Universidad Politécnica de Cartagena

Departamento de Tecnologías de la
Información y las Comunicaciones

Tesis Doctoral

Aprendizaje Máquina Multitarea mediante Edición de Datos y Algoritmos de Aprendizaje Extremo

Autor: **Andrés Bueno Crespo**
Director: **Dr. José Luis Sancho Gómez**

2013

A mis padres, a mi mujer, y a esos dos peques que espero que algún día entiendan "esas cosas raras que escribe papá".

Agradecimientos

Este apartado es algo que todo el mundo se lee, bien porque me conoce y espera encontrar su nombre en estas líneas o bien porque aunque no me conozca le lleva la propia curiosidad. Si eres del grupo de los que me conocen, espero no defraudarte, aunque cuando uno empieza a citar nombres corre el peligro de no nombrar a alguien, que no quiere decir que se ha olvidado, ya que son muchas las personas de las que de alguna manera u otra les pertenece algo de esta Tesis.

Quiero empezar por mi director, el Dr. José Luis Sancho, Profesor Titular de la *Universidad Politécnica de Cartagena*, quién durante años me ha ayudado con su amistad, consejos y apoyo incondicional que ha hecho posible todo este trabajo. Gracias por tu compromiso.

A Pedro, quién de primera hora hemos compartido las mismas inquietudes, por esos bocetos hechos en servilletas de cafetería, aun las guardo, gracias.

A Antonio, que compartió conmigo aquellas primeras investigaciones.

Como olvidar a mis compañeros de trabajo, también un trocito de esta Tesis se los debo a ellos. Un equipo fantástico donde todos reman en la misma dirección.

También una parte de este trabajo se la debo a mi familia: a mis padres, Andrés y Antonia, que siempre me han animado a seguir por este camino y a los que tanto debo. A Diego y Conchi, por tenerme como a un hijo más. A Juan, Mamen, Pedro y Gloria, por hacer que me sienta como en casa y a Diego, por

ser como un hermano.

A María José, por compartir día a día los buenos y malos momentos y sufrir esta Tesis, prácticamente, es casi suya. Y a mis hijos, Álvaro y Andrea, por todo este tiempo robado, espero ahora recuperar todas esas horas de parque.

A todos, gracias por ejercer un *sesgo inductivo positivo* en la realización de esta Tesis, aunque, claro, seguramente no sabrás que es eso del *sesgo inductivo*, pero si te animas y sigues leyendo, seguro que hasta te gusta...

Resumen

Cuando los seres humanos nos enfrentamos a un nuevo concepto que queremos aprender, nuestro cerebro no lo hace de forma aislada, sino que utiliza todo el conocimiento previamente aprendido para ayudarse en este nuevo aprendizaje. Además, nuestro cerebro es capaz de aislar lo que no va a beneficiarnos y a utilizar lo que realmente nos va ser útil, esto lo hace muy bien y de forma inconsciente. Sin embargo, cuando una maquina de aprendizaje es entrenada para resolver una determinada tarea, por ejemplo, a diagnosticar una determinada enfermedad, normalmente esta máquina aprende en solitario sólo con los datos disponibles sobre esa enfermedad.

Hay una metodología llamada Aprendizaje Multitarea, MTL ("Multi-Task Learning"), que se fundamenta en la idea inicialmente expuesta. De esta forma, la tarea a resolver (tarea principal) se aprende conjuntamente con otras tareas relacionadas (tareas secundarias), se produce una transferencia de información entre ellas que puede ser ventajosa para el aprendizaje de la primera. Sin embargo, en problemas reales, es difícil encontrar tareas que estén relacionadas o incluso, encontrándolas, es sumamente complejo determinar el grado en que se va a realizar esa ayuda, ya que una tarea puede contener información que puede ayudar pero también perjudicar.

Esta Tesis incorpora una nueva metodología que permite obtener tareas secundarias relacionadas con la que se pretende aprender (tarea principal). La segunda contribución de este trabajo se enmarca también dentro del MTL,

en este caso, diseñando de forma automática una máquina MTL que elimine todos aquellos factores que perjudiquen o no beneficien al aprendizaje de la tarea principal. Esta arquitectura es única y además se obtiene sin necesidad de metodologías de ensayo/error que aumentan la complejidad de cálculo.

Abstract

When humans faced with a new concept that we want to learn, our brain does not work in isolation, but rather uses all previously learned knowledge to assist in this new learning. In addition, our brain is able to isolate what is not going to benefit and use what will be really useful, it does very well and unconsciously. However, when learning machines are trained for solving an specific problem, for example, to diagnose a particular disease, usually this machine learns only the available data on this disease.

There is a methodology called Multi-Task Learning (MTL), which is based on the idea initially exposed. Thus, the task to be solved (main task) is learned together with other related tasks (secondary tasks), by producing a transfer of information among them which may be advantageous for learning of the main one. But in real problems, it is extremely difficult to determine how the simultaneous learning with other related tasks affects to the performance of the main one, because a task can contain information that can be helpful (i.e. the main task learning is improved) or harmful (i.e. the main task learning gets worse).

This PhD. Thesis, proposes a new methodology that allows to obtain related secondary tasks in order to be helpful to the main one. The second contribution of this work is also included in the MTL framework: the complete automatically removing those factors which harm or no benefit the learning of the main task. This architecture is also unique and it is obtained without the traditional test/fail

methodologies which usually increase the computational complexity.

Índice

Lista de Tablas

Lista de Figuras

Capítulo 1

Machine Learning is the study of computer algorithms that improve automatically through experience.

(Tom Mitchell)

Introducción

1.1. Introducción

Cada día son más los modelos computacionales que aprenden a través de la experiencia, esta rama de la inteligencia artificial recibe el nombre de aprendizaje máquina [71]. El objetivo de este tipo de aprendizaje es imitar la forma en la que aprende el ser humano. Nuestro cerebro no se enfrenta a un nuevo conocimiento sin tener en cuenta lo ya aprendido, y para ello creará nuevas conexiones sinápticas o inhibirá aquellas que considere innecesarias. Sin embargo, el aprendizaje máquina suele abordar un problema sin tener en cuenta otros problemas que pudieran estar relacionados.

Esta Tesis pretende profundizar en un tipo de aprendizaje máquina denominado aprendizaje multitarea, en el cual se aprenden diferentes tareas al mismo

tiempo, de tal forma que se produzca una transferencia de conocimiento entre ellas. Esta transferencia de conocimiento está condicionada por la relación que tengan las tareas entre ellas. De tal forma que encontrar esa relación o favorecerla desde el punto de vista de la arquitectura de la máquina de aprendizaje, así como crear tareas de forma artificial que ayuden a aprender una determinada tarea son las principales líneas de investigación de esta Tesis.

Este capítulo introductorio, continúa con tres secciones. La Sección 1.1 muestra en más detalle la definición del problema. La Sección 1.2 define los objetivos propuestos por esta Tesis y finalmente, la Sección 1.3 nos indica la estructura de los capítulos siguientes.

1.2. Definición del problema

Supongamos que tenemos una base de datos con información sobre un determinado tipo de cáncer. Nos encontramos con un problema de clasificación supervisado donde se pretende aprender a clasificar pacientes sanos de los enfermos. La mayoría de métodos de aprendizaje máquina, utilizarían estos ejemplos que tenemos en la base de datos para crear una máquina que sea capaz de aprender este modelo. En principio parece lo lógico, sin embargo, pensemos un momento que haría nuestro cerebro. Al empezar a comprender las posibles causas del cáncer que estamos estudiando, puede que nos llame la atención algo que ya vimos, por ejemplo, en otro tipo de cáncer y que es común en este. Por esa razón, parece más lógico abordar este problema mediante un aprendizaje máquina en un entorno multitarea, donde el aprendizaje de varios tipos de cáncer conjuntamente con el que queremos aprender facilite esta labor de aprendizaje frente a su aprendizaje en solitario. Dicho así, no parece

muy complejo, sin embargo, en problemas reales, usualmente no se dispone de tareas relacionadas para poder aprender una determinada tarea utilizando un esquema multitarea. También pudiera ocurrir que la relación entre las tareas sea incierta, con lo que haría difícil la selección de las tareas más convenientes para este tipo de esquema. Hay que tener en cuenta que una tarea que no esté relacionada con la que se quiere aprender no sólo no le va a ayudar, sino que incluso podría perjudicar su aprendizaje ya que incorporaría a este proceso información confusa.

1.3. Objetivos

Esta Tesis tiene dos objetivos:

1. Proporcionar un método para poder realizar un enfoque multitarea cuando esto no es posible por no disponer de tareas relacionadas con la que queremos aprender. De esta manera, a partir de una tarea a aprender, obtendremos tareas relacionadas con ella que facilitarán su aprendizaje.

2. Proporcionar un método de diseño, automático y único, de arquitecturas multitarea. Esto se consigue mediante la eliminación de las características de entrada que no benefician el aprendizaje, así como mediante la poda de neuronas y conexiones innecesarias.

Para realizar estos objetivos, las máquinas multitarea desarrolladas en esta Tesis Doctoral son esquemas de aprendizaje basados en redes neuronales artificiales tipo MLP ("Multi-Layer Perceptron") [14, 85, 43, 4]. Las máquinas implementadas se fundamentan en los modelos propuestos por Caruana [22], y posteriormente por Silver [94]. Como conjuntos de datos se han utilizado modelos artificiales para explicar la metodología utilizada [85, 94], y un conjunto

amplio de bases de datos reales con diferentes número de entradas, salidas y número de muestras, todos ellos disponibles en la UCI Machine Learning Repository [5].

Se han creado conjuntos de datos de forma artificial utilizando técnicas de Edición de Datos para poder utilizar esquemas multitarea y una metodología para crear una arquitectura para la red neuronal basada en el algoritmo OP-ELM que mejora el entrenamiento de un MLP. Todo ello para facilitar la transferencia de información entre tareas. Para ello, se presentan los resultados analíticos de los modelos presentados en esta Tesis que se comparan con el aprendizaje de la tarea en solitario y, en el caso de existir tareas relacionadas, con la utilización de un modelo multitarea clásico (totalmente interconectado). Los resultados obtenidos sobre los conjuntos de test, demuestran la validez de los modelos presentados en esta Tesis.

1.4. Estructura de la Tesis

Esta Tesis tiene un total de seis capítulos. El resto de capítulos que la componen son los siguientes.

El Capítulo 2 define los elementos necesarios para comprender y ubicar esta Tesis, esto es, una introducción al aprendizaje máquina y la forma en que se realiza la transferencia de información entre tareas. Se explican las diferencias entre un aprendizaje clásico, donde sólo se aprende una tarea y un esquema multitarea. También se analiza la importancia de la relación entre tareas y como una tarea puede influir de forma negativa en un aprendizaje.

El Capítulo 3 presenta un método para crear tareas artificiales para entornos multitarea. Este esquema favorecerá el aprendizaje frente a aprender la tarea

en solitario.

Los Capítulos 4 y 5 están bastante relacionados. Mientras que en el 4 se muestra una metodología para diseñar una arquitectura que elimina características de entrada, nodos y conexiones irrelevantes para un perceptrón multicapa, en el 5 se adapta esta nueva arquitectura para problemas con aprendizaje multitarea.

Finalmente en el Capítulo 6 se abordan las contribuciones, conclusiones y trabajos futuros. Otro aspecto importante de este capítulo es que incluye un listado de publicaciones tanto en congresos nacionales, internacionales, y revistas indexadas o no indexadas así como capítulos de libro que se han derivado directamente de este trabajo.

Capítulo 2

Las neuronas son células delicadas y elegantes, las misteriosas mariposas del alma, cuyo batir de alas quién sabe si esclarecerá el secreto de la vida mental.

(Santiago Ramón y Cajal)

Aprendizaje máquina e inductivo

2.1. Introducción

Cualquiera de nosotros, al enfrentarnos a un nuevo concepto a aprender, utiliza el conocimiento que ya posee sobre otros conceptos aprendidos previamente y que guardan relación con ese nuevo concepto. Esta utilización del conocimiento ya aprendido lo hacemos de forma inconsciente y además, nuestro cerebro lo hace muy bien. Sin embargo, cuando una máquina ha de aprender un nuevo concepto (que llamaremos tarea), normalmente se aprende en solitario, sin tener en cuenta ningún tipo de conocimiento aprendido previamente o a la vez que se aprende esa tarea. Este capítulo proporciona la información básica necesaria para comprender los conceptos del aprendizaje máquina

cuando se utiliza transferencia de información entre tareas. Inicialmente, en la Sección 2.2 se explica en qué consiste el aprendizaje máquina y el tipo de redes neuronales artificiales, ANN ("Artificial Neural Networks") que se van a utilizar en esta Tesis. En la Sección 2.3, se explicará qué es y cómo se produce la transferencia inductiva de información entre tareas y en la Sección 2.4 se verá cómo esa transferencia de información tiene lugar en las redes neuronales artificiales. Por último, la Sección 2.5, nos muestra una arquitectura para el aprendizaje de múltiples tareas, MTL ("Multi-Task Learning") mediante redes neuronales, así como un ejemplo de este tipo de aprendizaje máquina.

2.2. Aprendizaje máquina

Un sistema de reconocimiento de patrones para clasificación está formado por un conjunto de n características de entrada (X) y un conjunto de m valores para la salida (Y). Esas salidas se conocen como *salidas deseadas*, y el par formado por un conjunto de características de entrada y su salida deseada correspondiente se conoce como *patrón*. Estos patrones se han formado mediante una función $f : X \rightarrow Y$ que es la función conocida como *verdadera*. El objetivo del sistema de reconocimiento de patrones para clasificación es encontrar una función $h : X \rightarrow Y$ que es una aproximación de la función verdadera f. A esta función de aproximación también se conoce como hipótesis. Por ejemplo, para determinar si un paciente tiene una enfermedad cardiaca, tenemos que examinar un conjunto de atributos del paciente, como la presión de la sangre, el peso, el colesterol en sangre, etc. Pasamos estos atributos, junto con algunos datos personales del paciente, tales como sexo, edad, y otros a un sistema de aprendizaje máquina. Esperamos que la hipótesis generada por la máquina pueda

proporcionar un alto grado de precisión para predecir la existencia de enfermedades del corazón. Esta forma de aprendizaje se conoce como aprendizaje máquina, y por lo tanto, su objetivo es desarrollar algoritmos de aprendizaje para la generación de hipótesis que mejor se aproximen a la función verdadera. Desde este punto de vista, existen dos conceptos especialmente importantes: el espacio de hipótesis y el sesgo inductivo. A continuación, se explican cada uno de ellos.

2.2.1. Espacio de hipótesis

Un espacio de hipótesis H, está formado por todas las posibles hipótesis h de un sistema de aprendizaje. Estas hipótesis generalmente comparten las mismas características de entradas y el mismo conjunto de datos [95]. El objetivo del aprendizaje máquina es llevar a cabo una búsqueda a través del espacio de hipótesis del sistema y encontrar la hipótesis que mejor se adapte al conjunto de datos con los que se está aprendiendo. Al ser una búsqueda en el espacio de hipótesis, se tiene el inconveniente que si este espacio es muy grande, la búsqueda pueda requerir mucho tiempo computacional y la solución encontrada puede no ser óptima. Una posible solución es reducir este espacio de hipótesis, sin embargo, si esta reducción no se hace de una forma eficiente, podemos tener el inconveniente de no encontrar una solución óptima por haberla dejado fuera del espacio de hipótesis donde estamos buscando, conocido como *espacio de búsqueda*. Por ello, sobre un espacio de hipótesis grande, necesitaremos heurísticos o restricciones que nos permitan de una forma eficiente reducir este espacio inicial de búsqueda. A estos heurísticos o restricciones, se conocen como sesgos inductivos.

2.2.2. Sesgo inductivo

Se entiende por sesgo inductivo a todo aquello que induce a una máquina a preferir unas soluciones frente a otras en el espacio de soluciones (hipótesis) durante el aprendizaje de una tarea (concepto a aprender), con el objetivo de obtener mejores prestaciones en la resolución final de la misma [98].

El propósito de una máquina de clasificación es conseguir una elevada capacidad de generalización [13, 14, 30], es decir, una alta capacidad de que el modelo clasifique adecuadamente una tarea ante valores para las características de entrada que no se han utilizado en el aprendizaje de la misma, de esta forma, el sesgo inductivo permite encontrar hipótesis que favorecen la generalización para un conjunto de datos [70, 95]. El sesgo inductivo favorece a un subconjunto del espacio de hipótesis, este subconjunto es el *espacio de búsqueda* antes citado, que se supone que contienen mejores hipótesis. Esto hace que el proceso de búsqueda sea mucho más eficiente. La relación entre hipótesis, el espacio de búsqueda, y el espacio de hipótesis se muestra en la Figura 2.1. El espacio de hipótesis de un sistema de aprendizaje, H_A, contiene todas las posibles hipótesis h_i del sistema. Supongamos que una hipótesis particular h_0 es la que mejor se adapta al conjunto de datos con los que se está aprendiendo y que tenemos cuatro espacios de búsqueda: H_1, H_2, H_3 y H_4 para el aprendizaje del sistema. Tanto el tamaño del espacio de hipótesis y los cuatro espacios de búsqueda son finitos. Cada espacio de búsqueda es de menor tamaño que el espacio de hipótesis, $|H_i| < |H_A|$. Entre estos cuatro espacios de búsqueda, es de especial importancia H_1, seguido de H_3, ya que la hipótesis objetivo h_0 está dentro de H_1 y, a su vez H_1 está en H_3, pero no en H_2 ni en H_4. A pesar de que el H_4 es el más pequeño en tamaño, no debe ser utilizado, ni tampoco H_2 que aunque está dentro de H_3, la hipótesis objetivo no pertenece a él. Además, el

tamaño de H_3 es mayor que el de H_1 y con ello requiere más tiempo para la búsqueda. Por lo tanto, H_3 no debería utilizarse tampoco.

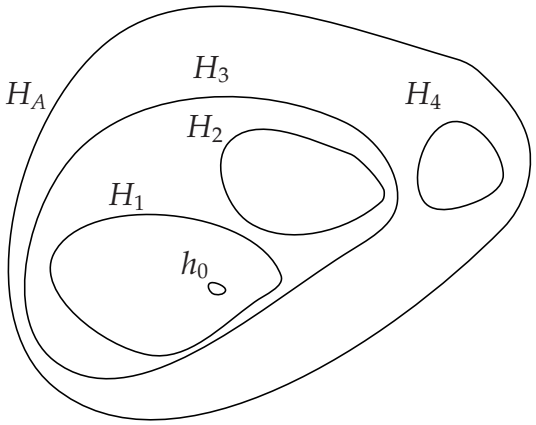


Figura 2.1: Relación entre hipótesis, espacio de búsqueda y espacio de hipótesis. La hipótesis buscada, h_0, está contenida en el espacio de búsqueda H_1 que a su vez lo está en H_3. H_A representa todo el espacio de hipótesis.

No hay sistema de aprendizaje inductivo universal que puedan garantizar que va a funcionar de manera eficiente para diferentes tareas [88]. Dependiendo de la máquina de aprendizaje, el sesgo inductivo será diferente debido a la propia forma en la que aprenden. Por ejemplo, para las redes neuronales artificiales, si en la inicialización de los pesos se aplica el conocimiento que se tiene del problema para hacer una elección orientada a lo que sería un buen punto de inicio de los pesos, esto sería un sesgo inductivo que favorece la búsqueda de la hipótesis ya que el punto inicial de donde parte la red neuronal está muy próximo al buscado. Normalmente, este conocimiento no está disponible en la mayoría de los casos.

Hay cinco categorías principales de sesgo inductivo [70]. En todas se utiliza el conocimiento aprendido previamente para facilitar el aprendizaje de la tarea

actual. Las categorías son las siguientes:

- Sesgo simple: para la resolución de un problema, los seres humanos preferimos una solución simple ante una compleja. En el aprendizaje máquina, este principio se refiere a la creencia común de que entre todas las hipótesis que pueden producir resultados similares, la hipótesis más simple suele ser la mejor (Occam's razor).

- Elección por conocimiento: el conocimiento de la hipótesis elegida puede ayudar a la elección del sesgo inductivo. Por ejemplo, en la predicción de un terremoto, se prefiere una hipótesis que produzca un falso positivo (la predicción de un terremoto, cuando en realidad no hay), en lugar de producir un falso negativo (no predecir el terremoto).

- Conocimiento de los datos de entrenamiento: si se conocen de antemano la fuente de los datos de entrenamiento, se puede realizar inferencias lógicas. Esto es útil para hacer restricciones. Por ejemplo, un alumno puede asumir que los ejemplos proporcionados por los profesores contienen características importantes para la tarea de aprendizaje actual.

- Dominio de la tarea: es razonable suponer que todas las tareas dentro del mismo dominio siguen las mismas reglas. Así, las normas generales y los conocimientos previamente aprendidos en el dominio de una tarea se pueden utilizar para limitar la búsqueda de otra. Por ejemplo, si el dominio de la tarea es aprender los diferentes estados emocionales de una persona, el aprendizaje de un tipo de gestos debe ser capaz de ayudar al aprendizaje de otros.

- Analogía con las tareas previamente aprendidas: si un sistema de aprendizaje se entrenó con varias tareas relacionadas, los sesgos de las tareas

anteriormente aprendidas serán útiles cuando se está aprendiendo una nueva tarea que esté relacionada con las anteriores. Por ejemplo, cuando se aprenden los gestos faciales, si hemos observado en un aprendizaje previo cierto gesto con respecto a un estado de ánimo, podemos asumir esa asociación gesto-estado para una nueva tarea a aprender.

2.2.3. Redes Multi-capa Unidireccionales

Dentro de los sistemas de aprendizaje máquina, las redes neuronales artificiales utilizan el aprendizaje inductivo para imitar los aspectos del aprendizaje humano, siendo sus parámetros de aprendizaje totalmente adaptativos [94].

Dentro de las redes neuronales artificiales, hay una variedad que es la red Multi-capa Unidireccional [80], la cual está compuesta por unas características de entrada, una capa de nodos ocultos, y una capa de nodos de salida, tal como se puede ver en la Figura 2.2. La información se transmite capa a capa desde la entrada, pasando por los nodos de la capa oculta, hasta la capa de salida (Figura 2.2. En esta Tesis, utilizaremos este tipo de red.

El algoritmo BP ("back-propagación") se utilizará para entrenar estas redes neuronales, corrigiendo los errores del aprendizaje, al igual que hacemos los seres humanos, cuando aprendemos, por ejemplo algún deporte, intentamos corregir nuestros errores. Este algoritmo funciona de forma similar, propagando hacia atrás el error producido por la salida de un patrón (conjunto de características de entrada y valor de la salida deseada) en la red neuronal. De acuerdo con [80], el algoritmo de BP es el algoritmo más utilizado en el aprendizaje de redes neuronales. Una vez que la red produce un valor de salida para un patrón, el algoritmo BP calcula el error entre la salida obtenida y la salida deseada, y ajusta los pesos para disminuir ese error. Este proceso se repetirá

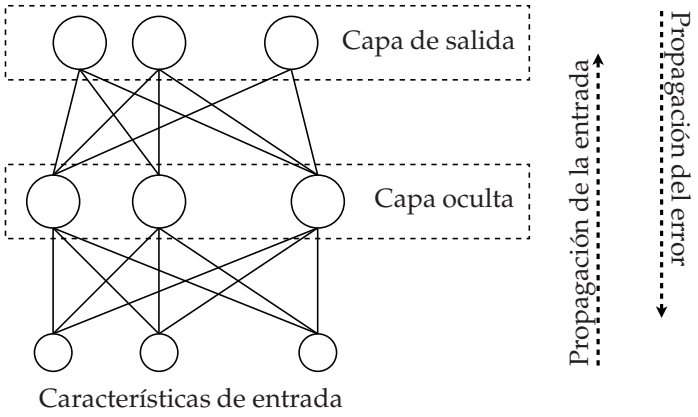


Figura 2.2: Red Multi-capa Unidireccional. La información se transmite desde los nodos de entrada hasta la capa de salida, mientras que el error producido al comparar la salida deseada con la obtenida se propaga en sentido inverso (BP).

hasta que se satisfaga el criterio de parada establecido (error por debajo de un determinado umbral o saturación de dicho error). Cuando el proceso de aprendizaje termine, se espera que la red produzca un modelo con la mayor generalización posible, esto es, que cuando la red neuronal clasifica un patrón con el que no se ha entrenado, de una salida que se clasifique de forma correcta. En esta Tesis, se presenta un modelo que modifica este algoritmo para solventar algunos problemas que presenta, como por ejemplo los mínimos locales. Por ello se verá más detenidamente en el siguiente capítulo.

Mínimo local

Si tenemos en cuenta todo el espacio de hipótesis de la red neuronal para un espacio de n-dimensiones, donde n es el número de pesos de la red, cada hipótesis se representa como un punto en este espacio. En cada dimensión, el valor del error E es una función de un peso en particular w_{ij}. Por ejemplo, en la Figura 2.3 se puede ver la función del error con respecto a los valores de un peso

genérico w_{ij}. Cuando el algoritmo BP realiza el descenso del gradiente en el proceso de actualización para un peso, su propósito es reducir el error aportado por ese peso para minimizar el error de la red. Por ejemplo, en la Figura 2.3, supongamos que el valor inicial para el peso w_{ij} es en el punto I_1. Entonces, el algoritmo BP intenta mover el modelo (el valor de los pesos) cuesta abajo para reducir el valor del error. En este proceso, hay una posibilidad de que el modelo se atasque en un mínimo local (L_1). Este mínimo local representa una solución parcial para la red en respuesta a los ejemplos de entrenamiento. Esto proporcionaría una solución no tan buena como la solución que se alcanzaría con el mínimo global (G).

Existen muchas propuestas para reducir la probabilidad de quedar atrapado en un mínimo local cuando entrenamos con algoritmos de gradiente tipo BP, [6, 39, 37]. Por ejemplo, inicializar los pesos con valores aleatorios, con lo que cada vez que ejecutemos el algoritmo, éste se situará en un punto diferente de la curva del error permitiendo evitar el mínimo local en alguna de las inicializaciones. Otro método consiste en añadir ruido estocástico en los pesos de la red. Esto permite trasladar el modelo a otro lugar de la superficie del error y evitar de esta forma quedarse atrapado en un mínimo local [65].

Estos métodos no son excluyentes uno de otros, sino que, por el contrario se suelen combinar entre sí. Aun así, no hay un método (o combinación de ellos) que permita evitarlos por completo.

En esta Tesis, se presenta un método para que una tarea evite mínimos locales mediante el sesgo inductivo proporcionado por tareas relacionadas, las cuales son más fáciles de aprender, convergen más rápido y ayudan a esa tarea desde sus primeras épocas de entrenamiento.

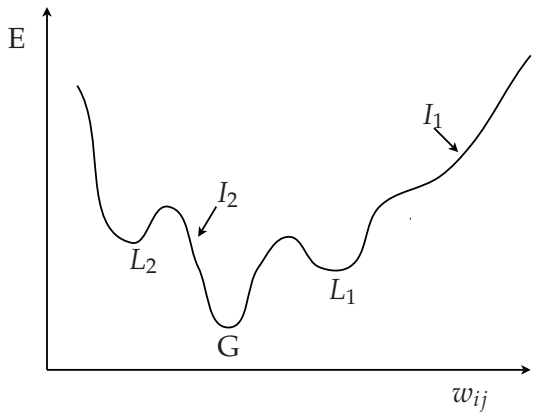


Figura 2.3: Relación entre mínimo local y global. Dependiendo del punto en que nos situemos de la curva del error tendremos más o menos posibilidades de alcanzar el mínimo global.

Sobreaprendizaje

Otro problema que presentan las ANNs es el sobreaprendizaje de los pesos de la red. Esto ocurre cuando se aprende en exceso los patrones con los que se entrena, con lo cual, la red clasifica muy bien esos patrones, pero generaliza mal para los nuevos con los que no hemos entrenado. Para evitar el sobreaprendizaje se utiliza otro conjunto independiente de patrones que se conoce como conjunto de validación. Mientras se realiza el aprendizaje con los patrones de entrenamiento, se mide el error con los de validación, de tal forma que el punto óptimo del aprendizaje se encuentra en donde los patrones de validación presentan el menor error (Figura 2.4). Para evitar caer en un mínimo local, se mantendrá cierto tiempo el aprendizaje. Una vez finalizado éste, los pesos se restaurarán al valor que tenían para el error más bajo que presentaban los patrones de validación.

Otra manera de evitar el sobreaprendizaje es utilizar métodos de regularización, con el objetivo de aumentar la capacidad de generalización de un red

neuronal. Existen muchas propuestas para las ANNs, sin embargo, uno de los métodos de regularización más conocido es el *Weight Decay*, que se basa en añadir términos adicionales en la función de error que penalizan los pesos con valores elevados [25, 15, 14, 88, 109].

También, los métodos de poda, permiten optimizar la capacidad de generalización ya que reducen el número de pesos y conexiones de la red manteniendo el error de aprendizaje tan bajo como sea posible y por lo tanto reduciendo el número de épocas. Entre los métodos de poda más conocidos, tenemos el OBM ("Optimal Brain Damage") y su variante OBS ("Optimal Brain Surgeon") que mediante el empleo de información de segundo orden de la derivada de la función de error, permiten mejorar la capacidad de generalización y la velocidad de aprendizaje de la red [27, 42, 110].

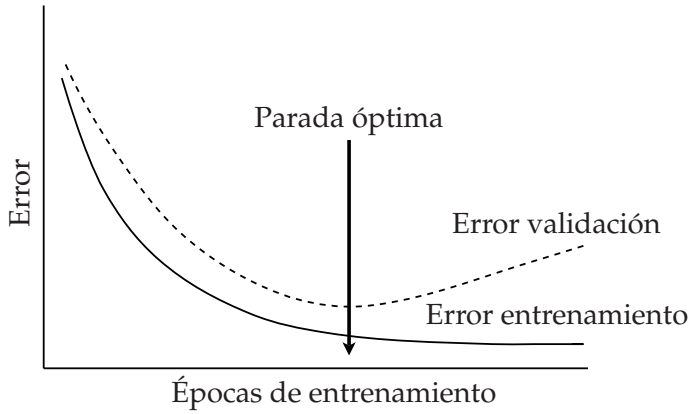


Figura 2.4: Relación entre el error proporcionado por las muestras de entrenamiento y las de validación. El punto de parada óptimo para evitar el sobreaprendizaje se produce para el menor valor del error de validación.

Selección de características

En algunos casos, puede ser necesario incluir una etapa previa para eliminar las características de entrada innecesarias. A este proceso se le conoce como selección de características. Téngase en cuenta, que en problemas reales, usualmente disponemos de muchas características de entrada, de las cuales, nos es difícil encontrar las que son relevantes para el aprendizaje de una determinada tarea y discriminar características redundantes con información contenida en otras. En estos casos, es importante quedarnos con un subconjunto de las características de entrada donde se han eliminado las redundantes e irrelevantes [91, 12, 9, 26, 35, 60].

2.3. Aprendizaje inductivo

En el aprendizaje humano, cuando aprendemos una determinada tarea, recibimos una gran cantidad de información procedentes de múltiples variables. Sin embargo, somos capaces de discriminar de una manera bastante eficiente lo que nos es realmente importante y descartar la información irrelevante. Sin embargo, para el aprendizaje máquina, este proceso es bastante costoso y no siempre se obtienen buenos resultados. Este es el caso de la clasificación de patrones, cuyo objetivo es construir un sistema capaz de etiquetar correctamente un patrón extraído del problema de clasificación a resolver. Dicho problema está representado por un conjunto de patrones etiquetados con la clase a la que pertenecen. Este conjunto de pares patrón-etiqueta que se van a utilizar para que una máquina aprenda se denomina conjunto de *entrenamiento*, mientras que los que se van a utilizar para comprobar la capacidad de generalización de la máquina se denomina conjunto de *test*. Como se ha definido antes,

todo el conjunto de las posibles soluciones a una determinada tarea se conoce como el conjunto de hipótesis y para el aprendizaje máquina es necesario reducir de una forma inteligente ese espacio utilizando algún tipo de estrategia o heurístico [45]. Cualquier restricción del espacio de hipótesis implica lo que se conoce como sesgo inductivo. Las Figuras 2.5 y 2.6 muestran la diferencia de aplicar sesgo inductivo en el proceso de aprendizaje. En la primera, se muestra un esquema de aprendizaje máquina inductivo, donde se construye un modelo de clasificación, entendiéndose como tal a una máquina entrenada para clasificar una tarea, siendo el *dominio* el conjunto de todas estas tareas. A partir de un problema a resolver por un conjunto de datos, se construye un modelo de clasificación a partir del conjunto de entrenamiento y un proceso aprendizaje. Este proceso de aprendizaje que permite construir el modelo es lo que se conoce como *aprendizaje inductivo*. Posteriormente, se evalúa su capacidad de generalización mediante los patrones de test. Sin embargo, en la segunda figura aparece el sesgo inductivo gracias al conocimiento del dominio y esto afectará al aprendizaje. Se puede observar como este conocimiento del dominio puede venir de las propias muestras con las que se entrena o/y del modelo de clasificación que se está creando durante el aprendizaje. Esta transferencia de información hace que una máquina prefiera unas soluciones (hipótesis) sobre otras. Este sesgo inductivo puede ser positivo o negativo, según ayude o no al aprendizaje [98]. El sesgo inductivo positivo es esencial para la obtención de una hipótesis con buena generalización en el proceso de aprendizaje.

2.3.1. “Hints”

El concepto de “hints”, fue utilizado por Abu Mostafa para referirse a toda información auxiliar acerca del problema a resolver que se pueda utilizar para

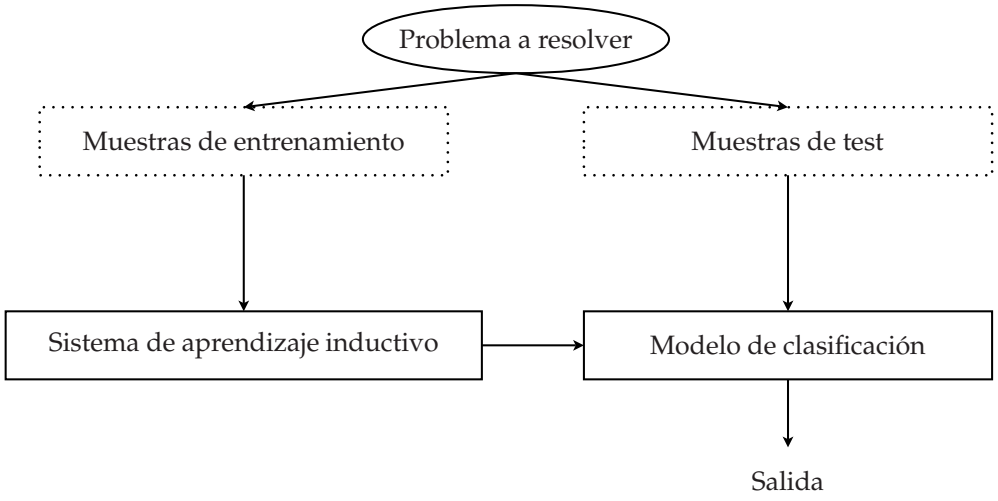


Figura 2.5: Esquema básico de aprendizaje inductivo. El aprendizaje inductivo permite crear un modelo de clasificación a partir de los patrones de entrenamiento. Los patrones de test nos permitirán evaluar la capacidad de generalización del modelo.

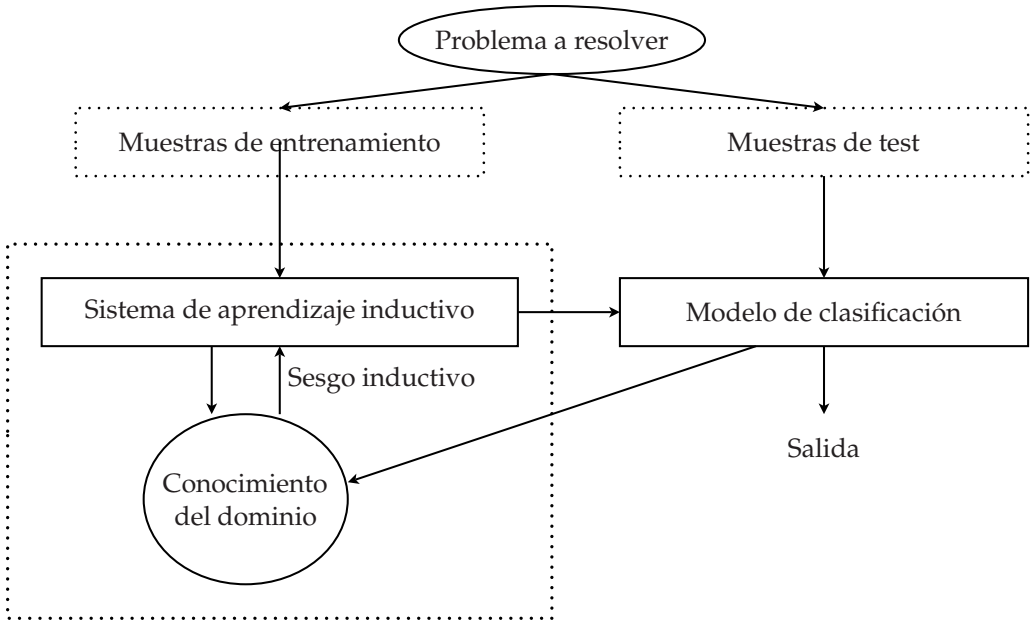


Figura 2.6: Esquema de aprendizaje utilizando sesgo inductivo. Este esquema está basado en la ayuda que proporciona el conocimiento auxiliar que incorpora el sesgo inductivo gracias al conocimiento del dominio.

guiar su aprendizaje [1, 2]. Por lo tanto, esa información auxiliar contribuirá de forma positiva hacia la obtención de una hipótesis correcta, incorporando un sesgo inductivo positivo a ese proceso de aprendizaje.

Esta idea de aprendizaje con "hints", fue formalizada por Baxter, denominándolo aprendizaje paralelo [11], y posteriormente, Caruana lo denominó Aprendizaje Multitarea, MTL ("Multi-Task Learning"), demostrando como un conjunto de tareas elegidas de un mismo dominio pueden ser utilizadas para aprender una representación interna común que será útil para el aprendizaje de tareas de ese mismo dominio [24, 21, 23, 20]. Para una máquina de aprendizaje, esta representación interna viene definida por la primera capa oculta, mientras que el dominio es el conjunto de tareas que utilizan los mismos atributos de entrada. Baxter y Caruana han demostrado el éxito del entrenamiento en paralelo de varias tareas sobre problemas reales y artificiales; así por ejemplo, lo han aplicado en ecuaciones booleanas, reconocimiento de objetos y diagnosis médica [20, 10, 22].

Para entrenar una red neuronal con "hints", se requiere la minimización de una función del error $E(h, f)$ que es una función de los errores individuales de cada "hints". Silver [94] utiliza medidas de relación para calibrar el efecto que produce en el aprendizaje de una tarea la utilización de varios "hints". Para poder extraer los "hints", se necesita un amplio conocimiento de las tareas y su dominio, por lo que no siempre es factible. En [2], Abu Mostafa formaliza la utilización de "hints" para reducir la dimensión Vapnik-Chervonenkis que mide el tamaño del espacio de hipótesis, [38]. Además, también desarrolla un modelo matemático para el aprendizaje de una tarea (que denomina *principal*) junto con "hints". Abu Mostafa define una estimación del error de la tarea principal como: $E = f(E_0, E_1, ..., E_k, ..., E_t)$ donde E_k es el error que presentan los

patrones de entrenamiento para cada "hints", siendo k=1,...t y E_0 el error de la tarea principal de los patrones de entrenamiento. De esta manera el error de la tarea principal se ve afectado por los de los "hints".

Tal y como lo hizo Caruana y posteriormente Silver, se utilizará un perceptrón multicapa (MLP) con múltiples salidas donde cada salida representará un "hint". En [97], Silver utiliza una forma de crear "hints" a partir de la tarea principal denominándolo "tareas de ensayo", TRM ("Task Rehearsal Method"). Silver se basa en [86] donde ya se presentaba un método pseudo-ensayo sobre ejemplos virtuales, y en los trabajos de Caruana sobre aprendizaje multitarea. EL TRM consiste en un esquema de aprendizaje multitarea donde cada "hints" es creado mediante el entrenamiento de un MLP con una única salida y las mismas entradas. Al entrenar todos los MLPs, con distinta inicialización de pesos, pero con los mismos conjuntos de entrada, producen salidas similares con un pequeño margen de error. Estas salidas se pueden utilizar como un "hint" ya que obviamente, por la naturaleza en que son creadas, están relacionadas con la tarea a aprender.

Siendo E la función objetivo a minimizar por el algoritmo de BP, para un nodo de salida k tenemos:

$$\triangle w_{ij} = \mu \frac{\delta \hat{E}}{\delta E_k} \frac{\delta E_k}{\delta w_{jk}} \tag{2.1}$$

donde μ es la tasa de aprendizaje, $\frac{\delta \hat{E}}{\delta E_k}$ es la tasa del cambio de la estimación del error verdadero para la tarea principal con respecto de la tasa de cambio del error para la tarea k. Esta tasa puede ser considerada como la medida de relación que cada "hints" aporta para el aprendizaje de la tarea principal (R_k). Esta medida se ve afectada por el valor de cada E_k, por lo Abu Mostafa explora métodos estáticos o dinámicos para determinar los valores apropiados

para cada E_k. Presenta un algoritmo llamado AMS ("Adaptative Minimization Schedule") que equilibra la información contenida dentro de los diferentes "hints" relacionando el entrenamiento de cada un de ellos con respecto al total del error estimado $\hat{E}$, de tal forma que se le da mayor importancia al "hint" que más contribuya al valor de $\hat{E}$. De esta manera, se asegura que todos los "hints" contribuyan favorablemente al aprendizaje de una tarea que denominamos tarea principal, consiguiendo una transferencia de información positiva desde los "hints" (que actúan como tareas secundarias) hacia ella. Posteriormente Silver amplia este trabajo profundizando notablemente sobre las medidas de relación entre tareas (R_k, donde k es el índice de la tarea), estableciendo valores para los R_k que permitan la transferencia de un sesgo positivo entre las tareas secundarias y la principal, así como el análisis del problema que presenta la elección de tareas no relacionadas [94]. De esta forma, se ponderan cada una de las tareas secundarias, dándole mayor importancia (mayor valor para R_k) a aquellas que guardan mayor relación con la tareas principal. Silver demuestra la mala influencia (sesgo negativo) que puede proporcionar una tarea que no guarde relación con la principal, en esos casos, la tasa R_k valdrá cero.

Un "hint" positivo reduce el espacio de soluciones centrando el aprendizaje en las mejores hipótesis; mientras que un "hint" negativo provocará que el espacio de soluciones sea mayor o no contenga solución válida y, por tanto, no beneficiará el proceso de aprendizaje. Un "hint" adecuado es aquel que le permite al algoritmo encontrar las soluciones del modelo que producen una mayor capacidad de generalización. En caso de no elegirse de forma adecuada, las prestaciones de la máquina serán probablemente sub-óptimas, ya que su representación interna es la que establece el espacio de posibles soluciones.

2.3.2. Tareas relacionadas

Al aprender un conjunto de tareas de forma simultánea, no tenemos garantía, a priori, que exista transferencia inductiva entre ellas, es decir, que una de ellas (la principal) se aprenda mejor, en relación a algún criterio, por el hecho de aprenderse conjuntamente con otras tareas (las secundarias). De igual manera, en el caso en que las tareas no estén relacionadas, se podría producir un sesgo inductivo negativo que dificultará el aprendizaje. Por lo tanto, si se va a realizar un aprendizaje multitarea, es esencial que las tareas que vamos a aprender estén relacionadas entre sí, y que esta relación se traduzca en un sesgo inductivo positivo que reduzca el espacio de soluciones. Esto suele ser un inconveniente en el mundo real ya que encontrar esa relación no siempre es una tarea fácil.

El éxito de la transferencia de información entre tareas vendrá determinado por el conocimiento que se tenga del dominio de las tareas y del método de aprendizaje empleado. Caruana explica la importancia de la relación entre tareas para el conocimiento basado en el aprendizaje inductivo. De esta forma, dos tareas a T_0 y T_K están relacionados si existe un algoritmo L que aprende T_0 mejor cuando se entrena conjuntamente con T_K [22].

Existe una conexión directa entre el concepto de sesgo inductivo positivo y relación entre tareas, ya que dos tareas están relacionadas si el aprendizaje simultáneo de ambas mediante una misma máquina produce un sesgo inductivo positivo [94]. Por otro lado, es importante distinguir entre relación y correlación estadística, ya que si las tareas a aprender están totalmente correladas no existirá una información adicional que guíe el aprendizaje de la tarea principal.

Silver y Mercer proponen métodos de selección de tareas antes y durante el aprendizaje de la tarea principal, [98, 94]. Estos métodos permiten medir

la relación entre ellas dentro de un determinado dominio, así se seleccionan aquellas que guardan más relación con respecto a la que se quiere aprender. Otros autores también proporcionan información sobre el cálculo de la relación entre tareas, [92, 32, 63]. Sin embargo, en la práctica no existe una única medida analítica que permita conocer el grado exacto de relación entre tareas.

La Figura 2.7 ilustra una representación conceptual de los diferentes espacios de soluciones o hipótesis en función de dos esquemas bastante diferenciados de aprendizaje: una tarea puede ser aprendida en solitario o conjuntamente con otras tareas. Cuando se aprende en solitario, la tarea no recibe ningún tipo de ayuda proveniente de otras tareas, este esquema recibe el nombre de aprendizaje STL ("Single-Task Learning"), mientras que cuando se aprenden varias tareas al mismo tiempo, estamos hablando de un esquema MTL tal y como se comentó en la Sección 2.3.1. En un esquema MTL, una de las tareas se considera tarea principal (T_P) mientras que el resto se consideran secundarias, siendo la misión de estas últimas favorecer el aprendizaje de la principal. Con ayuda de la Figura 2.7 es fácil comprender por qué las máquinas MTL poseen un espacio de posibles soluciones con mayor capacidad de generalización para la tarea principal, en el caso de emplear tareas secundarias relacionadas con la primera.

Considerar que H_{Opt} es la solución óptima para los parámetros de la máquina correspondiente al modelado de T_P. La Figura 2.7 muestra un ejemplo donde el espacio de soluciones o hipótesis (H_{STL}) de una máquina STL es reducido y puede no proporcionar una solución óptima para T_P, aunque proporcione una solución que a menudo tendrá un buen nivel de generalización. Por el contrario, una máquina MTL, con la misma representación interna, tendrá un espacio de soluciones mayor, llamado H_{MTL}, debido a la contribución de conocimiento

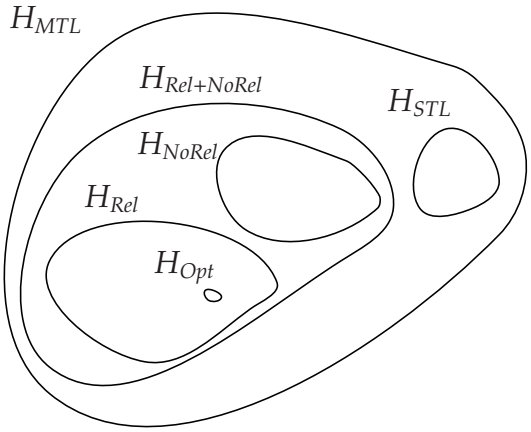


Figura 2.7: Espacio de soluciones para esquemas de aprendizaje STL y MTL. El uso de tareas relacionadas hace que el espacio de soluciones incluya la solución óptima para la tarea principal, mientras que las tareas no relacionadas dificultan el aprendizaje de dicha tarea.

que se produce entre las tareas que componen el dominio, incluyendo tareas relacionadas y no relacionadas con la tarea principal ($H_{Rel+NoRel}$). De esta manera, al modelar T_P junto con tareas secundarias relacionadas con la principal, se reduce el espacio de posibles soluciones al espacio llamado H_{Rel}, que incluye H_{Opt}. Por otro lado, el modelado con tareas secundarias no relacionadas con la principal lo reducen al espacio llamado H_{NoRel}. Este espacio de soluciones proporciona peores prestaciones finales para T_P, y no incluye la solución H_{Opt}. El uso de tareas secundarias ayuda a restringir el espacio de hipótesis, guiando y dirigiendo el aprendizaje a la solución correcta para la tarea principal. El aprendizaje será más orientado a H_{Opt} cuanto más relacionadas estén las tareas secundarias con la principal, y por el contrario, el uso de tareas no relacionadas generalmente dificulta el proceso de aprendizaje, obteniendo incluso peores resultados que en el caso en el que se aprende únicamente T_P. En algunas situaciones, puede ser necesario usar una representación interna mayor, es

decir, un mayor número de neuronas, para modelar problemas MTL que en el caso de utilizar un esquema STL.

Fundamentalmente, la transferencia inductiva de información puede mejorar los siguientes aspectos [19, 22]:

- La generalización y la velocidad de entrenamiento respecto a un esquema STL.
- Un esquema MTL presenta buenas prestaciones con un número de patrones reducido.
- Al aprender múltiples tareas simultáneamente, si una de ellas queda atrapada en un mínimo local, puede ser que la información que le aportan las demás tareas le proporcione suficiente inercia para abandonar el mínimo local y alcanzar el mínimo global.

Debido a que un esquema MTL elige unas soluciones frente a otras apoyándose en la información compartida de tareas comunes, es importante que para conseguir una transferencia efectiva de información entre tareas:

1. Las tareas secundarias deben estar tan relacionadas como sea posible con la tarea principal.
2. La relación existente debe ser tal que produzca un sesgo inductivo positivo.
3. La máquina MTL debe tener suficiente capacidad de representación interna, es decir, un número suficiente de neuronas en la capa oculta para resolver el problema planteado.

2.4. Transferencia inductiva en las ANNs

La transferencia inductiva es la transferencia de conocimientos por parte de otras tareas previamente aprendidas hacia una nueva que se quiere aprender. Por lo tanto, la transferencia inductiva entre tareas es una forma de sesgo inductivo. Si las tareas previamente aprendidas y la nueva pertenecen al mismo dominio, el conocimiento del dominio de las tareas previamente aprendidas se utiliza como sesgo inductivo que reduce el espacio de hipótesis favoreciendo al aprendizaje de la nueva tarea a aprender.

En los siguientes apartados, se explican dos formas de transferencia inductiva que presentan las redes neuronales.

2.4.1. Transferencia inductiva representacional y funcional

Hay dos formas de transferencia inductiva, que son la representacional y funcional, [94]. En la transferencia representacional, se hace un estudio previo de la tarea principal. Por ejemplo, aprender la tarea principal en solitario y los valores de sus pesos utilizarlos para otra ANN a entrenar. En consecuencia, la nueva red neuronal tiene un espacio de hipótesis donde se encuentran las mejores hipótesis para la resolución de la tarea principal. De esta manera, el conocimiento se transfiere antes del aprendizaje. La ventaja de usar este tipo de transferencia, es que puede reducir el coste computacional ya que se reduce el tiempo necesario para el aprendizaje de la red. Sin embargo, hay un problema con esta forma de transferencia inductiva: que la red se inicialice a un conjunto de pesos malos (por ejemplo, en la zona de atracción de un mínimo local de error elevado), lo que conducirá a un rendimiento ineficiente del modelo.

En contraste con la transferencia representacional, la transferencia funcional

no utiliza la representación previa de un aprendizaje anterior. En su lugar, se utilizan ejemplos adicionales de entrenamiento [2, 102], el aprendizaje paralelo de tareas relacionadas que comparten una representación interna común [11, 20], o la información obtenida de históricos de entrenamientos que nos permite utilizar ecuaciones para la actualización del los pesos o la modificación de la tasa de aprendizaje [72, 74, 105, 104]. La transferencia funcional también se utiliza para restringir el tamaño de espacio de búsqueda para el sistema de aprendizaje. En un aprendizaje en paralelo, cuando se inicia el aprendizaje, todas las tareas secundarias son aprendidas en paralelo con la tarea principal, de tal forma que todas las tareas contribuyen a la actualización de los pesos por igual. Por lo tanto, la tarea principal se deja influenciar por las demás, [95]. En la transferencia funcional, el conocimiento es transferido durante el entrenamiento. En esta Tesis, se utilizará la transferencia funcional bajo esquemas de aprendizaje MTL.

2.4.2. Enfoques para la transferencia inductiva usando ANNs

Los psicólogos descubrieron que, después de que los humanos desarrollan una función para resolver una determinada tarea, tienden a adquirir el conocimiento con una estructura que permita prepararse para el conocimiento de tareas futuras relacionadas. Este comportamiento es modelado como una red multicapa por Kehoe [58], consistente en dos funciones: una que representa un dominio de la tarea común que asigna atributos de entrada a una representación interna, y otra que representa una tarea específica que asigna la representación interna a los atributos de salida. Cada función representa una capa en la ANN. Para el aprendizaje de una tarea específica, el modelo elabora

un dominio común para satisfacer las necesidades de la tarea actual y hace ajustes para las tareas específicas. Este fue el concepto que desarrolló Caruana como aprendizaje MTL [19, 20]. Este aprendizaje utiliza la transferencia de conocimiento durante el aprendizaje de forma paralela entre varias tareas relacionadas, siendo la forma de transferencia inductiva mejor documentada de los métodos de aprendizaje en paralelo [23, 96].

2.5. Arquitectura de las ANNs Multitarea

La arquitectura para una red neuronal que presenta un esquema MTL es similar a la que presenta el esquema clásico STL, con la salvedad que en el esquema MTL tendremos una salida por cada tarea a aprender, mientras que en el STL se creará una red independiente por cada tarea (ver Figura 2.8). El modelo MTL no es diferente al que se vio en el aprendizaje con "hints", simplemente en el aprendizaje MTL se consideran que esos "hints" son tareas secundarias adicionales. De esta forma, cuando se habla de aprendizaje MTL nos estamos refiriendo a un aprendizaje donde hay una tarea principal y otras tareas que se consideran secundarias. Aunque todas las tareas se aprenden a la vez, sólo nos interesa una a la que llamaremos tarea principal, mientras que las otras se consideran tareas secundarias cuyo objetivo es ayudar y guiar el aprendizaje de la principal. Una vez entrenada la red MTL, las tareas secundarias se desechan y sólo se tiene en cuenta la tarea principal.

Es importante recordar, que en el esquema MTL, hay una parte común que comparten todas las tareas y una parte específica propia de cada una de ellas. La parte común, está formada por los pesos que conectan las características de entrada con la capa oculta, permitiendo una representación interna común para

todas las tareas, [19]. Gracias a esta representación interna, el aprendizaje puede beneficiarse del sesgo inductivo introducido al modelar simultáneamente dos o más tareas relacionadas, [22]. La parte específica, formada por los pesos que conectan la capa oculta con la capa de salida, permiten modelar específicamente cada tarea a partir de la representación común. El conjunto total de pesos, formado por la parte común y las partes específicas, constituye el espacio de posibles soluciones o hipótesis para los parámetros de la máquina MTL.

Aunque dependiendo del problema, puede ser interesante utilizar más de una capa oculta, esto añade un grado de complejidad que suele provocar grandes inconvenientes en la fase de diseño y entrenamiento de la arquitectura. De todas formas, como se ha demostrado ampliamente, una ANN con una capa oculta de neuronas es capaz de modelar cualquier función continua (aproximador universal), si los parámetros que definen la máquina son escogidos de manera correcta [46, 111].

Una parte importante de esta Tesis es la creación de un método que nos permita obtener una arquitectura para esquemas MTL en donde se eliminen las tareas secundarias no relacionadas, con lo que se evita la influencia negativa entre tareas. Esta arquitectura también realizará una selección de características, podará neuronas y conexiones irrelevantes favoreciendo el sesgo inductivo positivo hacia la tarea principal.

2.6. Ejemplo de aprendizaje MTL: Logic Domain

Con el objetivo de mostrar cómo trabaja un esquema MTL, se va a introducir en esta sección la definición y resultados de un problema de juguete que será utilizado en los siguientes capítulos. El conjunto original de datos del “Logic

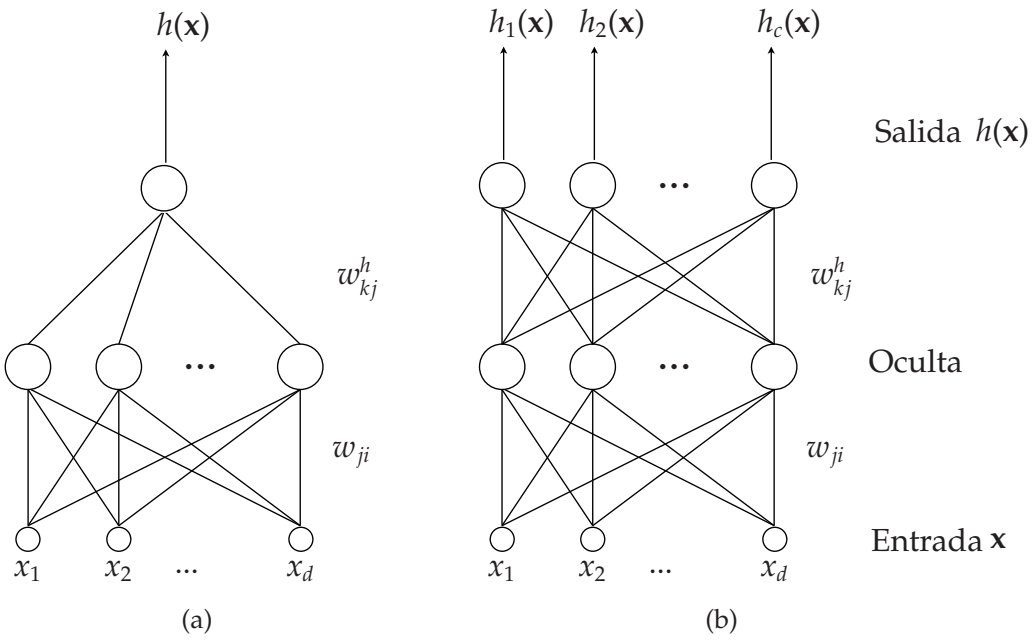


Figura 2.8: Diferentes esquemas de aprendizaje. En (a) se muestra un esquema donde una única tarea se aprende en solitario (STL), mientras que en (b) se aprenden un conjunto de tareas de forma simultánea (MTL), existiendo una zona común (desde la entrada a la capa oculta) y otra específica (desde la capa oculta a la de salida) de cada tarea.

Domain" introducido por Silver [98], consta de 8 funciones lógicas, siendo cada función una tarea a aprender. Cada tarea es una combinación lógica de 4 de entre 11 características de entrada binarias. La Tabla 2.1 muestra la expresión lógica de cada tarea. La tarea principal es f_0, mientras que el resto de tareas se consideran como secundarias. La tarea principal está compuesta por las 4 primeras características de entrada, las cuales son compartidas en diferente grado con las tres primeras tareas secundarias (f_1, f_2 y f_3). El resto de tareas no comparten ninguna entrada con la principal.

Tabla 2.1: Descripción de las tareas del "Logic Domain". Cada tarea es una expresión lógica formada por la combinación de cuatro características de entrada, estando cada una de ellas más o menos relacionada con la principal (T_0).

Tarea	Expresión Lógica
f_0	$(x_0 > 0.5 \vee x_1 > 0.5) \wedge (x_2 > 0.5 \vee x_3 > 0.5)$
f_1	$(x_1 > 0.5 \vee x_2 > 0.5) \wedge (x_3 > 0.5 \vee x_4 > 0.5)$
f_2	$(x_2 > 0.5 \vee x_3 > 0.5) \wedge (x_4 > 0.5 \vee x_5 > 0.5)$
f_3	$(x_3 > 0.5 \vee x_4 > 0.5) \wedge (x_5 > 0.5 \vee x_6 > 0.5)$
f_4	$(x_4 > 0.5 \vee x_5 > 0.5) \wedge (x_6 > 0.5 \vee x_7 > 0.5)$
f_5	$(x_5 > 0.5 \vee x_6 > 0.5) \wedge (x_7 > 0.5 \vee x_8 > 0.5)$
f_6	$(x_6 > 0.5 \vee x_7 > 0.5) \wedge (x_8 > 0.5 \vee x_9 > 0.5)$
f_7	$(x_7 > 0.5 \vee x_8 > 0.5) \wedge (x_9 > 0.5 \vee x_{10} > 0.5)$

Está claro que para que las tareas estén relacionadas con la principal, han de compartir atributos de entrada, y mientras más compartan, potencialmente serán mejores candidatos para aprenderse junto a la principal en el esquema MTL. Sin embargo, también es importante que compartan la representación interna de cada una de las tareas para la transferencia de información entre ellas. Esto se ve claramente observando que f_0 y f_2 comparten la expresión lógica ($x_2 > 0.5 \vee x_3 > 0.5$), mientras que f_0 y f_1, aunque comparten más características de entrada (x_1, x_2 y x_3), éstas entran en conflicto ya que cada

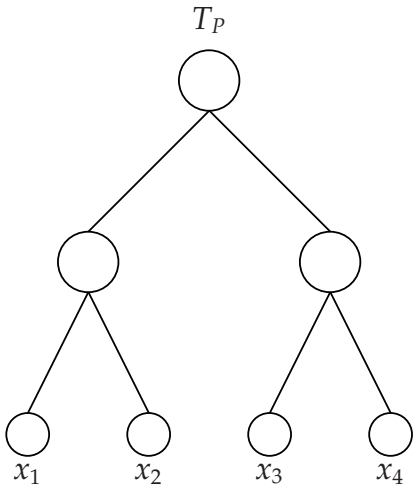


Figura 2.9: Arquitectura óptima para la tarea principal del "Logic Domain".

tarea realiza con ellas diferentes funciones lógicas.

Estas tareas no son linealmente separables y necesitan al menos dos nodos ocultos para formar una representación interna razonable (Figura 2.9). Esto sugiere que una red MTL óptima tendría diez nodos en la capa oculta [98], esto es, para cada tarea, se necesitarían dos nodos ocultos para realizar la disyunción y un nodo de salida para la conjunción. Observando la segunda tarea secundaria, se ve claramente que guarda más relación con la tarea principal, al compartir la estructura interna formada por las características de entrada x_2 y x_3. Esto hará que proporcione un sesgo inductivo positivo que favorezca el aprendizaje de la tarea principal.

Esta relación entre las diferentes tareas con respecto a la principal se puede ver claramente en la Tabla 2.2, donde se muestra la probabilidad de acierto para clasificación de tres esquemas diferentes de aprendizaje. Por un lado, se ha utilizado un esquema de entrenamiento de la tarea principal (f_0) en solitario (STL); después, se ha entrenado un esquema MTL clásico donde se han utilizado las ocho tareas de forma conjunta; y finalmente, se ha entrenado la tarea

principal con cada una de las otras tareas secundarias, de forma individual, en un esquema MTL. Se puede observar como el esquema MTL clásico no mejora el aprendizaje de la tarea principal. Esto se debe a que hay más tareas no relacionadas que relacionadas con la principal, causando interferencias en la transferencia de información hacia la tarea principal. También se puede ver como la utilización de un esquema MTL sólo con la segunda tarea secundaria es la que provoca menor error de clasificación, tal y como se ha mencionado antes, ya que es la que más similitud tiene con respecto a la tarea principal.

Este ejemplo permite ver, de una forma clara y sencilla, la importancia de la relación entre las tareas y la importancia que la arquitectura de la red puede ejercer en el aprendizaje MTL. En los siguientes capítulos volveremos a este ejemplo para ver cómo se comporta frente a los algoritmos propuestos en esta Tesis.

Tabla 2.2: Probabilidad de acierto para clasificación del conjunto de test (media ± desviación estándar) para una arquitectura de 6 neuronas en la capa oculta, para los esquemas *STL*, *MTL* y $MTL_{tareasecundaria}$ para el problema del "Logic Domain"

Esquema	Prob. acierto para clasif.
STL	0.428 ± 0.011
MTL	0.397 ± 0.068
MTL_{T_1}	0.498 ± 0.015
MTL_{T_2}	0.575 ± 0.032
MTL_{T_3}	0.420 ± 0.011
MTL_{T_4}	0.406 ± 0.009
MTL_{T_5}	0.302 ± 0.109
MTL_{T_6}	0.409 ± 0.035
MTL_{T_7}	0.285 ± 0.112

Capítulo 3

Hints are pieces of information about an unknown function that we wish to learn.

(Abu-Mostafa)

Generación artificial de "hints" basado en la Edición de Datos.

3.1. Introduction

En la clasificación de patrones, una tarea está representada por un conjunto de datos etiquetados. Las etiquetas representan las clases a la que pertenecen dichos datos. En el aprendizaje clásico, una máquina aprende en solitario una tarea sin ningún tipo de ayuda proveniente del conocimiento previo de otras tareas que guarden relación con ella, STL ("Single-Task Learning"). Sin embargo, en el aprendizaje Multi-Tarea, MTL ("Multi-Task Learning") se aprenden diferentes tareas al mismo tiempo [24, 21, 23, 20]. Este tipo de aprendizaje es un método de transferencia de conocimiento que aumenta la generalización

en el aprendizaje de la máquina que aprende la "tarea principal" usando la información implícita en otras tareas llamadas "tareas secundarias". Esta información adicional introducida en el aprendizaje de una tarea se conoce como "hint" y fue introducido por Abu-Mostafa [1, 2]. La transferencia inductiva de información hacia la tarea principal está condicionada por su relación con las tareas secundarias. De esta forma, la relación existente entre la tarea principal y las tareas secundarias debe ser tal que se produzca un sesgo positivo en la solución, al mismo tiempo que se reduce el espacio de soluciones [98].

En problemas reales, usualmente no se dispone de tareas relacionadas para poder aprender una tarea utilizando un esquema MTL. También pudiera ocurrir que la relación entre las tareas sea incierta, con lo que se haría difícil la selección de las tareas secundarias más convenientes para el mejor aprendizaje de la tarea principal. Hay que tener en cuenta que una tarea no relacionada con la principal no sólo no le va a ayudar, sino que incluso podría perjudicar su entrenamiento ya que incorporaría al proceso de aprendizaje un sesgo inductivo negativo.

En este capítulo, se presenta un nuevo método de generación de tareas secundarias relacionadas con la principal y que benefician su aprendizaje dentro de una arquitectura MTL. Estas tareas extras se obtienen mediante procedimientos de selección de muestras, en particular, mediante Edición de Datos. Los resultados obtenidos sobre problemas artificiales y reales, demuestran las ventajas del método propuesto. En particular, se consigue una convergencia más rápida así como una reducción en la probabilidad de quedar atrapado en un mínimo local.

El resto del capítulo está organizado de la siguiente manera. Las Secciones 2 y 3 introducen respectivamente los procedimientos de Edición de Datos y

el método propuesto de generación de tareas para aprendizaje MTL. En la Sección 4, se presentan los resultados del método propuesto para problemas de clasificación y la discusión de los mismos. La Sección 5 muestra el empleo del método propuesto en un esquema MTL clásico, y finalmente, en la Sección 6, las conclusiones y trabajos futuros.

3.2. Edición de Datos

En muchos problemas de clasificación, sólo necesitamos un pequeño subconjunto del problema original para poder obtener una solución precisa de la frontera de decisión. En estos casos, la Edición de Datos ("Data Editing") proporciona una herramienta eficaz para obtener estos subconjuntos, reduciéndose así los requerimientos de memoria y coste computacional.

La idea de la Edición de Datos fue introducida por Wilson [112] al intentar reducir el alto coste computacional de la regla *K*-Nearest Neighbors (*K*-NN). Posteriormente, se ha realizado muchos trabajos al respecto, de entre los que sobresalen [112, 107, 82, 29, 28, 85, 87]. Uno de los algoritmos de Edición de Datos más representativos es el Condensed-NN propuesto por Hart [29, 41], y que consta de las siguientes etapas:

1. Se dividen las muestras en dos conjuntos, los llamados almacén y cajón de sastre. Una posible partición consiste en poner una primera muestra elegida aleatoriamente en el almacén y el resto en el cajón.

2. Se clasifica cada muestra en el cajón por la regla 1-NN usando el almacén como conjunto de entrenamiento. Si la muestra no se ha clasificado correctamente se pasa al almacén.

3. Se vuelve al paso 2 a menos que no se transfiera nada o el cajón esté vacío.

4. Se devuelve el almacén como conjunto editado.

La selección de muestras proporcionada por el algoritmo Condensed-NN puede ser usada como una nueva tarea secundaria, ya que en la mayoría de las aplicaciones reales de clasificación de patrones existen zonas de solapamiento entre clases, y en estos casos, el algoritmo Condensed-NN eliminaría muestras innecesarias, desde el punto de vista de la clasificación situadas en esas zonas. Una forma de obtener un conjunto más representativo y reducido del problema, consiste en establecer una etapa previa a la condensación que elimine esas muestras de solape. Para ello, como el *K*-NN no se puede utilizar para quitar muestras en la zona de solape, se utiliza una extensión de ella proporcionada por el algoritmo {*K*,*L*}-NN, [44] . Dada una muestra **x**, este algoritmo comienza buscando las *K* muestras más cercanas a **x**. Entonces, **x** se mantiene si al menos hay *L* muestras de la misma clase que **x**. De lo contrario, la muestra se desecha. Por lo tanto, {*K*,*L*}-NN se utiliza previamente al algoritmo Condensed-NN.

La Figura 3.1 explica el procedimiento de Data Editing descrito. Cada vez que aplicamos el algoritmo Condensed-NN obtenemos un conjunto distinto de muestras. Por lo tanto, empleando diferentes valores de *K* y *L* para el algoritmo {*K*, *L*}-NN, se obtienen diferentes conjuntos de datos que utilizaremos como "hints". La siguiente Sección describe este método.

3.3. Método propuesto

Como se ha visto anteriormente, para que el aprendizaje MTL sea efectivo, es necesario que todas las tareas secundarias estén relacionadas con la principal, y de esta manera, actúen como sesgos inductivos positivos. En muchos casos,

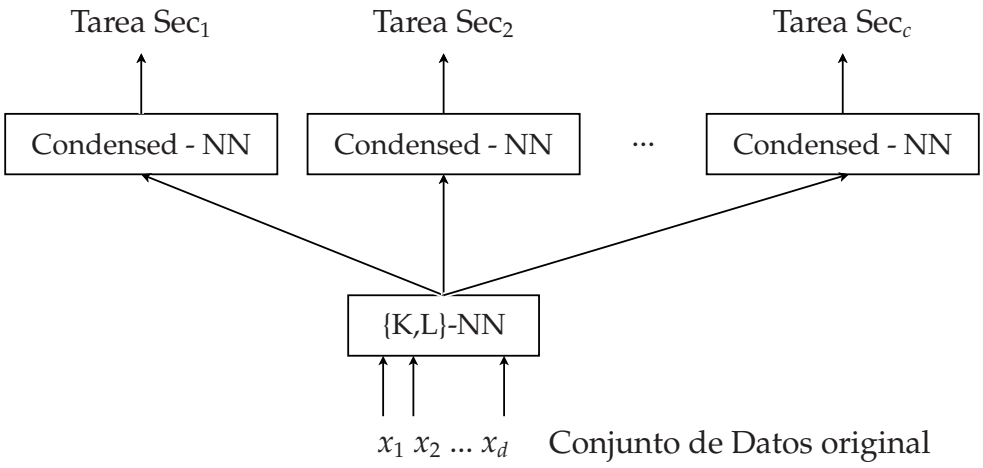


Figura 3.1: Procedimiento de Edición de Datos para obtener los "hints". Estos "hints" se obtienen a partir de la tarea principal usando los algoritmos {*K*, *L*}-NN y Condensed-NN.

encontrar tareas relacionadas es bastante difícil. La limitación en el número de muestras que presentan los problemas reales junto con las dificultades en reconocer la relación entre tareas, hacen que en muchas ocasiones se descarte un aprendizaje MTL, y finalmente, la tarea se aprenda en solitario.

Una forma de crear tareas artificiales asociada a una tarea principal es mediante el empleo de un procedimiento de Edición de Datos, combinando los algoritmos {*K*, *L*}-NN y Condensed-NN como se ha descrito en la sección anterior. Estos nuevos "hints" están representados por conjuntos de datos más pequeños que aproximan la frontera de decisión cuando se aplica la regla 1-NN. Además, la región de solape se ha reducido por medio de la utilización de la regla {*K*,*L*}-NN, por lo que, en general, estas muestras serán más fáciles de aprender, estando relacionadas con la tarea principal por la forma en que se obtuvieron [17, 18]. La información proporcionada por las tareas secundarias trabaja como un sesgo inductivo para la tarea principal, reduciendo así el conjunto de las posibles soluciones.

Dada una tarea (principal) a resolver, ésta estará determinada por un conjunto de muestras $\mathbf{x}^n$ y un conjunto de salidas deseadas $f(\mathbf{x}^n)$. A su vez, cada patrón $\mathbf{x}^n$ estará compuesto por d componentes, llamadas características, $\mathbf{x}^n = x_1^n, x_2^n, ..., x_d^n$.

El objetivo para STL es encontrar una hipótesis h, contenida en el espacio de hipótesis H_{STL}, que minimice la función objetivo:

$$\sum_{x \in S_{STL}} error[f(\mathbf{x}), h(\mathbf{x})], \tag{3.1}$$

donde S_{STL} representa un conjunto de entrenamiento para el esquema STL.

Análogamente, para un esquema con "hints", el objetivo consistirá en encontrar un conjunto de hipótesis $\mathbf{h} = h_1, h_2, ..., h_c$, siendo c el número de tareas en el espacio de hipótesis, H_{hint}, que minimice la función objetivo:

$$\sum_{x \in S_{hint}} \sum_{i=1}^{n} error[f_i(\mathbf{x}), h_i(\mathbf{x})], \tag{3.2}$$

donde S_{hint} representa un conjunto de entrenamiento para un esquema con "hints". La Figura 3.2 representa el esquema con "hints" utilizado en esta Tesis, donde todas las tareas (cada una de ellas está asociada a una salida de la red) comparten una única capa oculta de neuronas. Es una red multicapa unidireccional ("feed-forward") con una salida por cada tarea que ha de ser aprendida. Cada tarea secundaria trabaja como un "hint" que ayuda a la principal mediante la aportación de información adicional que favorecerá su aprendizaje. Para ello, es necesario que las tareas secundarias estén relacionadas con la principal, y que además, lo estén de forma que produzcan un sesgo inductivo positivo.

Un esquema con "hints" contiene múltiples hipótesis, $H_{hint} = \{H_{TP}, H_{Tsec}\}$, es decir, H_{hint} está representado por el conjunto de hipótesis de la tarea principal y por el de las tareas secundarias [99]. En nuestro esquema, por la forma de obtener las tareas secundarias, es de esperar que estas estén estrechamente relacionadas con la principal, contribuyendo positivamente hacia la obtención de una buena hipótesis.

En un esquema con "hints" clásico, a cada vector de entrada le corresponde un vector de salida deseada formado por una parte correspondiente a la tarea principal y otra a las tareas secundarias. Sin embargo, al utilizar la Edición de Datos para generar las tareas secundarias, nos encontramos con que hay vectores de entrada que no tienen representación en forma de salida deseada para algunas tareas secundarias, ya que no pertenecen a ella, es decir, el algoritmo de Edición de Datos no las seleccionó (Figura 3.3). Sin embargo, es obvio que toda muestra pertenece a la tarea principal por lo que hay que hacer algunas

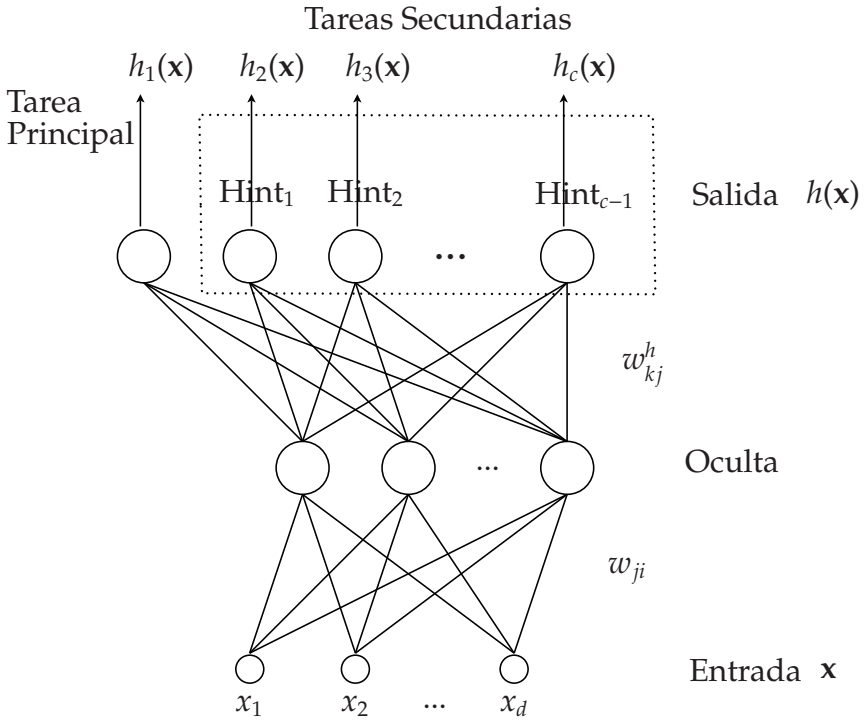


Figura 3.2: Aprendizaje con "hints". Cada tarea secundaria actúa como un "hints", ya que al forzar su aprendizaje simultáneamente al de la tarea principal, se producirá una transferencia de conocimiento de unas tareas a otras.

modificaciones del algoritmo original de BP ("Back-Propagation").

Para tener en cuenta esta particularidad, se utiliza la función de activación lineal en las unidades de salida, considerando como la función de error SSE ("Sum-of-Squares Error') para ser minimizada durante el proceso de aprendizaje de c tareas. El error correspondiente a la n-th muestra viene dado por:

$$E^n = \frac{1}{2}\sum_{k=1}^{c}(h(\mathbf{x}_k^n) - f(\mathbf{x}_k^n))^2, \tag{3.3}$$

donde $h(\mathbf{x}_k^n)$ es la salida producida por $\mathbf{x}_k^n$ correspondiente a la k-th tarea, $f(\mathbf{x}_k^n)$ son las salidas deseadas de las $\mathbf{x}^n$ correspondientes a las k-th tareas, y $c-1$ es

el número de tareas secundarias. Por simplicidad, se va a considerar que tanto la tarea principal como las secundarias van a tener una única salida.

Como ya se ha mencionado, los pesos de la red se calculan usando el algoritmo BP. El peso w_{ji} que va desde la unidad de entrada i a la unidad de la capa oculta j; w^h_{kj} es el peso de la segunda capa que va desde la neurona j de la capa oculta a la unidad k de la capa de salida. Las derivadas de (3.3) con respecto a los pesos de la primera capa y segunda capa son respectivamente,

$$\frac{\partial E^n}{\partial w_{ji}} = \delta^n_j x^n_i, \qquad \frac{\partial E^n}{\partial w^h_{kj}} = \delta^h_k z^n_j, \tag{3.4}$$

donde z^n_j es la salida de la j-th neurona de la capa oculta correspondiente a la muestra $\mathbf{x}^n$, δ^n's son los errores producidos en la capa de salida y δ^h's los ocurridos en la capa oculta. En ambas capas se ha utilizado la función de activación sigmoide (logística).

Para abordar la particularidad comentada anteriormente sobre el hecho de que hay $\mathbf{x}^n$ que no tienen valores en ciertas componentes del vector $f(\mathbf{x}^n)$, se introduce un parámetro α^n_k que representa la pertenencia de $\mathbf{x}^n$ a la tarea k-th. De esta forma, los pesos que conectan cada unidad de salida con las neuronas de la capa oculta se ven influenciados por el parámetro α^n_k antes mencionado, según

$$\delta^n_k = (h(\mathbf{x}^n_k) - f(\mathbf{x}^n_k))\alpha^n_k, \tag{3.5}$$

Este sesgo α^n_k toma el valor 0 ó 1 dependiendo si la muestra n-th pertenece o no al conjunto editado que forma la tarea k-th. Lógicamente, α^n_1 es igual a 1 para todo n, pues todas las muestras pertenecen a la tarea principal. La Figura 3.3 muestra este concepto.

Así, los pesos de la primera capa se actualizan dependiendo del error de

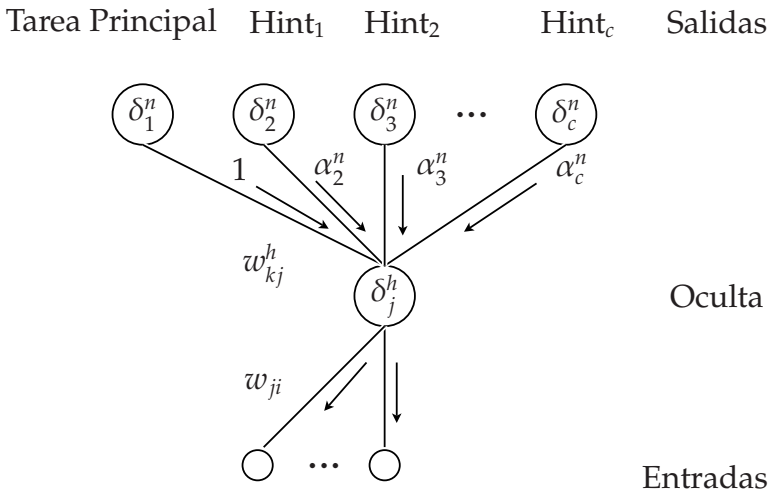


Figura 3.3: El método BP sufre algunas modificaciones. Las tareas secundarias son un subconjunto de la principal, por lo que algunas muestras no pertenecen a una determinada tarea secundaria. Cuando esto ocurre, α_k^n toma el valor 0 y el error δ_k^n no se propaga. Para la tarea principal α_1^n toma el valor 1 para cualquier muestra, ya que todas las muestras pertenecen a la tarea principal.

todas las tareas:

$$\delta_j^n = z_j^n(1 - z_j^n) \sum_{k=1}^{c} \frac{w_{kj}^h \delta_k^n}{l_k}, \tag{3.6}$$

donde l_k es la cardinalidad del conjunto de muestras que forma la tarea k-th, que se introduce como factor de normalización. Esta normalización es necesaria debido a que las tareas secundarias (conjuntos editados) son tareas con un número reducido de muestras con respecto a la tarea principal (conjunto original). De esta manera se normaliza el aprendizaje de las tareas secundarias respecto a la tarea principal ya que los conjuntos están claramente desbalanceados.

El empleo de tareas secundarias obtenidas mediante la selección de muestras procedentes de la Edición de Datos presenta tres ventajas relativas al diseño, velocidad de convergencia y calidad de la solución. Esto es, debido

a que las tareas secundarias están representadas por conjuntos pequeños de muestras próximas a la frontera de decisión, estas pueden ser utilizadas para seleccionar la arquitectura con un bajo coste computacional ya que la la frontera producida por estos conjuntos son una aproximación a la producida por el conjunto completo. La velocidad de convergencia también se ve reducida debido a que las tareas secundarias aceleran el aprendizaje de la principal, ya que estas convergen rápidamente y reducen considerablemente el número de épocas de entrenamiento de la tarea principal. La calidad de la solución se suele ver mejorada ya que el método propuesto reduce la probabilidad de caer en mínimos locales debido a la convergencia rápida de las tareas secundarias. A continuación, explicaremos en más detalle estas ventajas utilizando para ello problemas artificiales y reales para confirmar el buen funcionamiento del método.

3.4. Experimentos

Este trabajo ha sido testado sobre conjuntos de datos artificiales y reales. Para explicar en mayor detalle como se extraen las tareas secundarias mediante la Edición de Datos, se ha utilizado el conjunto de datos empleado por Ripley en [85]. Otro conjunto artificial se va a utilizar para comprobar cómo el método propuesto contribuye a evitar mínimos locales. Además, se han seleccionado seis problemas reales pertenecientes a la "UCI Machine Learning Repository" (Pima Indians Diabetes, Heart Diseases, Ionosphere, Wisconsin Diagnostic Breast Cancer, Abalon y Forest CoverType)[5].

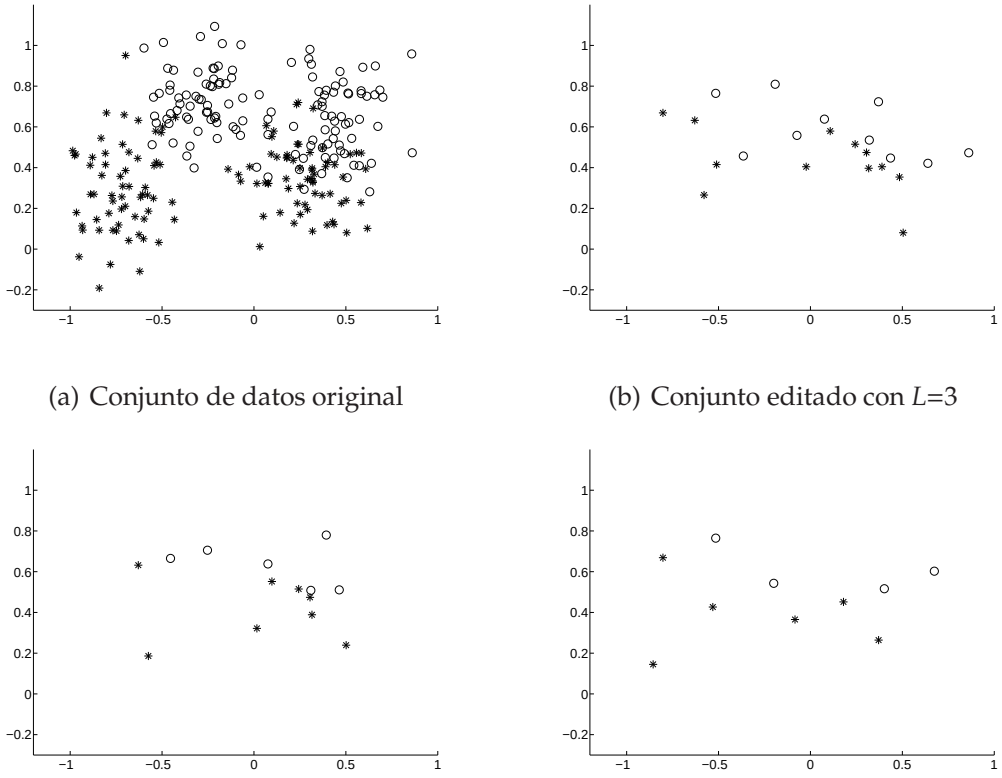
Para la elección de la arquitectura de cada esquema y conjunto de datos, se ha utilizado validación cruzada. El método propuesto ha sido comparado

con el aprendizaje de cada tarea principal por separado (STL) para evaluar su bondad. En todos los experimentos, se ha utilizado la función de coste MSE ("Mean Squared Error") minimizada por el algoritmo BP con momento, repitiendo cada simulación 30 veces y utilizando validación cruzada con 10-fold. La tasa de aprendizaje es adaptativa [13], con un valor inicial de 0.05. El parámetro del momento se ha establecido a 0.9. Los pesos de ambas capas se han inicializado de forma aleatoria en un rango desde -0.1 a 0.1 [98].

3.4.1. Problemas artificiales

El conjunto de muestras de Ripley, [85] corresponde a un problema de dos clases en dos dimensiones con una probabilidad de error teórica de 0.08 (Figura 3.4(a)). El conjunto completo de datos está formado por 1500 muestras pertenecientes a una única tarea a aprender, donde no disponemos de ninguna tarea relacionada. Por lo tanto, usaremos la Edición de Datos para poder entrenar nuestra red utilizando un esquema Multi-Tarea. De esta forma, obtenemos un nuevo conjunto reducido de muestras a partir del conjunto original (Figura 3.4(b-d)). Cada nuevo conjunto representa una nueva tarea que será empleada como tarea secundaria ("hint").

Estas nuevas tareas secundarias, obtenidas de la principal, presentan una frontera de decisión 1-NN similar a la que presenta la tarea principal, ya que se han obtenido mediante una selección inteligente de las muestras más representativas (Figura 3.5). Como las tareas secundarias están representadas por un subconjunto muy reducido del original que no presenta solape entre muestras, su aprendizaje resulta fácil y rápido. En el esquema MTL, el espacio de soluciones de la tarea principal se reduce al aprenderse conjuntamente con las tareas secundarias, con lo que acelera su convergencia al aportar a la principal



(a) Conjunto de datos original

(b) Conjunto editado con L=3

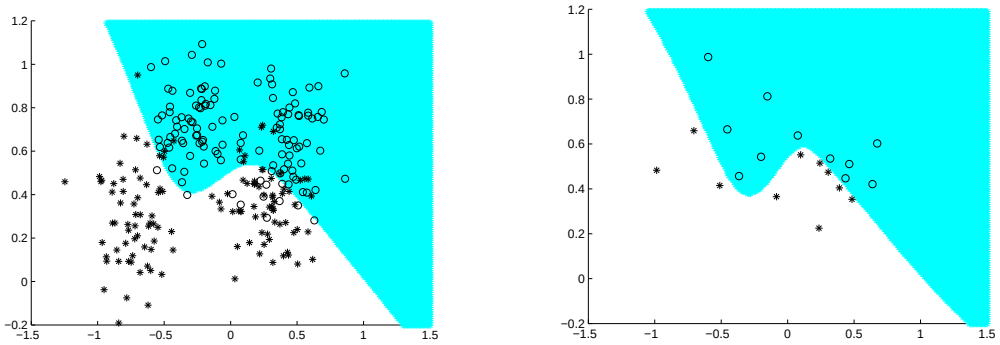
(c) Conjunto editado con L=5

(d) Conjunto editado con L=7

Figura 3.4: Ejemplo del procedimiento de Edición de Datos para la obtención de tareas secundarias. Se ha eliminado la zona de solapamiento mediante el algoritmo {K,L}-NN con diferentes valores de L (K=10). Posteriormente, se ha seleccionado un conjunto destacado de muestras utilizando el algoritmo Condensed-NN.

un sesgo inductivo positivo (Figura 3.6).

El hecho de que las nuevas tareas secundarias tengan una frontera de decisión 1-NN bastante parecida a la que ofrece la tarea principal puede ser utilizado para la selección de la arquitectura. La elección de la arquitectura apropiada en un MLP para la resolución de un problema de clasificación, implica, en general, un importante coste computacional proporcional al número de patrones. Las tareas secundarias que hemos obtenido mediante la Edición de Datos están



(a) (b)

Figura 3.5: Selección de la arquitectura. Para las dos simulaciones, conjunto completo de datos (a) y nuevo conjunto de datos obtenido mediante la Edición de Datos a partir del conjunto completo (b), se ha utilizado el mismo número de neuronas en la capa oculta, obteniéndose una frontera similar.

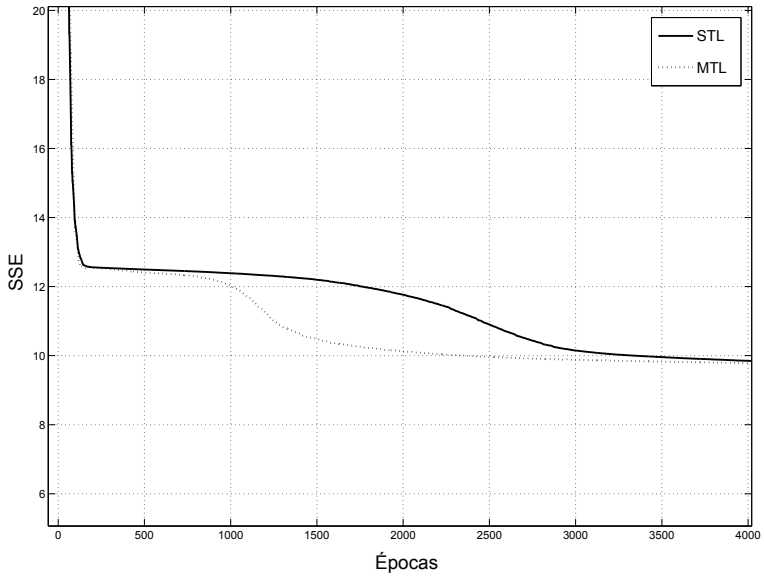


Figura 3.6: Relación del MSE (sobre el conjunto de datos de Ripley) de la tarea principal para los esquemas STL y MTL con Edición de Datos. Se puede apreciar como se acelera el aprendizaje de la tarea principal al utilizar un esquema MTL con tareas secundarias generadas mediante la Edición de Datos.

Tabla 3.1: Media de la probabilidad de acierto (sobre el conjunto de test) y desviación estándar sobre 30 simulaciones para diferentes arquitecturas entrenadas para aprender los datos de Ripley.

Esquema	Probab. de acierto± std	Épocas de entr.
STL	0.906 ± 0.003	4120
MTL con 1 Tarea Secundaria	0.912 ± 0.010	1524
MTL con 3 Tareas Secundarias	0.915 ± 0.008	927

determinadas por un conjunto muy reducido de patrones, lo cual supone un gran decremento en el coste computacional requerido para aprenderla. Como ya se ha comentado, en la Figura 3.5 se puede observar la similitud de las fronteras de decisión utilizando la misma arquitectura para ambos conjuntos de datos. De esta forma, el método propuesto utilizará las tareas secundarias para buscar el dimensionado de la capa oculta del esquema MTL.

En la Tabla 3.1 se han obtenido la media y desviación estándar de la probabilidad de acierto para clasificación sobre el conjunto de test sobre tres esquemas: un MLP donde se entrena en solitario (STL), y dos esquemas MTL con 1 y 3 conjuntos editados respectivamente ("hints" obtenidos de la tarea principal). Como se puede observar, las tareas secundarias ayudan al aprendizaje de la tarea principal y reducen el número de épocas de entrenamiento con respecto a un esquema STL, aumentando así la velocidad de convergencia.

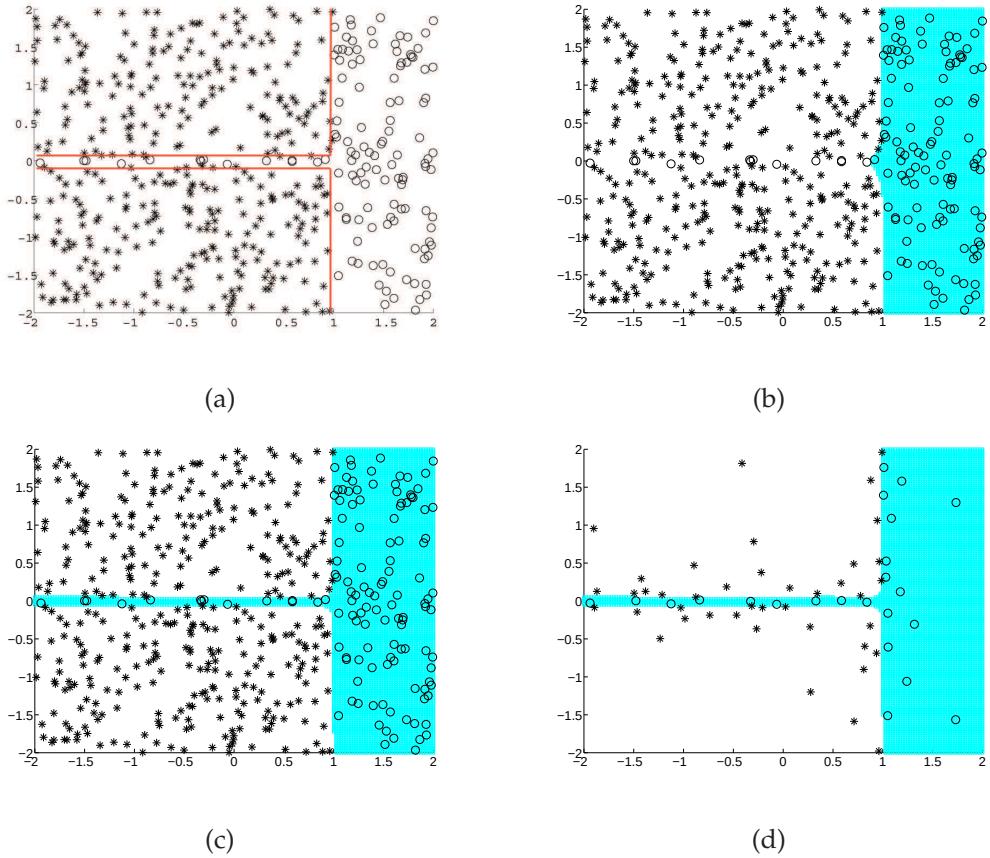
En [87], los autores proponen entrenar un MLP utilizando como conjunto de entrenamiento las muestras obtenidas mediante la Edición de Datos, en lugar de entrenar con el conjunto completo. Los resultados muestran una reducción del coste computacional mientras que la probabilidad de clasificación propuesta por el MLP es una muy similar utilizando todo el conjunto de datos. Estos métodos funcionan sólo con el conjunto editado, teniendo la desventaja de

que tienen que ajustarse sus parámetros mediante validación cruzada, lo cual supone un gran coste computacional. Sin embargo, el método propuesto en esta Tesis, no requiere de un proceso de búsqueda de parámetros (con el ahorro computacional que esto supone), mejorando, en general la probabilidad de clasificación. Además, otra ventaja que presenta es la eficacia a la hora de evitar los mínimos locales. Como ejemplo, se utilizará el problema mostrado en la Figura 3.7(a). Se trata de un problema de dos clases con distribución uniforme y en donde una de las clases se introduce en la otra mediante una ramificación delgada. Esto se traduce en un mínimo local en el que fácilmente queda atrapado el entrenamiento del MLP. Las tareas secundarias ayudarán a que el aprendizaje de la tarea principal no caiga en el mínimo local. En la Figura 3.7(b), se observa que en el entrenamiento STL, la red tiene problemas para no caer en dicho mínimo, mientras que con el método propuesto se reducen las probabilidades de caer en un mínimo local, obteniéndose un mejor aprendizaje de la tarea principal (Figura 3.7(d)).

La Tabla 3.2 presenta la probabilidad de acierto de clasificación para un esquema STL, un esquema con RBF ("Radial Basis Funcion") y un esquema con "hints". Se puede observar la capacidad de reducir la probabilidad de caer en mínimos locales (menor desviación estándar). El peor de los casos corresponde a un esquema STL sin "hints" (Figura 3.7(c)).

Tabla 3.2: Probabilidad de acierto (Media y desviación estándar) para diferentes arquitecturas sobre el conjunto de datos artificial de la Figura 3.7 sobre 30 simulaciones.

Esquema	Probabilidad de acierto ± std	Mejor
STL	0.984 ± 0.005	0.992
RBF	0.989 ± 0.002	0.991
MTL con 1 Hint	0.996 ± 0.001	0.997



(a) (b)

(c) (d)

Figura 3.7: En esta figura se muestra un conjunto de datos creado artificialmente junto con su frontera ideal (a) para mostrar la capacidad del método propuesto para evitar mínimos locales. En (b) se muestra una situación de mínimo local en donde frecuentemente quedan atrapados los pesos de la red STL. En (c) se muestra la frontera de clasificación obtenida mediante Edición de Datos del conjunto inicial. Este nuevo conjunto será utiliado como un "hint" creando un esquema de aprendizaje MTL (d) que mejora el aprendizaje de la tarea principal.

En el esquema MTL, la tarea secundaria ayuda a la tarea principal en su aprendizaje (mayor probabilidad de acierto en media y menor desviación estandar). Esto es porque la tarea secundaria reduce el espacio de soluciones de tal manera que encamina a la tarea principal hacia una solución mejor.

3.4.2. Conjunto de datos de la "UCI MLR"

En esta sección, se analizan los resultados obtenidos sobre problemas reales de diversos ámbitos. Se trata de seis conjuntos de la "UCI Machine Learning Repository" cuyas características se encuentran resumidas en la Tabla 3.3.

Heart Diseases contiene 76 atributos, pero en todos los experimentos publicados se utiliza un subconjunto de 14 de ellos (13 características y el objetivo). El objetivo es determinar la presencia de enfermedad cardiaca en el paciente. Es un valor entero de 0 (no presencia) a 4, aunque se va a tratar el problema como uno de clasificación que distinga entre presencia (valores 1,2,3,4) o ausencia (valor 0) de la enfermedad.

Pima Indian Diabetes representa a pacientes que son mujeres de al menos 21 años con herencia indígena Pima que viven cerca de Phoenix, Arizona. Las pruebas para la diabetes se hicieron según los criterios de la World Health Organization.

Las características de Breast Cancer Wisconsin se han calculado a partir de una imagen digitalizada de una punción de masa mamaria. En ellas se describen las características de los núcleos de las células presentes en la imagen.

Ionosfera es un conjunto de datos de radar, donde los objetivos son electrones libres en la ionosfera. Se catalogan como "bueno" si el radar devuelve indicios de algún tipo de estructura en la ionosfera, y se catalogan como "malos", en otro caso.

Abalon es un conjunto de datos para predecir la edad del árbol Abalón. La edad del Abalón se determina mediante el corte del tronco, contando el número de anillos a través de un microscopio. También se utilizan otras mediciones para predecir la edad (el clima, ubicación, etc.) Existen 10 categorías, por simplicidad se ha tratado el conjunto de datos como un problema de clasificación de 2 categorías (del 1 al 9 para una clase y mayor o igual a 10 para la otra).

Forest Covertype es una base de datos de cuatro áreas del desierto del norte de Colorado, para determinar el tipo de árbol que puede crecer en esas tierras mediante la información proporcionada por la elevación, el tipo de suelo, etc. El conjunto de datos se trata como un problema de clasificación de 2 categorías (distinguir entre la primera clase de las otras).

Tabla 3.3: Resumen de las base de datos utilizadas de la UCI Databases Repository.

Cjto de datos	Clases	Características	Tamaño	%Clase 1	%Clase 2
Heart Diseases	2	13	270	55.6	44.4
P. Indian Diabetes	2	8	786	34.9	65.1
Breast Cancer W.	2	9	683	65.2	34.8
Ionosphera	2	34	351	35.9	64.1
Abalon	2	8	4177	48.5	51.5
CoverType	2	54	581012	30.2	69.8

La Tabla 3.4 muestra la probabilidad de clasificación correcta para diferentes esquemas de aprendizaje. También muestra los porcentajes del número de muestras utilizados en los "hints" para L=3, 5 y 7 respectivamente, con K=10 (si sólo hay 1 "hint", L=7). Se puede observar, como se reduce notablemente el número de épocas de entrenamiento y se mejora la probabilidad de acierto. Especialmente, se reduce la desviación estándar del error, lo que muestra la reducción del número de veces que el entrenamiento se ha quedado estancado en un mínimo local. También se observa que al incrementar el número de tareas

Tabla 3.4: Probabilidad de clasificación para el conjunto de test (media ± desviación estándar y media de las épocas de entrenamiento) para los esquemas STL y "hints" para diferentes conjuntos de datos. La última columna (%Hints) representa el porcentaje de reducción de muestras de los "hints" (L=3,5,7) para K=10.

	STL	Tarea principal + 1 Hint	Tarea principal + 3 Hints
Conjunto	Clasif.±std (Ep.)	Clasif.±std (Ep.)	Clasif.±std (Ep.)(%Hints)
Heart	0.815±0.017 (341)	0.824±0.009 (217)	0.825±0.003 (212)(8,7,5)
Diabetes	0.742±0.010 (405)	0.748±0.004 (520)	0.753±0.005 (138)(9,7,4)
Cancer	0.979±0.003 (520)	0.981±0.002 (125)	0.982±0.002 (110)(9,8,6)
Ionosphere	0.881±0.008 (603)	0.891±0.010 (610)	0.891±0.010 (604)(7,6,5)
Abalon	0.776±0.012 (9928)	0.790±0.011 (3650)	0.790±0.009 (2204)(6,5,3)
CoverType	0.928±0.011 (9687)	0.936±0.009 (7398)	0.939±0.008 (6520)(5,4,2)

secundarias, se mejoran levemente los resultados (en algunos casos).

3.5. Data Editing para MTL

Hasta ahora, se ha utilizado la Edición de Datos para proporcionar "tareas secundarias" en esquemas donde no existían estas tareas o resultaba difícil identificarlas. De esta forma, estas tareas se pueden emplear como "hints" en el esquema de aprendizaje MTL.

Es fácil ver que este mismo esquema puede ser utilizado sobre un esquema multitarea clásico, utilizando la Edición de Datos sobre las tareas secundarias. De esta manera, tenemos tareas secundarias donde hemos eliminado solapamiento y muestras poco significativas. La utilización de la Edición de Datos sobre estas tareas secundarias tiene sentido en problemas complejos donde las tareas secundarias puedan presentar bastante ruido, facilitando así la transferencia de información hacia la tarea principal (a la que también se le puede hacer la Edición de Datos).

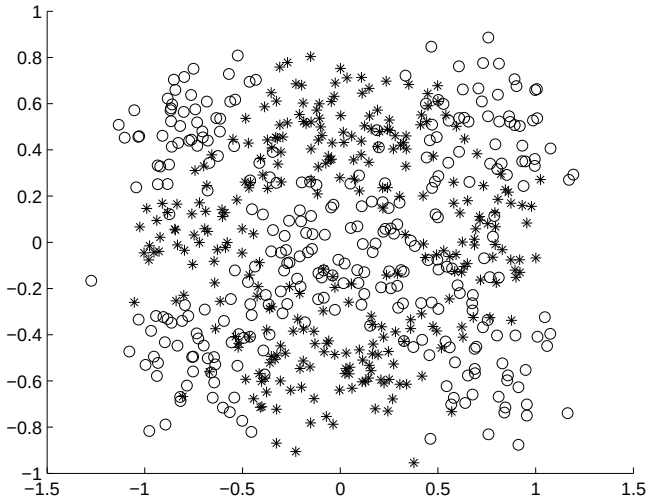
Vamos a ver este enfoque con un problema de dos dimensiones. Este problema se ha generado artificialmente, consta de dos conjuntos diferentes de los cuales uno de ellos se usa como la tarea principal y el otro conjunto como el secundario. La Figura 3.8(a) muestra el conjunto de datos correspondiente a la tarea principal, y en la Figura 3.8(b) los datos de la tarea secundaria. Se puede ver claramente que ambas tareas están relacionadas, y también, la mayor parte de los datos de entrada pertenecen a los dos conjuntos de datos. En cada caso, se tienen 600 muestras para el conjunto de entrenamiento, 600 muestras para el conjunto de validación, y 6000 muestras para test. Las dos clases son equiprobables en todas las series.

La Figura 3.9 muestra los conjuntos de datos resultantes una vez realizada la Edición de Datos tanto para la tarea principal como para la secundaria. De los 600 muestras que componen los conjuntos de entrenamiento (Figura 3.8), el número de muestras se han reducido a 88, donde 45 patrones pertenecen a la tarea principal (Figura 3.9(a)) y 49 patrones a la secundaria (Figura 3.9(b)). Como se puede apreciar, el conjunto editado (Figura 3.9) es un subconjunto del conjunto original (Figura 3.8) compuesto por las muestras cercanas a la frontera de decisión. Esta reducción del conjunto de datos hace que el rendimiento muy rápido.

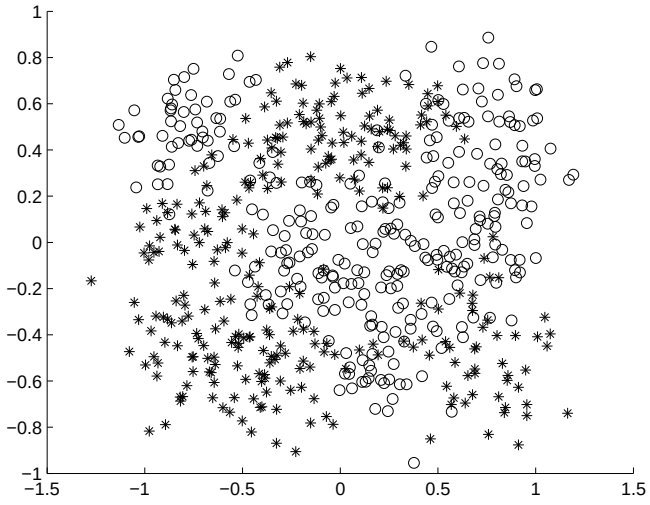
Las redes utilizadas tienen 12 nodos en la capa oculta y se ha seguido la misma dinámica de entrenamiento que en los casos anteriores.

La Figura 3.10 muestra la evolución de la probabilidad de clasificación sobre el conjunto de entrenamiento sobre las diferentes arquitecturas utilizadas. Esta figura muestra que el esquema MTL utilizando la edición de datos mejora el rendimiento del sistema STL.

En la Tabla 3.5, se muestran los resultados obtenidos para las diferentes

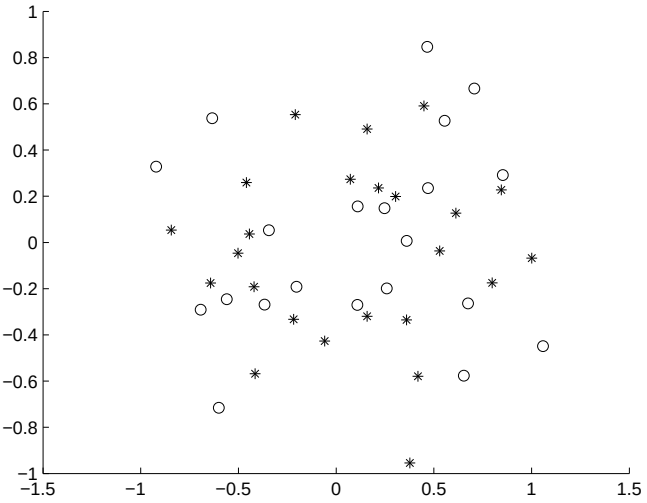


(a) Tarea principal.

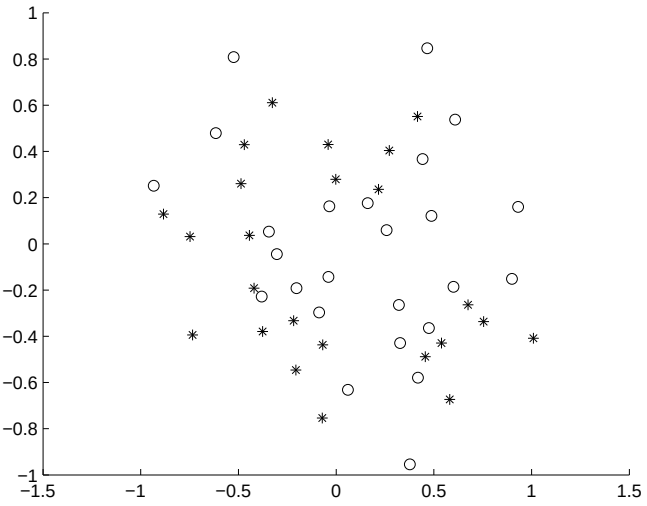


(b) Tarea secundaria relacionada.

Figura 3.8: Conjunto de entrenamiento de un problema de clasificación de dos clases.



(a) Muestras seleccionadas de la tarea principal.



(b) Muestras seleccionadas de la tarea secundaria relacionada.

Figura 3.9: Muestras seleccionadas utilizando la edición de datos para la tarea principal y secundaria.

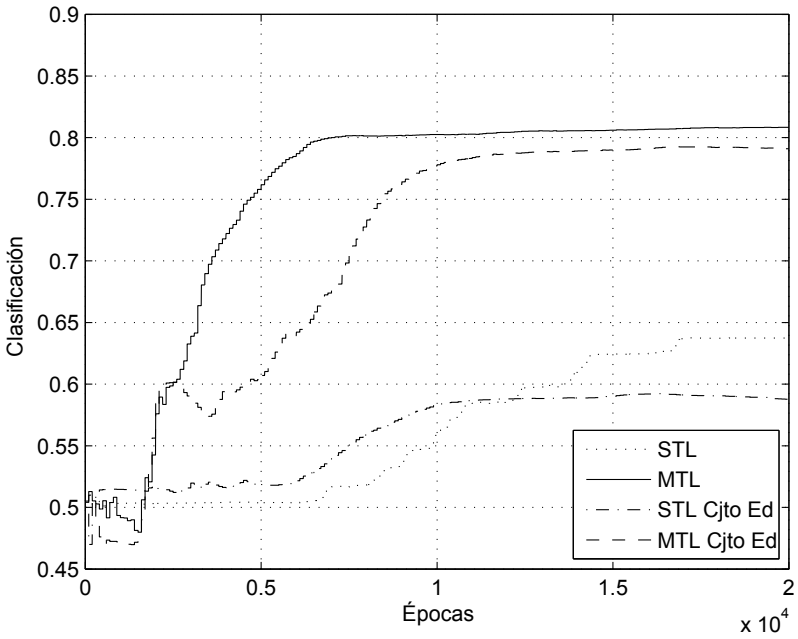


Figura 3.10: Épocas de entrenamiento frente a probabilidad de acierto (medido sobre el conjunto de test) para diferentes esquemas.

Tabla 3.5: Probabilidad de acierto (sobre el conjunto de test) y desviación estandar sobre varios esquemas.

Esquema	% Prob. de acierto	std	Épocas	Mejor
STL	0.637	0.001	19990	0.640
MTL	0.808	0.003	16742	0.814
STL con Cjto Editado	0.588	0.004	6762	0.594
MTL con Cjto Editado	0.761	0.025	7635	0.782

arquitecturas. Se puede observar como el mejor resultado se obtiene mediante la arquitectura MTL, sin embargo, el enfoque MTL con edición de datos es una buena alternativa si no disponemos de tareas secundarias ya que la única alternativa que nos quedaría sería un esquema STL. Al utilizar MTL con edición de datos, se obtienen buenos resultados (mejores que STL), reduciendo las épocas de entrenamiento y la complejidad del algoritmo debido a la gran reducción de muestras.

3.6. Conclusiones

El uso de "hints" con una tarea principal para entrenar una red neuronal artificial, actúa como un sesgo para esa tarea principal, siendo de vital importancia que los "hints" estén relacionados con esa tarea principal para que el sesgo sea positivo. En algunos casos, este tipo de arquitectura utilizando 'hints" no es posible debido a que no se dispone de tareas relacionadas. En esta Tesis, se utiliza un procedimiento de Edición de Datos, para obtener, de forma artificial, tareas relacionadas con la que se quiere aprender. La Edición de Datos es un procedimiento que obtiene un conjunto reducido de muestras críticas a partir de un conjunto original. Para problemas de clasificación, estas muestras seleccionadas, se encuentran en una posición cercana a la frontera de decisión,

por eso son críticas para incorporar un sesgo inductivo positivo en el aprendizaje de una tarea, considerada como tarea principal. En particular, hemos utilizado los algoritmos de selección de muestras {*K*,*L*}-NN y Condensed-NN como algoritmos de edición.

Este método siempre genera tareas relacionadas debido a que son subconjuntos de la principal que aproximan la frontera de decisión según la regla 1-NN. En particular, se ha observado que estas nuevas tareas secundarias hacen que la tarea principal evite mínimos locales, así como que se reduzcan el número de épocas de entrenamiento. Hay que tener en cuenta que las tareas secundarias son conjuntos de complejidad reducida con respecto a la tarea principal, por lo que aprenden su tarea mucho más rápido que ésta. Así, en pocas épocas de entrenamiento ya aportan un sesgo inductivo positivo a la tarea principal.

Hemos utilizado tres problemas de juguete para explicar las ventajas de utilizar la Edición de Datos en un esquema MTL, y seis problemas reales de la "UCI Machine Learning Repository" para, finalmente, testear el método propuesto. Aunque el aprendizaje MTL clásico suele obtener mejores resultados que el método propuesto, hay que destacar que la justificación de su uso es para cuando no es posible utilizar el aprendizaje MTL. En ese caso, sólo queda la posibilidad de utilizar un esquema STL. El método propuesto permite por lo tanto crear tareas relacionadas artificiales y poder utilizar un esquema MTL que mejora los resultados de uno STL. También se ha utilizado la Edición de Datos sobre tareas secundarias de un esquema MTL clásico reduciéndose considerablemente las épocas de entrenamiento.

Capítulo 4

Smaller networks that are able to develop hypotheses for a task are better than larger networks.

(Occams Razor)

Diseño de arquitecturas MLP mediante ELM

4.1. Introducción

En la vida real, cuando aprendemos una tarea, tenemos que procesar una gran cantidad de información proveniente de múltiples variables. Sin embargo, el aprendizaje humano es capaz de discriminar de una manera bastante eficiente lo que es realmente importante para aprender una tarea y descartar la información irrelevante. Del mismo modo, cuando una máquina de aprendizaje está intentando resolver una nueva tarea, necesita seleccionar las características de entradas relevantes o eliminar conexiones innecesarias [7, 61, 40, 91, 12, 31, 75]. Hay métodos de selección de características utilizados antes del entrenamien-

to, por ejemplo, los basados en la teoría de la información [91]. Otros métodos tratan de definir la relevancia de las entradas durante el aprendizaje, por ejemplo, mediante la medición de la sensibilidad de la salida con respecto a las entradas, repitiéndose este proceso hasta que no existan más características de entrada irrelevantes [12]. Este tipo de métodos son interesantes en problemas complejos con muchas entradas y neuronas en la capa oculta, ya que simplifican la arquitectura. Además del esfuerzo para reducir la dimensionalidad de los datos, se necesita un alto coste computacional en la selección de la arquitectura.

Este capítulo presenta un método nuevo para obtener de una forma rápida y eficiente una arquitectura de un MLP ("MultiLayer Perceptron") para resolver problemas de clasificación. El método propuesto utiliza las ventajas que presenta el algoritmo ELM ("Extreme Learning Machine"), y en particular, el OP-ELM ("Optimal Prunned ELM"), obteniendo una solución única para la arquitectura MLP y seleccionando automáticamente las características y conexiones relevantes para cada neurona de la capa oculta, así como el tamaño de la capa oculta. Los resultados obtenidos en la clasificación de diferentes problemas muestran las ventajas del método propuesto. En particular, se obtiene una convergencia más rápida y una reducción del coste computacional por la eliminación de neuronas o conexiones innecesarias; todo esto aportando una tasa de clasificación muy competitiva con otros métodos contrastados.

El resto del capítulo está organizado de la siguiente manera. La siguiente sección está dedicada a explicar el algoritmo ELM. En la Sección 4.3 se explican los métodos de poda para ELM. La Sección 4.4 explica una optimización que tiene el ELM para clasificación que nos va a permitir construir nuestro modelo. La Sección 4.5 muestra el método propuesto, terminando el capítulo con una sección de experimentos y conclusiones.

4.2. Extreme Learning Machine

El ELM ("Extreme Learning Machine") está basado en el concepto de que si los pesos de entrada y los sesgos de la capa oculta de una SLFN ("Single hidden Layer Feedforward neural Network"), son asignados al azar, la SLFN puede ser considerada como un sistema lineal y los pesos de salida se pueden determinar analíticamente mediante la inversa generalizada. Esta idea ya fue estudiada en [79, 57] y en [108], hay un estudio profundo acerca de las diferencias entre el ELM y estos trabajos previos. Sin embargo, Huang demuestra la capacidad que tiene el ELM para ser un aproximador univeral donde los nodos elegidos al azar pueden ser generados de acuerdo con cualquier distribución de probabilidad continua, sin ningún conocimiento previo [48, 54]. Esta asignación aleatoria se puede hacer si las funciones de activación en la capa oculta son infinitamente diferenciables [56]. Así, para N distintas muestras aleatorias $(\mathbf{x}_j, \mathbf{t}_j)$, donde $\mathbf{x}_j = [x_{j1}, x_{j2}, \ldots, x_{jn}]^T \in \mathbb{R}^n$ y $\mathbf{t}_j = [t_{j1}, t_{j2}, \ldots, t_{jm}]^T \in \mathbb{R}^m$, las SLFNs con M neuronas ocultas y función de activación $f(\cdot)$ son matemáticamente resueltas mediante

$$\mathbf{o}_j = \sum_{i=1}^{M} \boldsymbol{\beta}_i f(\mathbf{w}_i \cdot \mathbf{x}_j + b_i), \quad j = 1, \ldots, N; \tag{4.1}$$

donde $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ es el vector de pesos que conecta la i-ésima neurona oculta y las características de entrada; $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ es el vector de pesos que conecta la i-ésima neurona oculta y las unidades de salida, y b_i es el sesgo (bias) de la i-ésima neurona oculta. Además, $\mathbf{w}_i \cdot \mathbf{x}_j$ es el producto escalar de $\mathbf{w}_i$ y $\mathbf{x}_j$. Obsérvese que las unidades de salida de la red son lineales. La función de activación más popular para las unidades de la capa oculta es la sigmoide y, en este caso, la SLFN se conoce como MLP ("Multi-Layer Perceptron"). También se pueden utilizar otras funciones de activación, como

por ejemplo las de base radial [52, 53] y, en este caso, la SLFN se conoce como RBF ("Radial Basis Function").

En general, las SLFNs estándar con M neuronas en la capa oculta y función de activación f, pueden aproximar estas N muestras con error cero. Esto es:

$$\sum_{j=1}^{M} \|\mathbf{o}_j - \mathbf{t}_j\| = 0, \tag{4.2}$$

en otras palabras, debe existir $\boldsymbol{\beta}_i$, $\mathbf{w}_i$ y b_i tal que

$$\sum_{i=1}^{M} \boldsymbol{\beta}_i f(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \, j = 1, \ldots, N. \tag{4.3}$$

Y de forma compacta, se puede escribir:

$$\mathbf{HB} = \mathbf{T} \tag{4.4}$$

donde

$$\begin{aligned} &\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_M, b_1, \ldots, b_M, \mathbf{x}_1, \ldots, \mathbf{x}_N) \\ &\qquad = \begin{bmatrix} f(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \ldots & f(\mathbf{w}_M \cdot \mathbf{x}_1 + b_M) \\ \vdots & \ldots & \vdots \\ f(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \ldots & f(\mathbf{w}_M \cdot \mathbf{x}_N + b_M) \end{bmatrix}_{N \times M} \end{aligned} \tag{4.5}$$

$$\mathbf{B} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_M^T \end{bmatrix}_{M \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{4.6}$$

donde $\mathbf{H}$ es la matriz de salida de la capa oculta del SFLN; la i-ésima columna

de $\mathbf{H}$ es la i-ésima salida de la neurona oculta con respecto a las entradas $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. Huang et al. [56] prueban que si la función de activación f es infinitamente diferenciable, el número requerido de nodos ocultos satisface que $M \leq N$.

Para introducir de una manera formal el algoritmo ELM, Huang et al. prueban en [55, 56] los siguientes teoremas:

Teorema 1. *Dada una SLFN estándar con N nodos ocultos y una función de activación* $f : \mathbb{R} \to \mathbb{R}$ *la cual es infinitamente diferenciable en cualquier intervalo, para N muestras aleatorias distintas* $(\mathbf{x}_j, \mathbf{t}_j)$*, donde* $\mathbf{x}_j \in \mathbb{R}^n$ *y* $\mathbf{t}_j \in \mathbb{R}^m$*, para cualquier* $\mathbf{w}_i$ *y* b_i *elegidos aleatoriamente de cualquier intervalo de* $\mathbb{R}^n$ *y* $\mathbb{R}$*, respectivamente, de acuerdo con cualquier distribución de probabilidad continua, entonces, con probabilidad uno, la matriz de salida de la capa oculta de salida* $\mathbf{H}$ *de la SLFN es invertible y* $\|\mathbf{H}_{N\times M}\mathbf{B}_{M\times m} - \mathbf{T}_{N\times m}\| = 0$.

Teorema 2. *Dado un valor positivo pequeño* $\varepsilon > 0$ *y una función de activación* $f : \mathbb{R} \to \mathbb{R}$ *la cual es infinitamente diferenciable en cualquier intervalo, existe* $M \leq N$ *tal que para N muestras aleatorias distintas* $\mathbf{x}_j \in \mathbb{R}^n$ *y* $\mathbf{t}_j \in \mathbb{R}^m$*, para cualquier* $\mathbf{w}_i$ *y* b_i *elegidos aleatoriamente de cualquier intervalo de* $\mathbb{R}^n$ *y* $\mathbb{R}$*, respectivamente, de acuerdo con cualquier distribución de probabilidad continua, entonces, con probabilidad uno,* $\|\mathbf{H}_{N\times M}\mathbf{B}_{M\times m} - \mathbf{T}_{N\times m}\| < \varepsilon$.

Para un valor fijado de $\mathbf{w}_i$ y b_i, el entrenamiento de una SLFN es equivalente a encontrar una solución de mínimos cuadrados $\hat{\mathbf{B}}$ del sistema lineal $\mathbf{HB} = \mathbf{T}$:

$$\hat{\mathbf{B}} = \arg\min_{\mathbf{B}} \|\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_M, b_1, \ldots, b_M)\mathbf{B} - \mathbf{T}\| \tag{4.7}$$

Si el número de neuronas ocultas es igual al número N de muestras de entrenamiento ($M = N$), $\mathbf{H}$ es cuadrada e invertible, cuando $\mathbf{w}_i$ y b_i son elegidos

aleatoriamente y, por lo tanto, una SLFN puede aproximar las muestras de entrenamiento con error cero. Sin embargo, en la mayoría de los casos, el tamaño de la capa oculta es mucho menor que el número de vectores de entrenamiento ($M \ll N$) y, entonces, $\mathbf{H}$ no es una matriz cuadrada y no pueden existir $\boldsymbol{\beta}_i$ tal que $\mathbf{HB} = \mathbf{T}$.

Siguiendo el trabajo [56], y como es bien sabido, el Teorema 3 establece:

Teorema 3. *Sea una matriz* $\mathbf{G}$ *tal que* $\mathbf{Gy}$ *es una solución con menor norma de mínimos cuadrados de un sistema lineal* $\mathbf{Ax} = \mathbf{y}$. *Entonces es necesario y suficiente que* $\mathbf{G} = \mathbf{A}^{\dagger}$, *la matriz inversa generalizada de Moore-Penrose de* $\mathbf{A}$.

De esta forma la solución con menor norma de mínimos cuadrados del anterior sistema lineal es

$$\hat{\mathbf{B}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{4.8}$$

donde $\mathbf{H}^{\dagger}$ es la matriz inversa generalizada de Moore-Penrose $\mathbf{H}$, [90, 83].

Huang et al. [47] demuestran que para una SLFN, a lo sumo con N nodos ocultos y cualquier función de activación no lineal, se pueden aprender N distintas muestras.

Así, a modo de resumen, se presenta a continuación el algoritmo ELM para las SLFNs.

Como ya se mencionó, $f(\cdot)$, además de poder ser función sigmoidal o de base radial, puede ser sinusoidal, exponencial, etc. Por ejemplo, la salida de una SLFN con nodos RBFs está representada por

$$\mathbf{o}_j = \sum_{i=1}^{M} \boldsymbol{\beta}_i f\left(\frac{\|\mathbf{x}_j - \mathbf{w}_i\|}{b_i}\right), \quad j = 1, \ldots, N \tag{4.9}$$

donde $f(\cdot)$ es usualmente un núcleo (kernel) gaussiano, y $\mathbf{w}_i$ y b_i son, respectivamente, el centro y la anchura i-ésima del núcleo de nodo oculto. En el caso de

Algoritmo 1 Extreme Learning Machine (ELM).

Dado un conjunto de entrenamiento $\mathcal{D} = \left\{(\mathbf{x}_j, \mathbf{t}_j) | \mathbf{x}_j \in \mathbb{R}^n, \mathbf{t}_j \in \mathbb{R}^m, j = 1, \ldots, N\right\}$, una función de activación f y un número de neuronas en la capa oculta M,

1: Asignar aleatoriamente los pesos de la capa de entrada $\mathbf{w}_i$ y sesgos b_i, $i = 1, \ldots, M$.
2: Calcular la matriz de la capa de salida oculta $\mathbf{H}$ usando (4.5).
3: Calcular la matriz de pesos de la capa de salida: $\mathbf{B} = \mathbf{H}^{\dagger}\mathbf{T}$, donde $\mathbf{B}$ y $\mathbf{T}$ están definidos en (4.6) y siendo $\mathbf{H}^{\dagger}$ la matriz inversa generalizada de Moore-Penrose de $\mathbf{H}$.

las RBFs, parecido al estándar MLP, el algoritmo ELM asigna aleatoriamente los centros y anchos de los núcleos y, entonces, los pesos de la capa de salida son determinados analíticamente a través de la inversa generalizada de la matriz de salida de la capa oculta [52, 53].

Es importante mencionar que hay varios métodos para calcular la inversa generalizada de Moore-Penrose. Algunos de estos métodos están basados en proyecciones ortogonales, métodos iterativos, descomposición de valor singular, etc. [77].

4.3. Métodos de poda para las ELM

Para el diseño de la arquitectura, en esta Tesis se utiliza un enfoque basado en la poda [84]. Inicialmente se crea una SLFN totalmente conectada cuyo tamaño es mayor de lo necesario y se reduce el tamaño mediante la eliminación de nodos no relevantes. Así, en un diseño de poda [93], se entrena una SLFN sobredimensionada y a continuación, los nodos o pesos menos útiles se eliminan de acuerdo a una medida predefinida.

Hay muchos algoritmos para realizar poda en redes neuronales [93, 81, 84].

Por ejemplo, algunos de estos métodos eliminan individualmente un peso cada vez, lo cual puede resultar sub-óptimo. Otros métodos eliminan uno o varios pesos en un sólo paso y, por lo tanto se reducen significativamente el tamaño de la red reduciendo la complejidad. En particular, en esta Tesis se utiliza una poda bastante eficiente realizada sobre los nodos ocultos de una SLFN entrenada con el algoritmo ELM. Como ya hemos mencionado, en el algoritmo ELM estándar, los pesos de la capa de salida (**B**) se calculan teniendo en cuenta los pesos de entrada prefijados aleatoriamente. Una vez calculados, la información de sus valores se emplea para descubrir nodos ocultos poco relevantes para la tarea a resolver [55, 56].

Con el fin de mejorar el esquema ELM, se han propuesto esquemas de podas automáticas [89, 66, 69, 64, 67, 68]. Por lo general, estos métodos se aplican sobre redes ELM en un gran número de nodos de la capa oculta cuyos pesos se inicializan aleatoriamente. Posteriormente, se estudia la relevancia de dichos nodos para proceder a la poda. Se calcula la relevancia de cada nodo de la capa oculta en relación a la salida deseada ($\left\{\mathbf{t}_j\right\}_{j=1}^{N}$). De acuerdo a esa medida de relevancia, se podan o se mantienen los nodos ocultos. Así, los nodos de la capa oculta con baja relevancia se eliminan para obtener una red más compacta que aumenta la capacidad de generalización. El concepto de relevancia se define mediante una medida estadística que es diferente en cada algoritmo de poda [89, 66, 64, 67]. Principalmente hay propuestos dos enfoques distintos: P-ELM (Poda ELM "Pruned-ELM") [89] y OP-ELM (Poda Óptima ELM "Optimally Pruned-ELM") [66, 68]. A continuación, se analizan estos dos métodos.

4.3.1. Poda ELM: P-ELM

Este método de poda P-ELM se ha propuesto para resolver tareas de clasificación mediante ELM [89]. En particular, el P-ELM calcula la relevancia estadística de los nodos ocultos utilizando dos criterios: el Chi-cuadrado (χ^2) [59] y las medidas de Ganancia de Información (GI), [26]. Antes de utilizar estos métodos estadísticos, y dado que **T** es una tarea de decisión (es decir, datos discretos), las respuestas de los nodos ocultos se discretizan empleando el método de discretización de Fayyad [34]. Esto se hace para poder comparar estas salidas con **T**.

El criterio χ^2 mide las diferencias entre las distribuciones de las variables categóricas [59], mientras que el criterio GI mide la reducción de la incertidumbre sobre el valor del objetivo **T** dada la respuesta discretizada de cada nodo oculto [26]. Se va a utilizar r_i para denotar la relevancia obtenida para la i-ésima neurona oculta determinada por χ^2 o GI. En base a r_i, los nodos ocultos se ordenan en orden descendente. Nótese que cuanto mayor sea el valor de r_i, mayor relevancia del nodo oculto. Una vez que r_i se ha calculado, se define un parámetro umbral γ. La elección de γ define la arquitectura final y la capacidad de generalización de la red P-ELM. El valor óptimo de γ se determina mediante el criterio de información de Akaike, AIC ("Akaike Information Criterion") [3] sobre un conjunto de validación elaborado a partir de las muestras de entrenamiento [89]. A continuación, se presenta un esquema del algoritmo P-ELM.

Como inconvenientes, el valor umbral elegido es bastante sensible a la correcta selección del subconjunto de validación. Esto aumenta el coste computacional debido al hecho de que hay que realizar varias repeticiones. También se debe señalar que el subconjunto de validación hace que tengamos menos

Algoritmo 2 Poda ELM (P-ELM).

Dado un conjunto de entrenamiento $\mathcal{D} = \left\{(\mathbf{x}_j, \mathbf{t}_j)|\mathbf{x}_j \in \mathbb{R}^n, \mathbf{t}_j \in \mathbb{R}^m, j = 1, \ldots, N\right\}$, una función de activación predefinida f (sigmoide o gaussiana), una capa oculta más grande de lo necesario, M, y un conjunto de q umbrales $\{\gamma_s\}_{s=1}^{q}$,

1: Dividir $\mathcal{D}$ en dos subconjuntos no solapados para entrenamiento (N_T muestras) y validación (N_V muestras), con $N = N_T + N_V$
2: Asignar aleatoriamente los pesos de la capa de entrada $\{\mathbf{w}_i, b_i\}_{i=1}^{M}$.
3: Calcular la matriz de los pesos de la capa oculta $\mathbf{H}$ usando el subconjunto de aprendizaje.
4: Calcular la relevancia estadística r_i para las neuronas ocultas mediante χ^2 or GI.
5: Ordenar las neuronas ocultas en orden descendiente de acuerdo a r_i.
6: **for** $s = 1$ to q **do**
7: Encontrar el subconjunto de relevancia de neuronas ocultas, con cardinalidad S_s, que satisfaga $r_k < \gamma_s$, $k = 1, 2, \ldots, S_s$
8: Obtener el error de clasificación en el conjunto de validación: E_V.
9: Computar AIC: $AIC\,(s) = 2N_V \log\left(\frac{E_V^2}{N_V}\right) + S_s$
10: **end for**
11: Seleccionar la red con M^* nodos acorde al $min(AIC)$
12: Calcular la matriz de la capa oculta $\mathbf{H}^*$ usando $\mathcal{D}$
13: Calcular la matriz de pesos de salida: $\mathbf{B}^* = (\mathbf{H}^*)^{\dagger}\mathbf{T}$

muestras disponibles durante el proceso de aprendizaje, que pueden ser un inconveniente en algunos problemas definidos por conjuntos de datos escasos. Además, como se ha dicho anteriormente, P-ELM sólo puede aplicarse en problemas de clasificación.

4.3.2. Poda Óptima ELM: OP-ELM

A diferencia del método anterior de poda, la OP-ELM ("Optimally Pruned Extreme Learning Machine") [66, 68] puede ser aplicada tanto a la clasificación como a tareas de regresión. Además, en lugar de utilizar un subconjunto de

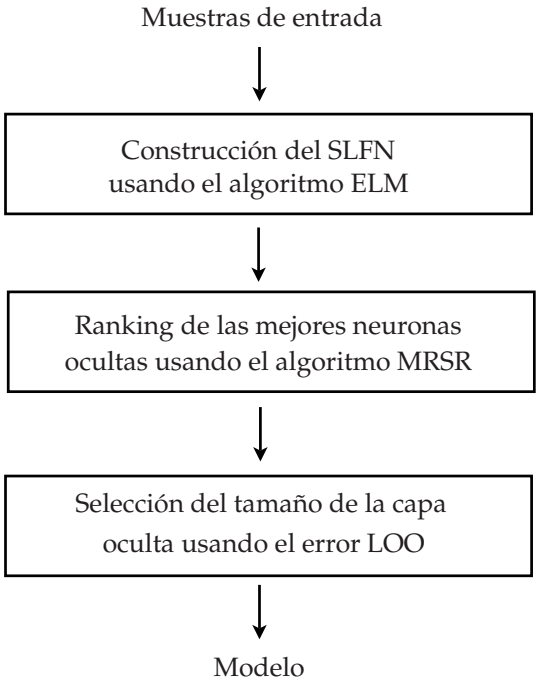


Figura 4.1: Etapas del algoritmo OP-ELM.

validación, este procedimiento utiliza el método LOO ("Leave-One-Out") como criterio para la selección de las unidades ocultas. Por otra parte, el OP-ELM utiliza el algoritmo MRSR ("MultiResponse Sparse Regression") para la medición de la relevancia de las unidades ocultas [100]. La principal ventaja del MRSR es que realiza una valoración de las neuronas en la capa oculta. Como se muestra en [68], el OP-ELM ofrece un modelo extremadamente rápido y logra en la mayoría de los casos el mismo nivel de precisión que otros métodos de aprendizaje bien conocidos, como por ejemplo, las SVM ("Support Vector Machine"), GP ("Gaussian Processes") o MLP. A continuación, se detallan las tres etapas principales del método OP-ELM que, a modo de resumen, se muestran en la Figura 4.1 .

Fase 1. Inicialización aleatoria

Este primer paso se realiza utilizando el algoritmo ELM estándar para un número de neuronas M. Desde el punto de vista práctico, se aconseja que M esté por encima del número de características de entrada: M >> n. Aunque la metodología ELM utiliza un solo tipo de función de activación o núcleo (por ejemplo, funciones sigmoidales), en OP-ELM se utilizan tres tipos de funciones (gaussiana, sigmoide y lineal) que pueden usarse en combinación para una mejor robustez y generalidad. Así, mientras que las funciones sigmoidales, los pesos son elegidos aleatoriamente según una distribución uniforme en un intervalo de los datos de entrada previamente normalizados (media cero y varianza uno), para los núcleos gaussianos, sus centros y anchuras son elegidos al azar entre los percentiles 20 y 80 de la distribución de las distancias de los datos de entrada [68].

Fase 2. Ranking de unidades ocultas: Algoritmo MRSR

Como segunda etapa, el algoritmo MRSR se utiliza para hacer una clasificación de las neuronas ocultas en función de su relevancia [100]. MRSR es una extensión del conocido algoritmo LARS ("Least Angle Regression") y, por lo tanto, es una técnica mediante ranking [33].

Los fundamentos del algoritmo MRSR en un modelo ELM son los siguientes, para la matriz de salida $\mathbf{H}_{N\times M}$ de los nodos ocultos, cada columna $\mathbf{h}_i$ (vector de las N salidas que produce la neurona i-ésima para las N entradas) se añade una a una al modelo en sucesivos pasos del algoritmo. De esta manera, para el paso k ($k = 1, \ldots, M$), el modelo se define como

$$\tilde{\mathbf{O}}^k_{N\times m} = \mathbf{H}_{N\times M}\mathbf{B}^k_{M\times m} \tag{4.10}$$

donde $\tilde{\mathbf{O}}^k$ y $\mathbf{B}^k$ son, respectivamente, la matriz de salida del modelo y la matriz de pesos de salida para el paso k-ésimo. $\mathbf{B}^k$ tiene k filas distintas de cero en el paso k-ésimo del algoritmo MRSR. Con cada nuevo paso y nueva fila distinta de cero en $\mathbf{B}^k$, se introduce la nueva columna ($\mathbf{h}_k$) al modelo. En [100], Smila et al. definen un criterio que mide la suma de las correlaciones entre los residuos del modelo y el regresor considerado (k-ésimo nodo oculto del vector de salida) a través de todas las variables objetivo. MRSR selecciona regresores en una forma gradual en cada iteración. Los detalles sobre la definición de una correlación acumulativa se pueden encontrar en el artículo original sobre el MRSR [100]. Es importante tener en cuenta que el MRSR obtiene una clasificación exacta para problemas separables linealmente. Puesto que la salida de un esquema ELM es lineal con respecto a las unidades ocultas inicializadas de forma aleatoria, la clasificación obtenida de las neuronas por MRSR es exacta [66, 69].

Fase 3. Selección de las unidades ocultas

Una vez que las neuronas estén ordenadas y por consiguiente $\mathbf{H}$, se seleccionan las mejores unidades ocultas para el modelo ELM (M^* nodos ocultos). Para ello, como se hace en el P-ELM, se divide al azar el conjunto de datos en aprendizaje y validación, por cada una de esas divisiones, el modelo se ajusta a los datos de aprendizaje y la precisión se evalúa a través de los datos de validación [13]. En este caso, los resultados varían si el análisis se repite con otro conjunto diferente generado de forma aleatoria. Una solución mejor es utilizar LOO ("Leave-One-Out"): utiliza una única muestra del conjunto de aprendizaje original como validación. Esto se repite de tal manera que cada observación en el conjunto de muestras se usa una vez como dato de validación, y por lo tanto, todas las muestras se utilizan para el aprendizaje y la validación. El principal

inconveniente con el método LOO es que puede ser extremadamente costoso computacionalmente cuando el conjunto de datos tiene un gran número de muestras [13].

Sin embargo, el error LOO puede calcularse para modelos lineales mediante el uso del estadístico PRESS (PREdiction Sum of Squares) [8]. Considerando que $\hat{\mathbf{B}}_{(j)}$ es la solución mediante el error por mínimos cuadrados cuando la j-ésima muestra (j-ésima fila de $\mathbf{H}$) es omitida, es decir,

$$\hat{\mathbf{B}}_{(j)} = \mathbf{H}_{(j)}^{\dagger}\mathbf{T}_{(j)}, \tag{4.11}$$

donde $\mathbf{H}_{(j)}$ y $\mathbf{T}_{(j)}$ son, respectivamente, $\mathbf{H}$ y $\mathbf{T}$ sin su j-ésima fila. Entonces, el error LOO es dado por

$$\epsilon_{PRESS} = \frac{1}{N}\sum_{j=1}^{N} \|\mathbf{h}_j\hat{\mathbf{B}}_{(j)} - \mathbf{T}_{(j)}\|^2 \tag{4.12}$$

donde $\mathbf{h}_j$ es la j-ésima fila de $\mathbf{H}$. Para seleccionar un modelo usando el criterio del error mediante LOO, se computa el ϵ_{PRESS} para cada modelo y se elige el modelo con un mínimo ϵ_{PRESS}. Usando directamente la fórmula anterior, la computación de ϵ_{PRESS} sería extremadamente ineficaz y computacionalmente prohibitivo ya que el modelo se ha calculado tantas veces como el número de muestras existente. Sin embargo, hay una fórmula bien conocida para realizar el estadístico PRESS de forma no iterativa y precisa:

$$\epsilon_{PRESS} = \frac{1}{N}\|\Gamma\left(\mathbf{I} - \mathbf{P}\right)^{-1}\left(\mathbf{P} - \mathbf{I}\right)\|_{\mathcal{F}}^2, \tag{4.13}$$

con $\mathbf{P} = \mathbf{H}\mathbf{H}^{\dagger}$, $\mathbf{I}$ es la matriz identidad y $\Gamma(\mathbf{A})$ mantiene sólo las entradas diagonales de la matriz cuadrada $\mathbf{A}$. Siendo $\|\mathbf{A}\|_{\mathcal{F}}$ la norma de Frobenius de la

matriz **A**. Una demostración de la fórmula (4.13) se da en [73].

Para redes OP-ELM, el estadístico PRESS se computa de forma iterativa añadiendo un nodo oculto al modelo. El modelo con M^* neuronas ocultas obtiene el valor más bajo del estadístico PRESS, considerándose este modelo como óptimo, es decir,

$$\epsilon_{PRESS}^{M^*} < \epsilon_{PRESS}^{k}, \quad \forall k \in (1, 2, \ldots, M). \tag{4.14}$$

Notese que este procedimiento es bastante mejor que el P-ELM, el cual requiere seleccionar un umbral. De acuerdo a [66, 69, 67, 68], la etapa de selección de los nodos tiene dos efectos importantes: se consigue una convergencia más rápida y los modelos ELM son más pequeños ($M^* < M$). A continuación, en esta sección se incluye un resumen del algoritmo OP-ELM.

Algoritmo 3 Poda Óptima ELM (OP-ELM).

Dado un conjunto de entrenamiento $\mathcal{D} = \left\{(\mathbf{x}_j, \mathbf{t}_j) | \mathbf{x}_j \in \mathbb{R}^n, \mathbf{t}_j \in \mathbb{R}^m, j = 1, \ldots, N\right\}$, varias funciones de activación (sigmoides, gaussianas y lineales), y un gran número de neuronas M,

1: Asignar al azar los pesos de entrada $\{\mathbf{w}_i, b_i\}_{i=1}^{M}$.
2: Calcular la matriz de la capa de salida de los nodos ocultos $\mathbf{H}$ con $\mathbf{X}$ y los pesos de la capa de entrada.
3: Clasificar las salidas ocultas usando el algoritmo MRSR, es decir, clasificar $\mathbf{H}$, donde el conjunto $\mathbf{H}^0$ representa una matriz vacía.
4: **for** $k = 1$ to N **do**
5: Añadir el nodo k-ésimo al modelo $\rightarrow \mathbf{H}^k = \left[\mathbf{H}^{k-1}, \mathbf{h}_k\right]$, siendo $\mathbf{h}_k$ la k-ésima columna de $\mathbf{H}$.
6: Calcular el error mediante LOO (ϵ_{PRESS}^{k}) usando (4.13) con $\mathbf{H}^k$.
7: **end for**
8: Seleccionar el tamaño de la red según $M^* = argmin_k(\epsilon_{PRESS}^{k})$.
9: Calcular la matriz de los pesos de la capa de salida: $\mathbf{B}^* = (\mathbf{H}^*)^{\dagger}\mathbf{T}$

Con el fin de evitar la inversión de la matriz y los costosos productos matriciales (4.13), hay trabajos que utilizan diferentes criterios de selección de

variables , tales como el Criterio de Información de Hannan-Quinn [67] o el Test de Delta [64]. Estos trabajos proporcionan procedimientos más fáciles para la selección de modelos que el método original de LOO.

4.4. Optimización para clasificación

Un resultado importante del trabajo de Huang tiene que ver con los valores de los sesgos [51] y que deriva de trabajos previos [50]. Se puede demostrar que los sesgos b_j no son necesarios en la optimización del ELM para clasificación ya que el hiperplano de separación del espacio del ELM pasa a través del origen. Esta manera de trabajar será utilizada en el método que aquí se presenta.

A diferencia de los trabajos anteriores sobre ELM para SVM [62, 36], los cuales se basan en aplicar kernels ELM en SVM, Huang [51] estudia la optimización del ELM para clasificación demostrando que el ELM puede ser linealmente extendido a SVMs (pero con menos limitaciones de optimización) en lugar de sólo reemplazar kernels SVM con kernels ELM y por lo tanto el aprendizaje y la implementación de las SVMs se puede hacer mucho más simple y eficiente, ya que de acuerdo a las teorías del ELM [55, 56, 50, 48, 49] todas las muestras de entrenamiento son linealmente separables por un hiperplano que pasa a través del origen con probabilidad uno en el espacio de características del ELM, por lo que ELM para la clasificación tiende a lograr una mejor generalización que las SVMs.

4.5. Diseño automático de MLP's

Uno de los objetivo de esta Tesis es obtener automáticamente un diseño único para la arquitectura de una red para aprendizaje multitarea mediante el uso del algoritmo OP-ELM visto anteriormente. El paso previo para obtener esto es encontrar una arquitectura para un MLP. Cabe señalar que el método propuesto no utiliza el OP-ELM para clasificar, sino como una herramienta que nos permite obtener la nueva arquitectura deseada para un MLP. En particular, el método propuesto elimina entradas irrelevantes, pesos y nodos ocultos innecesarios en una arquitectura MLP. Este método será referenciado como ASELM ("Architecture Selection using Extreme Learning Machine") [16]. El diseño automático de la red se consigue mediante la introducción de dos cambios principales en el diseño de la arquitectura de red inicial obtenida por OP-ELM. En primer lugar, los pesos de la capa de entrada no se inicializan al azar: son los valores binarios generados por todas las combinaciones posibles del número de características de entrada. Cada neurona oculta está conectada a una de estas posibles combinaciones. Por lo tanto, el segundo cambio es que el valor inicial para M se fija en $2^n - 1$, que corresponde al número de combinaciones posibles de las características de entrada (2^n) con la eliminación del caso en el que todos los pesos son 0. Por ejemplo, si el problema viene definido por 4 características de entrada, la arquitectura inicial estará compuesta por 15 neuronas ocultas con 4 conexiones de entrada, siendo sus correspondientes pesos entradas $\mathbf{w}_1 = [0, 0, 0, 1]$, $\mathbf{w}_2 = [0, 0, 1, 0]$, $\mathbf{w}_3 = [0, 0, 1, 1]$, ..., $\mathbf{w}_{15} = [1, 1, 1, 1]$. Se ha de tener en cuenta que el caso $[0, 0, 0, 0]$ no se considera y el bias se considera cero de acuerdo al trabajo de Huang comentado previamente en la sección 4.4.

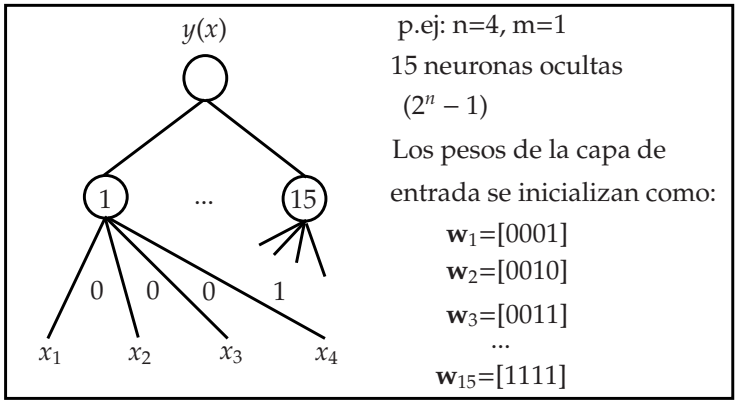
Una vez que se define la arquitectura MLP inicial, se entrena con el algoritmo OP-ELM. A través del uso combinado de LARS y validación cruzada mediante

LOO, se descartan de manera óptima las neuronas ocultas cuya combinación de variables de entrada no es relevante para la tarea objetivo. Vale la pena mencionar que de forma automática se obtiene una arquitectura simplificada donde sólo las características de entrada y las conexiones de la capa oculta relevantes se conservan, eliminándose por tanto las conexiones y variables innecesarias. El uso de OP-ELM en el método propuesto permite el diseño de la arquitectura MLP de una manera rápida y eficiente en comparación con otros métodos tradicionales que tienen un coste computacional excesivo, ya que requieren procedimientos de prueba/error para la selección de características, métodos secuenciales de añadir/podar neuronas ocultas y métodos iterativos de validación cruzada, etc. Además, el método ASELM crea un esquema que permite ver las dependencias entre las características de entrada en la arquitectura del MLP.

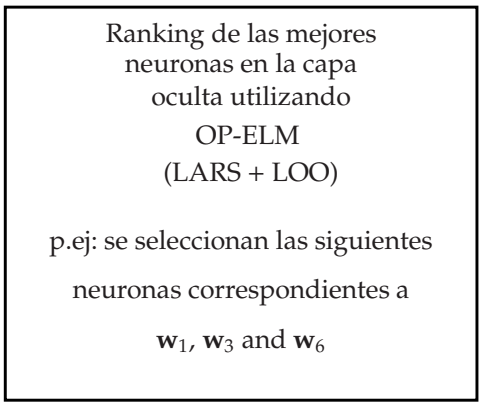
Después de que finalice el proceso de diseño del MLP, la red se entrenará mediante BP ("Back-Propagation") basado en el gradiente donde los pesos de las conexiones que se consideran activas se inicializan aleatoriamente. La Figura 4.2 muestra un diagrama de flujo del proceso descrito anteriormente. A modo de resumen, se presenta a continuación el algoritmo ASELM. En la siguiente sección, el problema del "Logic Domain" se utilizará para explicar en profundidad el método propuesto.

4.6. Experimentos. Diseño de arquitecturas MLP

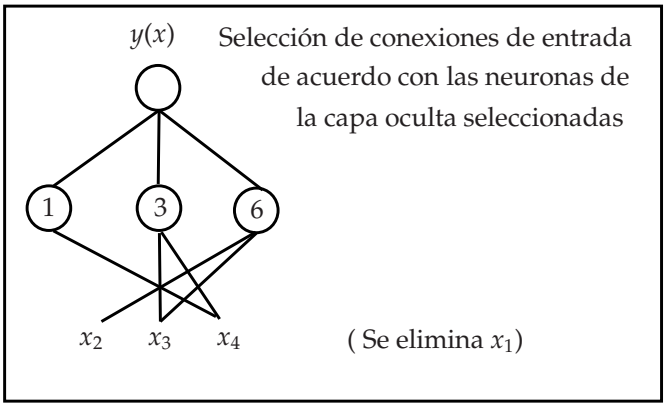
Con el fin de mostrar la bondad del método ASELM para el diseño adecuado de un MLP, se han comparado los resultados de clasificación sobre el conjunto de test en cinco esquemas MLP. En primer lugar, el MLP obtenido con ASELM



(a)



(b)



(c)

Figura 4.2: Ejemplo de los tres pasos del método ASELM para resolver un problema de clasificación de 4 dimensiones. a) Inicialización de los pesos de la red; b) Entrenamiento con el algoritmo OP-ELM; y c) Arquitectura final.

Algoritmo 4 Selección de la arquitectura mediante ELM (ASELM)

Dado un conjunto de entrenamiento $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \ldots, N\}$, una función de activación f, un número de neuronas ocultas $M = 2^n - 1$, donde n es el número de características de entradas, el procedimiento es el siguiente:

1: Los pesos de la capa de entrada se inicializan con valores binarios teniendo en cuenta todas las combinaciones posibles de las entradas. Se descarta el caso en el que todos los pesos son cero.
2: Entrenar la red MLP mediante el algoritmo OP-ELM.
3: Para el esquema obtenido del MLP, seleccionar aquellos pesos de la capa de entradas cuyo valor sea 1 y su correspondiente neurona oculta haya sido seleccionada por el OP-ELM.
4: Inicializar aleatoriamente los pesos del MLP cuya arquitectura queda determinada por las neuronas ocultas y las conexiones de entradas seleccionadas en el paso anterior.
5: Entrenar el esquema MLP mediante el algoritmo BP (u otro algoritmo de la literatura).

se ha entrenado con el algoritmo BP; en segundo lugar, se entrena un MLP con el algoritmo BP seleccionando la dimensión de su capa oculta mediante el procedimiento de validación cruzada, CV ("Cross Validation"); después, se analizan los resultados de un MLP mediante dos métodos de poda clásicos, como son el OBS ("Optimal Brain Surgeon") [42, 110], y OBD ("Optimal Brain Damage") [27]. La poda del MLP mediante lo métodos OBS y OBD serán referenciados en esta Tesis mediante MLP-OBS y MLP-OBD, respectivamente; y, por último, se entrena un MLP mediante el método OP-ELM.

Hay que tener en cuenta que una vez que el MLP totalmente interconectado ha sido finalmente diseñado de acuerdo al procedimiento de CV, los métodos MLP-OBS y MLP-OBD se utilizan para podar las conexiones que no sean relevantes para el entrenamiento del MLP. Las simulaciones del MLP-OBS[1],

[1] http://eu.wiley.com/WileyCDA/Section/id-105036.html

MLP-OBD[2] y OP-ELM[3] han sido realizadas utilizando las "Toolboxes" de Matlab desarrolladas en [101],[76] y [66].

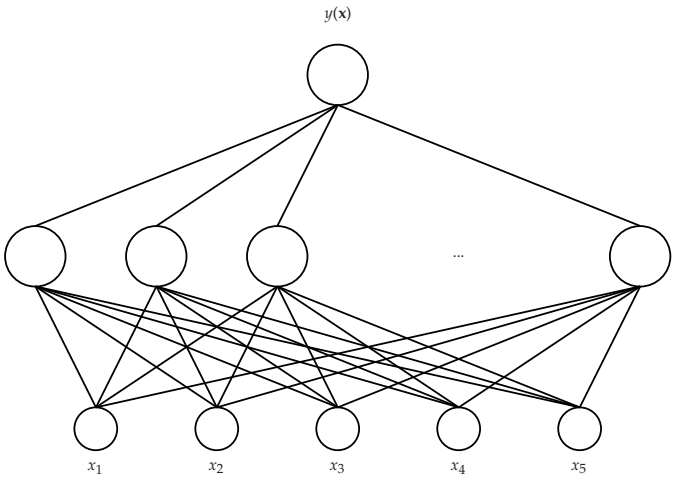
Como función de coste se ha utilizado el MSE ("Mean Squared Error"), y un 10-fold CV con 30 repeticiones (inicializaciones de los pesos) con el fin de obtener la probabilidad de clasificación para las cinco redes. Se descartan aquellas que han obtenido una mala convergencia, por ejemplo, debido a inicializaciones inapropiadas, mediante el uso de la distribución t-student con un intervalo de confianza del 95 %. Todas las simulaciones se han llevado a cabo en el mismo equipo (Intel Xeon a 2,93 GHz, 8 GB de RAM).

Un total de catorce conjuntos diferentes se han seleccionado con el criterio de tener una gran variabilidad, esto es, diferentes dimensiones en la entrada y salida, y diferente número de patrones. La Tabla 4.1 muestra un resumen de las principales características de estos conjuntos, los cuales están disponibles en el repositorio de la UCI MLR [5], excepto el problema del "Logic Domain" [94] que fue presentado en la Sección 2.6.

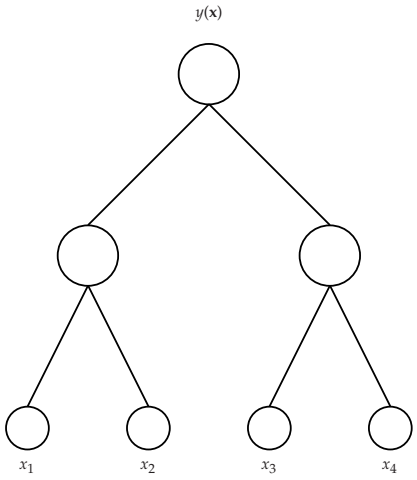
Una versión reducida del problema del "Logic Domain" se va a utilizar con el fin de explicar en más detalle como el ASELM crea la arquitectura para el MLP. Sólo utilizaremos la primera tarea con cinco características de entrada. Esta tarea presenta la combinación lógica de cuatro características de entradas binarias: $t = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$. Además, hay otra característica de entrada, x_5, que no es relevante para aprender la función lógica, pero se utiliza para demostrar cómo el método descarta características de entradas irrelevantes.

El OP-ELM se inicializa con 31 neuronas (es decir, $M = 31 = 2^5 - 1$) con valores para los vectores de la capa oculta desde [0 0 0 0 1] hasta [1 1 1 1 1]. Después de entrenar la red inicial con el método OP-ELM, éste selecciona sólo dos neuronas

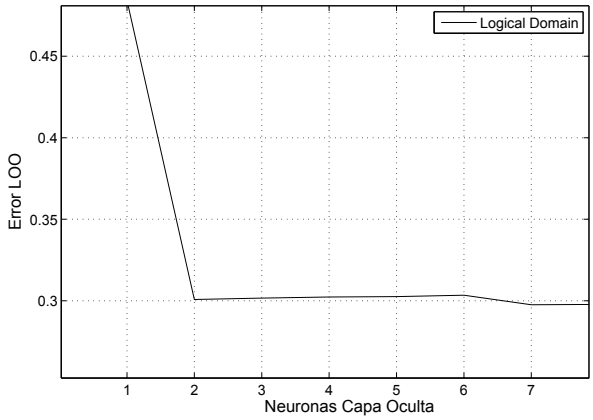
[2]http://www.iau.dtu.dk/research/control/nnsysid.html
[3]http://research.ics.aalto.fi/eiml/software/OPELM.zip



(a)



(b)



(c)

Figura 4.3: Solución del método ASELM del problema "Logic Domain". a) Selección de la arquitectura; b) Selección de las neuronas de la capa oculta mediante el error LOO obtenido por validación cruzada del método OP-ELM. Se puede observar como el error se satura después de añadir la segunda neurona oculta.

Tabla 4.1: Características de entrada y salida, número de muestras de entrenamiento y test de los diferentes conjuntos de datos.

Conjunto	Entradas	Salidas	Entrenamiento	Test
Logic	5	1	20	12
Iris	4	3	150	50
Pima	7	1	300	232
CoverType	10	4	2265	1132
Abalon	8	3	3133	1044
Cancer	9	1	480	202
Monk 1	6	1	288	144
Monk 2	6	1	288	144
Monk 3	6	1	288	144
Breast Tissue	10	4	70	36
Balance Scale Weight	4	3	416	209
Mammographic Mass	5	1	550	280
MAGIC Gamma Telescope	10	1	12680	6339
Wine (Red) recognition	11	1	1066	533

en la capa oculta con pesos de entrada $[1,1,0,0,0]$ y $[0,0,1,1,0]$. La primera aprende la función $(x_1 \vee x_2)$ y la otra $(x_3 \vee x_4)$. También, se puede observar que la característica irrelevante x_5 no ha sido seleccionada por el método (ver Figura 4.3). Esta nueva arquitectura proporciona un esquema MLP más compacto donde se han eliminado las conexiones innecesarias y características de entrada irrelevantes. Como se muestra en la simulación que presenta la Figura 4.4, esta nueva arquitectura necesita, para obtener la convergencia, un número mucho menor de épocas que el MLP diseñado mediante la validación cruzada.

La Figura 4.5(a) representa la arquitectura obtenida por ASELM para el problema "Iris Data", problema de tres clases. Se puede observar en esta nueva arquitectura que la característica de entrada x_1 ha sido eliminada. La Figura 4.5(b) representa el error de LOO mediante validación cruzada para el método OP-ELM para las tres clases con respecto al número de neuronas ocultas. Sólo dos neuronas son necesarias para clasificar la primera clase (Setosa) y cinco

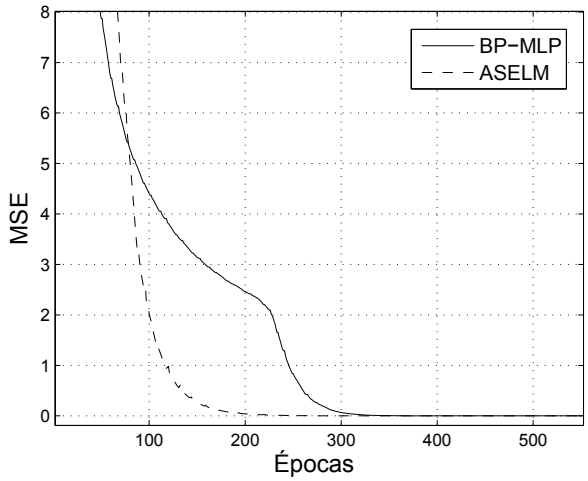
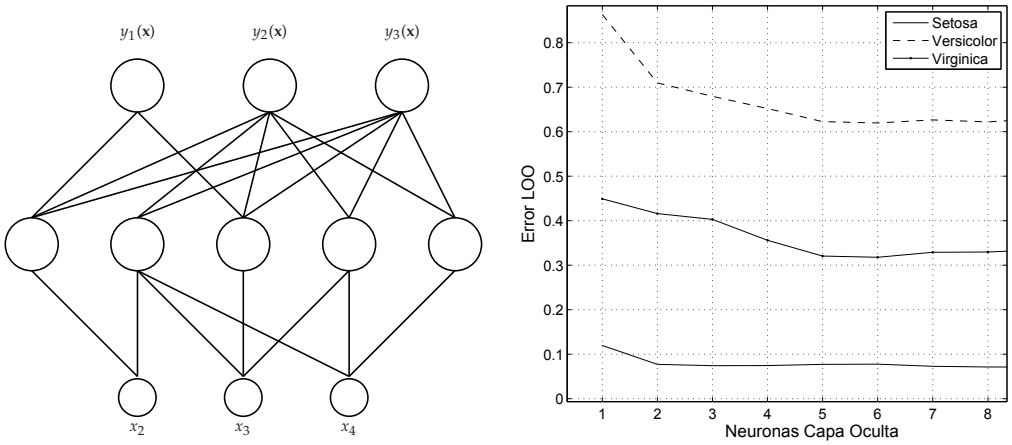


Figura 4.4: Reducción de épocas de entrenamiento del esquema ASELM frente al esquema estándar BP-MLP para el conjunto de datos "Logic Domain".



(a) (b)

Figura 4.5: (a) Arquitectura obtenida para el conjunto de datos "Iris Data" donde las tres salidas son respectivamente Setosa, Versicolor y Virginica. En (b) se puede observar el número de neuronas ocultas seleccionadas para clasificar cada clase mediante el error LOO obtenido por validación cruzada del método OP-ELM.

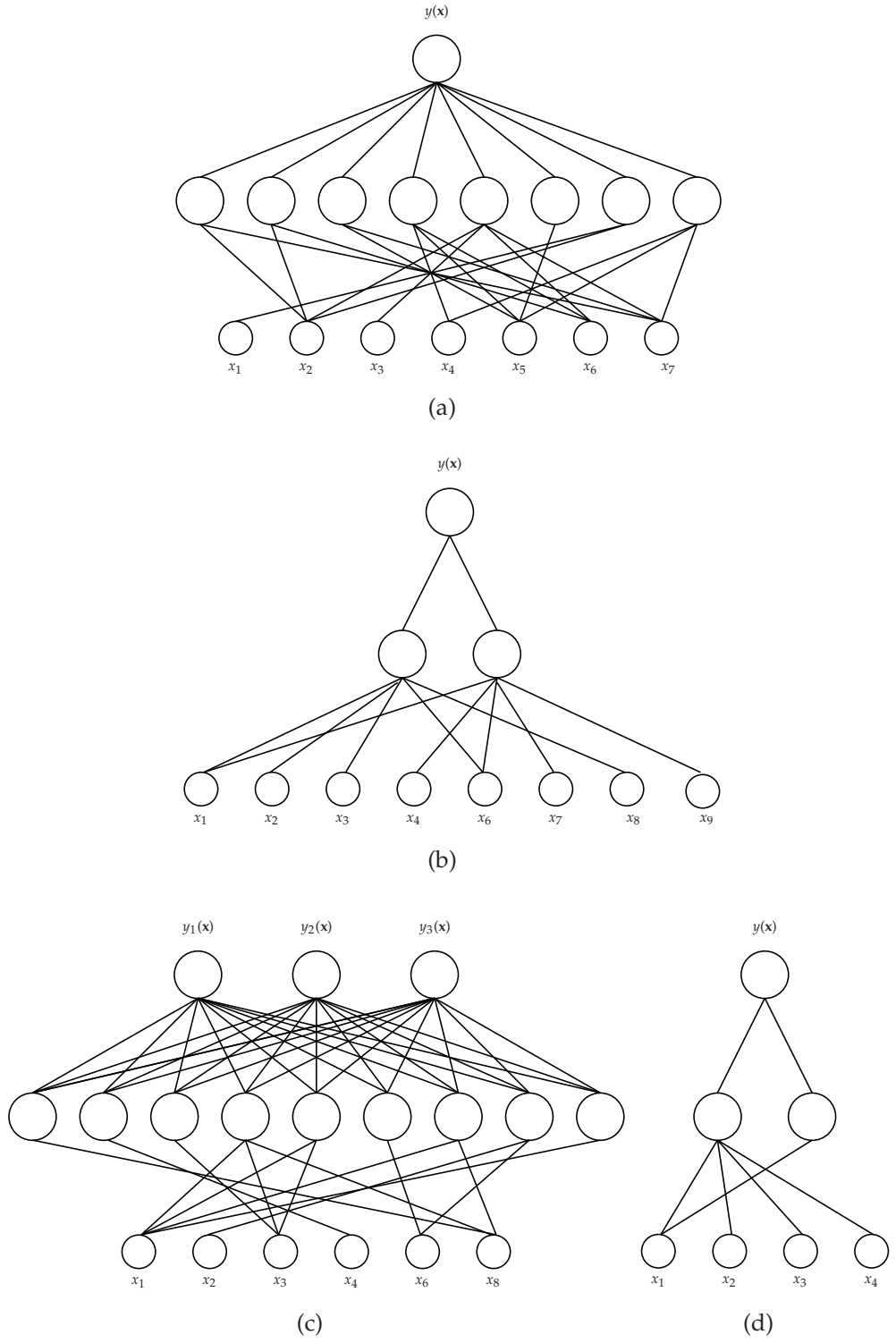


Figura 4.6: Arquitectura obtenida mediante el método ASELM para los problemas Pima Indian Diabetes, Breast Cancer Wisconsin, Abalon y Mammographic Mass, respectivamente.

Tabla 4.2: Test de clasificación "TC" (media ± desv. estándar) para diferentes métodos y conjuntos de datos, "NN" representa el número de neuronas en la capa oculta, "NC" es el número de conexiones que tiene la arquitectura, "TE" es el tiempo de ejecución en segundos (proceso completo: selección de la arquitectura y entrenamiento), y "CD" es la característica (o características) de entrada que el método ASELM ha descartado.

Data set	Method	CA(mean ± std)	HN	NC	RT	RF
Logic	*ASELM*	1.000±0.000	2	4	13	x_5
	MLP – BP	0.941±0.059	7	35	143	
	MLP – OBS	0.854±0.064	7	32±1	143+14	
	MLP – OBD	0.911±0.072	7	32±2	143+8	
	OP – ELM	0.838±0.045	6±1	30±5	<1	
Iris	*ASELM*	0.986±0.001	5	8	144	x_1
	MLP – BP	0.982±0.006	6	24	658	
	MLP – OBS	0.967±0.021	6	10±5	658+30	
	MLP – OBD	0.979±0.011	6	16±4	658+20	
	OP – ELM	0.947±0.026	8±2	32±8	3	
Pima	*ASELM*	0.791±0.003	8	24	43	none
	MLP – BP	0.791±0.006	21	147	14879	
	MLP – OBS	0.775±0.007	21	137±5	14879+337	
	MLP – OBD	0.764±0.015	21	128±8	14879+496	
	OP – ELM	0.768±0.015	7±2	49±14	4	
CoverType	*ASELM*	0.924±0.004	12	42	5044	x_5
	MLP – BP	0.926±0.003	15	150	25532	
	MLP – OBS	0.832±0.008	15	138±3	25532+4707	
	MLP – OBD	0.867±0.004	15	112±5	25532+9216	
	OP – ELM	0.871±0.006	45±2	450±20	204	
Abalon	*ASELM*	0.762±0.010	9	14	4248	x_5 and x_7
	MLP – BP	0.775±0.027	19	152	75592	
	MLP – OBS	0.791±0.024	19	140±7	75592+5974	
	MLP – OBD	0.775±0.015	19	137±6	75592+9587	
	OP – ELM	0.760±0.017	33±3	264±24	270	
Cancer	*ASELM*	0.966±0.009	2	10	128	x_5
	MLP – BP	0.973±0.010	10	90	2080	
	MLP – OBS	0.975±0.015	10	63±8	2080+795	
	MLP – OBD	0.952±0.016	10	61±14	2080+238	
	OP – ELM	0.955±0.023	9±3	90±30	6	
Monk 1	*ASELM*	0.998±0.005	8	21	899	none
	MLP – BP	0.986±0.017	8	48	42465	
	MLP – OBS	0.993±0.022	8	40±3	42465+75	
	MLP – OBD	0.984±0.028	8	16±5	42465+18	
	OP – ELM	0.705±0.035	22±5	132±30	7	
Monk 2	*ASELM*	0.754±0.032	21	60	1020	none
	MLP – BP	0.820±0.043	26	156	49794	
	MLP – OBS	0.689±0.032	26	126±6	49794+929	
	MLP – OBD	0.816±0.045	26	146±6	49794+711	
	OP – ELM	0.682±0.038	19±7	114±42	7	
Monk 3	*ASELM*	0.970±0.008	6	13	477	x_1
	MLP – BP	0.990±0.005	5	30	42730	
	MLP – OBS	0.835±0.006	5	22±3	42730+57	
	MLP – OBD	0.945±0.046	5	10±3	42730+12	
	OP – ELM	0.834±0.043	25±6	150±36	7	
Balance	*ASELM*	0.933±0.007	10	20	529	none
	MLP – BP	0.917±0.003	14	56	27528	
	MLP – OBS	0.901±0.007	14	45±5	27528+168	
	MLP – OBD	0.898±0.018	14	38±8	27528+478	
	OP – ELM	0.909±0.010	33±7	132±28	11	
Breast T.	*ASELM*	0.918±0.018	27	86	82	none
	MLP – BP	0.919±0.021	19	190	10032	
	MLP – OBS	0.928±0.028	19	129±12	10032+328	
	MLP – OBD	0.904±0.031	19	135±6	10032+720	
	OP – ELM	0.920±0.024	24±12	216±108	3	
Mammogr.	*ASELM*	0.811±0.010	2	5	281	x_5
	MLP – BP	0.782±0.052	9	45	9612	
	MLP – OBS	0.799±0.018	9	35±6	9612+167	
	MLP – OBD	0.786±0.012	9	28±9	9612+84	
	OP – ELM	0.808±0.011	24±8	120±40	14	
MAGIC	*ASELM*	0.858±0.011	44	310	1820	none
	MLP – BP	0.855±0.014	9	90	193018	
	MLP – OBS	0.795±0.018	9	48±8	193018+10231	
	MLP – OBD	0.861±0.004	9	57±9	193018+9261	
	OP – ELM	0.815±0.006	50±6	500±60	659	
Wine	*ASELM*	0.744±0.007	32	127	8112	none
	MLP – BP	0.738±0.004	21	231	74061	
	MLP – OBS	0.719±0.015	21	199±8	74061+4468	
	MLP – OBD	0.717±0.010	21	226±4	74061+1636	
	OP – ELM	0.722±0.011	34±7	374±77	68	

para las otras dos, debido a que se satura el error LOO. La Figura 4.6 muestra otros ejemplos de obtención de la arquitectura mediante el método ASELM.

Por otro lado, la Tabla 4.2 muestra la probabilidad de clasificación para los catorce conjunto de datos usando los cinco esquemas descritos: ASELM, MLP-BP, MLP-OBS, MLP-OBD y OP-ELM. La columna NC representa el número de conexiones entre la entrada y la capa oculta. Esto nos permite comprobar como ASELM, MLP-OBS y MLP-OBD reducen el número de conexiones relevantes frente a un esquema totalmente conectado (MLP-BP y OP-ELM). La columna RT representa el número de segundos necesarios para el diseño y entrenamiento de la red ("RunTime"). Por lo tanto, RT incluye el total de 10-fold mediante validación cruzada (CV) con 30 simulaciones para cada "fold" y para cada arquitectura del MLP. Se ha de tener en cuenta que para el caso de los métodos MLP-OBS y MLP-OBD, el valor de RT viene dado por el valor de RT para obtener el MLP-BP más el tiempo correspondiente de la poda. En el caso del MLP-BP, este proceso (10-fold CV con 30 simulaciones) se repite con diferentes números de neuronas en la capa oculta para obtener la arquitectura más eficiente. Como se puede observar, ASELM proporciona una tasa de clasificación bastante similar a la de los métodos MLP-BP, MLP-OBS y MLP-OBD, pero con una reducción de tiempo bastante significativa y con una arquitectura más reducida. Además, se mejora claramente los resultados dados por el procedimiento OP-ELM tanto en clasificación como en tamaño de la red. Es también un hecho relevante que ASELM proporciona una solución única, realizando una selección de características, una selección de conexiones y una poda de los nodos ocultos. Se crea así una arquitectura simplificada de la red neuronal que converge de forma más rápida y eficiente.

4.7. Conclusiones

Para el diseño de la arquitectura de un MLP, hay implícito un coste computacional debido a procedimientos de prueba y error, y validación cruzada. Además, pueden existir características de entrada irrelevantes o ruidosas que dificultan seriamente el aprendizaje. Para ellos, algunos métodos, como por ejemplo el OBD y OBS, se han desarrollado para la eliminación de entradas, conexiones y nodos ocultos innecesarios. Aun así, para una arquitectura completamente conectada esto implica tiempo de entrenamiento adicional y además, no lo hace asegurando, en términos de las capacidades de generalización, una mejora del MLP completamente interconectado.

En esta Tesis, se presenta un nuevo algoritmo basado en el algoritmo OP-ELM para construir automáticamente una arquitectura simplificada de un MLP. Este nuevo método de selección de la arquitectura (denominado ASELM) descarta características irrelevantes de entrada, y también elimina conexiones y neuronas de la capa oculta que son innecesarias para el aprendizaje. En particular, el método propuesto utiliza el algoritmo OP-ELM, basado en LARS para la clasificación de neuronas ocultas y la fórmula de Allen del error LOO con validación cruzada para la poda de neuronas inútiles. ASELM proporciona automáticamente una solución única para el MLP con una buena capacidad de generalización y sin la intervención del usuario: no hay ningún parámetro a configurar.

En la sección de experimentos, se ha empleado un problema de juguete para explicar en más detalle el método propuesto y trece problemas reales del repositorio “UCI ML” se han utilizado para testear el método propuesto y compararlo con cuatro métodos conocidos para el diseño de redes MLP: MLP-BP, MLP-OBS, MLP-OBD y OP-ELM. En general, la arquitectura simpli-

ficada obtenida por ASELM para estos conjuntos de datos han mejorado el rendimiento con respecto a arquitecturas MLP totalmente interconectadas y arquitecturas MLP podadas mediante OBS y OBD en términos de tiempo de cálculo y capacidad de generalización.

Capítulo 5

The measures of relatedness are unable to eliminate all negative inductive bias caused by unrelated tasks.

(Daniel L. Silver)

Diseño de arquitectura MTL mediante ELM

5.1. Introducción

En el capítulo anterior, se propone una arquitectura para un MLP (denominada ASELM), donde se han eliminado características de entrada y conexiones irrelevantes. Aprovechando las ventajas de esta arquitectura, en este capítulo se muestra cómo podemos utilizarla en un enfoque multitarea, MTL ("Multi-Task Learning").

Como ya se ha visto anteriormente, en un esquema MTL, una de las tareas es considerada como tarea principal mientras que las otras se consideran tareas secundarias. Al aprender estas mediante un MLP, hay una zona común y una

zona específica de cada tarea. En esta Tesis empezamos trabajando sobre un método de regularización para la obtención de una arquitectura para MTL, y comprobamos que es posible obtenerla. En esta nueva arquitectura se eliminan las tareas secundarias que no ayudan al aprendizaje de la tarea principal, así como las características de entrada y neuronas de la capa oculta irrelevantes. Sin embargo, esta arquitectura no es única ya que es dependiente de la inicialización de los pesos. Además, estos métodos utilizan un parámetro de regularización que debe de ser ajustado con lo que se incrementa notablemente el coste computacional. Esto nos llevó a la utilización de ASELM para crear una arquitectura para MTL única y sin ningún tipo de intervención por parte del usuario. Este capítulo explica estos estudios utilizando regularización para obtener una arquitectura para MTL, y continua con la utilización de ASELM para la obtención de una arquitectura MTL única. Finalmente, se incluye una sección de experimentos donde se analiza esta metodología en detalle y otra sección de conclusiones.

5.2. Diseño mediante regularización

Los métodos de regularización se propusieron inicialmente con el objetivo de aumentar la capacidad de generalización de un red neuronal [25, 15, 14]. Uno de los métodos de regularización más conocido es "Weight Decay", que se utiliza para podar pesos irrelevantes, lo que en principio permite eliminar entradas innecesarias. Se basa en introducir un término aditivo (β) en la función de error original (E_0)

$$E = E_0 + \frac{1}{2}\mu\beta, \tag{5.1}$$

donde μ es el parámetro que mide la importancia del término β. Existen varias alternativas para la elección de β, como por ejemplo:

$$\beta = \sum_{ij} w_{ij}^2, \tag{5.2}$$

donde w_{ij} son los pesos del nodo j al nodo i. El efecto de introducir ese término es evitar que la magnitud de los pesos crezca indefinidamente, situación que indica un sobreajuste de los datos. En este sentido, los pesos de la red muy pequeños no son útiles, tomarán un valor próximo a cero, y pueden ser eliminados. Esta elección de β, es la más simple, y hace que decaigan tanto los pesos grandes como los pequeños. Otra elección de β, sugerida en [109], más eficiente es:

$$\beta = \sum_{ij} (w_{ij}^2/w_0^2)/(1 + w_{ij}^2/w_0^2), \tag{5.3}$$

donde w_0 es un parámetro libre que hay que elegir. Dependiendo de su valor, la red puede preferir varios pesos pequeños frente a uno grande o al revés. Haciendo w_0 igual a 1, obtenemos:

$$\beta = \sum_{ij} w_{ij}^2/(1 + w_{ij}^2), \tag{5.4}$$

que presenta la ventaja de disminuir los pesos con valores pequeños más rápidamente que los grandes. Este es el término β que utilizaremos, ya que precisamente buscamos resaltar las conexiones con valores altos.

"Weight Decay" presenta el problema de que al algoritmo BP ("Back-Propagation"), le cuesta más trabajo encontrar una buena solución porque el término de regularización introduce mínimos locales. También, la determi-

nación del valor del parámetro μ es una dificultad añadida. Sin embargo, como lo que se pretende es simplemente detectar zonas comunes de aprendizaje dentro del MLP con respecto a la tarea principal, no es un valor que se necesite ajustar con precisión.

Para implementar el "Weight Decay", se utilizará la siguiente función de coste para el algoritmo BP:

$$E = \frac{1}{2}\sum_i (f_i - y_i)^2 + \frac{1}{2}\mu \sum_{ij} w_{ij}^2/(1 + w_{ij}^2), \tag{5.5}$$

el primer término es la función original de error para el BP estándar, donde f es la salida deseada e y es la salida obtenida. El segundo término es el definido por la ecuación 5.4. Utilizando la función sigmoide, la actualización de los pesos w_{ij} quedaría como

$$\Delta w_{ij} = -\eta(\delta E/\delta w_{ij}), \tag{5.6}$$

por lo que

$$\Delta W_{ij} = \eta X_j (f_i - y_i) y_i (1 - y_i) - \mu \sum_{ij} w_{ij}^2/(1 + w_{ij}^2)^2, \tag{5.7}$$

donde X_{ij} es la salida del nodo j. La segunda expresión es el término de regularización. Esta ecuación 5.7 actualiza los pesos entre la capa oculta y la capa de salida. La actualización de los pesos entre las entradas y la capa oculta es similar, ya que el término de regularización no cambia y sólo se modificaría el primer término según el algoritmo BP.

Se utilizará este método para encontrar conexiones comunes entre neuronas en un MLP con varias tareas a aprender. Es importante destacar, que no se trata de una poda de pesos sobre un MLP, sino que partiendo de una estructura que aprende la tarea principal, se van añadiendo conexiones y neuronas que

tienen entradas comunes con tareas secundarias. Esta nueva arquitectura, evita la influencia negativa que proporcionan las tareas no relacionadas, ya que las elimina, dejando la parte de información que consideramos relevante de una tarea, descartando la que no lo es.

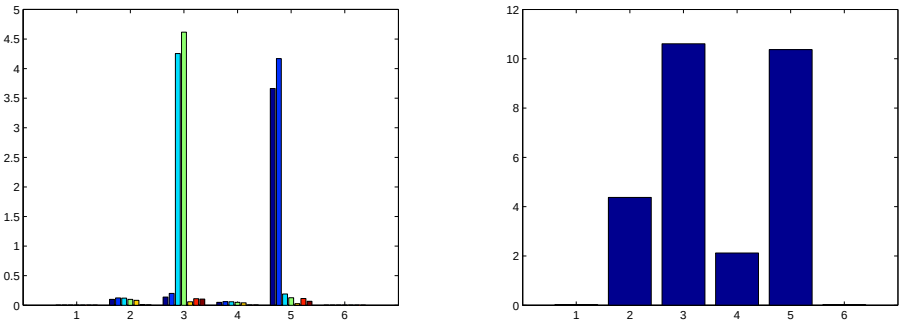
Para explicar el procedimiento en detalle, volveremos al ejemplo del "Logic Domain" presentado en el Capítulo 2 (Sección 2.6). En este caso, se utilizarán las cuatro primeras funciones ya que son suficientes para mostrar la bondad del método (extenderlo a más funciones es obvio). Cada una de las cuatro tareas es una combinación lógica de 4 características de entradas binarias. Como recordatorio, la Tabla 5.1 muestra la expresión lógica de cada una de las cuatro tareas. La primera tarea se considerará como tarea principal (T_P), mientras que el resto se considerarán como secundarias.

Tabla 5.1: Descripción de las tareas del "Logic Domain". Cada una de las tareas es una combinación lógica de cuatro características de entrada.

Tarea	Expresión Lógica
T_P	$(x_1 > 0.5 \vee x_2 > 0.5) \wedge (x_3 > 0.5 \vee x_4 > 0.5)$
T_{Sec_1}	$(x_2 > 0.5 \vee x_3 > 0.5) \wedge (x_4 > 0.5 \vee x_5 > 0.5)$
T_{Sec_2}	$(x_3 > 0.5 \vee x_4 > 0.5) \wedge (x_5 > 0.5 \vee x_6 > 0.5)$
T_{Sec_3}	$(x_4 > 0.5 \vee x_5 > 0.5) \wedge (x_6 > 0.5 \vee x_7 > 0.5)$

En un estudio previo, Silver y Mercer comprueban que no todas las tareas secundarias ayudan al aprendizaje de la tarea principal, [98]. Por ejemplo, se puede ver en la Tabla 5.1, como la tarea principal y la primera tarea secundaria comparten tres entradas (x_2, x_3 y x_4), sin embargo, esta tarea secundaria no ayuda a la tarea principal con un sesgo inductivo positivo porque implementan funciones diferentes. Se ve claramente que la segunda tarea secundaria es la que más va a ayudar a la principal, ya que hay una parte del aprendizaje de

ambas tareas que es común para las dos, concretamente (x_3 >0.5 $\vee$ x_4 >0.5). La utilización de un esquema MTL clásico con todas tareas secundarias no va a mejorar mucho el aprendizaje de la principal ya que esta tarea recibe tanto un sesgo inductivo positivo como negativo, lo que en definitiva, perjudica su aprendizaje.



(a) (b)

Figura 5.1: Valor de los pesos utilizando "Weight Decay" para la primera función (T_P) del problema "Logic Domain". En (a) se muestran los pesos que conectan las características de entrada con la capa oculta, mientras que (b) muestra los que conectan la capa oculta con la de salida.

Para entender cómo se crea la nueva arquitectura, la Figura 5.1 muestra el valor absoluto de los pesos tras el aprendizaje del MLP utilizando "Weight Decay" con una capa oculta de seis neuronas. En (a) se muestran los valores de estos pesos entre las características de entrada y la capa oculta, mientras que en (b) son los valores de los pesos entre la capa oculta y la de salida. Si nos fijamos en la Figura 5.1(b), se observa claramente como de las seis neuronas de la capa oculta, sólo dos de ellas tienen valores altos, y si vemos las características de entrada de esas neuronas en la Figura 5.1(a), se ve claramente como en la tercera neurona intervienen las características de entrada 3 y 4, mientras que en la quinta intervienen la 1 y 2. Esto es porque están aprendiendo las

subtareas ($x_3 > 0.5 \vee x_4 > 0.5$) y ($x_1 > 0.5 \vee x_2 > 0.5$) respectivamente. Finalmente, la nueva arquitectura para la tarea principal se muestra en la Figura 5.2(a). Análogamente, se obtienen resultados similares para cada una de las otras tareas (5.2(b-d)).

Finalmente, podemos observar como hay una neurona oculta que comparte entrada entre la tarea principal (Figura 5.2(a)) y la tarea secundaria representada por la T_{Sec_2} (Figura 5.2(c)). Esta estructura común, formada por la neurona que aprende la función ($x_3 > 0.5 \vee x_4 > 0.5$) será el nexo común que una la tarea principal con la secundaria. En este proceso, la primera y segunda tarea secundaria no se seleccionarían ya que no tienen nada en común con la tarea principal (Figura 5.2(b) y 5.2(c)).

El esquema obtenido para la arquitectura MTL se puede ver representado en la Figura 5.3.

Ahora, el aprendizaje de la tarea principal se ve influenciado por la parte común que tiene con la segunda tarea secundaria, que aportará un sesgo inductivo positivo a su aprendizaje. De la misma forma, se han eliminado los posibles sesgos negativos, proporcionados por la primera y tercera tareas secundarias. Esto proporciona un esquema MTL más robusto donde se eliminan las tareas no relacionadas y se dejan las conexiones relevantes de las tareas relacionadas con la principal.

Este método, aunque proporciona una solución eficiente para un esquema MTL con pocas dimensiones y fácil de resolver, presenta el inconveniente de que en problemas más complejos, como por ejemplo alto dimensionado, proporciona una arquitectura distinta por la inicialización aleatoria de los pesos de entrada, con lo que dificulta el encontrar neuronas con estructuras comunes. Esto hace que aunque proporcione una arquitectura para MTL, hay que uti-

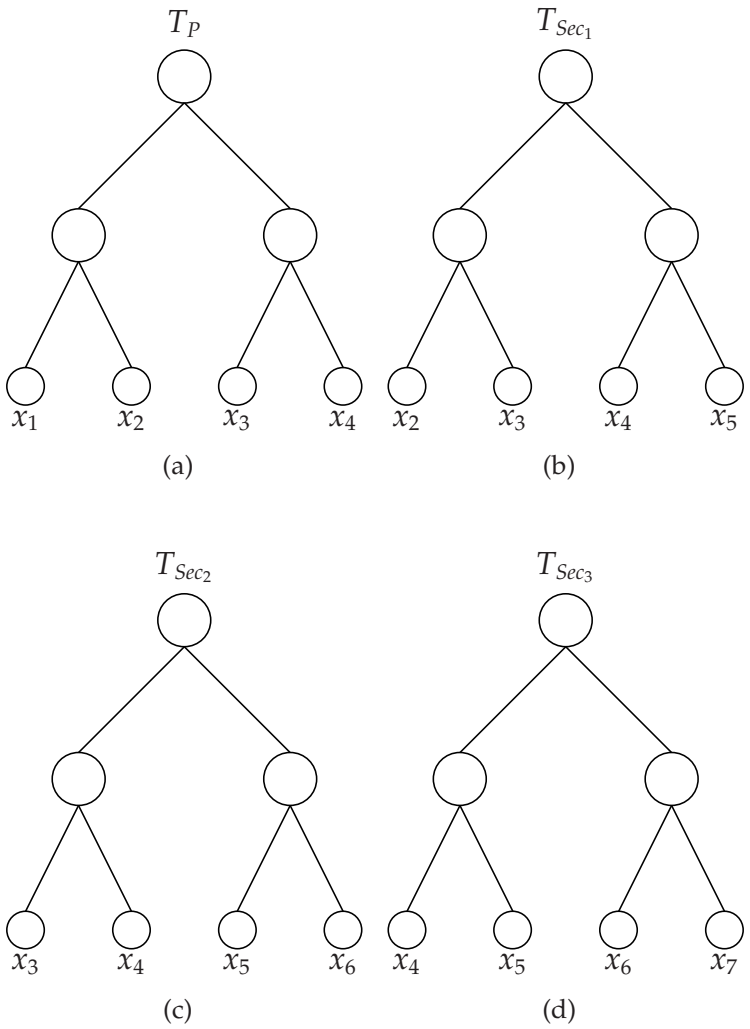


Figura 5.2: Arquitectura obtenida mediante el método ASELM para las cuatro tareas (T_P, T_{Sec_1}, T_{Sec_2} y T_{Sec_3}) del conjunto de datos del "Logic Domain" entrenadas por separado.

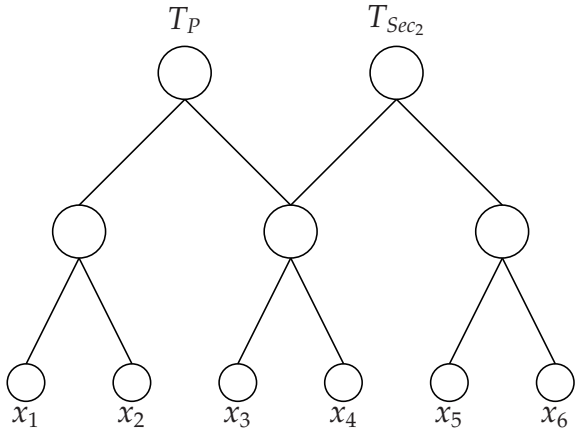


Figura 5.3: Esquema final MTL propuesto para el "Logic Domain" utilizando "Weight Decay". La tarea principal se conecta a la segunda tarea secundaria mediante una neurona que aprende la parte común de ambas.

lizar métodos de validación cruzada tal y como se hace en un MLP clásico para obtener la arquitectura apropiada.

Por todo esto, y con la finalidad de proporcionar un método eficiente y de solución única para el diseño de arquitecturas MTL, se presenta un método basado en el método ASELM.

5.3. Arquitectura MTL basada en ASELM

En el capítulo anterior se utilizó el método ASELM para encontrar una arquitectura que facilitara el aprendizaje de una tarea, eliminando aquellas neuronas o conexiones que no intervienen o dificultan el aprendizaje. El ASELM favorece el sesgo inductivo positivo al eliminar aquellos elementos que dificultan el aprendizaje de la red neuronal. En esta sección, se propone el uso de la metodología ASELM para diseñar una arquitectura MTL. Como se explica en detalle más adelante, será necesario introducir un par de modificaciones al método original para adaptarlo al aprendizaje MTL.

Hay que destacar que se podría utilizar directamente la arquitectura que nos proporciona ASELM en un contexto multitarea, es decir, como el MTL clásico (totalmente interconectado) pero con la ventaja de haber realizado una poda de conexiones, neuronas y selección de características irrelevantes. Evidentemente, esta arquitectura favorecerá la transferencia de sesgo inductivo positivo. Sin embargo, esta arquitectura sólo elimina características de entrada, poda neuronas y conexiones que van desde las entradas a la capa oculta. Para que el método propuesto proporcione una arquitectura más eficiente, debería también realizar una selección de tareas secundarias relevantes, así como eliminar las conexiones que van desde la capa oculta a la capa de salida. Cuando se analizó la obtención de este tipo de arquitectura utilizando "Weight Decay", se observó que se podía obtener esta poda en todas las capas de la red neuronal. Sin embargo, ASELM no proporciona poda en la capa de salida. Para ello, los "targets" (salidas deseadas) de las tareas secundarias se utilizarán como nuevas entradas (quitándolas de las salidas), de tal forma que ASELM proporcione información sobre conexiones comunes con respecto a la tarea principal.

Esta idea de intercambiar salidas por entradas o viceversa, no es nueva. Caruana propuso la idea de que hay entradas que pueden funcionan mejor como salidas [22], demostrando cómo la transferencia del sesgo inductivo se puede realizar utilizando características de entrada como salidas deseadas (como si fuesen una nueva tarea secundaria). Esto es muy interesante para el caso de características perdidas o ausentes a la hora de clasificar un nuevo patrón. Por ejemplo, Caruana lo utiliza para ordenar una lista de espera de pacientes de los cuales no se dispone de toda la información en el proceso de ingreso, pero sí dispone de una amplia base de datos de los ya ingresados [24]. Posteriormente, Silver usa esta idea para crear su modelo MTL sensible

al contexto, *cs*-MTL ("Context Sensitive MTL") [99]. En este trabajo, Silver crea un modelo MTL donde las tareas secundarias se utilizan como entradas. Sin embargo, ese modelo de Silver es totalmente diferente al método propuesto, ya que él propone entrenar ese MLP donde los pesos de esas nuevas entradas se van *congelando*, de tal forma que permita de una forma secuencial ver que tareas van mejor que otras.

El método que se propone en esta sección, que se denominará MTL_{ASELM}, presenta dos sustanciales diferencias con respecto a ASELM:

1. Los "targets" de las tareas secundarias se quitan como salidas y se añaden como entradas, dejando, por lo tanto, una única salida que corresponde con la tarea principal.

2. Entrenar con ASELM.

3. Cuando ASELM finaliza, se crea el MLP pasando nuevamente las tareas secundarias que ASELM haya seleccionado (de igual forma que selecciona entradas y conexiones).

Estas dos diferencias con respecto al método original, permiten que la poda se realice tanto a nivel de la capa oculta como en la de salida. En la siguiente sección, entramos en más detalle sobre la metodología MTL_{ASELM} y cómo, paso a paso, se obtiene esta arquitectura.

5.4. Experimentos

Con el fin de mostrar la bondad del método para el diseño de una arquitectura MTL, se han comparado los resultados de clasificación del conjunto de test obtenidos con el aprendizaje monotarea (STL) y el aprendizaje multitarea

clásico (MTL). Para ver en más detalle la creación de la arquitectura propuesta, vamos a utilizar cuatro conjuntos de datos: "Logic Domain", "Iris Data", "Monk's Problems" y "Telugu". La forma de trabajar va a ser la misma que en capítulos anteriores (10-fold x 30 inicializaciones), con la salvedad de que aquellas que han obtenido una mala convergencia no se han descartado ya que queremos comprobar cómo se adaptan los esquemas a problemas como la adición de ruido en la tarea principal o caída en mínimos locales. Para los esquemas STL y MTL clásico se han seleccionado la mejor arquitectura mediante validación cruzada. A continuación, entramos en detalle en cada uno de estos problemas.

5.4.1. Logic Domain

Para este conjunto de datos, utilizaremos el modelo de cuatro funciones visto anteriormente cuando se analizó la utilización del "Weight Decay" (Tabla 5.1).

La arquitectura inicial se puede observar en la Figura 5.4, donde la única salida es la tarea principal (T_P), mientras que las otras tres tareas secundarias se utilizan como entradas extras. Una vez que este modelo se entrena con el método ASELM, el resultado es bastante significativo, el modelo selecciona dos neuronas, que vienen representadas por [0 0 1 1 0 0 0 0 1 0] y [1 1 0 0 0 0 0 0 0 0]. Se puede observar como la primera neurona seleccionada está compuesta por las características de entradas x_3 y x_4, y a la vez se activa la entrada correspondiente al "target" de la segunda tarea secundaria (ver Figura 5.5). Esto significa que esa tarea secundaria está influyendo en el aprendizaje de la neurona que aprende con las entradas x_3 y x_4, con lo que en el modelo final, habrá una conexión desde esta neurona hasta la salida de la tarea secundaria representada por T_{Sec_2}. La segunda neurona seleccionada, estará formada por

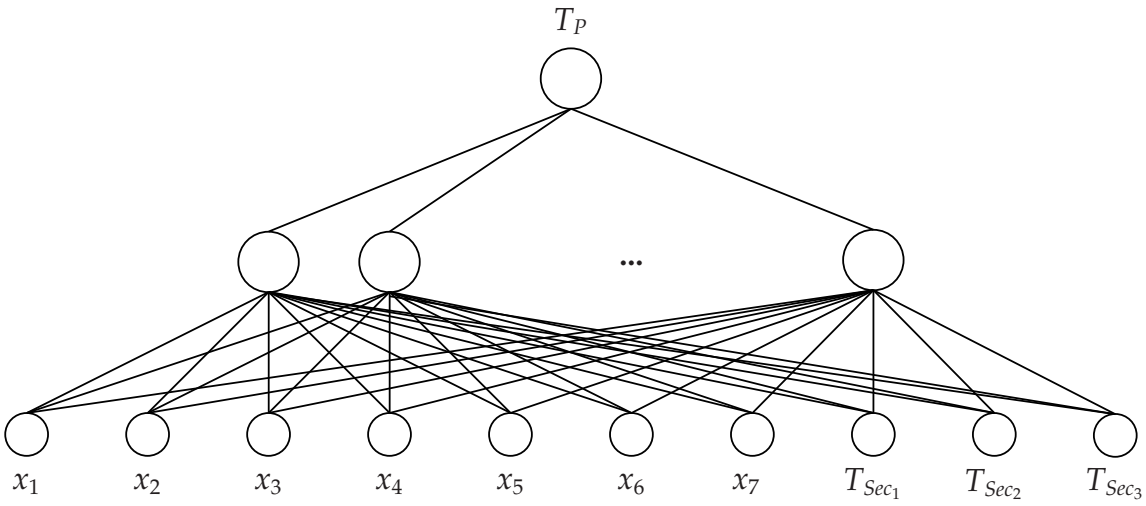


Figura 5.4: Logic Domain. Esquema inicial de MTL_{ASELM} para aprender la tarea principal utilizando las tareas secundarias como entradas.

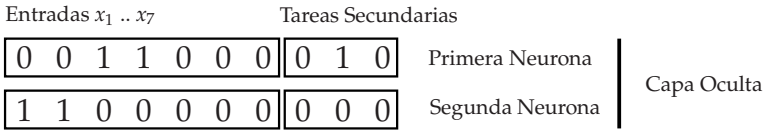


Figura 5.5: Logic Domain. Neuronas seleccionadas por MTL_{ASELM}. La primera neurona tendrá dos conexiones correspondientes a las entradas (x_3 y x_4). La segunda neurona viene representada por conexiones de la capa de entrada (x_1 y x_2). Las valores a "1" de las entradas correspondientes a los "targets" de las tareas secundarias representan conexiones que van desde la capa oculta a la de salida.

las características de entradas x_1 y x_2, sin ninguna conexión adicional de tareas secundarias.

La Figura 5.6 representa el esquema previo a la arquitectura final, en donde se han eliminado las conexiones y entradas irrelevantes para el aprendizaje de la tara principal. Finalmente en la Figura 5.7 se puede ver el esquema final para el aprendizaje MTL.

Los resultados de clasificación (Tabla 5.2) muestran como al tratarse de un conjunto muy fácil de aprender, para los tres esquemas planteados se obtiene un acierto de un 100 %.

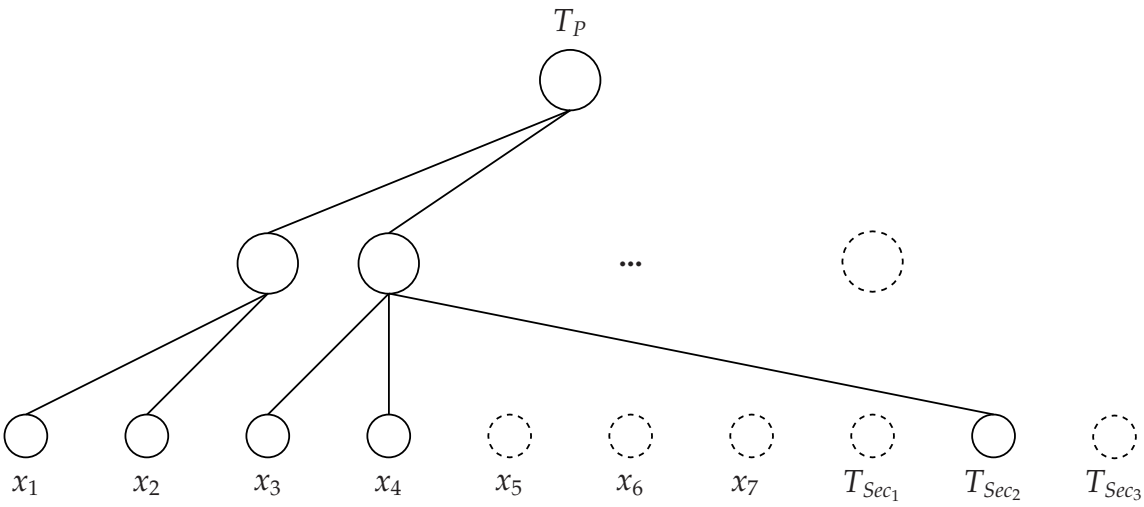


Figura 5.6: Logic Domain. Esquema intermedio donde se han podado los pesos. Se puede comprobar cómo las características de entrada x_5 y x_6, y las tareas secundarias T_{Sec_1} y T_{Sec_3} no son relevantes para el aprendizaje de la tarea principal.

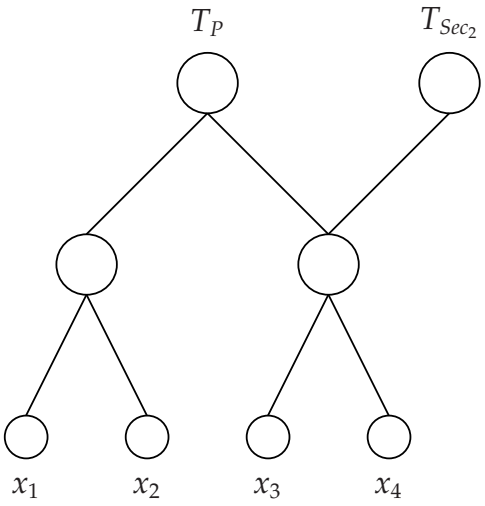


Figura 5.7: Logic Domain. Arquitectura final obtenida con MTL_{ASELM}. La T_P comparte una neurona de la capa oculta con la T_{Sec_2} que aprende la parte común de ambas tareas.

Tabla 5.2: Test de clasificación "TC" (media ± desv.estandard) con diferentes arquitecturas sobre el conjunto de datos "Logic Domain", donde se han penalizado muestras añadiendo ruido en las salidas deseadas.

Método	**Ruido**	**TC(mean ± std)**
STL	Sin ruido	1.000±0.000
	Ruido al 10 %	0.902±0.048
	Ruido al 20 %	0.774±0.052
MTL	Sin ruido	1.000±0.000
	Ruido al 10 %	0.877±0.055
	Ruido al 20 %	0.791±0.065
MTL_{ASELM}	Sin ruido	1.000±0.000
	Ruido al 10 %	0.972±0.037
	Ruido al 20 %	0.886±0.051

Para que se vea, en más detalle, la influencia de las tareas secundarias, al conjunto de datos se le ha añadido ruido en las salidas deseadas, en diferentes proporciones (10 % y 20 % de las muestras) de la tarea principal. Los resultados muestran que a medida que añadimos más ruido (dificultamos el aprendizaje de la tarea principal), el método MTL_{ASELM} mejora bastante los resultados con respecto al multitarea clásico y al aprendizaje en solitario. Además, se puede observar, como con poco ruido (10 %), el aprendizaje STL mejora al MTL clásico, ya que éste presenta un sistema totalmente interconectado que se ve influenciado positivamente por la tarea relacionada (T_{Sec_2}) y negativamente por las tareas no relacionadas (T_{Sec_1} y T_{Sec_3}). Sin embargo, al meter más ruido (20 %), el esquema MTL clásico mejora con respecto el aprendizaje STL, que aunque no recibe ninguna influencia de otras tareas, la alta tasa de ruido dificulta el aprendizaje en solitario. El método propuesto es el que menos se ve afectado por el ruido y obtiene mejores valores de clasificación.

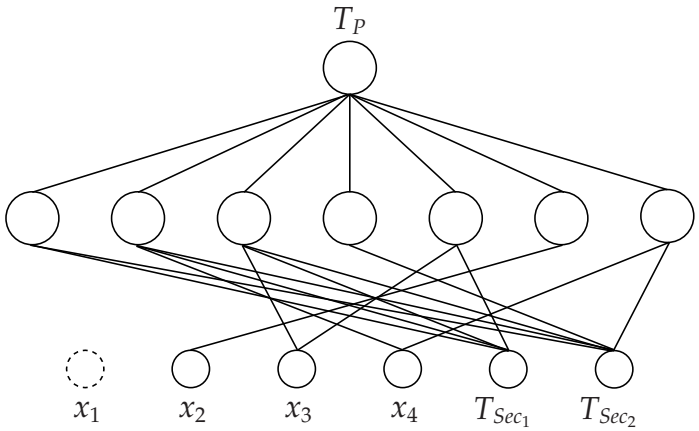


Figura 5.8: Iris. Esquema previo al final donde se han eliminado las conexiones y entradas irrelevantes.

5.4.2. Iris

Este conjunto trata de clasificar tres tipos de flor: Iris Setosa, Iris Versicolor e Iris Virgínica. La Iris Setosa es linealmente separable con respecto a las otras dos. Se va a utilizar como tarea principal el reconocimiento de la Iris Versicolor, por lo que la clasificación de Iris Setosa e Iris Virgínica serán utilizadas como tareas secundarias. Las Figuras 5.8 y 5.9 representan respectivamente el paso previo (red entrenada con ASELM) y final del método MTL_{ASELM}. En el primero, donde las tareas secundarias se utilizan como entradas para obtener la arquitectura, se ve que la entrada x_1 no presenta ninguna conexión con las neuronas de la capa oculta y por lo tanto se puede eliminar. Sin embargo, en la segunda figura, en donde utilizamos las dos tareas secundarias como salidas para crear el modelo final, se puede observar que además de eliminar la entrada x_1, se eliminan la primera y tercera neurona de la capa oculta, ya que estas sólo estaban conectadas con las tareas secundarias (utilizadas como entradas) y no con la tarea principal.

La Tabla 5.3 compara las tres arquitecturas. Por un lado, la clasificación de la

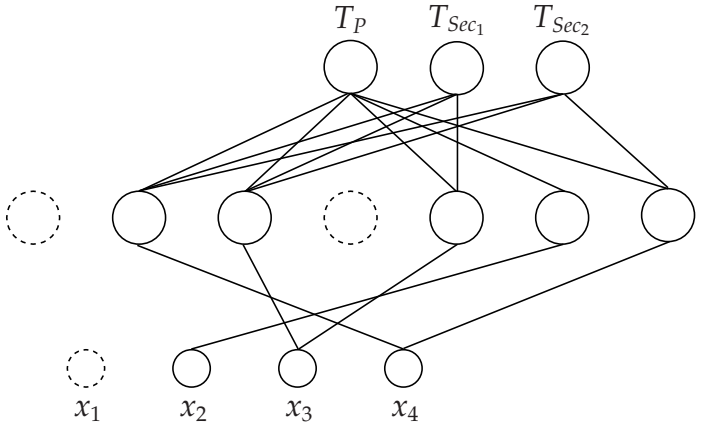


Figura 5.9: Iris. Esquema final para la arquitectura MTL. La característica de entrada x_1 se ha eliminado y las tareas secundarias se han pasado de entradas a salidas.

Tabla 5.3: Test de clasificación "TC" (media ± desv.estandard) con diferentes arquitecturas sobre el conjunto de datos "Iris", donde se han penalizado muestras añadiendo ruido en las salidas deseadas de la tarea principal.

Método	**Ruido**	**TC(mean ± std)**
STL	Sin ruido	0.978±0.005
	Ruido al 10 %	0.945±0.026
	Ruido al 20 %	0.912±0.030
MTL	Sin ruido	0.973±0.014
	Ruido al 10 %	0.963±0.014
	Ruido al 20 %	0.928±0.031
MTL_{ASELM}	Sin ruido	0.970±0.003
	Ruido al 10 %	0.961±0.012
	Ruido al 20 %	0.947±0.017

Iris Versicolor en solitario, es decir, utilizando un modelo STL donde no aparece ninguna tarea secundaria; por otro lado, el esquema multitarea clásico (MTL) y finalmente, el método MTL_{ASELM}. Los resultados muestran que a medida que añadimos más ruido, el método MTL_{ASELM} mejora bastante los resultados con respecto a STL y MTL. Además, también se puede observar como el método propuesto presenta una arquitectura mucho más simplificada (Figura 5.9).

5.4.3. Monk's Problems

"Monks's Problems" presenta un conjunto de tres tareas distintas a resolver donde las tres comparten el mismo dominio. La primera tarea es la que consideraremos como principal, mientras que las otras dos serán consideradas como secundarias. Se compone de seis características de entrada que toman valores numéricos enteros (x_1, x_2 y x_3 toman valores de 1 a 3, mientras que x_3 y x_6 toman valores de 1 ó 2, y x_5 los toma del 1 al 4). Las tareas vienen representadas por las siguientes funciones:

- Tarea principal: ($x_1 = x_2$) or ($x_5 = 1$)
- Primera tarea secundaria: exactamente dos coincidencias ($x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$, $x_6 = 1$)
- Segunda tarea secundaria: (x_5= 3 and $x_4 = 1$) or (x_5 <>4 and x_2 <>3). Esta tarea presenta un 5 % de ruido.

El método MTL_{ASELM} selecciona un total de 14 neuronas en la capa oculta. La Figura 5.10 presenta las cinco primeras neuronas y la última para la arquitectura seleccionada. Por ejemplo, se puede observar como la primera neurona presenta sólo conexión con la característica de entrada x_5 y con las salidas de la T_P y la

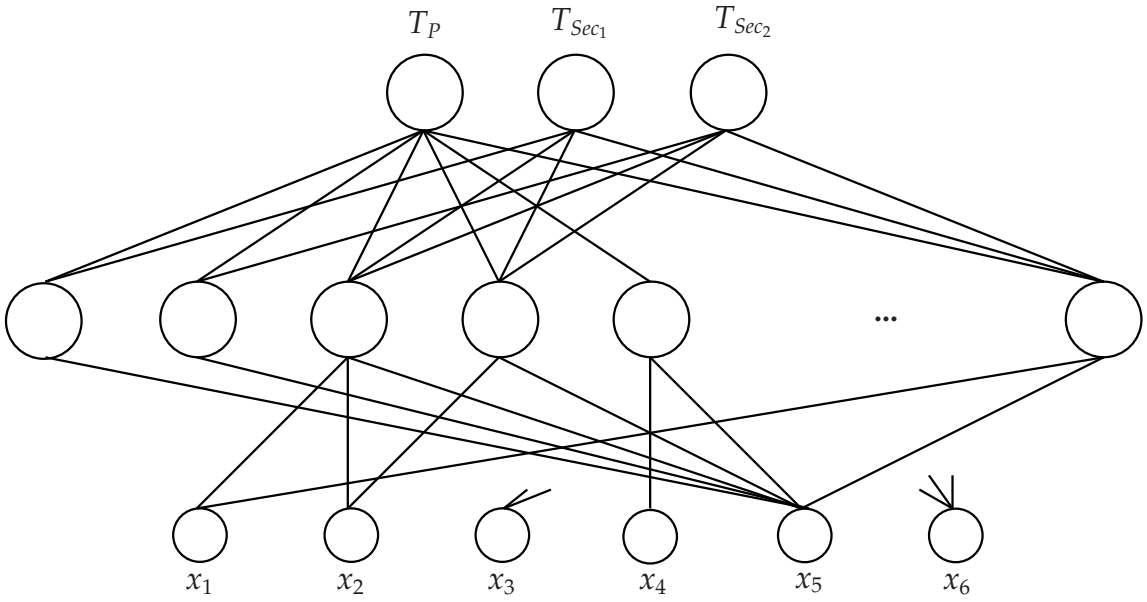


Figura 5.10: Monk. Esquema final.

T_{Sec_1} pero no con la T_{Sec_2}. Si vemos cómo se forma cada una de las tareas, se observa que tanto la T_P como la T_{Sec_1} coinciden en el valor de x_5, mientras que para la T_{Sec_2}, x_5 ha de ser diferente de 4, pero además x_2 ha de ser diferente de 3, con lo que como para x_2 no tenemos conexión, no se selecciona para la T_{Sec_2}.

La Tabla 5.4 muestra la probabilidad de acierto para clasificación de la tarea principal mediante los tres esquemas utilizados. Los resultados obtenidos son similares a los obtenidos en los ejemplos analizados anteriormente, MTL_{ASELM} obtiene una arquitectura más simplificada y eficiente para MTL.

5.4.4. Telugu

El dialecto hindú Telugu, es la segunda lengua materna oficial más hablada de la India. Se ha creado, a partir de los tres primeros formantes de cada vocal, un conjunto de datos de 871 muestras, [78]. Estos tres formantes corresponden

Tabla 5.4: Test de clasificación "TC" (media ± desv.estandard) con diferentes arquitecturas sobre el conjunto de datos "Monk's Problems", donde se han penalizado muestras añadiendo ruido en las salidas deseadas de la tarea principal.

Método	**Ruido**	**TC(mean ± std)**
STL	Sin ruido	0.954±0.016
	Ruido al 10 %	0.924±0.021
	Ruido al 20 %	0.766±0.066
MTL	Sin ruido	0.982±0.026
	Ruido al 10 %	0.935±0.037
	Ruido al 20 %	0.822±0.068
MTL_{ASELM}	Sin ruido	0.991±0.016
	Ruido al 10 %	0.930±0.028
	Ruido al 20 %	0.876±0.076

a tres características de entrada (x_1, x_2 y x_3). Aunque realmente este dialecto utiliza catorce vocales, en este conjunto de datos sólo se consideran seis: /a:/, /e/, /i/, /o/, /u/ y /d/.

El conjunto de datos está formado por 871 patrones. Se considerará la primera vocal (/a:/) como tarea principal, mientras que las otras cinco se considerarán como secundarias. Cabe destacar que al ser un problema de seis clases al ser nuestro objetivo aprender a clasificar la primera vocal con respecto a las demás, no es un problema balanceado. Nos encontramos que de los 871 patrones, sólo 72 pertenecen a la primera vocal. Para evitar los problemas que nos puede ocasionar el que los datos no estén balanceados, como la no convergencia, se balancearán los datos seleccionando para las tares secundarias las muestras más relevantes mediante la Edición de Datos, concretamente utilizando el algoritmo Condensed-*NN* visto en el Capítulo 3. Esto nos va a permitir reducir el número de patrones de las tareas secundarias manteniendo todos los de la tarea principal. El nuevo conjunto estará formado por 293 patrones. En la Tabla 5.5 se muestran el número de patrones finalmente obtenido por tarea secundaria.

Tabla 5.5: Número de patrones (NP) seleccionados mediante edición de datos para las tareas secundarias del problema Telugu.

Tarea Sec.	NP
T_{Sec1}	16
T_{Sec2}	39
T_{Sec3}	36
T_{Sec4}	69
T_{Sec5}	61

El método MTL_{ASELM} selecciona 4 neuronas en la capa oculta. La Figura 5.11 presenta la arquitectura final obtenida. Se puede observar que esta arquitectura utiliza sólo dos de los tres características de entrada, lo cual es bastante interesante ya que en dialectos con menos de seis vocales sólo son necesarios dos formantes para clasificarlos, sin embargo, aunque esta base de datos sólo contiene seis clases (correspondientes a seis vocales), el dialecto original está formado por catorce vocales, de ahí que el conjunto de datos esté formado por tres características de entrada (formantes).

La Tabla 5.6 muestra la probabilidad de acierto para clasificación de la tarea principal mediante los tres esquemas utilizados. Se puede apreciar claramente como el uso del MTL clásico mejora la probabilidad de clasificación de la tarea principal, sin embargo, al utilizar el método MTL_{ASELM} se obtienen mejores valores de clasificación que la arquitectura clásica.

5.5. Conclusiones

En este capítulo se presenta una forma de diseñar la arquitectura de un esquema MTL. Se ha utilizado un método de regularización como es el "Weight Decay" que nos ha permitido ver de una forma muy clara las relaciones entre

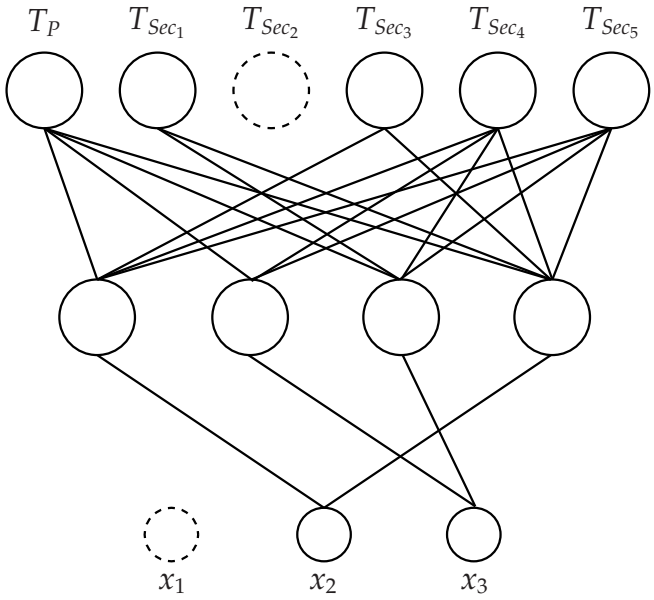


Figura 5.11: Telugu. Arquitectura MTL obtenida mediante ASELM. Se han eliminado la primera característica de entrada y la segunda tarea secundaria.

Tabla 5.6: Test de clasificación "TC" (media ± desv.estandard) con diferentes arquitecturas sobre el conjunto de datos "Telugu", donde se han penalizado muestras añadiendo ruido en las salidas deseadas de la tarea principal.

Método	**Ruido**	**TC(mean ± std)**
STL	Sin ruido	0.836±0.012
	Ruido al 10 %	0.832±0.014
	Ruido al 20 %	0.821±0.025
MTL	Sin ruido	0.841±0.011
	Ruido al 10 %	0.838±0.012
	Ruido al 20 %	0.823±0.016
MTL_{ASELM}	Sin ruido	0.866±0.026
	Ruido al 10 %	0.854±0.014
	Ruido al 20 %	0.843±0.021

tareas y la posibilidad de crear una arquitectura para MTL. Sin embargo, este método, por su naturaleza, no ofrece una arquitectura única. Con el objetivo de diseñar un método que solvente este inconveniente, se ha utilizado el método ASELM presentado en el capítulo anterior. Modificando este modelo se ha presentado una metodología denominada MTL_{ASELM} para la obtención de esta arquitectura única y simplificada. Este esquema permite utilizar la ventaja que presenta ASELM para simplificar la arquitectura de un MLP para utilizarlo en esquemas multitarea. Esta nueva arquitectura presenta la ventaja de obtener un esquema MTL donde se han eliminado características de entrada y conexiones irrelevantes para el aprendizaje de una tarea. De esta manera, se elimina uno de los principales problemas del aprendizaje multitarea, como es la influencia negativa de tareas no relacionadas. Además, al modificar el método ASELM para adaptarlo a multitarea, se ha conseguido no sólo la eliminación de conexiones de las entradas a la capa oculta sino también de la capa de salida. Este esquema es único, no depende de inicializaciones aleatorias y no hay que estar realizando pruebas de ensayo y error para determinar la mejor arquitectura, y no necesita ningún parámetro a configurar por parte del usuario.

En la sección de experimentos, se ha podido observar cómo este método permite obtener una arquitectura única para la clasificación de la tarea principal mediante un esquema multitarea. En problemas reales donde la dificultad intrínseca del problema dificulta encontrar esa relación entre las conexiones y las tareas a aprender, se observa que se obtiene una solución bastante simplificada y con buena capacidad de generalización respecto a la solución totalmente conectada que presenta el multitarea clásico. Además, se ha dificultado el aprendizaje de la tarea principal añadiendo ruido y se ha visto como el método propuesto produce con una buena capacidad de generalización, cosa que no

ocurre ni con el STL ni con el MTL clásico.

Capítulo 6

Contribuciones, conclusiones y trabajos futuros

6.1. Introducción

Cuando se aprende una determinada tarea, la utilización de un esquema multitarea MTL ("Multi-Task Learning") produce mejores soluciones que un esquema en donde se aprende la tarea en solitario. Esto ocurre siempre y cuando las tareas estén relacionadas y esa relación sea positiva. Sin embargo, encontrar esta relación no es trivial. En los problemas reales es difícil encontrar tareas relacionadas que mejoren el aprendizaje, e incluso, una vez encontradas, es difícil analizar el grado en que estas tareas benefician a la tarea que se considera principal. En esta Tesis, se aborda esta problemática desarrollando tareas relacionadas, que se utilizarán como secundarias en un esquema MTL, mediante procedimientos de Edición de Datos. Además, también se estudia otra problemática que presentan los esquemas multitareas: encontrar una ar-

quitectura óptima para la transferencia de información entre tareas. Mediante algoritmos de poda basados en el ELM ("Extreme Learning Machine"), se consigue una arquitectura única que mejora el aprendizaje MTL eliminando nodos y conexiones irrelevantes al aprendizaje de la tarea principal. También, esta forma de obtener la arquitectura nos permite descartar características de entrada que no aportan información relevante para el aprendizaje, por lo que también se consigue una selección de características de entrada. Las siguientes secciones muestran las contribuciones que aporta esta Tesis así como las conclusiones y trabajos futuros.

6.2. Contribuciones

Las principales contribuciones que presenta esta Tesis son la generación de tareas secundarias a partir de la tarea principal de forma artificial, para su uso en esquemas MTL. Además, se ha desarrollado un método para la obtención de una arquitectura que mejore el aprendizaje de un MLP. Y posteriormente, se ha utilizado esa arquitectura en esquemas MTL. A continuación se presenta un resumen, por temas, de las contribuciones de esta Tesis:

Capítulo 3:

1. Se utiliza un procedimiento de Edición de Datos, para obtener, de forma artificial, tareas relacionadas con la que se quiere aprender.

2. Se ha demostrado que estas tareas obtenidas de forma artificial permiten mejorar el aprendizaje de la tarea principal.

3. Se ha utilizado la generación de tareas artificiales sobre dos esquemas bien diferenciados. Por un lado, se ha hecho sobre la propia tarea principal, de

esta forma, se garantiza la relación con la tarea principal y permite utilizar un esquema similar al multitarea cuando no se tienen tareas secundarias. Por otro lado, se ha realizado este mismo procedimiento de generación de tareas sobre tareas secundarias reales en esquemas MTL, de tal forma, que se reduce considerablemente el número de épocas de entrenamiento frente al MTL clásico.

4. Las tareas secundarias generadas a partir de las tareas secundarias originales, son mejores para un enfoque MTL ya que en ellas se les ha eliminado el solapamiento y las muestras no significativas que, por estar muy alejadas de la frontera de decisión optima, no aportan información que permita ayudar a la tarea principal.

5. Este nuevo esquema multitarea reduce la probabilidad de caer en mínimos locales aumentando además la tasa de convergencia. La reducción del solapamiento en las nuevas tareas secundarias, hacen que se evite, en la mayoría de los casos, el caer en un mínimo local ayudando a la principal a encontrar la frontera de decisión óptima. Esto se debe a que las tareas secundarias actúan como guías ("hints") para el aprendizaje de la tarea principal, reduciendo el espacio de soluciones.

6. En el esquema propuesto se reducen las épocas de entrenamiento frente a un esquema multitarea clásico debido a que estas tareas secundarias convergen rápidamente ayudando desde épocas muy tempranas a la tarea principal.

Capítulo 4:

1. Se presenta un nuevo algoritmo basado en el algoritmo OP-ELM para construir automáticamente una arquitectura simplificada de un MLP. Es-

ta arquitectura proporciona automáticamente una solución única más fácil de resolver para el MLP produciendo una buena capacidad de generalización.

2. Esta solución única no requiere de la configuración de ningún parámetro adicional.

3. La nueva arquitectura se obtiene con una reducción considerable del coste computacional ya que no requiere de métodos de prueba y error, ni validación cruzada para encontrar el número óptimo de neuronas en la capa oculta.

4. Esta arquitectura permite también eliminar características de entrada irrelevantes o ruidosas, por lo que además de ser un método de poda (de conexiones y nodos ocultos) lo es de extracción de características.

Capítulo 5:

1. El algoritmo presentado en el Tema 4 se utiliza para crear una arquitectura para aprendizaje multitarea. Esta nueva arquitectura facilita la transferencia inductiva positiva de información.

2. Esta arquitectura multitarea elimina las tareas secundarias no relacionadas, eliminando uno de los principales problemas del aprendizaje multitarea, como es la influencia negativa de tareas no relacionadas.

3. Se presenta un método que proporciona una solución única, que poda tanto la capa oculta como la de salida, y que mejora la probabilidad de clasificación respecto a la solución totalmente conectada que presenta el multitarea clásico.

A continuación se presentan las publicaciones originadas por este trabajo de investigación, tanto en revistas como en congresos, así como en fase de revisión.

Publicaciones originadas por este Trabajo de Investigación

6.2.1. Capítulos de Libro

1. Pedro J. García-Laencina, Andrés Bueno-Crespo y José-Luis Sancho-Gómez, "Design and Training of Neural Architectures using Extreme Learning Machine", Neurocomputing: Learning, Architectures and Modeling, ISBN: 978-1-61324-699-3, Nova Science Publishers, 2011.

6.2.2. Revistas

Indexadas en ISI-JCR

1. Andrés Bueno-Crespo, Pedro J. García-Laencina y José-Luis Sancho-Gómez, "Neural architecture design based on Extreme Learning Machine.", Neural Networks 48: 19-24. 2013. ISSN: 0893-6080. Índice de impacto: 1,93. Base: JCR. Posición: 28/114. Área: Computer Science, Artificial Intelligence. Cuartil en la categoría: Q1.

2. Andrés Bueno-Crespo, Antonio Sánchez-García y José-Luis Sancho-Gómez, "Improving Learning using Artificial Hints.", Neurocomputing 79: 18-25 (2012). ISSN: 0925-2312. Índice de impacto: 1,58. Base: JCR. Posición: 39/111. Área: Computer Science, Artificial Intelligence. Cuartil en la categoría: Q2.

No indexadas

1. Andrés Bueno-Crespo, José Luis Sancho-Gómez, Rafael Verdú, "Artificial Construction of Task for Multitask Learning.", IADAT Journal of Advanced Technology on Imaging and Graphics, 1, 50-52. 2005. ISSN: 1885-6411

En revisión

1. Pedro J. García-Laencina, José-Luis Roca-González, Andrés Bueno-Crespo, José-Luis Sancho-Gómez, "Exploiting Diversity of Neural Network Ensembles based on Extreme Learning Machine.", Neural Network World. ISSN 1210-0552.

6.2.3. Congresos

Internacionales

1. Andrés Bueno-Crespo, Antonio Sánchez-García, Juan Morales-Sánchez, José-Luis Sancho-Gómez, "Multitask Learning with Data Editing". Bio-inspired Modeling of Cognitive Tasks, Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007. Lecture Notes in Computer Science, Cartagena (España), 4527: 320-326. 2007.

2. Andrés Bueno-Crespo y José-Luis Sancho-Gómez, "Artificial Construction of Tasks for Multitask Learning.", IADAT micv 2005. International Conference on Multimedia, Image Processing and Computer Vision. Palacio de Congresos - Madrid (España). 31/03/2005.

Nacionales

1. Andrés Bueno-Crespo, "Artificial tasks in medical diagnosis context.", Simposio Doctoral de la TICBioMed 2010. Universidad de Murcia. 16/06/2010.

2. Andrés Bueno-Crespo y José-Luis Sancho-Gómez, "Aprendizaje Multitarea en problemas con un número reducido de datos.", XX Simposium Nacional de la Unión Científica Internacional de Radio, URSI 2005. Gandía - Valencia. 15/09/2005.

6.2.4. Otras Publicaciones

Aquí se presenta otras publicaciones que aunque han sido realizadas durante este periodo de investigación, no está directamente relacionada con la Tesis. En concreto, una publicación indexada en ISI-JCR, un congreso internacional y tres capítulos de libro:

1. Antonio Sánchez-García, Andrés Bueno-Crespo, José-Luis Sancho-Gómez, "An efficient statistics-based method for the automated detection of sperm whale clicks", Applied Acoustics; 71, 451-459. 2009. ISSN: 0003-682X. Índice de impacto: 0,78. Base: JCR. Posición: 15/28. Área: Acoustics. Cuartil en la categoría: Q3.

2. Alberto Caballero, Miguel Angel Guillén, Andrés Bueno-Crespo, Belén López. "Experiencias en la impartición a distancia de varias asignaturas del Grado en Ingeniería Informática.", Congreso Internacional de Innovación Docente. Cartagena. 06/07/2011.

3. P. J. García Laencina, A. Bueno Crespo, J. L. Roca González, J. Roca González y G. Rodríguez Bermúdez, "Sistemas inteligentes: casos prácticos para la sociedad, la industria y la defensa", Las Tecnologías de Doble Uso: La Investigación y el Desarrollo al Servicio de la Sociedad Civil y Militar. 61-72, Primera Edición, Julio 2011. ISBN 978-84-939010-1-1.

4. J. L. Roca González, J. Roca González, G. Rodríguez Bermúdez, P. J. García Laencina, A. Bueno Crespo y J. Roca Dorda, "Las técnicas de organización industrial en contextos civiles y militares", Las Tecnologías de Doble Uso: La Investigación y el Desarrollo al Servicio de la Sociedad Civil y Militar. 73-82, Primera Edición, Julio 2011. ISBN 978-84-939010-1-1.

5. J. Roca González, G. Rodríguez Bermúdez, J. L. Roca González, P. J. García Laencina y A. Bueno Crespo, "Instrumentación y análisis de bioseñales en entornos BCI: Aplicaciones civiles y militares", Las Tecnologías de Doble Uso: La Investigación y el Desarrollo al Servicio de la Sociedad Civil y Militar. 109-120, Primera Edición, Julio 2011. ISBN 978-84-939010-1-1.

6.3. Conclusiones

Las conclusiones vienen derivadas de los experimentos que se han realizado en los capítulos 3, 4 y 5. Aunque al finalizar cada capítulo se ha incluido una sección de conclusiones, aquí se presenta un breve resumen de las mismas:

En el Capítulo 3, mediante procedimientos de Edición de Datos, se han creado tareas artificiales que serán utilizadas como nuevas tareas secundarias. Se han obtenido las siguientes conclusiones:

1. La obtención de tareas secundarias a partir de la tarea principal, mediante

la Edición de Datos, nos permite utilizar un esquema multitarea cuando no se dispone de tareas secundarias.

2. Por lo general, en un esquema multitarea con tareas secundarias que están relacionadas con la principal, y esta relación produce sesgo inductivo positivo, se obtiene una mejor clasificación que si las tareas secundarias se obtienen de la principal de forma artificial. Sin embargo, un esquema multitarea con tareas secundarias obtenidas de la principal producirá una mejor clasificación que entrenar la tarea principal en solitario o con tareas secundarias que no estén relacionadas.

3. Cuando las tareas secundarias son muy complejas, porque por ejemplo presentan solapamiento, ruido o simplemente alta dimensionalidad de muestras o características de entrada, puede ocurrir que la transferencia de información (sesgo inductivo positivo) hacia la tarea principal se vea dificultada. En ese caso, la utilización de la Edición de Datos para simplificar esas tareas secundarias eliminando muestras poco significativas y muestras solapadas hace que la información procedente de esas tareas secundarias ayuden a la principal a obtener una mejor convergencia.

4. La Edición de Datos utilizada para generar tareas secundarias, permite evitar, en la mayoría de los casos, mínimos locales y reducir el número de épocas de entrenamiento.

5. La arquitectura de la red neuronal suele ser similar cuando se utiliza el conjunto original y el editado, ya que la frontera de decisión de ambos conjuntos es similar. Esto nos permite reducir considerablemente el coste computacional, en la búsqueda de la arquitectura óptima, al tratarse de

un conjunto con un número de muestras bastante reducido con respecto al original.

Los Capítulos 4 y 5 presentan la obtención de la arquitectura para un MLP y posteriormente su adaptación para un esquema multitarea. Se han obtenido las siguientes conclusiones:

1. La nueva arquitectura obtenida mediante OP-ELM permite la transferencia inductiva positiva de información, por lo que es válida para entrenar un MLP y para un esquema multitarea.

2. La mayoría de los métodos de poda, obtienen una arquitectura diferente cada vez que se ejecutan debido a diferentes inicializaciones de pesos o por algún parámetro que ha de ser regulado. El método presentado en esta Tesis presenta una solución única y reduce considerablemente el coste computacional por no utilizar técnicas de prueba y error.

3. En problemas de alta dimensionalidad, encontrar tareas relacionadas o realizar poda, es un trabajo bastante costoso computacionalmente y la red puede tener dificultad a la hora de encontrar estas tareas relacionadas o las conexiones que ayudarán a la tarea principal. El método propuesto en esta Tesis, permite descartar características irrelevantes de entrada, eliminar conexiones y neuronas de la capa oculta que son innecesarias para el aprendizaje, y todo en un sólo paso, sin necesidad de validación cruzada.

4. Un esquema multitarea clásico se ve afectado por tareas secundarias que no están relacionadas, características de entrada irrelevantes o ruido. Por ello, antes de entrenar la red, hay que realizar técnicas de selección de

características e identificar las posibles relaciones entre tareas. El método presentado, hace todo esto de manera implícita, dando una arquitectura final única para el MLP para ser entrenado en un esquema multitarea.

6.4. Trabajos futuros

Como trabajos futuros se pretende estudiar diferentes métodos para medir la relación entre tareas y así, poder seleccionar la tarea secundaria (o tareas) que más eficientemente ayude en el aprendizaje de la tarea principal. También se pretende extender el método ASELM a otros modelos de máquinas de aprendizaje, tales como Redes de Funciones de base Radial (RBF), utilizarlo para resolver problemas de regresión (en esta Tesis se ha hecho para clasificación debido a la eliminación del "bias" que es una característica de optimización del algoritmo ELM para clasificación), así como extender este modelo al aprendizaje de conjuntos de datos con alta dimensionalidad. Esta limitación del método actual, es debida a la propia naturaleza del método ELM que está basado en el cálculo de la pseudoinversa. En este sentido, estamos trabajando en utilizar la aceleración proporcionada por la GPU y el cálculo secuencial de la pseudoinversa de Moore-Penrose [106, 103].

Apéndice A

Tabla de acrónimos

1-NN: 1-Nearest Neighbor.

AIC: Akaike Information Criterion.

AMS: Adaptative Minimization Schedule.

ASELM: Architecture Selection using Extreme Learning Machine.

ANN: Artificial Neural Network.

BP: Back-propagation.

Condensed-NN: Condensed Nearest Neighbor.

cs-**MTL:** Context Sensitive Multi-Task Learning.

CV: Cross Validation.

ELM: Extreme Learning Machine.

GP: Gaussian Processes.

GI: Ganancia de Información.

{*k*,*L*}-**NN:** {*k*, *L*}-Nearest Neighbor

k-**NN:** *k*-Nearest Neighbor.

LARS: Least Angle Regression.

LOO: Leave-One-Out.

MLP: Multi-Layer Perceptron.

MRSR: MultiResponse Sparse Regression.

MSE: Mean Squared Error.

MTL: Multi-Task Learning.

OBM: Optimal Brain Damage

OBS: Optimal Brain Surgeon

OP-ELM: Optimally Pruned Extreme Learning Machine.

P-ELM: Pruned Extreme Learning Machine.

PRESS: PREdiction Sum of Squares.

RBF: Radial Basis Function.

SLFNs: Single hidden Layer Feedforward neural Networks.

SSE: Sum-of-Squares Error

STL: Single-task Learning.

SVM: Support Vector Machine.

TRM: Task Rehearsal Method.

UCI MLR: University of California, Irvine, Machine Learning Repository.

Bibliografía

[1] Y. S. Abu-Mostafa. Learning from hints in neural networks. *Jounal of Complexity*, 6:192–198, 1990.

[2] Y. S. Abu-Mostafa. Hints. *Neural Computation*, 7:639–671, 1995.

[3] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[4] D. Andina, A. Vega-Corona, J. I. Seijas, and J. Torres. Neural networks historical review. *Computational Intelligence for Engineering and Manufacturing*, pages 39–65, 2007.

[5] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[6] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning form examples and local minima. *Neural Networks*, 2:53–58, 1989.

[7] E. B. Bartlett. Self determination of input variable importance using neural. *Neural, Parallel & Scientific Computations*, 2:103–114, 1994.

[8] A. Bartoli. On computing the prediction sum of squares statistic in linear least squares problems with multiple parameter or measurement sets. *International Journal on Computer Vision*, 85:133–142, November 2009.

[9] R. Battiti. Using the mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[10] J. Baxter. The evolution of learning algorithms for artificial neural networks, complex systems. *Complex Systems, IOS Press*, 1995.

[11] J. Baxter. Learning internal representations. In *Proceedings of the Eight International Conference on Computational Learning Theory*, Santa Cruz, CA, 1995. ACM Press.

[12] L. M. Belue and K. W. Baver. Determining input features for multilayer perceptrons. *Neurocomputing*, 7(2):111–121, 1995.

[13] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.

[15] S. Bos and E. Chug. Using weight-decay to optimize the generalization ability of a perceptron. *Proceedings of IEEE International Conference on Neural Networks*, 1:241–246, 1996.

[16] A. Bueno-Crespo, P.-J. García-Laencina, and J.-L. Sancho-Gómez. Neural architecture design based on extreme learning machine. *Neural Networks*, 48:19–24, 2013.

[17] A. Bueno-Crespo, J.-L. Sancho-Gómez, and R. Verdú. Artificial construction of task for multitask learning. *IADAT Journal of Advanced Technology on Imaging and Graphics*, 1:50–52, 2005.

[18] A. Bueno-Crespo, A. Sánchez-García, and J.-L. Sancho-Gómez. Improving learning using artificial hints. *Neurocomputing*, 79:18–25, 2012.

[19] R. Caruana. Multitask connectionist learning. In *In Proceedings of the 1993 Connectionist Models Summer School*, pages 372–379, 1993.

[20] R. Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664. Morgan Kaufmann, 1995.

[21] R. Caruana. Algorithms and applications for multitask learning. In *International Conference on Machine Learning*, pages 87–95, 1996.

[22] R. Caruana. Multitask Learning. *Phd Thesis, School of Computer Science, Carnegie Mellon University, Pittsburg, PA*, 1997.

[23] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[24] R. Caruana, S. Baluja, and T. Mitchell. Using the future to 'sortout' the present: Rankprop and multitask learning for medical risk evaluation. In *Advanced in Neural Information Processing Systems*, volume 8, pages 959–965, 1996.

[25] T. Cibas, F. F. Soulié, P. Gallinari, and S. Raudys. Variable selection with neural networks. *Neurocomputing*, 12:223–248, 1996.

[26] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[27] L. Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. *In Advances in Neural Information Processing Systems*, 2:598–605, 1990.

[28] B. V. Dasarathy. Nearest neighbor (NN) norms: NN pattern classification techniques. *Los Alamitos, CA: IEEE Computer Society Press*, 1991.

[29] P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[30] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[31] B. Durbin, S. Dudoit, and M. L. van der Laan. A deletion-substitution-addition algorithm for classification neural networks, with applications to biomedical data. Journal of Statistical Planning and Inference, 138:464–488, 2008.

[32] E. Eaton, M. desJardins, and T. Lane. Modeling transfer relationships between learning tasks for improved inductive transfer. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *ECML/PKDD (1)*, volume 5211 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2008.

[33] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[34] U. Fayyad and K. Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceedings of the International Joint Conference on Uncertainty in Artificial Intelligence*, pages 1022–1027, 1993.

[35] D. Francois, F. Rossi, V. Wertz, and M. Verleysen. Resampling methods for parameter-free and robust feature selection with mutual information. *Neurocomputing*, 70(7-9):1276–1288, 2007.

[36] B. Frénay and M. Verleysen. Using svm swith randomised feature spaces: an extreme learning approach. In *Proceedings of The 18th European Symposium on Artificial Neural Networks (ESANN).*, pages 315–320, Bruges, Belgium, 28-30 April 2010.

[37] Y. Fukuoka, H. Matsuki, H. Minamitani, and I. Akimasa. A modified back-propagation method to avoid false local mminima. *Neural Networks*, 11(2):1059–1072, 1998.

[38] S. A. Goldman. Computational learning theory, 1992.

[39] M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:76–86, 1992.

[40] M. Hagiwara. A simple and effective method for removal of hidden units and weights. *Neurocomputing*, 6(3):207–218, 1994.

[41] P. E. Hart. The condensed nearest neighbour rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[42] B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299, 1993.

[43] S. Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, July 1998.

[44] M. E. Hellman. The nearest neighbour classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6:179–185, (Reprinted in Dasarathy, 1991) 1970.

[45] J. Hertz, A. Krogh, and R. G. Palmer. Introduction to the theory of neural computation. *Adddison-Wesley Pub. Co., Redwood-City, CA.*, 1991.

[46] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[47] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Transactions on Neural Networks*, 9(1):224–229, 1998.

[48] G.-B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70:3056–3062, 2007.

[49] G.-B. Huang and L. Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71:3460–3468, 2008.

[50] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.

[51] G.-B. Huang, X. Ding, and H. Zhou. Optimization method based extreme learning machine for classification. *Neurocomputing*, 74(1-3):155–163, December 2010.

[52] G.-B. Huang and C.-K. Siew. Extreme learning machine: RBF network

case. In *8th ICARCV Control, Automation, Robotics and Vision Conference, 2004.*, volume 2, pages 1029–1036, 2004.

[53] G.-B. Huang and C.-K. Siew. Extreme learning machine with randomly assigned rbf kernels. *International Journal of Information Technology*, 11(1):16–24, 2005.

[54] G.-B. Huang, D. H. Wang, and Y. Lan. Extreme learning machines: A survey. *International Journal of Machine Leaning and Cybernetics*, 2(2):107–122, 2011.

[55] G.-B. Huang, Q. Zhu, and C.-K. Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 985–990, 2004.

[56] G.-B. Huang, Q. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[57] B. Igelnik and Y. H. Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 8(2):452–454, 1997.

[58] E.J. Kehoe. A layered network model of associative learning: Learning to learn and configuration. *Psychological Review*, 95(4):411–433, 1988.

[59] S. Kullback. *Information theory and statistics*. Wiley, 1959.

[60] N. Kwak and C.-H. Choi. Input feature selection by mutual information based on parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1667–1671, 2002.

[61] T. Y. Kwok and D. Y. Yeung. Constructive algorithms for structure learning in feedforward neural. *IEEE Transactions on Neural Networks*, 8(3):630–645, 1997.

[62] Q. Liu, Z. He, and Z. Shi. Extreme support vector machine classifier. *Lecture Notes in Computer Science.*, 5012:222–233, 2008.

[63] J. Madrid-Sánchez, E. Parrado-Hernández, and A. Figueiras-Vidal. Selective multitask learning by coupling common and private representations. In *NIPS2008, Workshop on Learning from Multiple Sources*, 2008.

[64] F. Mateo and A. Lendasse. A variable selection approach based on the delta test for extreme learning machine models. In *Proceedings of the European Symposium on Time Series Prediction (ESTP)*, pages 57–66, September 2008.

[65] P. J. McCracken. *Selective Representational Transfer Using Stochastic Noise*. PhD thesis, 2003.

[66] Y. Miche, P. Bias, C. Jutten, O. Simula, and A. Lendasse. A methodology for building regression models using extreme learning machine: Op-elm. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, pages 247–252, 2008.

[67] Y. Miche and A. Lendasse. A faster model selection criterion for OP-ELM and OP-KNN: Hannan-quinn criterion. In *Proceeding of the European Symposium on Artificial Neural Networks (ESANN)*, pages 177–182, April 22-24 2009.

[68] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse. OP-

ELM: Optimally Pruned Extreme Learning Machine. *Neural Networks, IEEE Transactions on*, 21(1):158–162, December 2009.

[69] Y. Miche, A. Sorjamaa, and A. Lendasse. OP-ELM: Theory, experiments and a toolbox. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, LNCS 5163, pages 145–154, 2008.

[70] T. M. Mitchell. The need for biases in learning generalizations. Technical report, 1980.

[71] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[72] T. M. Mitchell and S. B. Thrun. Explanation-based neural network learning for robot control. In *Advances in Neural Information Processing Systems 5*, pages 287–294. Morgan Kaufmann, 1993.

[73] R. Myers. *Classical and Modern Regression with Applications*. Duxbury Press, 2 edition, 1990.

[74] D.K. Naik and R. J. Mammone. Learning by learning in neural networks. In *Articial Neural Networks for Speech and Vision*, Chapman and Hall, London, 1993. ed. Richard J. Mammone.

[75] P. L. Narasimhaa, W. H. Delashmitb, M. T. Manrya, J. Li, and F. Maldonado. An integrated growing-pruning method for feedforward network training. Neurocomputing, 71:2831–2847, 2008.

[76] M. Norgaard, O. Ravn, L. K. Hansen, and N. K. Poulsen. The nnsysid toolbox - a matlab toolbox for system identification with neural networks. In *Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design*, 1996.

[77] J.M. Ortega. *Matrix Theory*. Springer, New York, London, 1987.

[78] S. K. Pal and D. Dutta Majumder. Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(8):625–629, 1977.

[79] Y. H. Pao, G. H. Park, and D. J. Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[80] CORPORATE PDP Research Group. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*. MIT Press, Cambridge, MA, USA, 1986.

[81] M. Pelillo and A. Fanelli. A method of pruning layered feed-forward neural networks. In J. Mira, J. Cabestany, and A. Prieto, editors, *New Trends in Neural Computation*, volume 686 of *Lecture Notes in Computer Science*, pages 278–283. Springer Berlin / Heidelberg, 1993.

[82] C. Penrod and T. Wagner. Another look at the edited nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*.

[83] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications*. Wiley, New York, 1971.

[84] R. Reed. Pruning algorithms - a review. *IEEE Transactions on Neural Networks*, 7:740–747, 1993.

[85] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, USA, 1996.

[86] A. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7:123–146, 1995.

[87] P. Rockett and S. Choi. The training of neural classifiers with condensed datasets. *IEEE Transactions on Systems, Man, and Cybernetics - part B: Cybernetics*, 32(2):202–206, April 2002.

[88] S. G. Romaniuk. Learning to learn: automatic adaptation of learning bias. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, AAAI'94, pages 871–876, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[89] H.-J. Rong, Y.-S. Ong, A.-H. Tan, and Z. Zhu. A fast pruned-extreme learning machine for classification problem. *Neurocomputing*, 72(1-3):359–366, 2008.

[90] D. Serre. *Matrices: Theory and Applications*. Springer, New York, 2002.

[91] R. Setiono and H. Liu. Improving backpropagation learning with feature selection. *Applied Intelligence: The Internation Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, 6(2):129–139, 1996.

[92] B.-D. Shai and R. Schuller. Exploiting task relatedness for multiple task learning. In *COLT*, pages 567–580, 2003.

[93] J. Sietsma and R. Dow. Creating artificial neural networks that generalize. *Neural networks*, 4:67–79, 1991.

[94] D. L. Silver. *Selective Transfer of Neural Network Task Knowledge*. PhD thesis, University of Western Ontario, 2000.

[95] D. L. Silver and P. McCracken. Selective transfer of task knowledge using stochastic noise. In *Proceedings of the 16th Canadian society for computational studies of intelligence conference on Advances in artificial intelligence*, AI'03, pages 190–205, Berlin, Heidelberg, 2003. Springer-Verlag.

[96] D. L. Silver and R. E. Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science Special Issue: Transfer in Inductive Systems*, 8(2):277–294, 1996.

[97] D. L. Silver and R. E. Mercer. The task rehearsal method of sequential learning, May 1998.

[98] D. L. Silver and R. E. Mercer. Selective functional transfer: Inductive bias from related tasks. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*, pages 182–191, Cancun, Mexico, May 2001. ACTA Press.

[99] D. L. Silver, R. Poirier, and D. Currie. Inductive transfer with context-sensitive neural networks. *Mach Learn*, 73:313–336, 2008.

[100] T. Similä and J. Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005*, LNCS 3697, pages 97–102, 2005.

[101] D. G. Stork and E. Yom-Tov. *Computer Manual in MATLAB to Accompany Pattern Classification, Second Edition*. Wiley-Interscience, 2004.

[102] S. C. Suddarth and Y. L. Kergosien. Rule-injection hints as a means of improving network performance and learning time. In *Proceedings of the*

EURASIP Workshop 1990 on Neural Networks, pages 120–129, London, UK, UK, 1990. Springer-Verlag.

[103] J. Tapson and A. van Schaik. Learning the pseudoinverse solution to network weights. *Neural Networks*, Available online 13 March 2013.

[104] S. Thrun. Lifelong learning: A case study. Technical report, CMU-CS-95-208, Carnegie Mellon University, 1995.

[105] S. Thrun and T. M. Mitchell. Learning one more thing. Technical report, CMU-CS-94-184, Carnegie Mellon University, 1994.

[106] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse. Gpu accelerated and parallelized elm ensembles for large-scale regression. *Neurocomputing*, 74(16):2430–2437, 2011.

[107] T. Wagner. Convergence of the edited nearest neighbor. *IEEE Transactions on Information Theory*.

[108] L. Wang and W. Chunru. Comments on "the extreme learning machine". *IEEE Transactions on Neural Networks*, 19(8):1494–1495, 2008.

[109] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight elimination with application to forecasting. Advances in Neural Information Processing System, III:875–882, 1991.

[110] W. Weishui Wan, S. Mabu, K. Shimada, K. Hirasawa, and J. Hu. Enhancing the generalization ability of neural networks through controlling the hidden layers. *Applied Soft Computing*, 9:404–414, 2009.

[111] H. White. Learning in artificial neural networks: a statistical perspective. *Neural Computation*, 1(4):425–464, 1989.

[112] D. Wilson. Asymptotic properties of nearest neighbor rules using editing data. *IEEE Transactions on Systems, Man and Cybernetics*, 2:408–421, 1972.