para comparar el propuesto en este trabajo con otros métodos existentes
- ✓ Analizar el efecto de otros métodos de ponderación de rasgos en la construcción de prototipos, en particular CMA-ES (*Covariance Matrix Adaptation - Evolution Strategy*) y Evolución Diferencial.

REFERENCIAS BIBLIOGRÁFICAS

AHA, D. & KIBLER, D. 1991. Instance-based learning algorithms. *Machine learning.*

AREERACHAKUL, S. & SANGUANSINTUKUL, S. 2010. Classification and Regression Trees and MLP Neural Network to Classify Water Quality of Canals in Bangkok, Thailand. *International Journal of Intelligent Computing Research (IJICR),* 1.

BARANDELA, R. Year. The nearest neighbor rule and the reduction of the training sample size. *In:* Proceedings 9th Symposium on Pattern Recognition and Image Analysis, 2001 Castellon, España. 103-108.

BARANDELA, R., GASCA, E. & ALEJO, R. 2002. Correcting the Training Data. *Pattern Recognition and String Matching.*

BELLO, R. & OSORIO, J. J. G. *Tomando Decisiones Basadas en el Conocimiento*

BEZDEK, J. C., KELLER, J., KRISHNAPURAM, R. & PAL, N. P. 1999. Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. *Pattern Recognition and Inage Processing*.

BEZDEK, J. C. & KUNCHEVA, L. I. 2004. Nearest Prototype Classifier Designs: An Experimental Study. *International journal of Intelligent systems,* 16**,** 1445-1473.

BEZDEK, J. C., REICHHERZER, T. R., LIM, G. S. & ATTIKIOUZEL, Y. 1998. Multiple prototype classifier design. *IEEE Trans. on System, Man, and Cybernetics,* C28**,** 67-79.

BISHOP, C. 1995. Extremely well-written, up-to-date. Requires a good mathematical background, but rewards careful reading, putting neural networks firmly into a statistical context. *Neural Networks for Pattern Recognition*.

BONET, I. 2008. *Modelo para la clasificación de secuencias, en problemas de la bioinformática, usando técnicas de inteligencia artificial.* . PhD, Universidad Central "Marta Abreu" de Las Villas.

BORGELT, C. & KRUSE, R. 2003. Neural Networks, Working Group Neural Networks and Fuzzy Systems. University of Magdeburg.

BRATTON, D. & KENNEDY, J. Year. Defining a Standard for Particle Swarm Optimization. *In:* IEEE Swarm Intelligence Symposium (SIS 2007). 2007.

BREIMAN, L., FRIEDMAN, J., STONE, C. J. & OLSHEN, R. A. 1984. *The classic Classification and Regression Tree (C&RT) algorithm.*

CABALLERO, Y. 2007. *Aplicación de la Teoría de los Conjuntos Aproximados en el Preprocesamiento de los Conjuntos de Entrenamientos para Algoritmos de Aprendizaje Automatizado.* Doctor en Ciencias Técnicas . Universidad central de Las Villas.

CEVERON, V. & FERRI, F. J. 2001. Another move towards the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule. *IEEE Trans. on System, Man, and Cybernetics, Part B:Cybernetics,* 31.

CORTIJO, F. J. & BLANCA, N. P. D. L. 1997. A comparative study of some non-parametric spectral classifiers. Applications to problems with high-overlapping training sets *International Journal of remote Sensing,* 18**,** 1259-1275.

COVER, T. M. & HART, P. E. 1967. Nearest neighbour pattern classification. *Institute of Electronical and Electronics Engineers Transactions on Information Theory.,* 13**,** 21-27.

CHANG, E. I. & LIPPMANN, R. P. Year. Using genetic algorithms to improve pattern classification performarce. *In:* Advances in Neural Information ProcessingSystem, 1991 San Mateo. CA: Morgan Kaufmann, 797-803.

DASARATHY, B. V. 1991. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.,* Los Alamitos,California, IEEE Computer Society Press.

DEMSAR, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research***,** 1-30.

DEVIJVER, P. A. & KITTLER, J. Year. On the edited nearest neighbor rule. *In:* 5th International Conference on Pattern Recognition, 1980 Los Alamitos,California. IEEE Computer Society Press, 72-80.

DEVIJVER, P. A. & KITTLER, J. 1982. Pattern Recognition: A Statistical Approach.

EFRON, B. & GONG, G. 2008. A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation.

FAUSETT, L. 1994. A well-written book, with very detailed worked examples to explain how the algorithms function. *Fundamentals of Neural Networks*.

FAWCETT, T. 2005. An introduction to ROC analysis.

FERRI, F. J. 1998. Combining adaptive vector quantization and prototype selection techniques to improve nearest neighbor classifiers. 34**,** 405-410.

FILIBERTO, Y. 2012. *Métodos de aprendizaje para dominios con datos mezclados basados en la teoría de los conjuntos aproximados extendida.,* Universidad central de Las Villas.

FILIBERTO, Y. & BELLO, R. Year. A method to built similarity relations into extended Rough set theory. *In:* Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA2010), 2010a Cairo, Egipto.

FILIBERTO, Y. & BELLO, R. 2010b. Using PSO and RST to Predict the Resistant Capacity of Connections in Composite Structures. *In International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) Springer***,** 359-370.

FILIBERTO, Y. & BELLO, R. 2011. Improving the MLP Learning by Using a Method to Calculate the Initial Weights of the Network Based on the Quality of Similarity Measure. *Lecture Notes in Computer Science (LNAI)* 7095**,** 351-362.

GARCÍA-DURAN, FERNÁNDEZ, F. F. & BORRAJO, D. 2010. A prototype-based method for classification with time constraints: a case study on automated planning.

GARCÍA-MARTÍNEZ 2008. Global and local real-coded genetic algorithms based on parent-centric crossover operators. *European Journal of Operational Research,* 185**,** 1088-1113.

GARCÍA, M. M. & BELLO, R. 2003. *Redes Neuronales Artificiales.*

GATES, G. W. 1972. The reduced nearest neighbor rule. *IEEE Trans. on Information Theory***,** 431-433.

GEVA, S. & SITTE, J. 1991. Adaptative nearest neigbor pattern classifier. *IEEE Trans. on Neural Networks,* 2**,** 318-322.

GONZÁLES, H. M. & ARAUJO, L. I. 2006. *Extensiones al Ambiente de Aprendizaje Automatizado WEKA.* Lic, Universidad Central "Marta Abreu " de Las Villas.

GONZÁLEZ, J. R. H. 1995. Redes neuronales artificiales. Fundamentos,modelos y aplicaciones. *In:* IBEROAMERICANA., A.-W. (ed.).

GRAU, R. 1994. *Estadística aplicada utilizando paquetes de software.* .

GRBOVIC, M. & VUCETIC, S. 2009. Regression Learning Vector Quantization. *IEEE International Conference on Data Mining*.

GUEVARA, L. 2007. *Extensión del sistema Weka con la incorporación de Redes Neuronales Recurrentes.*

GYORFI, L. 1981. The Rate of Convergence of k-NN Regression Estimates and Classification Rules. *IEEE Trans. on Information Theory,* 27**,** 362-364.

HAMMER, B., STRICKERT, M. & VILLMANN, T. 2005. On the Generalization Ability of GRLVQ Networks. *Neural Processing Letters,* 21**,** 109-120.

HAMMER, B. & VILLMANN, T. Year. Mathematical Aspects of Neural Networks. *In:* European Symposium on Artificial Neural Networks 2003, 2003. 59-72.

HART, P. E. 1968. The condensed nearest neighbor rule. *IEEE Trans. on Information Theory,* IT-14**,** 515-516.

HERRERA, F. & LOZANO, M. 2003. Taxonomy for the Crossver Operator for Real Coded Genetic Algorithms: An Experimental Study. *International Journal of Intelligent Systems* 18**,** 309-338.

HU, X. & EBERHART, R. Year. Multiobjective optimization using dynamic neighborhood Particle Swarm Optimization. *In:* Computational Intelligence, 2002 Hawaii. IEEE.

JIN, X.-B., LIU, C.-L. & HOU, X. 2010. Regularized margin-based conditional log-likelihood loss for prototype learning. *Pattern Recognition* 43**,** 2428-2438.

KARAYIANNIS, N. B. & BEZDEK, J. C. 1996. Repairs to GLVQ: A new family of competitive learning schemes. *IEEE Trans. on Neural Networks,* 7**,** 1062-1071.

KENNEDY, J. & EBERHART, R. C. Year. Particle swarm optimization. *In:* In Proceedings of the 1995 IEEE International Conference on Neural Networks, 1995 Piscataway, New Jersey. IEEE Service Center.

KHELIFI, FOUAD & JIANMIN, J. 2011. K-NN Regression to Improve Statistical Feature Extraction for Texture Retrieval. *IEEE Transactions on Image Processing,* 20**,** 293-298.

KIM, S.-W. & OOMMEN, B. J. 2003. A brief taxonomy and ranking of creative prototype reduction schemes. *Pattern Anal Applications* 6**,** 232-244.

KOGGALAGE, R. & HALGAMUGE, S. 2004. Reducing the Number of Training Samples for Fast Support Vector Machine Classification. 2.

KOHAVI, R. 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.

KOHONEN, T. 1990. The Self-organizing Map. *Proceedings of the IEEE,* 78.

KOHONEN, T. 1995. *Self-Organizing Maps.,* Germany.

KUNCHEVA, L. I. 1995. Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters, Special Issue on Genetic Algorithms,* 16**,** 809-814.

KUNCHEVA, L. I. 1997. Fitness function in editing k-NN reference set by genetic algorithms. *Pattern Recognition,* 30**,** 1041-1049.

KUNCHEVA, L. I. 2005. *Combining Pattern Classifiers. Methods and Algorithms.,* New York, Wiley Interscience.

KUNCHEVA, L. I. & BEZDEK, J. C. 1998. On prototype selection: Genetic algorithms or random search? *IEEE Trans. on System, Man, and Cybernetics, Part B:Cybernetics,* C28**,** 160-164.

LARRAÑAGA, P. & LOZANO, J. A. 2001. *Estimation of Distributions Algorithms. A new Tool for Evolutionary Computation*, Kluwer Academic Publishers.

LARRAÑAGA, P. & PUERTA, J. M. 2003. Algoritmos de Estimación de Distribución.

LIPPMANN, R. P. 1988. *An introduction to computing with neural nets,* ACM.

LÓPEZ, M. 2010. *Estudio del comportamiento de losas compuestas con lámina colaborante mediante la combinación de técnicas de experimentación y simulación numérica.*, Universidad Central de Las Villas.

LOZANO, M. & SÁNCHEZ, J. S. 2003. Training Set Size Reduction by Replacing Neighbouring Prototypes.

LOZANO, M., SOTOCA, J. M., SÁNCHEZ, J. S., PLA, F., PERKALSKA, E. & DUIN, R. P. W. 2006. Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. *Pattern Recognition* 39**,** 1827-1838.

MADERA, J. C. 2009. *Hacia una generación más eficiente de algoritmos evolutivos con estimación de distribuciones: prueba de independencia+paralelismo.* Doctor en Ciencias Matemáticas., Instituto de Cibérnetica; Matemática, y Física.

MITCHELL, T. M. 1997. *Machine Learning,* Portland, OR, USA, McGraw Hill.

MITRA, S. 2010. Shadowed c-means: Integrating fuzzy and rough clustering. *Pattern Recognition,* 43**,** 1282-1291.

MÜHLENBEIN, H. 1998. The equation for the response to selection and its use for prediction Evolutionary Computation. *Evolutionary Computation,* 5**,** 303-346.

MÜHLENBEIN, H., MAHNIG, T. & OCHOA, A. 1999. Schemata, distributions and graphical models on evolutionary optimization. *Journal of Heuristics,* 5**,** 215-247.

NAVOT, A., SHPIGELMAN, L., TISHBY, N. & VAADIA, E. 2005. Nearest Neighbor Based Feature Selection for Regression and its Application to Neural Activity. *NIPS.*

ODORICO, R. 1997. Learning vector quantization with training counters(LVQTC). *Neural Networks* 10**,** 1083-1088.

PARDO, A. & RUIZ, M. A. 2002. *SPSS 11. Guía para el análisis de datos,* Madrid, McGraw-Hill.

PARSOPOULOS, K. E. & PLAGIANAKOS, V. P. Year. Stretching technique for obtaining global minimizers through Particle Swarm Optimization. *In:* Particle Swarm Optimization Workshop, 2001 Indianapolis, USA.

PARSOPOULOS, K. E. & VRAHATIS, M. N. 2002. Recent approaches to global optimization problems through Particle Swarm Optimization. *Natural Computing,* 1**,** 235-306.

QUINLAN, J. R. 1986. Induction of decision trees. *Machine Learning.*

QUINLAN, J. R. Year. C4.5: Programs for Machine Learning. *In,* 1993 San Mateo, California.: Morgan Kaufmann.

QUINLAN, J. R. 2000. C5.0: An Informal Tutorial.

RIAHI, Z. 2007. Performance-based seismic models for confined masonry walls. England, Universidad de British Columbia.

RIPLEY, B. D. 1996. A very good advanced discussion of neural networks, firmly putting them in the wider context of statistical modeling. *Pattern Recognition and Neural Networks*.

RUIZ, J. 2010. *Reconocimiento lógico combinatorio de patrones: teoría y aplicaciones.* Doctor en Ciencias, Universidad Central de Las Villas.

RUMELHART, D. E. 1986. Learning representations by back-propagating errors. *Nature,* 323**,** 533-536.

RUSSELL, S. J. & NORVIG, P. 1996. *Inteligencia artificial:un enfoque moderno.,* Mexico, Prentice Hall Hispanoamericana S.A.

RYNKIEWICZ, J. 2012. General bound of overfitting for MLPregression models. *Neurocomputing,* 90**,** 106-110.

SAEYS, Y. & INZA, I. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics,* 23**,** 2507-2517.

SERRA, R. & CUCCHIARA, R. 2009. The Necessity of Machine Learning and Epistemology in the Development of Categorization Theories: A Case Study in Prototype-Exemplar Debate. *Lectures Notes on Artificial Intelligence* 5883**,** 182-191.

SHULCLOPER, J. R. & ARENAS, A. G. 1999. ENFOQUE LÓGICO COMBINATORIO AL RECONOCIMIENTO DE PATRONES.

SIMON, H. 1983. Why should machines learn?

. *Machine Learning, An Artificial Intelligence approach.*

SKABAR, A. & CLOETE, I. 2001. Neural Networks, Financial Trading and the Efficient Markets Hypothesis.

SKALAK, D. B. Year. Prototype and feature selection by sampling and random mutation hill climbimg algorithms. *In:* 11th Internacional Conference on Machine Learning, 1994 Los Alamitos. 293-301.

TORRES-REYNA, O. 2009. Regression Analysis. *Data Analysis 101*. University Princeton.

WEBER, R. & FOIX, C. 2007. Pronóstico del precio del cobre mediante redes neuronales. *Revista Ingenieria de Sistemas,* XXI**,** 75-76.

WERBOS, P. J. 1974. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD, Harvard University.

WILSON, D. L. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. on System, Man, and Cybernetics,* SMC-2**,** 408-421.

WILSON, R. & MARTÍNEZ, T. 1998. Reduction Techniques for Exemplar Based Learning Algorithms. *Machine Learning. Computer Science Department, Brigham Young University. USA.*

WILSON, R. & MARTÍNEZ, T. 2000. Reduction Techniques for Instance Based Learning Algorithms. *Machine Learning. Computer Science Department, Brigham Young University. USA,* 38**,** 257-286.

WITTEN, I. & FRANK, E. Year. Data Mining: Practical Machine Learning Tools and Techniques. *In,* 2005 San Francisco.

WU, X. & KUMAR, V. 2008. Top 10 algorithms in data mining. *Knowledge Information System,* 14**,** 1-37.

XIE, Q., LASSLO, C. A. & WARD, R. K. 1993. Vector quatization technique for nonparametric classifier design. *IEEE Trans. on Pattern Analysis and Machine Learning Intelligence,* 15**,** 1326-1330.

XU, M. & SETIONO, R. 2003. Gene selection for cancer classification using a hybrid of univariate and multivariate feature selection methods. *Applied Genomic and Proteomics* 2**,** 79-91.

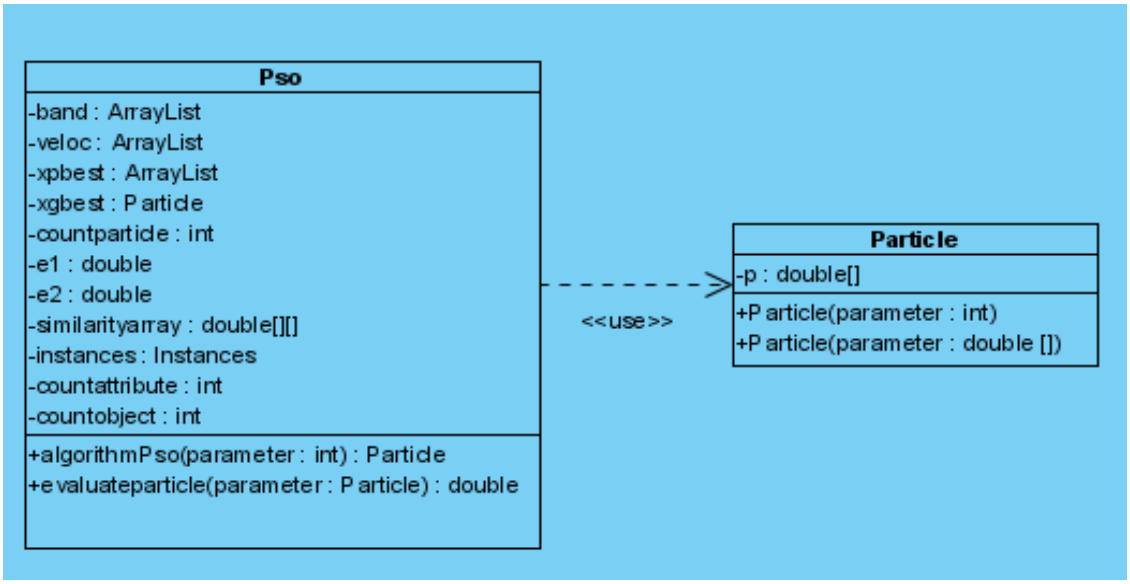
YU, L. & LIU, H. 2004a. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research,* 5**,** 1205-1224.

YU, L. & LIU, H. Year. Redundancy based feature selection for microarry data. *In:* 10th ACM SIGKDD Internacional Conference on Knowledge Discovery and Data Mining, 2004b.

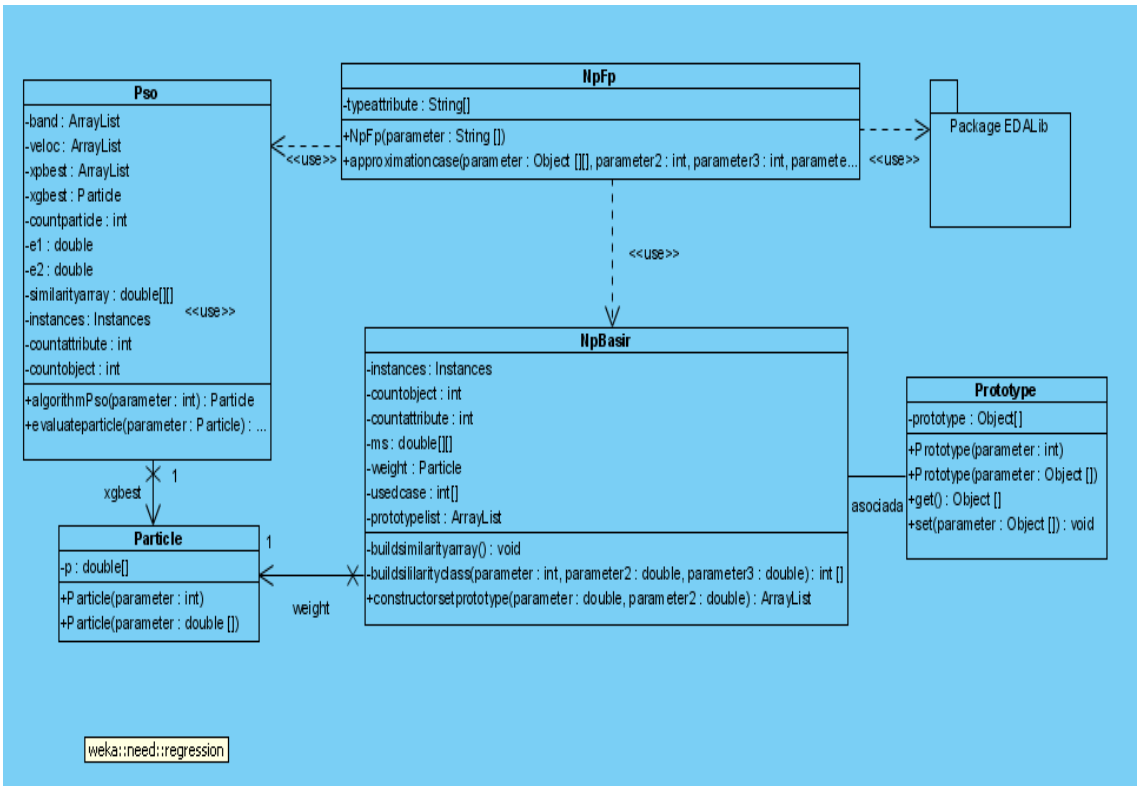
YU, L. & LIU, H. 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. On knowledge and data enginnering,* 17**,** 1-12.

ANEXOS

Anexo 1: Diagrama de clases fundamentales para la implementación de la metaheurística PSO.



Anexo 2: Diagrama de clases fundamentales del paquete *weka.need.regression*.



Anexo 3: Redefinición de los métodos *buildClassifier(), classifyInstance(), listOptions(), setOptions(), getOptions()* haciendo uso del paquete *weka.need.regression*.

```
buildClassifier(Instances data) throws Exception {

        typeattribute = new String[data.numAttributes()];
        for (int t = 0; t < data.numAttributes(); t++) {
            if (data.attribute(t).isNumeric()) {
                typeattribute[t] = "is numeric";
            } else {
                typeattribute[t] = "is nominal";
            }
        }
        if (weight_calculator == PSO) {

System.out.println("..................pso..................");
            Pso pso = new Pso(data, data.numAttributes(),
data.numInstances(), individual_count, epsilon1, epsilon2);
            weight = pso.algorithmPso(generation_count);
        } else if (weight_calculator == UMDA) {

System.out.println("..................umda..................");
            Umda umda = new Umda(data, data.numAttributes(), epsilon1,
epsilon2,              generation_count, individual_count,
selection_threshold);
            weight = umda.calculate_weight();
        } else {

System.out.println("..................none..................");
            int tamp = data.numAttributes() - 1;
            double[] w = new double[tamp];
            for (int wi = 0; wi < w.length; wi++) {
                w[wi] = 1.0 / tamp;
            }
            weight = new Particle(w);
        }
        NpBasir np = new NpBasir(data, data.numInstances(),
data.numAttributes(), weight);
        list = np.constructorsetprototype(epsilon1, epsilon2);
        prototypelist = convert(list);

}

classifyInstance(Instance instance) throws Exception {

        NpFp a = new NpFp(typeattribute);
        double value = a.approximationcase(prototypelist,
prototypelist.length, instance.numAttributes(), instance, weight, k);
        return value;

}

listOptions() {

        Vector newVector = new Vector();
        newVector.addElement(new Option("\tEscoger el calculador de
weight. " +
```

```
                "0 = PSO, 1 = UMDA, 2=NONE" +
                ".\n" +
                "\t(Default 0 = PSO)",
                "C", 1, "-C <calculador de weight>"));
        newVector.addElement(new Option(
                "\tEpsilon 1.",
                "E", 1, "-E <epsilon1>"));
        newVector.addElement(new Option(
                "\tEpsilon 2.",
                "P", 1, "-P <epsilon2>"));
        newVector.addElement(new Option(
                "\tSelection Threshold.",
                "S", 1, "-S <SelectionThreshold>"));
        newVector.addElement(new Option(
                "\tCantidad de ciclos.",
                "A", 1, "-A <cantidad de ciclos>"));
        newVector.addElement(new Option(
                "\tCantidad de ciclos.",
                "N", 1, "-N <cantidad de particulas>"));
        newVector.addElement(new Option(
                "\tCantidad de ciclos.",
                "K", 1, "-K <k semejantes>"));
        return newVector.elements();

}

setOptions(String[] options) throws Exception {

        String weight_calculatorString = Utils.getOption('C',
options);
        if (weight_calculatorString.length() != 0) {
            setWeight_calculator(new
SelectedTag(Integer.parseInt(weight_calculatorString),
TAGS_SELECTION));
        } else {
            setWeight_calculator(new SelectedTag(PSO,
TAGS_SELECTION));
        }
        String epsilon1String = Utils.getOption('E', options);
        if (epsilon1String.length() != 0) {
            setEpsilon1((new Double(epsilon1String)).doubleValue());
        } else {
            setEpsilon1(0.85);
        }
        String epsilon2String = Utils.getOption('P', options);
        if (epsilon2String.length() != 0) {
            setEpsilon2((new Double(epsilon2String)).doubleValue());
        } else {
            setEpsilon2(0.85);
        }
        String selection_thresholdString = Utils.getOption('S',
options);
        if (selection_thresholdString.length() != 0) {
            setSelection_threshold((new
Double(selection_thresholdString)).doubleValue());
        } else {
            setSelection_threshold(0.5);
        }
        String generation_countString = Utils.getOption('A', options);
        if (generation_countString.length() != 0) {
```

```
            setGeneration_count((new
Integer(generation_countString)).intValue());
        } else {
            setGeneration_count(100);
        }
        String individual_countString = Utils.getOption('N', options);
        if (individual_countString.length() != 0) {
            setIndividual_count((new
Integer(individual_countString)).intValue());
        } else {
            setIndividual_count(40);
        }
        String kString = Utils.getOption('K', options);
        if (kString.length() != 0) {
            setK((new Integer(kString)).intValue());
        } else {
            setK(1);
        }
        Utils.checkForRemainingOptions(options);

}

getOptions() {

        String[] options = new String[100];
        int current = 0;
        options[current++] = "-C";
        options[current++] = "" + getWeight_calculator();
        options[current++] = "-E";
        options[current++] = "" + getEpsilon1();
        options[current++] = "-P";
        options[current++] = "" + getEpsilon2();
        options[current++] = "-S";
        options[current++] = "" + getSelection_threshold();
        options[current++] = "-A";
        options[current++] = "" + getGeneration_count();
        options[current++] = "-N";
        options[current++] = "" + getIndividual_count();
        options[current++] = "-K";
        options[current++] = "" + getK();
        while (current < options.length) {
            options[current++] = "";
        }
        return options;

}
```