Metodología para la Optimización de la Estimación del Desarrollo de Software Utilizando Redes Neuronales

A. García, I. González, R. Colomo, J.L. López and B. Ruiz

***Abstract*— One of the most important tasks of a software development project manager is to produce accurate time and effort estimations. Improving the estimation accuracy is a widely recognized benefit for several software development processes. In order to achieve these objectives, there are proposals based on Artificial Intelligence techniques and specifically artificial neural networks. This paper proposes an optimization methodology for searching the best neural model applicable to the effort estimation of software projects. This will draw on a set of known factors in the early stages of development, outside the complex calculation of function points, which would entail a high level of maturity and definition of the project. This methodology has allowed, on the one hand, ensure the adequacy of the proposed neural network model and, on the other hand, optimize the performance, both in time and accuracy.**

***Keywords*—Neural Networks, Software Engineering, Effort Estimation, Optimization Methodology.**

I. Introducción

El SOFTWARE se ha convertido en un elemento fundamental en la vida de los ciudadanos y las empresas. Algunos investigadores (e.g. [1], [2]) tildan de esencial o crítico esta importancia. Así, en [3] se indica que el desarrollo de software representa la mayor partida presupuestaria dentro de un presupuesto de un departamento de sistemas. Teniendo en cuenta esta importancia, contar con planes ajustados en tiempo, esfuerzo, costes y calidad representa una necesidad para los gestores de proyecto de desarrollo de software en todo el globo [4], además de un objetivo [3], o incluso un reto [5].

Una estimación inadecuada de los esfuerzos y tiempos en un proyecto de desarrollo de software decrementa la eficiencia del final del proyecto, pudiendo, incluso resultar en una imposibilidad de alcanzar los objetivos globales del mismo [6]; no en vano todas actividades del desarrollo, la asignación de recursos, el precio, la planificación y el seguimiento del proyecto la dependen de la estimación realizada [7]. Una estimación inexacta puede materializarse en una infraestimación o una sobreestimación. La primera puede tener efectos nocivos sobre la calidad del software, del proceso, del personal y, de forma eventual, afectar a la reputación de la compañía debido a la falta de cumplimiento de los plazos; la sobreestimación, por su parte, puede suponer una falta de competitividad de la empresa debido a sus altos costes y a la imposibilidad de utilizar recursos asignados incorrectamente a otro proyecto [8], [9]. Adicionalmente, e incrementando el alcance del fallo, los problemas de estimación pueden suponer riesgos para el portfolio de proyectos [10]. Por el contrario, una estimación adecuada proporciona una poderosa ayuda en el momento en el que las decisiones relativas a la gestión del software son tomadas [11], las etapas iniciales del proceso de desarrollo de software, allí donde se aprueban los presupuestos [12]. Así, atendiendo a [13], una mejora en la predicción puede reducir los costes de los errores de planificación de forma drástica, contribuyendo de esta forma a la mejora de los procesos de software [14], que, de acuerdo a [15], constituyen el proceso de negocio más importante dentro de la industria del desarrollo de software. Por último, las estimaciones defectuosas son, atendiendo a un estudio empírico realizado por Chen y Huang [16], uno de los factores claves que afectan a la mantenibilidad del software.

De acuerdo a [9], los cuatro pasos básicos para una estimación de un proyecto software son:

1. Estimar el tamaño del producto (en líneas de código, puntos de función u otras unidades de medida).
2. Estimar el esfuerzo en personas-mes (u otras medidas).
3. Estimar la duración del proyecto.
4. Estimar el coste.

Este trabajo se centrará en la estimación del esfuerzo a partir de un conjunto de factores conocidos en las fases iniciales del desarrollo, ajeno al cálculo de los puntos de función que conllevaría un nivel de definición y maduración del proyecto elevado. Trabajos sobre la estimación del esfuerzo en proyectos de desarrollo de software se pueden encontrar en los últimos cuarenta años [17], existiendo, a resultas de esta profusión, multitud de métodos conocidos. Así, los citados métodos se pueden clasificar en las siguientes categorías:

a) Juicio de expertos. Esta técnica [18], [19], a partir de

A. García, Universidad Carlos III de Madrid, Departamento de Informática, Leganés 28911, Madrid, angel.garcia@uc3m.es

I. González, Universidad Carlos III de Madrid, Departamento de Informática, Leganés 28911, Madrid, israel.gonzalez@uc3m.es

R. Colomo, Universidad Carlos III de Madrid, Departamento de Informática, Leganés 28911, Madrid, ricardo.colomo@uc3m.es

J.L. López, Universidad Carlos III de Madrid, Departamento de Informática, Leganés 28911, Madrid, joseluis.lopez.cuadrado@uc3m.es

B. Ruiz, Universidad Carlos III de Madrid, Departamento de Informática, Leganés 28911 Madrid, belen.ruiz@uc3m.es

un conjunto de expertos en la materia, permite obtener estimaciones de forma sencilla y sin necesidad de complementos técnicos adicionales. Sin embargo, debido a su naturaleza humana y subjetiva, resulta a veces también difícil de repetir [10].

b) Estimación por analogía. Esta técnica se ha propuesto desde tiempo atrás [20] como alternativa a los métodos algorítmicos y al juicio de expertos [8]. La estimación por analogía compara el proyecto con uno o varios proyectos anteriores de características similares para llevar a cabo la estimación [21]. De acuerdo a [8], este método es una manera sistemática de afrontar el juicio de expertos.

c) Métodos algorítmicos. Son probablemente los métodos más conocidos y tratados en la literatura [9]. Los métodos algorítmicos implican el uso de métodos de análisis estadístico para establecer ecuaciones de cálculo de esfuerzo [8]. Ejemplos de estos sistemas son COCOMO [22], [23] y SLIM [24].

Con el propósito de ampliar las capacidades de los métodos algorítmicos, así como para manejar de forma más correcta las características dinámicas de los entornos (en muchos casos presentando relaciones no lineales y altamente complejas [11]), muchos investigadores han aplicado técnicas de inteligencia artificial, tales como redes neuronales o modelos de lógica difusa, para llevar a cabo estimaciones [6].

Siguiendo esta línea de investigación en este trabajo se presenta una metodología para la optimización de la estimación del esfuerzo del desarrollo de software utilizando una de las técnicas expuestas: las redes neuronales.

El resto del artículo está organizado de la siguiente manera. El apartado segundo repasa el estado del arte relevante de la literatura incluyendo elementos relativos a las redes de neuronas y su aplicación en entornos de estimación del desarrollo del software. El epígrafe titulado "Descripción de la solución" presenta la aportación principal del presente trabajo incluyendo descripciones sobre el diseño, la arquitectura y la implementación de la herramienta desarrollada. El cuarto epígrafe muestra la validación de la propuesta tomando como base un conjunto de datos disponibles relativos a proyectos de desarrollo de software. Finalmente, el último apartado resume el trabajo y plantea los trabajos futuros que emanan del mismo.

II. Estado del arte

A. Redes de Neuronas (RNA)

La teoría y modelado de las Redes de Neuronas Artificiales (RNA) está inspirada en la estructura y funcionamiento de los sistemas nerviosos biológicos donde la neurona es el elemento fundamental. Las bases del sistema nervioso, tal y como se conocen en la actualidad, fueron fijadas por el científico español Santiago Ramón y Cajal, el cual a lo largo de su carrera investigadora consiguió demostrar los mecanismos que gobiernan la morfología y los procesos conectivos de las células nerviosas. Según estas premisas, el cerebro humano continuamente recibe señales de entrada de muchas fuentes y las procesa para crear una salida apropiada. El cerebro cuenta con millones de neuronas que se interconectan para elaborar redes neuronales. Estas redes ejecutan los millones de instrucciones necesarias para mantener una vida normal [1]. Dentro de las redes, las neuronas son las células que forman la corteza cerebral de los seres vivos, cada una está formada por varios elementos llamados cuerpo, axón y dendritas.

Las RNA son un paradigma de aprendizaje y procesamiento automático, inspirado en el funcionamiento del sistema nervioso biológico [33]. Son además consideradas como un modelo matemático, compuesto por un gran número de elementos o unidades de procesamiento, llamados neuronas, que trabajan al mismo tiempo para encontrar solución a problemas específicos. En cuanto a su estructura se trata de un sistema de interconexión de neuronas, organizado en red y distribuido en niveles, que colaboran para producir un estímulo de salida. Por último, y aunque existen numerosos tipos de RNA, clasificadas en función de diversas características como la topología, el tipo de aprendizaje o la clase de entrada que reciben, las de tipo Perceptrón MultiCapa (PMC) entrenadas con el algoritmo de RetroPropagación (RP) han sido las más utilizadas dentro de las literatura. Esto es debido a su carácter de aproximador universal de funciones junto a su facilidad de uso y aplicabilidad respecto de otras arquitecturas mucho más complejas [34]. Gracias a esto el PMC se ha aplicado con éxito a la resolución de problemas en campos tan diversos como el reconocimiento del habla, el control de procesos, la conducción de vehículos, predicción de series temporales o los diagnósticos médicos.

B. Uso de las RNA en ámbitos de estimación del desarrollo de software

Recientes investigaciones en el ámbito de la Inteligencia Artificial han demostrado su aplicabilidad en el dominio de la Ingeniería del software y más precisamente en el campo de la estimación [3]. Los resultados de sus trabajos demuestran una consistencia notable y una exactitud notable en sus resultados. En el ámbito concreto de las RNA y la estimación relativa al desarrollo de software, los trabajos que se pueden encontrar en la literatura son considerablemente profusos. Diversas predicciones relativas al desarrollo de software se encuentran ligadas a las redes neuronales. Así, en el trabajo de [5] se puede encontrar un ejemplo de la regularización bayesiana de una RNA para estimar el tamaño del software. La propuesta de [25] propone la estimación del coste del desarrollo software utilizando RNA de tipo Wavelet o la fiabilidad del software [26] utilizando modelos combinados, citando algunos de los trabajos más representativos.

Particularizando más el ámbito de uso de las RNA en la predicción y estimación del esfuerzo de los proyectos de desarrollo de software, éstas suponen también un prolífico ámbito de estudio. En este caso, la combinación de este tipo de técnicas de inteligencia artificial con otros métodos ha sido empleada con profusión: algoritmos genéticos [27], lógica difusa [12], [7] o el método de las analogías [8]. Trabajos genuinos de RNA también se encuentran disponibles en la

literatura, destacando entre los trabajos [28] y [9]. En el ámbito de la validación de RNA, los estudios de [6] y [29] basados en puntos de función se puede considerar un antecedente válido para el trabajo que se presenta en el presente artículo. Sin embargo, si bien estas investigaciones demostraron una dirección prometedora a seguir en el software de estimación de costes, estos enfoques basados en redes neuronales o en lógica difusa están lejos de su madurez [30]. Como aportación, el presente trabajo se centra en la estimación del esfuerzo de un proyecto mediante un sistema de RNA, pero implementando una metodología orientada a asegurar la idoneidad del modelo neuronal propuesto y optimizar el rendimiento final del mismo, tanto en tiempo como en precisión.

III. Descripción de la solución

De acuerdo a [9], uno de los pilares básicos para una correcta estimación de un proyecto software es la estimación del esfuerzo asociado a su desarrollo. Por tanto disponer de esta información a priori facilitará al líder del proyecto una gestión más eficiente del mismo tanto a nivel de costes como de plazos, facilitando la distribución de recursos y disminuyendo los riesgos o zonas críticas.

Hasta el momento, y en muchas ocasiones, las estimaciones se hacen valiéndose de la experiencia pasada como única guía. Si un proyecto es bastante similar en tamaño y funcionamiento a otro anterior, es probable que el nuevo proyecto requiera aproximadamente la misma cantidad de esfuerzo y que dure aproximadamente lo mismo. Desafortunadamente si el proyecto es totalmente distinto entonces puede que la experiencia obtenida no sea suficiente, por lo que se hace necesaria otra forma de conseguir obtener una estimación fiable. Las métricas algorítmicas expuestas al comienzo de este artículo presentan problemas de aplicación en ciertos dominios dinámicos donde se requiere una alta adaptación y flexibilidad [34].

Partiendo de estas hipótesis se ha diseñado un sistema basado en RNA que permitirá predecir el esfuerzo que conlleva un determinado proyecto partiendo de una serie de características o parámetros significativos. Se ha recurrido a estas estructuras computacionales basadas en el aprendizaje automático ya que permiten una inducción del conocimiento, i.e. son capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Además debido a su flexibilidad pueden aplicarse en una amplia variedad de dominios, de alta complejidad y no linealidad, gracias a la variedad de alternativas de diseño que aparecen durante la creación de un modelo neuronal. Sin embargo, esto a veces representa un inconveniente, ya que la gran cantidad de posibilidades y la falta de directrices y guías de uso general hace que el diseñador realice una toma de decisiones a veces arbitraria o basada en técnicas de fuerza bruta. En algunos casos se han propuesto una serie de aproximaciones teóricas que pueden facilitar en parte este proceso, pero que no se consideran soluciones analíticas al no poder aplicarse en todos los casos [51].

Por ello, se ha planteado una metodología de optimización orientada a guiar la búsqueda del mejor modelo neuronal para el problema estudiado y en consecuencia mejorar el rendimiento, tanto en tiempo como en precisión. Para homogeneizar el gran número de alternativas existentes y simplificar su aplicación éstas se han agrupado en tres grandes bloques o fases siguiendo la premisa *modelo = patrones + arquitectura + algoritmo.* En cada uno de los términos de la ecuación se estudiarán diferentes técnicas y métodos orientados a mejorar el rendimiento final.

A. Utilización de modelos neuronales para la estimación del desarrollo software.

1) Parámetros que definen el escenario

Este trabajo se centra en la estimación del esfuerzo final de un proyecto a partir de una serie de factores característicos del mismo. Para obtener una descripción fiable se ha recurrido a un subconjunto de datos adaptado del conjunto original obtenido de la organización International Software Benchmarking Standards Group[1] (ISBSG).

Para esta realizar esta investigación se ha filtrado el conjunto de características disponibles, excluyendo aquellas referentes al cálculo de puntos de función lo que necesita un nivel de definición elevado del proyecto. Por tanto se incluyen únicamente aquellas que aportan información del proyecto pero sin necesidad de una definición completa del sistema en cuestión. En consecuencia los patrones disponibles se corresponden con datos de 214 proyectos caracterizados por los factores mostrados en la Tabla I. Una descripción más detallada de cada uno de los campo se puede encontrar en [36].

TABLA I. Listado de variables de entrada y salida empleadas

	Característica	***Acrónimo***
Entradas (V_{ent})	Calidad de los Datos (Data Quality Rating)	DQ
	Nivel de Recursos (Resource Level)	RL
	Tipo de Desarrollo (Development Type)	DT
	Plataforma de Desarrollo (Development Platform)	DP
	Tipo de Lenguaje (Language Type)	LT
	Utiliza Sistema Gestor de Base de Datos (DBMS Used)	DB
	Emplea herramientas CASE (Upper CASE Used)	UC
	Duración del proyecto en meses (Project Elapsed Time)	ET
	Líneas de Código (Lines of code)	LC
Salida (V_{sal})	Esfuerzo Final en horas (Summary Work Effort)	SW

2) Estructura de la red neuronal

Para el diseño de la topología de la red, la arquitectura elegida es un PMC de tres niveles o capas. Según los estudios de Lippman, este tipo de estructura permite resolver la mayoría de los problemas, siendo capaz de formar

[1] http://www.isbsg.org/

cualquier límite complejo y aleatorio de decisión [37]. El número de entradas en un PMC viene determinado por el número de características que definen el problema en cuestión, mientras que el número de salidas es el mismo que el número de valores a predecir. Por tanto el único elemento dependiente del diseñador de la red es el número de capas ocultas y el número de neuronas de estas capas. Sin embargo, no hay un método o heurística de uso universal para determinar el número óptimo para resolver un problema dado. En esta investigación se ha elegido la regla matemática descrita por Tarassenko para aproximar el valor de los elementos ocultos [38]:

$$J = \sqrt{IK} = \sqrt{9 \times 1} = 3 \quad (1)$$

donde I, J y K son el número de neuronas de las capas de entrada, oculta y salida respectivamente.

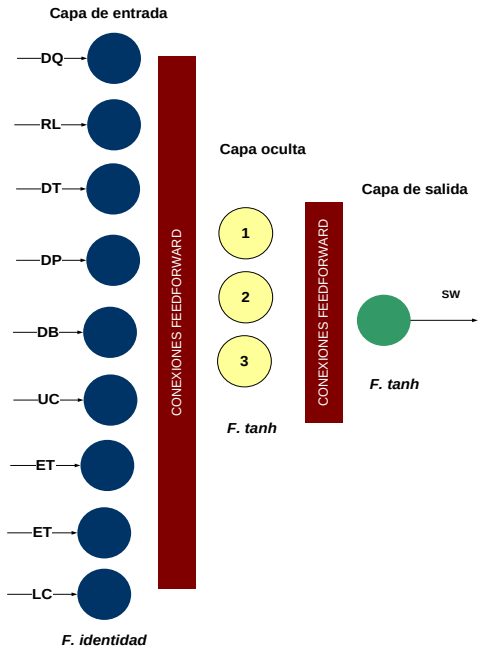

Figura 1. Estructura del PMC base diseñado.

Otro aspecto relativo a la arquitectura de la red, que es necesario fijar antes de empezar su entrenamiento, es la función de activación asociada a cada neurona. Típicamente todas las neuronas de una capa comparten la misma función de activación, siendo requisito imprescindible al utilizar el algoritmo de aprendizaje que la función elegida sea derivable. Sin embargo las ventajas asociadas a cada una de las posibles alternativas no han sido demostradas matemáticamente [42]. Por ello, durante la metodología de optimización se estudiará la influencia de las diversas alternativas dentro de cada escenario. Inicialmente se ha utilizado una función identidad para la capa de entrada y la función tangente hiperbólica en las capas oculta y de salida. En resumen, la arquitectura de red diseñada como punto de partida se muestra en la Fig. 1 y contiene las siguientes características:

- Nueve neuronas en la capa de entrada correspondientes a los elementos del vector de entradas V_{ent} con función de transferencia identidad.
- Tres neuronas en la capa oculta con función de transferencia tangente hiperbólica.
- Una neurona en la capa de salida asociada con la variable a predecir, el esfuerzo. La función de transferencia elegida es la función de transferencia tangente hiperbólica

3) Datos de entrada

El siguiente paso necesario es determinar el conjunto de datos de entrada que va a ser utilizado como parte del proceso de aprendizaje. Es importante hacer una estimación de la información necesaria para formar a la red de una manera adecuada. Para ello, es esencial disponer de una serie de datos lo suficientemente grande como para poder asegurarlo. Mientras se entrena la red, es necesario reconocer y responder a la amplia gama de condiciones que pueden aparecer durante el proceso de validación. Si el número de datos es demasiado pequeño, la completa gama de conexiones que la red neuronal debe aprender no será cubierta.

Del conjunto de patrones disponible, 214 datos de proyectos, una parte de ellos será asignada al conjunto de entrenamiento y el resto al conjunto de validación o test. Como distribución de partida se elegirá la clásica 50/50 para train/test. La búsqueda de la distribución óptima de los patrones se realizará mediante el estudio matemático expuesto Boonyanunta y Zeephongsekul en [44] y las tesis comentadas por Crowther y Cox en [45]. De acuerdo con las directrices dictadas en [47],[48] y [38], el número total de patrones disponible puede ser suficiente para obtener resultados satisfactorios. Sin embargo, en los trabajos de González et al. [49][50] se demostró que existen una serie de heurísticas encaminadas a maximizar y optimizar el uso que un PMC realiza sobre los patrones disponibles en escenarios con información reducida.

4) Algoritmo de aprendizaje

Para mejorar el rendimiento del PMC, el tercer componente que se puede ajustar es el algoritmo de aprendizaje. En este caso como punto de partida se ha elegido el algoritmo RP clásico al tratarse del más extendido dentro de las redes PMC y su fácil adaptación a un amplio abanico de problemas. Además para corregir las carencias del algoritmo RP: tasa de convergencia lenta y caída en mínimos locales, se han incluido los parámetros tasa de aprendizaje (η) y el coeficiente momentum (μ). El inconveniente asociado a estos parámetros es que no existe una regla general para determinar los mejores valores de η y μ. Sin embargo, varios investigadores han desarrollado diversas heurísticas para su aproximación. En este caso, se han seguido las propuestas por Wythoff [5], Swingler [6] y Principe et al. [43]. Siguiendo estas pautas, los valores de η y μ deben estar en el intervalo [0,1], siendo conveniente y habitual utilizar valores altos para

acelerar el aprendizaje y evitar la inestabilidad. Los valores iniciales de η y μ se han aproximado a partir del trabajo de Principe et al.: 0,7 y 1 para la capa oculta y 0,3 y 0,4 para la capa de salida, respectivamente. El modo de entrenamiento seleccionado fue el modo batch, ya que presenta un comportamiento equilibrado entre la precisión y velocidad [6]. En este caso, la actualización de los pesos sinápticos se retrasa hasta que han sido presentados todos los patrones, i.e., al finalizar cada ciclo o epoch.

Por último, el criterio de parada, para todas las alternativas estudiadas, fue para realizar una serie de epochs dentro del rango [1000-81000] junto con alcanzar un valor umbral para el error MSE, en este caso 0,01. De esta manera, el proceso de aprendizaje no utiliza un conjunto de validación, ya el entrenamiento se detiene al alcanzar un número de epochs determinado o valor mínimo de MSE y no por el error de validación cruzada.

A. Metodología de optimización basada en modelos neuronales.

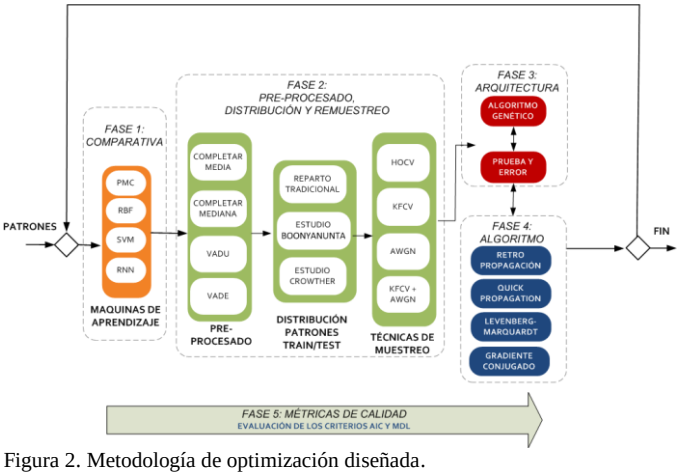

Figura 2. Metodología de optimización diseñada.

Partiendo de la premisa modelo neuronal = patrones + arquitectura + algoritmo y teniendo en cuenta el elevado número de parámetros, técnicas y métodos disponibles para entrenar una RNA, se hace necesaria un serie de pasos para guiar, estructurar y priorizar las diferentes alternativas existentes. Además muchas de estas alternativas no tienen justificación teórica o matemática por lo que son dependientes del problema lo que implica que el rendimiento de una RNA depende de una gran cantidad de factores que necesitan ser probados y evaluados tanto de forma individual como conjunta.

Por ello, la optimización de los modelos neuronales se llevará a cabo mediante la definición de una metodología de investigación, basada en una serie de fases y tareas orientadas a mejorar su rendimiento, tanto en tiempo como en precisión. Para homogeneizar el gran número de alternativas existentes y simplificar su aplicación, éstas se han agrupado en tres grandes bloques o fases: patrones, arquitectura y algoritmo. En cada uno de ellos se incluyen las variantes que según la literatura analizada mejor se adaptan a problemas ingenieriles similares al tratado en este estudio.

Este estudio describe el proceso de construcción y optimización de sistema basado en RNA, y su aplicación en el ámbito de la estimación del desarrollo de software para la predicción esfuerzo asociado a un proyecto. El objetivo fundamental es mejorar el rendimiento de un modelo neuronal buscando la mejor configuración posible para el problema estudiado. Para lograr este objetivo, se ha definido un proceso divido en cinco fases que se reflejan en la Fig. 2.

La primera fase tiene como objetivo responder a la pregunta de porque se ha utilizado un PMC y no otra alternativa basada en aprendizaje automático. Para ello se estudiarán y comparará el rendimiento de varias propuestas dentro del dominio de la investigación. La segunda fase se centra en las diferentes técnicas y métodos que permiten realizar maximizar el uso de los datos disponibles por parte del modelo neuronal. La tercera fase permitirá determinar la mejor arquitectura de red mediante la aplicación de un Algoritmo Genético (AG) para localizar la configuración de elementos ocultos, capas y neuronas, para el problema estudiado. La cuarta etapa estudiará la influencia de diferentes algoritmos de aprendizaje, de primer y segundo orden. Por último la quinta fase tiene como meta asegurar la calidad de los resultados mediante la introducción de una serie de indicadores basados en la teoría de la información.

El procedimiento seguido en cada simulación realizada dentro de este proceso de optimización se describe a continuación:

1. Elegir la mejor alternativa de máquina de aprendizaje dentro del dominio estudiado.
2. Comparar diferentes técnicas de pre-procesado y muestreo de los patrones y estudiar la mejor distribución de los mismos.
3. Localizar la mejor configuración de elementos ocultos (aplicando un algoritmo genético)
4. Comparar diferentes algoritmos de aprendizaje (se repite 15 veces con los pesos inicializados aleatoriamente).
5. Calcular en los pasos anteriores los criterios de calidad necesarios para asegurar la validez de los resultados.

En resumen, gracias a esta metodología, se posibilita la sistematización de los diferentes métodos y técnicas existentes a la hora de configurar y parametrizar un modelo neuronal basado en una topología PMC y algoritmo RP.

1) Fase 1. Comparativa de alternativas al PMC.

Esta etapa tiene como objetivo dilucidar cuál es la mejor solución dentro de las alternativas computacionales basadas en la teoría de las máquinas de aprendizaje. Para ello, se realizará un estudio comparativo que permita asegurar la viabilidad de la utilización de un PMC para este dominio concreto. Las alternativas que se han incluido para esta fase son las siguientes:

- Red de Función de Base Radial (Radial Basis Function Network o RBFN). Se trata de un tipo de RNA híbrida y no lineal que utiliza funciones de base radial como funciones de activación. Se emplean en diversos ámbitos como la aproximación de funciones, clasificación de patrones y la predicción de series temporales. Su arquitectura estándar suele estar formada por tres capas: una de entrada, una oculta

con una función de activación no lineal de base radial y una capa de salida lineal. En su forma básica la norma empleada es la distancia euclídea y la función base empleada es la Gaussiana. Las funciones de base radial son funciones cuya salida depende de la distancia a un punto denominado centro [54][55].

- Máquinas de Vectores de Soporte (Support Vector Machines o SVM). Se utilizan con frecuencia tanto para clasificar patrones como para resolver problemas de regresión no lineal. La idea principal asociada a un SVM es construir un hiperplano como dimensión de decisión que maximice el margen de separación entre los ejemplos positivos y negativos dentro del conjunto de datos. Este principio de inducción está basado en el hecho de que el coeficiente de error sobre el conjunto de test, i.e. el coeficiente del error de generalización, está limitado por la suma de los coeficientes del error de entrenamiento. Este término depende de la dimensión Vapnik-Chervonenkis [56]. Un SVM facilita una generalización precisa para problema de clasificación de patrones a pesar de no incorporar un dominio de conocimiento del problema.
- Red de Neuronas Recurrente (Recurrent Neural Network o RNN). En estas redes existen dos tipos de conexiones, las de tipo feedforward (hacia delante) y las de tipo (hacia atrás), que permiten que la información se propague en las dos direcciones, desde las neuronas de entrada hacia las de salida y viceversa. Las RNNs han atraido la atención de los investigadores durante los últimos años, especialmente en el campo de la predicción de series temporales, la identificación dinámica de sistemas y la clasificación temporal de patrones [57].

2) Fase 2. Preprocesado, distribución de la información y remuestreo.

Esta segunda fase se centra en primer lugar en realizar pre-procesamiento de los patrones para asegurar la plenitud de los datos empleados. En esta investigación el término "datos incompletos" hace referencia a la no disponibilidad de determinada información en los sujetos que componen la muestra. Para solventar estas situaciones se han estudiado diferentes métodos alternativos de imputación directa de valores ausentes [58][59]: la media, la mediana, un valor aleatorio de distribución de probabilidad uniforme (VADU) o un valor aleatorio de distribución de probabilidad estimada para la variable (VADE).

En segundo lugar, uno de los apartados que más atención ha recibido por la comunidad investigadora ha sido el estudio de la relación existente entre la capacidad de generalización de la red, la cantidad de datos disponibles y su distribución en los conjuntos de entrenamiento y test [46]. Para asegurar una correcta distribución de los patrones entre cada conjunto se empleará el modelo matemático detallado en [44] lo que permitirá explicar la relación entre el tamaño del conjunto de entrenamiento y la capacidad predictiva de la red. A raíz de este estudio y del realizado en [45] se establece que la capacidad de predicción de una RNA aumenta hasta un valor umbral. A partir de este punto no compensa la utilización de nuevos patrones, y por tanto su obtención, ya que la reducción del error es mínima y conlleva un mayor coste de computación durante el entrenamiento.

Por último, una de las características más atrayentes de estas estructuras computacionales es su capacidad de aprender a partir de ejemplos o de experiencias anteriores. Sin embargo esto en algunos dominios representa un inconveniente ya que no siempre se puede disponer de toda la información necesaria para realizar un entrenamiento correcto, lo que dificulta su utilización. En el dominio de esta investigación, el proceso de recogida y etiquetado de los datos es en algunas circunstancias complejo, por lo que la cantidad de datos disponible para el gestor puede ser limitada. Por otra parte, para realizar un entrenamiento adecuado y su posterior validación requiere dividir los datos en diferentes conjuntos lo que reduce aún más la cantidad de datos. Para solventar esta carencia, se han desarrollado varias técnicas estadísticas que se basan en el remuestreo múltiple de los datos. En esta investigación se han seguido las directrices propuestas por Priddy y Keller para hacer frente a cantidades limitadas de datos [51] y las normas dictadas por Bishop [52] y Holmstrom y Koistinen [53] para la generación de ruido aditivo.

La comparativa realizada en esta fase incluye las técnicas de validación cruzada Hold-Out (HOCV) y K-fold (KFCV), la introducción de ruido blanco gaussiano (AWGN) y la técnica combinada de ruido y validación k-fold (KFCV+AWGN). Una explicación detallada de estas alternativas y un ejemplo de su utilización sobre modelos neuronales con información reducida puede encontrarse en [49].

3) Fase 3. Arquitectura de red.

Si se analiza la manera en que la red realiza su trabajo, se puede ver que el tamaño de la misma (también conocido como complejidad del modelo) presenta una gran relación con el rendimiento: muy pocos grados de libertad (pesos) afectan la capacidad de la red para lograr un buen ajuste sobre la función objetivo. Si la red es demasiado grande, sin embargo, no generalizará correctamente, porque el ajuste es demasiado específico sobre el conjunto de datos de entrenamiento (memorización de los datos). Una red de tamaño intermedio es siempre la mejor opción. Por lo tanto, para lograr un rendimiento óptimo, es indispensable incluir en la metodología de diseño del PMC algún método de control de la complejidad.

El problema del tamaño de la red se puede describir de forma simplificada adaptando el postulado de Occam[2] a este dominio: *"Cualquier máquina de aprendizaje automático debe ser suficientemente grande para resolver un problema dado, pero no más grande".*

Para resolver la cuestión de que arquitectura es la más idónea para el problema estudiado, se ha incluido una heurística basada en AG que tiene como objetivo localizar la mejor configuración de elementos ocultos, tanto de capas

[2] En su forma original *"entia non sunt multiplicanda praeter necessitatem"* o *"no ha de presumirse la existencia de más cosas que las absolutamente necesarias"*

como de neuronas. Los AG son métodos sistemáticos para la resolución de problemas de búsqueda y optimización, que aplican a éstos los mismos métodos de la evolución biológica: selección basada en la población, reproducción y mutación.

En esta investigación el criterio usado para evaluar la capacidad de cada solución potencial es el coste más bajo obtenido durante el entrenamiento. Los criterios y valores usados se han elegido o fijado en función de las pautas dictadas por Lefebvre et al. [60] y su aplicación en problemas prácticos, como la optimización de sistemas eléctricos [61]. Los valores empleados para la configuración del AG se muestran en la Tabla II.

Para ratificar las conclusiones obtenidas por el AG, en este caso se ha optado por técnicas de fuerza bruta, i.e. prueba y error de diferentes combinaciones de arquitecturas de red.

TABLA II. Parámetros del AG empleado

Población	50
Generaciones máximas	100
Tiempo máximo de evolución (minutos)	60
Criterio Finalización	Umbral
Criterio Selección	Ruleta
Crossover	De un punto
Crossover	0.9
Probabilidad Mutación	0.01

4) Fase 4. Algoritmo de aprendizaje.

El entrenamiento de las RNA se basa en la minimización de una función del error E al variar el conjunto de pesos que definen la arquitectura de la red:

$$\frac{\partial E}{\partial W} \equiv 0 \Leftrightarrow \nabla E(w^*) = 0 \qquad (2)$$

Se trata por tanto de una optimización multivariable sin restricciones ya que no hay ningún otro requisito adicional con respecto a la función o a las variables de entrada. El problema de optimizar una función derivable, continua, multivariable y sin restricciones ha sido estudiado ampliamente fuera del campo de las RNA, y las distintas aproximaciones que se han seguido pueden ser aplicadas casi directamente al problema de minimización del error. Sin embargo, hay dos aspectos a tener en cuenta en este problema específico y que marcan la diferencia respecto a las técnicas convencionales de optimización: el alto número de variables a optimizar en este caso y la gran cantidad de derivadas que hay que calcular en cada paso de la iteración, que hacen los cálculos especialmente costosos.

El algoritmo RP clásico calcula el Jacobiano, matriz formada por las derivadas parciales de primer orden de la función del error, con respecto a los pesos para localizar la mejor dirección hacia donde ajustarlos. Posteriormente se aplica una actualización de los pesos constante en dicha dirección.

Por otro lado las técnicas de segundo orden de gradiente por su parte emplean el Hessiano, matriz cuadrada de las segundas derivadas parciales del error, con respecto a los pesos para adaptar el tamaño de paso, step-size, en la dirección de la actualización óptima:

$$\frac{\partial^2 E}{\partial W^2} \equiv 0 \qquad (3)$$

Dentro de las variantes de segundo orden destaca el algoritmo de propagación rápida o QuickPropagation (QP). Se trata de una variación del algoritmo RP estándar, y fue desarrollado por Fahlman en 1988 [62]. Este asume que la superficie de error tiene un comportamiento local cuadrático y emplea una aproximación a la derivada de segundo orden para actualizar los pesos, lo que acelera la búsqueda. El cómputo de la ecuación asociada la minimización del error, es similar al del algoritmo RP, pero al tratarse de una variante de los métodos de Newton la actualización de los pesos se realiza mediante derivadas de segundo orden y no empleando el método del descenso del gradiente. Se ha demostrado que este algoritmo es más rápido que el RP para algunas aplicaciones, pero no en todos los escenarios. Al igual que éste, también puede quedar atrapado en mínimos locales y presenta los mismos problemas de inestabilidad. Por tanto, aunque no se le considera un método de propósito general, si es especialmente útil para entrenar de forma rápida una RNA en ciertos dominios.

El algoritmo de Levenberg-Marquardt (LM) es otro algoritmo de optimización no lineal basado en el uso de derivadas de segundo orden [63][64]. Se trata de la variante más utilizada en RNA dentro de las diferentes variaciones del método de Newton, siendo un algoritmo iterativo basado en los multiplicadores de Lagrange. Fue diseñado para minimizar funciones que sean la suma de los cuadrados de otras funciones no lineales, por lo que puede utilizar en aplicaciones de estimación. Por ello el algoritmo LM tiene un excelente desempeño en el entrenamiento de RNA donde el rendimiento de la red esté determinado por el error cuadrático medio. Con el método anterior comparte su escalado en función del número de pesos, por lo que su empleo queda restringido a redes pequeñas, del orden de pocos cientos de conexiones. El algoritmo LM es una combinación de las características del gradiente descendiente, extraídas del algoritmo RP y el algoritmo de Newton. Asume que la función subyacente es lineal y que su error mínimo se puede encontrar en un solo paso.

Por último, el método del Gradiente Conjugado (GC) fue inventado por Hestenes y Stiefel en 1952 [65] y desde su aparición se han producido múltiples variaciones. Como se trata de un técnica de optimización, se puede aplicar durante el entrenamiento de una RNA para adaptar los valores de los pesos de igual forma que lo hace el algoritmo RP. Entre sus ventajas destaca la posibilidad de trabajar sobre con un elevado número de pesos, al contrario que la versión clásica del algoritmo RP. Se trata de un método intermedio entre los métodos del gradiente clásicos y los métodos de Newton, ya que por una parte intentan mejorar la convergencia de los métodos del descenso del gradiente y por otro evitan el cálculo y el almacenamiento de la información requerida en un método de Newton. La característica principal de este tipo de métodos

es que, asumiendo que la función de error es cuadrática, intentan conseguir direcciones de búsqueda que no interfieran con las realizadas en las iteraciones anteriores. Este tipo de métodos suelen requerir el doble del cálculo de gradientes que los métodos de Quasi-Newton (QN), pero son capaces de reducir el tiempo y la memoria requerida para determinar la matriz del Hessiano para problemas de altas dimensiones.

5) Fase 5. Métricas de calidad

Esta última fase tiene como objetivo determinar la validez y calidad de los resultados obtenidos. Se trata por tanto de una fase transversal a la que se recurrirá en diferentes puntos de la metodología para contrastar y evaluar la precisión y rendimiento de cada unas de las propuestas estudiadas.

El PMC diseñado se ajusta a un escenario de regresión, i.e. predice el valor numérico de una salida en función de un conjunto de entradas. En este caso se calcula para la variable de salida el coeficiente de correlación lineal (r) que existe entre los valores reales de esa variable y los obtenidos por la red neuronal. Este coeficiente permite conocer el grado de similitud y de exactitud de la respuesta basándose en la ecuación:

$$r = \frac{\sum_i x_i - \bar{x}(d_i - \bar{d})}{\sqrt{\frac{\sum_i (d_i - \bar{d})^2}{N}} \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{N}}} \quad (4)$$

donde el numerador se corresponde con la covarianza y el denominador con el producto de las desviaciones típicas de las variables x (valor obtenido) y d (valor esperado). Además para encontrar una media y desviación de la medida fiable se repetirá cada experimento 20 veces con diferentes patrones en cada conjunto y valores aleatorios para los pesos. Con esto se asegura la independencia de los resultados respecto de los datos o de los valores de los pesos.

No obstante si bien la evaluación del desempeño y rendimiento de un modelo neuronal de regresión concreto se realiza mediante el coeficiente anterior, es usual encontrarse con varios modelos alternativos con resultados parejos, lo que no permite dilucidar cuál es la mejor elección. Para facilitar esta tarea se han incluido una última fase que permite evaluar y comparar desde el punto de vista estadístico la calidad y capacidad de generalización de los modelos neuronales diseñados. Por ello se incluyen dos estimadores basados en la teoría de la información que indican la medida de la bondad del ajuste de un modelo estadístico estimado. Estos indicadores, medidas de la calidad en base a la información, son el criterio de información de Akaike (Akaike's Information Criterion o AIC) y el principio de longitud de descripción mínima (Minimum Description Length o MDL). Ambos criterios se ajustan al criterio general usado para el cómputo del error de predicción (EP), expresado como la suma de dos términos [52]:

$$EP = error\ entrenamiento + termino\ complejidad \quad (5)$$

donde el término de complejidad representa una penalización que crece a medida aumentan con los grados de libertad del modelo neuronal. Por tanto se basan en dos elementos de cómputo, uno que disminuye al mejorar el ajuste a los datos (término de fidelidad o ajuste) y otro que crece con el número de parámetros (término de penalización de la complejidad del modelo).

La utilización de estos criterios o indicadores en el ámbito de las RNA ha permitido seleccionar, entre varios candidatos, el modelo neuronal óptimo para un problema dado. El indicador AIC ha sido utilizado por los investigadores con diferentes finalidades, como reducir la subjetividad de la elección entre una arquitectura u otra [66], determinar el número de neuronas ocultas [67] e incluso diseñar comités de redes [61]. Por su parte el indicador MDL ha permitido reducir la complejidad de diferentes arquitecturas minimizando el número de pesos [68], y determinar el número de neuronas [70] y capas ocultas [69] para el modelo óptimo.

El criterio de información de Akaike es un factor estadístico que permite decidir el orden de un modelo. Fue ideado por Akaike en [71], y toma en consideración tanto la medida en que el modelo se ajusta a las series observadas como el número de parámetros utilizados en el ajuste. Este criterio se utiliza para medir el compromiso entre el rendimiento y el tamaño de la red. El objetivo en este caso es minimizar este factor para producir la red con una mejor capacidad de generalización. Para su cálculo dentro de una RNA se utiliza la siguiente ecuación:

$$AIC(k) = N\ln(ME) + 2k \quad (6)$$

donde k es el número de pesos de la red, N es el número de ejemplares del conjunto de entrenamiento y MSE es el error cuadrático medio obtenido. Utilizar el criterio de Akaike para seleccionar un modelo neuronal supone decantarse por aquel que presenta un menor valor del AIC lo cual es representativo de un modelo más parsimonioso (menos parámetros) y que mejor se ajusta a los datos (menor error de predicción). Un AIC pequeño es indicativo de que el estimador representa fielmente a la muestra a la vez que castiga el posible sobreajuste derivado del empleo de un excesivo número de parámetros. Con más parámetros es más fácil conseguir un buen ajuste a la muestra de referencia, pero los resultados del mismo no suelen adaptarse bien a otras muestras distintas, i.e. el modelo no es bueno y sus conclusiones no son extrapolables.

Por su parte, el principio de longitud de descripción mínima introducido por Rissanen en 1978 [72] establece que la mejor teoría para describir un conjunto de datos es aquella que minimiza la longitud de descripción del conjunto de datos. El principio MDL ha sido empleado con éxito para la inducción en árboles de decisión, procesamiento de imágenes y modelos de aprendizaje en dominios ingenieriles no homogéneos. Este indicador facilita un criterio para la selección de modelos, teniendo en cuenta su complejidad sin la restricción de asumir que los datos constituyen una muestra de una distribución verdadera. Se trata de un criterio similar a AIC en la medida en que intenta combinar el modelo de error con el número de grados de libertad para determinar el nivel de generalización. Por tanto el objetivo es minimizar este término a partir de la siguiente ecuación:

$$MDL(k) = N\ln(MSE)+0.5\ln(N) \quad (7)$$

donde k es el número de pesos de la red, N es el número de ejemplares del conjunto de entrenamiento y MSE es el error cuadrático medio obtenido.

IV. VALIDACIÓN

1) *Fase 1. Comparativa de alternativas al PMC.*

Una de las preguntas que pueden surgir al comenzar la fase de evaluación es porque se ha elegido un PMC y no otra alternativa basada en la teoría de las máquinas de aprendizaje o aprendizaje automático. En esta fase previa se comparará su calidad frente a otras propuestas dentro del escenario de la estimación del desarrollo de un proyecto software.

TABLA III. Resultados iniciales del PMC base con HOCV

Epochs	1000	3000	9000	27000	81000
PMC con HOCV	0,6750	0,6725	0,6616	**0,6844**	0,6811

Los resultados obtenidos después de 20 simulaciones para el PMC base se muestran en la Tabla III. En este caso el mejor comportamiento se presenta con un máximo 270000 epochs, destacando la estabilidad de las predicciones para las simulaciones realizadas. El número medio de epochs necesario es inferior a la cota de 27000, 26542 epochs, ya que en algunos casos se cumple el criterio adicional de valor mínimo de MSE<0,01.

TABLA IV. Listado de comparativas estudiadas en la fase 1

Propuesta	Descripción
RBF	Regla competitiva Conscience full. Métrica Euclidea. Topología 9-0-2
SVM	Algoritmo Kernel Adatron. Paso 0.01. Topología 9-0-2
RNN	Parcialmente recurrente. Regla de aprendizaje Momentum. Topología 9-1-2
PMC	Regla de aprendizaje Momentum. Topología 9-3-2

En la Tabla IV se muestran las máquinas de aprendizaje incluidas en la comparativa y sus características más importantes. La Fig. 3 muestra la precisión media de cada una de las alternativas estudiadas después de veinte repeticiones. Como se puede observar el PMC es el que mejores resultados en todos los casos, alcanzando el umbral de precisión para 27000 epochs. En segundo lugar destaca la RNN, sólo un -0,02 por debajo del PMC con 9000 epochs pero con un comportamiento más inestable en conjunto. El resto de propuestas, RBF y SVM se adaptan peor al problema propuesto, con unas cotas de precisión alejadas. Por último, en todos los clasificadores analizados la predicción decae de forma clara a partir de 27000 epochs, lo que indica signos de sobreaprendizaje.

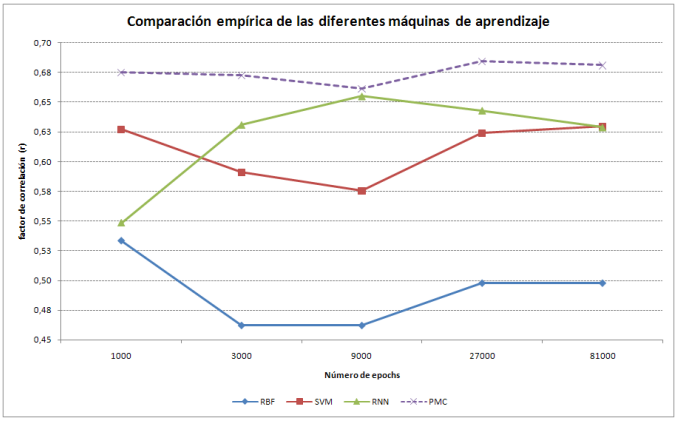

Figura 3. Fase 1. Comparativa de máquinas de aprendizaje.

Para corroborar las conclusiones anteriores se ha realizado un análisis enfocado a medir la calidad de cada máquina de aprendizaje. Para comparar la bondad del ajuste de cada una sobre los datos, se detallan en la Tabla V los criterios estadísticos basados en la teoría de la información AIC y MDL. Como se observa, el modelo neuronal basado en el PMC presenta un mejor comportamiento y ajuste sobre los datos, al obtener los valores más bajos en los indicadores estadísticos.

TABLA V. Resultados obtenidos en la fase 1, criterios AIC y MDL

Propuesta	AIC	MDL
RBF	623,7315	616,3639
SVM	137,2494	143,5810
RNN	131,8489	130,8091
PMC	**89,2494**	**88,5810**

Como ha quedado demostrado el PMC, además de ser superior en la media de los resultados, presenta un mejor ajuste sobre los datos desde punto de vista de los indicadores AIC y MDL. Por tanto, en los siguientes apartados se trabajará sobre esta propuesta con 27000 epochs para optimizar su utilización dentro de este dominio

2) *Fase 2. Preprocesado, distribución de la información y remuestrreo.*

2.1) Técnicas de imputación de datos

Esta fase se centra en primer lugar en realizar pre-procesamiento de los patrones para asegurar la plenitud de los datos empleados. En las simulaciones anteriores se ha empleado la media para completar los datos faltantes y poder realizar las pruebas correspondientes, ya que a pesar de ser la más sencilla suele presenta un comportamiento correcto en la mayoría de problemas. En la Tabla VI se muestran los resultados obtenidos para cada una de las alternativas estudiadas para completar la información del conjunto de patrones.

TABLA VI. Resultados obtenidos en la fase 2, pre-procesado de la información.

Propuesta	Mejor resultado
Media	**0,6838**
Mediana	0,6510
VADU	0,6456
VADE	0,6289

En este caso, la técnica que mejor resultado obtiene sigue siendo la que emplea la media para completar la información faltante. Se trata de una forma simple de imputación no aleatoria de un valor desconocido y consiste en asignar el valor promedio de la variable que lo contiene, calculado en los casos que tienen valor. Para las variables categóricas se ha imputado la moda de la distribución.

2.2) Técnicas de distribución de patrones

La distribuciones típicas de 50/50 o 80/20 para train y test expuestas en [41] no siempre se ajusta correctamente a todos los problemas. Por ello y para buscar la mejor distribución de patrones se seguirán las pautas descritas en [45] y se validarán los resultados mediante el modelo matemático de [44]. Además se realizarán para cada simulación 20 repeticiones con diferentes patrones en cada conjunto asegurando así la independencia del clasificador sobre los datos usados.

La Fig. 4 muestra la evolución de la precisión del PMC en función del tamaño del conjunto de train. Los resultados obtenidos coinciden con los postulados de Crowther y Fox. A partir de la línea de tendencia asociada se verifica la tesis expuesta en las investigaciones anteriores; el poder predictivo crece rápidamente hasta el punto A, a continuación empieza a decelerar hasta el punto B. Sin embargo en la última etapa, desde B hacia delante, no se observan mejoras significativas produciéndose cierto estancamiento e incluso empeorando la tasa de acierto.

En este caso el valor de máximo acierto coincide con las aserciones de Boonyanunta y Zeephongsekul, al fijarse el ratio para el conjunto de train cerca del 70% de los patrones disponibles. A partir de este punto B el rendimiento del clasificador se estanca y los resultados se estabilizan. La ventaja de realizar este análisis es que facilita conocer la cantidad de datos con que los resultados son aceptables, ya que la mejora conseguida a partir de este punto será mínima. Esto permite por un lado reducir el coste de conseguir datos adicionales para aumentar el tamaño del conjunto de entrenamiento y por otro acelerar la convergencia del aprendizaje. En este sentido es necesario razonar si compensa el coste asociado a generar más ensayos para ganar un porcentaje reducido de acierto ≈1%. Sin embargo, el problema asociado a esta técnica es la subjetividad en determinar que representa un nivel de mejora insignificante. Por ello, es necesario tener en cuenta otros criterios tales como el coste de extracción de la información, tiempo requerido para desarrollar y entrenar el modelo, etc.

TABLA VII. Valores aproximados por el modelo matemático en la fase 2 para la distribución de patrones

T(r)	p en T	k	P0(r)	Max.(r)	p en Max.
0,7512	501	0,002951	0,5223	0,7121	149

A partir de los resultados obtenidos por el PMC, y para probar la evolución del acierto del clasificador según aumenta el número de patrones más allá de los disponibles, se ha recurrido al modelo matemático descrito por Boonyanunta y Zeephongsekul en [44]. Dicho modelo se resume en la ecuación:

$$P(p) = T(1 - e^{kp}) + P(0)e^{kp} \quad (8)$$

donde T es el umbral de eficiencia, k la tasa de mejora en la capacidad de predicción por unidad de aumento de la eficiencia y P(0) es el poder predictivo cuando no se facilitan datos de entrenamiento al clasificador. Se han elegido tres puntos para aproximar el modelo descrito, 50, 100 y 149 patrones. A partir de los valores alcanzados por el clasificador y del método de optimización implementado en Microsoft Excel 2003 (método Solver), se han obtenido los valores mostrados en la Tabla VII..

El método Solver muestra el umbral máximo que podría alcanzar el clasificador (T(%)) al aumentar el número de patrones (p). La columna Max. (%) indica el valor máximo alcanzado por el clasificador y p en Max. el número máximo de patrones usados en ese caso. No obstante, al igual que ocurre con la investigación de referencia el modelo presenta cierto desfase en sus predicciones y no se ajusta correctamente para pocos datos. La línea de tendencia sobrepuesta en la Fig. 4 muestra que el valor umbral no se ha alcanzado. Además la elección de los puntos de referencia usados para ajustar el modelo se presenta fundamental a la hora de usar el método de optimización. Si bien la mejora prevista al aumentar el número de ensayos es interesante, hay que tener en cuenta el coste asociado a cada simulación de un solo caso balístico.

Las líneas de tendencia que aparecen son de tipo logarítmico, ya que se trata de una línea curva que se ajusta perfectamente a los datos, siendo de gran utilidad cuando el índice de cambios de los datos aumenta o disminuye rápidamente y después se estabiliza. Los puntos A y B se han aproximado mediante el cálculo de la media de variaciones y la desviación típica que se ha obtenido en las simulaciones.

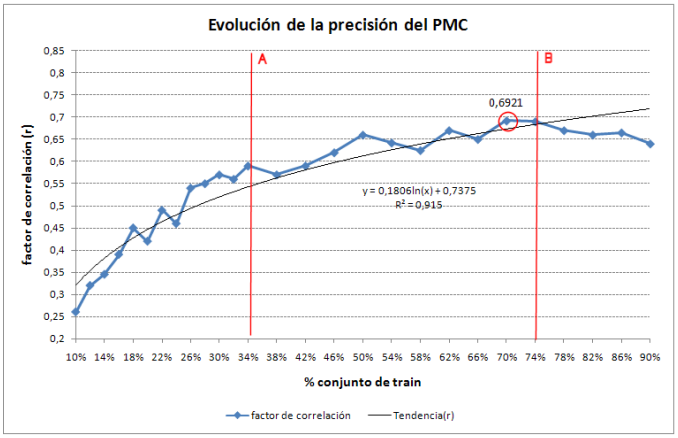

Figura 4. Fase 2. Precisión del PMC en función del conjunto de train.

Finalmente y una vez analizados los resultados la distribución óptima es la 70/30 para los conjuntos train/test.

2.3) Técnicas de remuestreo

TABLA VIII. Listado de experimentos para la fase 2, técnicas de muestreo

Propuesta	Descripción
HOCV	Validación cruzada Hold-Out.
$KFCV_5$ y $KFCV_{10}$	Validación cruzada K-fold con 5, 10 y 15 conjuntos.
$AWGN_{50}$ y $AWGN_{100}$	Introducción de 50 y 150 patrones con ruido.
$K_{10}AWGN_{100}$	Combinación de $KFCV_{10}$ y $AWGN_{100}$.

En algunos escenarios debido a las características del proceso que recrea los datos, la obtención y etiquetado de información reseñable se convierte en un proceso intensivo y complicado. Para mejorar los resultados sin necesidad de obtener más ensayos balísticos, se han analizado diversas técnicas provenientes del ámbito estadístico, utilizadas en problemas de estimación y clasificación, que permiten el remuestreo del conjunto original aumentando su tamaño. En estos casos la cantidad de datos disponibles que el diseñador de una RNA puede utilizar es limitada. En la Tabla VIII se detallan las diferentes alternativas incluidas en esta investigación.

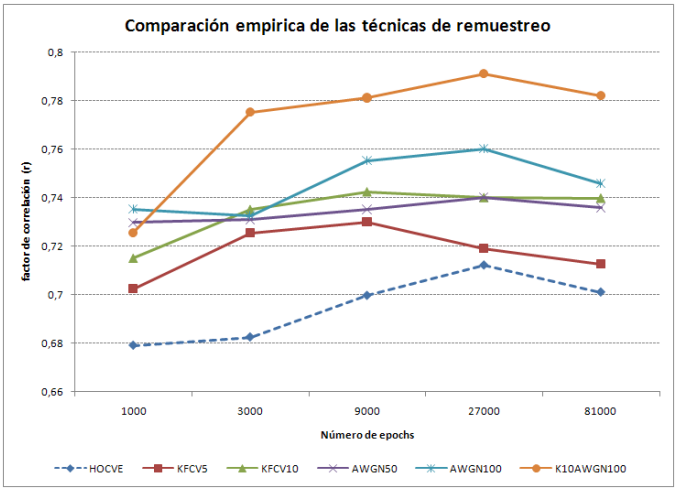

Figura 5. Fase 2. Comparativa de técnicas de remuestreo.

Los resultados de las simulaciones realizadas en este estudio aparecen en la Fig. 5. Aunque el método HOCV debido a su velocidad suele ser el más empleado para entrenar RNA, éste desperdicia una gran cantidad de información ya que solo una parte de los datos se usa para entrenar la red. Cuando la cantidad de datos disponible es reducida, como sucede en este dominio, es necesario estudiar otras alternativas que maximicen y optimicen el uso de los datos. La Fig. 5 muestra la comparación, por número de epochs y factor de correlación, de las técnicas analizadas. Como se puede observar, la propuesta $K_{10}AWGN_{100}$ es la que presenta los mejores resultados, con un comportamiento superior a partir de 1000 epochs. La mejora máxima respecto a HOCVE se obtiene con un máximo de 27000 epochs (el número medio de epochs necesario es de 21547), coincidiendo con el punto de máximo precisión del PMC: 0,7911. Del resto de alternativas destaca la introducción de 100 patrones de ruido y la variante KFCV de 10 conjuntos, siendo estas variantes las que presentan un comportamiento más estable.

Por ello para las siguientes fases se empleará el PMC bajo la configuración $K_{10}AWGN_{100}$. Gracias a esto se consigue paliar la influencia de la partición aleatoria de la técnica KFCV, al repetir cada simulación de la alternativa $K_{10}AWGN_{100}$ varias veces. Esto da como resultado la técnica 20*10cv, también denominada repeated K-fold cross validation, que indica que se ejecutará 20 veces cada simulación con 10 conjuntos de datos de similar tamaño, empleando 9 para entrenar 1 para testar la red [73].

3) Fase 3. Arquitectura de red

Esta fase es la encargada de optimizar la arquitectura del PMC diseñado, controlando los elementos ocultos, capas y neuronas, con el objetivo de mejorar su rendimiento y calidad. En primer lugar se utilizará un método adaptativo, del tipo AG, para localizar dentro del espacio de topologías aquellas que mejor comportamiento presenten. Para validar los candidatos anteriores se recurrirá a un proceso de prueba y error destinado a conocer el rendimiento de cada una de las propuestas.

Desde el punto de vista de la eficacia del aprendizaje no es conveniente crear topologías con muchas capas, siempre es mejor resolver un problema con redes de poca profundidad (i.e. un PMC de una o dos capas ocultas) para favorecer un entrenamiento más rápido. En este caso se parte de la arquitectura PMC básica por lo que inicialmente solo tendrá una capa oculta. Si el error no se considera adecuado se procederá a incluir una segunda capa. Dado que un PMC con dos capas ocultas se considera un aproximador universal, la utilización de más capas no suele ser necesaria.

El objetivo del primer paso, emplear un AG, es optimizar la arquitectura mediante la localización de aquella configuración que permita reducir al mínimo el error de validación. La topología de partida incluye una sola capa oculta y tres neuronas en ella: C_{1}N_{3}. En la Tabla IX se indican las mejores topologías localizadas por el AG para las capas y neuronas ocultas (señalizadas con el acrónimo AG). La primera de ellas, C_{1}N_{9}, utiliza una capa oculta con nueve neuronas mientras que la segunda, C_{1}N_{5}-C_{2}N_{7}, presenta dos capas ocultas de cinco y nueve neuronas. En este caso, para dos capas ocultas, como los resultados obtenidos por la última configuración no son mejores, y el tiempo de computación es más elevado, no se ha seguido buscando topologías más complejas, i.e con más capas ocultas. Para verificar las propuestas anteriores se ha estudiado su comportamiento real mediante un proceso de prueba y error, incluyendo además un rango mayor de configuraciones. En la Tabla IX se detallan los resultados de todas las arquitecturas estudiadas.

TABLA IX. Resultados obtenidos en la fase 3, comparativa de arquitecturas

Arquitectura	Acierto medio r	Epochs	AIC	MDL
9-3-1	0,7911	18652	89,2494	88,5810
9-9-1(AG)	**0,8252**	**15382**	**42,1789**	**40,6682**
9-10-1	0.7987	17265	77,3325	67,1231
9-3-6-1	0,6854	19892	154,5432	131,1397
9-5-7-1 (AG)	0,6912	16887	102,9325	127,8781
9-5-4-1	0,6511	20554	98,7843	95,1214

4) Fase 4. Algoritmo de aprendizaje

En esta fase se analizan diferentes alternativas orientadas a la optimización numérica del algoritmo. Las variantes del algoritmo de aprendizaje incluidas pretenden ayudar a localizar el mínimo global sobre la superficie del error del espacio de pesos. Cada una de estas alternativas, de primer o segundo orden, pretenden optimizar y acelerar dicha búsqueda mediante diferentes aproximaciones, como la forma de actualizar los pesos, la adaptación dinámica de los parámetros o la utilización de las segundas derivadas del error. En la Tabla X se detallan todas las propuestas estudiadas dentro de este apartado.

TABLA X. Listado de algoritmos de aprendizaje estudiados en la fase 4

Algoritmo	Descripción
RP	Método con parámetros (capa oculta-salida) η =1.0-0.4 y μ =0.7-0.4
LM	**Método de mejora de segundo orden para el gradiente**
QP	Método de mejora de segundo orden con parámetros (capa oculta-salida) η =1.0-0.4 y μ =0.7-0.4
GC	**Método de mejora de segundo orden para el gradiente**

Los resultados obtenidos se muestran en la Tabla XI. La mejor propuesta es el algoritmo GC, con un rendimiento en precisión y velocidad ligeramente superior al algoritmo clásico RP.

TABLA XI. Resultados obtenidos en la fase 4, algoritmo de aprendizaje

Algoritmo	Acierto mínimo r	Acierto promedio r	Acierto máximo r	Epochs promedio
RP	0,7610	0,8248	0,8812	15232
LM	**0,5189**	**0,5677**	**0,6452**	**24502**
QP	0,6063	0,6625	0,7383	19058
GC	0,7917	0,8525	0,8810	12566

5) Resumen final.

Una vez aplicadas todas las fases de la metodología se pueden obtener las siguientes conclusiones:

- Una correcta distribución de los patrones en los conjuntos de train y test mejora los resultados que obtienen las configuraciones clásicas.
- De todas las alternativas analizadas para maximizar la información, se ha seleccionado la combinación de la introducción de ruido AWGN y la técnica KFCVE, bajo la configuración $K_{10}AWGN_{100}$.
- La mejor topología de red localizada gracias al uso de un AG ha sido: C_{1}N_{9}, i.e. una sola capa oculta con nueve neuronas.
- El algoritmo de aprendizaje de Gradiente Conjugado permite mejorar el rendimiento del PMC respecto al algoritmo RP clásico para este dominio.
- Las métricas de calidad introducidas a lo largo de todas las fases de la metodología añaden un componente adicional para asegurar la validez de las conclusiones alcanzadas.
- La evolución de la precisión y del número de epochs completados se detallan en la Tabla XII. Las columnas "% Var." y "% Var. Total" muestran el porcentaje de variación respecto a la fase anterior y el porcentaje total de variación entre la última fase y el PMC base. En este caso, la mejora media total es del 24,56% y la reducción del número de epochs alcanza el 52.65%.

TABLA XII. Sumario y comparativa de los resultados obtenidos en cada fase

	PMC base	Fase 2	% Var.	Fase 3	% Var.
r	0,6844	0,7911	*+15,59*	0,8252	*+4,31*
Epochs	26542	21547	-18,81	15382	-28,61
	Fase 4	% Var.	% Var Total		
r	0,8525	*+3,30*	***+24,56***		
Epochs	12566	*-18,30*	***-52,65***		

V. CONCLUSIONES

Una de las líneas de investigación relativas a la estimación del esfuerzo que puede suponer el desarrollo de un proyecto software, consiste en la utilización de RNA. Esta técnica supone la consideración de un conjunto de factores de diseño elevado, así como un conjunto de datos de prueba suficiente, tal que permitan realizar un entrenamiento de la red para la obtención de resultados óptimos. Conjugar ambos aspectos es una tarea difícil que requiere conocimientos y experiencia. En este trabajo se ha propuesto una metodología de optimización de modelos de RNA para la estimación de esfuerzo del desarrollo de software. La metodología propuesta guía en la búsqueda del mejor modelo neuronal para el problema estudiado y en consecuencia mejora el rendimiento, tanto en tiempo como en precisión. El objetivo fundamental es lograr mejorar el rendimiento de un modelo neuronal buscando la mejor configuración posible para el problema estudiado.

Para validar la aproximación propuesta se ha diseñado una red de neuronas artificial basada en un conjunto de parámetros que se encuentran disponibles en los momentos iniciales del proyecto, excluyendo aquellos que necesitan un conocimiento detallado de la solución. De esta forma se ha comprobado la validez de la aproximación para las etapas iniciales del proyecto, en las que no se conocen en profundidad los detalles del proyecto. Además, la metodología propuesta maximiza la información a partir de un conjunto dado de datos de proyecto. Este aspecto de la metodología es de especial importancia dado que el número de proyectos de los que una compañía puede tener información puede ser reducido.

Los resultados obtenidos muestran como aplicando la metodología propuesta se ha conseguido configurar una arquitectura de RNA capaz de conseguir, a partir del conjunto de patrones de prueba, un factor de correlación promedio de 0,8525. La estimación del esfuerzo de un proyecto software se encuentra sujeto a un conjunto muy elevado de variables que pueden influir en el mismo. Conseguir una estimación adecuada y fiable a partir de un conjunto de datos reducido en las fases iniciales del desarrollo supone una ventaja competitiva y aporta una información muy valiosa para poder tomar decisiones.

Las líneas futuras de trabajo siguen dos vertientes fundamentales. Por un lado es posible continuar el estudio aplicando la metodología de optimización a un conjunto más amplio de arquitecturas de redes neuronales a fin de conseguir demostrar la aplicabilidad de la metodología a un espectro más amplio de posibilidades. Por otro lado, considerando que en fases más avanzadas del proyecto se dispone de información más detallada, la aplicación de esta metodología de optimización ofrece perspectivas muy prometedoras en cuanto a fiabilidad y eficiencia, las cuales deben ser contrastadas con futuras experimentaciones.

REFERENCIAS

[1] E. Y. Nakagawa, and J. C. Maldonado, "Requisitos Arquiteturais como Base para a Qualidade de Ambientes de Engenharia de Software," *IEEE Latin America Transactions*, vol. 6, no. 3, pp. 260-266, July 2008.

[2] R. Colomo-Palacios, Á. García-Crespo, J.M. Gómez-Berbís, and I. Puebla-Sánchez, "Social Global Repository: using semantics and social web in software projects", *International Journal of Knowledge and Learning*, vol. 4, no. 5, pp.452-464, 2008.

[3] H. Al-Sakran, "Software Cost Estimation Model Based on Integration of Multi-agent and Case-Based Reasoning", *Journal of Computer Science*, vol. 2, no.3, pp. 276-282, 2006.

[4] Á. García-Crespo, R. Colomo-Palacios, J.M. Gómez-Berbís, and M. Mencke, "BMR: Benchmarking Metrics Recommender for Personnel issues in Software Development Projects," *International Journal of Computacional Intelligence Systems*, vol. 2, no. 3, pp.257-267, October 2009.

[5] K.K. Aggarwal, Y. Singh, P. Chandra and M. Puri, "Bayesian Regularization in a Neural Network Model to Estimate Lines of Code Using Function Points", *Journal of Computer Science*, vol. 1, no.4, pp. 505-509, 2005

[6] H. Park, and S. Baek, "An empirical validation of a neural network model for software effort estimation", *Expert Systems with Applications*, vol. 35, no. 3, pp. 929–937, October 2008.

[7] S.J. Huang, and N.H. Chiu, "Applying fuzzy neural network to estimate software development effort", *Applied Intelligence*, vol.30 no.2, pp.73-83, April 2009.

[8] S.J. Huang, and N.H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation", *Information and Software Technology*, vol. 48, no. 11,pp. 1034–1045, November 2006.

[9] I. F. de Barcelos Tronto, J. D. Simoes da Silva, and N. Sant'Anna, "An investigation of artificial neural networks based prediction systems in software project management", *The Journal of Systems and Software* , vol. 81, no. 3, pp. 356–367, March 2008.

[10] S.J. Huang, N.H. Chiu, and L.W. Chen, "Integration of the grey relational analysis with genetic algorithm for software effort estimation", *European Journal of Operational Research*, vol. 188, no. 3, pp. 898–909, August 2008.

[11] X. Huang, D. Ho, J. Ren, and L.F. Capretz, "Improving the COCOMO model using a neuro-fuzzy approach", *Applied Soft Computing*, vol. 7, no. 1, pp.29–40, January 2007.

[12] M. A. Ahmeda, M. O. Saliub,and J. AlGhamdia, "Adaptive fuzzy logic-based framework for software development effort prediction", *Information and Software Technology*, vol. 47, no. 1, pp. 31–48, January 2005.

[13] L. F. Capretz, and V. Marza, "Improving Effort Estimation by Voting Software Estimation Models", *Advances in Software Engineering*, vol. 2009, doi:10.1155/2009/829725

[14] B. Mora, F. García, F. Ruiz, M. Piattini, A. Boronat, A. Gómez, J. Á. Carsí, I. Ramos, "JISBD2007-08: Software generic measurement framework based on MDA", *IEEE Latin America Transactions*, vol. 6, no. 4, pp. 363-370, August 2008.

[15] F. A. Zorzan, and D. Riesco, "Transformation in QVT of Software Development Process based on SPEM to Workflows", *IEEE Latin America Transactions*, vol. 6, no. 7, pp. 655-660, December 2008.

[16] J.C. Chen and S.J. Huang, "An empirical analysis of the impact of software development problem factors on software maintainability", *The Journal of Systems and Software*, vol. 82, no. 6,981–992, June 2009.

[17] M. Jørgensen, B. Boehm, and S. Rifkin, "Software Development Effort Estimation: Formal Models or Expert Judgment?" *IEEE Software*, vol. 26, no. 2, pp. 14-19, March/April 2009.

[18] M. Jørgensen, and D.I. Sjøberg, "The impact of customer expectation on software development effort estimates", *International Journal of Project Management*, vol. 24, no. 4, pp.317–325, May 2004.

[19] M. Jørgensen, "A review of studies on expert estimation of software development effort", *The Journal of Systems and Software*, vol. 70, no. 1/2, pp. 37–60, February 2004.

[20] R. Sternberg, "Component processes in analogical reasoning". *Psychological Review*, vol. 84 no. 4, pp.353–378, July 1977.

[21] N.H. Chiu and S.J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances", *Journal of Systems and Software*, vol. 80, no. 4, pp. 628–640, April 2007.

[22] B.W. Boehm, "*Software Engineering Economics*", Prentice-Hall, Englewood Cliffs, NJ, 1981.

[23] B.W. Boehm, E. Horowitz, R. Madachy, et al., "*Software Cost Estimation with COCOMO II*", Prentice-Hall, Upper Saddle River, NJ, USA, 2000.

[24] L. H. Putnam, "A general empirical solution to the macro sizing and estimating problem", *IEEE Transaction on Software Engineering*, vol. 4, no. 4, pp. 345–361, 1978.

[25] K. V. Kumar, V. Ravi, M. Carr, N. Raj Kiran, "Software development cost estimation using wavelet neural networks". *Journal of Systems and Software*, vol. 81, no. 11, pp. 1853–1867, November 2008.

[26] Y.S. Su, C.Y. Huang, "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models". *Journal of Systems and Software*, vol. 80, no. 4, pp. 606-615, April 2007.

[27] K.K. Shukla. "Neuro-genetic prediction of software development effort", *Information and Software Technology*, vol. 42, no. 10, pp. 701–713, July 2000.

[28] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort", *Information and Software Technology*, vol. 44, no. 15, pp. 911–922, December 2002.

[29] G. R. Finnie, G. E. Wittig, and J. M. Desharnais, "A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models", *Journal of Systems and Software*, vol. 39, no. 3, pp. 281- 289, December 1997.

[30] X. Huang, D. Ho, J. Ren and F. Capretz, "A soft computing framework for software effort estimation", *Soft Comput.*, vol. 10, no. 2, pp. 170-177, January 2006.

[31] H. Al-Sakran, "Software Cost Estimation Model Based on Integration of Multi-agent and Case-Based Reasoning", *Journal of Computer Science*, vol. 2, no. 3, pp. 276-282, April 2006.

[32] S. Ramon-y Cajal. "The croonian lecture: La fine structure des centres nerveux". *Proceedings of the Royal Society of London*, vol. 55, pp. 444–468, 1894.

[33] M. Arbib. "The Handbook of Brain Theory and Neural Networks". MIT Press, 1995.

[34] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[35] R. Lippmann. "An introduction to computing with neural nets". *ASSP Magazine, IEEE* [see also IEEE Signal Processing Magazine], vol. 4, no. 2, pp. 4–22, 1987.

[36] ISBSG, International Software Benchmarking Standards Group. ""Repository Data CD Release 8", 2003, http://www.isbsg.org.au/.

[37] R. Lippmann. "An introduction to computing with neural nets". *ASSP Magazine, IEEE* [see also IEEE Signal Processing Magazine], vol. 4, no. 2, pp. 4–22, 1987.

[38] L. Tarassenko. "A guide to neural computing applications". Arnol / NCAF, 1998.

[39] M. Hassoun. "Fundamentals of Artificial Neural Networks." The MIT Press, 1995.

[40] B.J. Wythoff. "Backpropagation neural networks: A tutorial". *Chemometrics and Intelligent Laboratory Systems*, vol. 18, pp. 115–155, 1993.

[41] K. Swingler. "Applying Neural Networks. A Practical Guide." Academic Press, 1996.

[42] M. Hassoun. "Fundamentals of Artificial Neural Networks." The MIT Press, 1995.

[43] J.C. Principe, N.R. Euliano, and W.C. Lefebvre. "Neural and Adaptive Systems: Fundamentals through Simulations". John Wiley & Sons, Inc., 1999.

[44] N. Boonyanunta and P. Zeephongsekul. "Predicting the relationship between the size of training sample and the predictive power of classifiers". *Knowledge-Based Intelligent Information and Engineering Systems*, vol. 3215, pp. 529–535, 2004.

[45] P. Crowther and R. Cox. "Accuracy of Neural Network Classifiers as a Property of the Size of the Data Set", *Knowledge-Based Intelligent Information and Engineering Systems*, vol. 4253, pp. 1143–1149. 2006.

[46] G. Foody, M.B. McCulloch, and W.B. Yates. "The effect of training set size and composition on artificial neural network classification". *International Journal of Remote Sensing*, vol. 16, no. 9, pp. 1707–1723, 1995.

[47] T.M. Cover. "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition". *IEEE Transactions on Electronic Computers*, vol. EC-14, no. 3, pp. 326–334, 1965.

[48] D. Foley. "Considerations of sample and feature size" , *IEEE Transactions on Information Theory*, vol. 18, no. 5, pp. 618–626, 1972.

[49] Gonzalez-Carrasco, I., Garcia-Crespo, A., Ruiz-Mezcua, B. and Lopez-Cuadrado, J.L. "*Dealing with limited data in ballistic impact scenarios: An empirical comparison of different neural network approaches"*. Applied Intelligence, (in press), 2009.

[50] Gonzalez-Carrasco, I., Garcia-Crespo, A., Lopez-Cuadrado, J.L. and Ruiz-Mezcua, B. *"An optimization methodology for machine learning strategies and regression problems in ballistic impact scenarios"*, Applied Intelligence, (in press), 2010.

[51] K. Priddy and P.E. Keller. "Artificial Neural Networks: An Introduction". SPIE Press, 2005.

[52] C. Bishop. "Neural Networks for Pattern Recognition". Oxford University Press, USA, 1996.

[53] L. Holmstrom and P. Koistinen. "Using additive noise in back-propagation training". *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 24–38, 1992.

[54] C. Harpham and C.W. Dawson. The effect of different basis functions on a radial basis function network for time series prediction: A comparative study". *Neurocomputing*, vol. 69, no. 16-18, pp. 2161–2170, 2006.

[55] Du, H., Zhang, N.: "Time series prediction using evolving radial basis function networks with new encoding scheme". *Neurocomput*ing, vol. 71, no. 7-9, pp. 1388–1400, 2008.

[56] V.N. Vapnik. "Statistical Learning Theory". John Wiley & Sons, Inc., 1998.

[57] E. Leclercq, F. Druaux, D. Lefebvre, and S. Zerkaoui. "Autonomous learning algorithm for fully connected recurrent networks". *Neurocomputing*, vol. 63, pp. 25– 44, 2005.

[58] R. J. A. Little and D. B. Rubin, *Statistical analysis with missing data*, John Wiley & Sons, Inc, 1986.

[59] E. M. L. Beale and R. J. A. Little, "Missing Values in Multivariate Analysis", *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 37, no. 1, pp. 129-145, 1975.

[60] Lefebvre, C., Fancourt, C., Principe, J. and Gerstenberger, J. *NeuroSolutions Documentation*. 2007.

[61] Z. Zhao, Y. Zhang, and H. Liao, "Design of ensemble neural network using the akaike information criterion", *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1182– 1188, 2008.

[62] S.E. Fahlman, "Faster-learning variations on back-propagation: an empirical study", *Connectionist Models Summer School*, pp. 38–51, 1988.

[63] K. Levenberg, "A method for the solution of certain non-linear problems in least squares", *Quarterly Journal of Applied Mathematics*, vol. II, no. 2, pp. 164–168, 1944.

[64] D.W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", *SIAM. Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[65] R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems", *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436, December 1952.

[66] D.B. Fogel, "An information criterion for optimal neural network selection", *Conference Record of Twenty-Fourth Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 998–1002, November 1990.

[67] N. Murata, S. Yoshizawa, and S. I. Amari, "Network information criterion — determining the number of hidden units for an artificial neural network model", *IEEE Transactions on Neural Networks*, vol. 5, no. 865–872, 1994.

[68] G.E. Hinton and D. Van-Camp, "Keeping the neural networks simple by minimizing the description length of the weights", *In COLT '93:*

Proceedings of the sixth annual conference on Computational learning theory, pp. 5–13. ACM, July 1993.
[69] S. Xu and L. Chen, "A novel approach for determining the optimal number of hidden layer neurons for fnns and its application in data mining", *In 5th International Conference on Information Technology and Applications*, pp. 23–26, June 2008.
[70] Y.I. Zhao and M. Small, "Minimum description length criterion for modeling of chaotic attractors with multilayer perceptron networks", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 3, pp. 722–732, March 2006.
[71] H. Akaike, "A new look at the statistical model identification", *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.
[72] J. Rissanen, "Modeling by shortest data description", *Automatica*, vol. 14, no. 5, pp. 465–471, September 1978.
[73] T.G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms", *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, October 1998.

Ángel García-Crespo. Responsable del grupo de investigación SOFTLAB ubicado en el Departamento de Informática de la Universidad Carlos III de Madrid. Es adicionalmente Subdirector de Ordenación Docente de la Escuela Politécnica Superior de la misma Universidad y secretario del Instituto Universitario Pedro Juan de Lastanosa. Es Doctor Ingeniero Industrial por la Universidad Politécnica de Madrid (premio del Instituto J.A. Artigas a la mejor tesis doctoral) y tiene un Executive MBA por el Instituto de Empresa. El Dr. García Crespo ha liderado y participado en proyectos de colaboración universidad-empresa, proyectos nacionales y europeos. Es autor de más de cien publicaciones en revistas, libros y congresos tanto nacionales como internacionales.

Israel González-Carrasco Ingeniero Informático y Doctor en Ciencia y Tecnología Informática por la Universidad Carlos III de Madrid. En la actualidad es Profesor Ayudante Doctor del Departamento de Informática de dicha Universidad. Es autor de varios artículos para revistas de impacto internacional, congresos nacionales e internacionales. Está involucrado en varios proyectos internacionales y es miembro del consejo revisor de la revista International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies.

Ricardo Colomo-Palacios. Ingeniero Superior en Informática y Doctor en Informática, adicionalmente, posee un Executive M.B.A. por el Instituto de Empresa. En la actualidad es Profesor Titular Interino del Departamento de Informática de la Universidad Carlos III de Madrid. Como parte de su labor investigadora ha participado en más de quince proyectos de investigación competitivos a nivel nacional y europeo y ha publicado más de sesenta contribuciones en revistas, libros científicos y congresos. Ha sido miembro del comité organizador, revisor y organizador de numerosos eventos científicos y en la actualidad desempeña el rol de editor en jefe de la revista International Journal of Human Capital and Information Technology Professionals.

José Luis López-Cuadrado es profesor ayudante en el Departamento de Informática de la Universidad Carlos III de Madrid. Es Ingeniero en Informática y Doctor en Ciencia y Tecnología Informática por la Universidad Carlos III de Madrid y su investigación está centrada en integración de sistemas inteligentes con sistemas de información en la web, redes de neuronas artificiales, mejora de procesos e ingeniería del software. Además es coautor de varios artículos en revistas de impacto internacional, congresos nacionales e internacionales.

Belén Ruiz-Mezcua. Doctora en físicas por la ETSI de Telecomunicación de la Universidad Politécnica de Madrid. Actualmente es profesa Titular en el departamento de Informática de la Universidad Carlos III de Madrid, directora técnica del Centro Español de Subtitulado y Audiodescripción (CESYA) y Vicerrectora adjunta de investigación para el Parque Científico Tecnológico. Es también miembro del Centro de Innovación Tecnológica de Discapacidad y Personas Mayores. Ha dirigido varios proyectos nacionales e internacionales Es autora y coautora de varias publicaciones nacionales e internacionales.