Universidad Nacional de Educación a Distancia
Departamento de Física Fundamental

Trabajo de fin de Máster

Predicción del rendimiento académico en las Matemáticas de la educación Secundaria mediante Redes Neuronales

Autor:
Roi Álvarez Rodríguez

Tutora:
Elka Radoslavova Koroutcheva
Curso 2019/2020

Resumen

Las redes neuronales son sistemas de procesamiento de información, que se enmarcan dentro del aprendizaje automático o *machine learning*. Por medio de esta técnica se pueden obtener predicciones o resultados sobre problemas cuya resolución implicaría un elaborado y complejo plan. La educación, es uno de los pilares fundamentales de nuestra sociedad, y es permeable al empleo de las nuevas tecnologías, como el *machine learning* para mejorar. Desde hace años se vienen utilizando técnicas de *data mining* en este ámbito para analizar los datos que generan, por ejemplo, los cursos universitarios *online* para obtener una predicción del rendimiento del alumnado en este tipo de cursos.

En el presente trabajo se emplea esta técnica para la predicción del rendimiento académico, con datos obtenidos en un entorno docente real, con alumnos de secundaria de un colegio. Con una muestra de unos pocos registros, y un conjunto pequeño de datos para cada uno (notas, identificador de grupo y medidas de atención a la diversidad) se han entrenado una serie de redes neuronales multicapa *feedforward* con el algoritmo ***backpropagation*** (propagación de errores hacia atrás), con el fin de comprobar si estos sistemas son capaces de predecir el rendimiento académico. También se quiere comprobar la calidad de esa predicción en función de la información de la entrada, del número de unidades de capa oculta y si se mejora el resultado complicando la configuración de la red con una segunda capa oculta o combinando las posibles salidas.

Con los resultados obtenidos se puede concluir que la predicción del rendimiento académico en la etapa secundaria, con los datos obtenidos en un entorno docente real, es posible. La precisión que ofrecen los resultados es razonablemente buena, en muchos casos superior al 90 %, pero que disminuye a la hora de dar una nota numérica. El incluir una estimación de las medidas de atención a la diversidad resulta adecuado y refina en muchos casos los resultados obtenidos.

Abstract

Artificial Neural Networks are information processing systems that belong to machine learning. Through this technique predictions or results can be obtained to solve problems that would require an overly complicated plan to reach a solution. Education, one of the fundamental cornerstones of our society, is especially receptive to the use of new technologies, such as *machine learning*. Over the last years, *data mining* techniques are being used in this context to analyze data generated by, for example, online university courses in order to obtain a performance prediction of its students.

This technique is used in the present thesis to predict academic performance, using data obtained from a real educational context with secondary level students from a single school. Using a sample of a few students and a small set of data for each (grades, group identification and attention to diversity measures) a series of feedforward multilayered neural networks have been trained with a ***backpropagation*** algorithm in order to test if these systems are capable of predicting academic performance. Furthermore, a series of additional objectives are tested, such as: the prediction quality based on the input information, the number of hidden layer units, and if results can improve increasing the network's configuration difficulty introducing a second hidden layer or combining the possible outputs.

The conclusion that can be drawn given the results and obtained data from a real educational context, is that academic performance for secondary level students is possible. The accuracy offered by the results is reasonably good, reaching 90 % in numerous cases but showing a decrease when using numerical marks. The inclusion of diversity attention measures proves to be adequate, refining the majority of the results.

Índice general

Capítulo 1

Introducción

El presente trabajo de fin de máster tiene como objetivo utilizar técnicas de aprendizaje automático (*machine learning*), como las redes neuronales artificiales, en el ámbito educativo para obtener una predicción de notas finales y de medidas de atención a la diversidad, con datos obtenidos en un entorno educativo real, un colegio gallego. Las técnicas que engloba el aprendizaje automático se han utilizado para clasificación y predicción de rendimiento académico en cursos *online* de diverso tipo, como el preuniversitario[1], universitario [2, 3, 4] o en cursos *elearning* [5]. La mayoría de trabajos están englobadas en lo que se conoce como la Minería de Datos en Educación (*EDM* en sus siglas en inglés)[6, 7, 8, 9]. En el transcurso de la fase de recopilación de información del presente trabajo, se encontraron algunos artículos sobre estudios de campo dentro de los tramos básicos de la educación, como la etapa secundaria [10, 11], que comprende la franja de entre los 12 y 16 años.

Dentro de la bibliografía es difícil encontrar trabajos sobre las matemáticas, específicamente, como tema principal de un estudio, como por ejemplo el de Lye [1]. En otros trabajos, su importancia está más difuminada, siendo una un parámetro más dentro del diseño de algún modelo propuesto como objeto de estudio, como por ejemplo en el trabajo de Cortez [10] o en otros trabajos como [4, 11].

La principal característica del presente trabajo es que cuenta con un pequeño estudio de campo, con datos obtenidos del rendimiento académico reglado de alumnos de un colegio de secundaria: notas medias de exámenes presenciales y otras características medibles como las dificultades de aprendizaje o el grupo de un curso específico. No son métricas obtenidas de una plataforma *online* (*clicks* o tiempo de conexión de una sesión), ni de pruebas digitales. No es mi intención desmerecer este tipo de registros obtenidos en otros trabajos, ya que en muchos casos son los únicos disponibles, sino poner en valor la ventaja que tiene el presentar los datos de este trabajo. Datos con los que se realizará una predicción del rendimiento académico obtenido en el curso 2018/19.

1.1. Machine Learning

El aprendizaje automático, o *machine learning*, se puede definir como "el campo de estudio que dota a los ordenadores de la capacidad de aprender a resolver problemas para los que no han sido explícitamente programados", cita que se atribuye a Arthur Samuel [12], aunque de difícil localización por no estar escrita explícitamente de esta manera. El aprendizaje automático se enmarca dentro de la Inteligencia Artificial, donde se entiende que el aprendizaje es un proceso por el que un ordenador es capaz de mejorar su habilidad de resolución de un problema a través de la adquisición de conocimiento, el cual se obtiene a través de la experiencia [13]. Esta experiencia se puede obtener a partir de datos, que con la reciente explosión del *Big Data* en los últimos años, ha sufrido un importante empuje. Con esto surgió la necesidad de trabajar de manera eficiente sobre conjuntos extensos de datos, lo que provocó el desarrollo de la minería de datos (*data mining*) que antes se mencionó en el ámbito de la educación [7].

El aprendizaje automático es una ayuda en tareas tediosas susceptibles de automatizar, que conllevan cierta complicación elaborar un modelo y la programación de algoritmos específicos [14]. Esto es una de las muchas ventajas de esta manera de resolver problemas, ya que simplemente bastaría con suministrar un conjunto de datos de entrada, aunque en ocasiones, para obtener un resultado aceptable, el tamaño de este conjunto puede ser considerable. Resulta útil en tareas de reconocimiento de patrones [15, 16, 17] como identificación de objetos, reconocimiento por voz, etc. el aprendizaje automático empieza a acercarse (y posiblemente superar) la precisión humana.

Ejemplos actuales y que han pasado a formar parte de nuestro entorno (incluso en el virtual) donde se emplea el aprendizaje automático son, entre otros, en los asistentes digitales (incluidos o no en dispositivos físicos diseñados específicamente para tal uso), las recomendaciones personalizadas de algunas plataformas de *streaming*, aplicaciones domóticas, aplicaciones de smartphones, el coche autónomo, etc.[14] Cabe indicar otras, como los sistemas de reconocimiento y vigilancia de autoridades de otros países.

Para hacer una primera clasificación sobre el aprendizaje hay que fijarse en el proceso de adquisición del conocimiento, con lo que habría dos tipos principalmente [14]:

- Aprendizaje Supervisado [18, 19, 20]. A partir de ejemplos (también se denomina "con profesor"), estos (ejemplos de entrenamiento) van acompañados de la salida correcta que el sistema debería ser capaz de reproducir. El entrenamiento consiste en ajustar los parámetros del modelo de aprendizaje para que sea capaz de reproducir una salida lo más parecida posible a la deseada. Una vez ajustados los parámetros del modelo se busca que sea capaz de generalizar o que el modelo proporcione, para un conjunto de datos distinto al empleado en el entrenamiento, unas salidas satisfactorias

- Aprendizaje No Supervisado [21, 17, 16]. Es por observación (también llamado

"sin profesor"), donde se construyen hipótesis a partir de un conjunto de hechos u observaciones, sin que exista información adicional acerca de cómo deberían clasificarse los ejemplos del conjunto de entrenamiento. Será el propio método el que realice el agrupamiento de los datos de entrenamiento (*clustering*) o el tipo de patrones más interesantes dentro de los datos de entrenamiento.

Existen otros tipos, minoritarios, como el de imitación (cuyo uso es anecdótico) o como el aprendizaje por refuerzo [14], relacionado con el condicionamiento clásico (experimento de Pavlov): una serie de respuestas involuntarias debidas a la aplicación de una serie de estímulos como las recompensas. En este aprendizaje se van ajustando las predicciones para que encajen mejor con el futuro. Se van comparando las recompensas, la esperada con la observada, lo que sirve como señal de error. Dependiendo si la recompensa real es superior a la esperada o inferior, se incentiva el comportamiento aumentando las expectativas o rebajándolas, para que la predicción se acerque más al resultado observado.

1.1.1. Redes Neuronales Artificiales

El eje central del trabajo radica en la predicción o clasificación del rendimiento de alumnos/as por medio de redes neuronales artificiales. Estas están englobadas, dentro del aprendizaje automático, y reciben su nombre del hecho de que imitan la manera de trabajar de nuestro cerebro [14, 22, 23, 24].

Los elementos básicos de una sistema neuronal biológico son las neuronas, que se agrupan en conjuntos compuestos por millones de ellas organizadas en capas constituyendo un sistema con funcionalidad propia [14, 22, 23, 24]. Un conjunto de estos subsistemas da lugar a un sistema global, como el nervioso en el caso biológico. En la realización de un sistema neuronal artificial puede establecerse una estructura jerárquica similar. El elemento esencial de partida sería la neurona artificial, que se organizará en capas; varias capas constituirán una red neuronal; y, por último, una red neuronal (o un conjunto de ellas), junto con las interfaces de entrada y salida, más los módulos convencionales adicionales necesarios, constituirán el sistema global de proceso [14, 22, 23, 24].

Las redes neuronales artificiales son sistemas de procesamiento de la información donde su estructura y funcionamiento están inspirados en las redes neuronales biológicas. Consisten en un número de elementos simples de procesamiento llamados nodos, unidades o neuronas que están organizados en capas [14, 22, 23, 24]. Cada neurona está conectada con otras neuronas mediante enlaces de comunicación, cada uno de los cuales tiene asociado un peso. En los pesos se encuentra el conocimiento que tiene la red acerca de un determinado problema. La idea que subyace en estos sistemas, es que puede resultar conveniente el construir algún tipo de sistema que copie la estructura de las redes neuronales biológicas, con la finalidad de obtener un rendimiento parecido al del cerebro, a la hora de abordar ciertos problemas que este resuelve eficientemente [14].

Según el grupo *Parallel Distributed Processing Research Group*, de la Universidad de California en San Diego (***PDP***) un sistema neuronal está compuesto por los

siguientes elementos [25, 26]:

- Un conjunto de procesadores elementales o neuronas artificiales.
- Un patrón de conectividad o arquitectura.
- Una dinámica de activaciones.
- Una regla o dinámica de aprendizaje.
- El entorno donde opera.

Neurona biológica

El cerebro es uno de los elementos importantes de nuestro sistema nervioso, que contiene alrededor de cien mil millones de neuronas [22, 23, 24]. Estas constituyen procesadores de información sencillos y, aunque pueden presentarse de muchas formas, la gran mayoría tienen un aspecto similar observadas al microscopio, como esquemáticamente se representa en la figura 1.1. En todas las neuronas se identifican las siguientes partes principales [22, 23, 24]:

- El cuerpo celular o soma, de entre 10 y 80 μm de longitud. Centro de síntesis de la célula, que procesa las señales que le llegan de otras células, en forma de impulsos, generando un nuevo impulso si se cumplen ciertas condiciones.
- Las dendritas. Son ramas fibrosas que surgen del cuerpo celular. Son las conexiones de entrada de la neurona. Una prolongación del cuerpo de la célula nerviosa, que conduce el impulso nervioso hacia el cuerpo de la neurona.
- El axón. Es la fibra principal que emana del cuerpo celular, su longitud varía entre los 100 μm y el metro en el caso de las neuronas motoras. Es el canal transmisor de los impulsos generados por la célula. Se ramifica en su extremo final para conectar con otras neuronas, a través de las dendritas de estas, que actúan como canales receptores de información. Cuando el axón está cerca de sus células de destino, se divide en muchas ramificaciones que forman sinapsis con el soma o axones de otras células. Esta unión puede ser inhibidora o excitativa según el neurotransmisor que libere.

La unión entre dos neuronas se denomina sinapsis. En el tipo más común no existe un contacto físico entre las neuronas, permanecen separadas por un vacío de unos 0,2 μm [22]. Las sinapsis son direccionales, la información fluye en un único sentido desde las neuronas presinápticas (las que envían las señales) a las postsinápticas (las que las reciben).

Las señales nerviosas se pueden transmitir de manera eléctrica o de manera química [22]. La transmisión química prevalece fuera de la neurona, mientras que la eléctrica lo hace en el interior de la neurona. Las transmisión química se basa en el intercambio de neurotransmisores, mientras que la eléctrica hace uso de descargas que se producen en el cuerpo celular y se propagan por el axón [22].

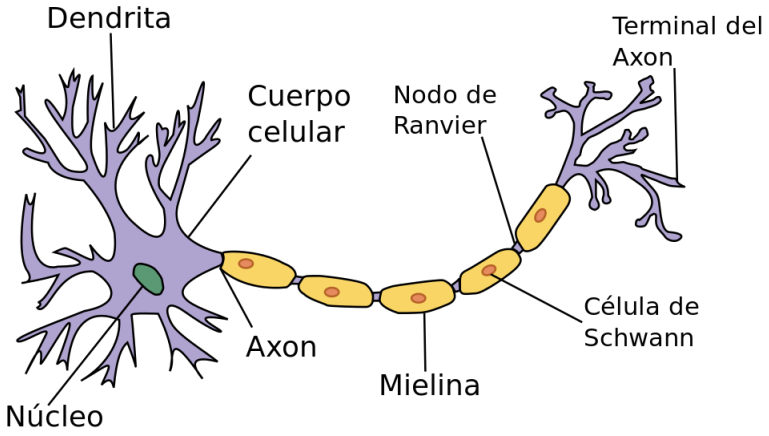


Figura 1.1: Modelo de neurona biológica. Fuente: Wikimedia Commons.

La cantidad de señal transferida depende de la cantidad química aportada por el axón y recibida por las dendritas. La intensidad sináptica es la que resulta modificada cuando decimos que el cerebro aprende. Las sinapsis combinadas con el proceso de información de la neurona, forman el mecanismo básico de la memoria. A la intensidad de una conexión se le llama peso sináptico. En esas uniones especiales, se produce una transformación del impulso eléctrico en un mensaje neuroquímico, mediante la liberación de neurotransmisores [22].

El resultado de estos neurotransmisores sobre la neurona receptora puede ser excitativo o inhibitorio, y es variable, de manera que podemos hablar de la fuerza o efectividad de una sinapsis [22]. Las señales excitativas o inhibitorias se combinan y en función de la estimulación recibida, la neurona toma un cierto nivel de activación que se traduce en la generación de breves impulsos nerviosos con una determinada frecuencia o tasa de disparo, y su propagación a lo largo del axón hacia las neuronas a las que transmite el impulso.

De esta manera, la información se transmite de unas neuronas a otras y va siendo procesada a través de las conexiones sinápticas y las propias neuronas. El aprendizaje de las redes neuronales se produce mediante la variación de la efectividad de las sinapsis; de esta manera cambia la influencia que unas neuronas ejercen sobre otras, de aquí se deduce que la arquitectura, el tipo y la efectividad de las conexiones en un momento dado, representa en cierto modo la memoria o estado de conocimiento de la red [22].

Biológicamente, se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con ellas mismas [27, 14, 22, 24].

Modelo de neurona artificial

Se denomina neurona a un dispositivo simple de cálculo, un procesador de información sencillo, que a partir de una señal de entrada procedente del exterior, u otras neuronas, proporciona una respuesta única o salida [22, 23, 24]. Los elementos

que constituyen la neurona i se muestra esquemáticamente en la figura 1.2 y son los siguientes [22]:

- Conjunto de entradas $x_j(t)$
- Pesos sinápticos de la neurona i, w_{ij}, que representan la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i. Cuando recibe una entrada positiva, si el peso es igualmente positivo, tenderá a excitar a la neurona postsináptica, si el peso es negativo tenderá a inhibirla. Así se habla de sinapsis excitativas (cuyo peso es positivo) e inhibidoras (cuyo peso es negativo).
- Regla de propagación $\sigma(w_{ij}, x_j(t))$, que proporciona el valor del potencial postsináptico $h_i(t) = \sigma(w_{ij}, x_j(t))$ de la neurona i en función de sus pesos y entradas. La función más habitual es de tipo lineal, y se basa en la suma ponderada de las entradas con los pesos sinápticos

$$h_i(t) = \sum_j w_{ij} x_j \tag{1.1}$$

que puede interpretarse como el producto escalar de los vectores de entrada y el de los pesos sinápticos.

Una regla de tipo no lineal sería:

$$h_i(t) = \sum_{j_1, j_2, \ldots, j_p} w_{ij} x_{j_1} x_{j_2} \cdots x_{j_p} \tag{1.2}$$

que implica una interacción de tipo multiplicativo entre las entradas de las neuronas, lo que implica una mayor complejidad.

Otra regla de propagación es la distancia euclídea

$$h_i^2(t) = \sum_j (x_j - w_{ij})^2 \tag{1.3}$$

que representa la distancia cuadrática existente entre el vector de entradas y el de los pesos.

- Función de activación $f_i(a_i(t-1), h_i(t))$, que proporciona el estado de activación actual $a_i(t) = a_i(t-1), h_i(t)$ de la neurona i, en función de su estado anterior $a_i(t-1)$ y de su potencial postsináptico actual. En muchos casos, se considera que el estado actual de la neurona no depende del estado anterior, sino únicamente del actual $a_i(t) = f_i(h_i(t))$. La función de activación se suele considerar determinista, y en la mayor parte de los modelos es monótona creciente y continua, como se observa en las neuronas biológicas. En ocasiones los algoritmos de aprendizaje requieren que la función de activación cumpla la condición de ser derivable. Las más empleadas en este sentido son las de tipo sigmoideo. Se puede consultar en el apéndice (A) los tipos de funciones de activación más comunes.

- Función de salida $F_i(a_i(t))$, que proporciona la salida actual $y_i(t) = F_i(a_i(t))$ de la neurona i en función de su estado de activación. En la mayoría de las ocasiones se suele considerar la función identidad, con lo que el estado de activación de la neurona es la propia salida. Se emplean también funciones del tipo estocásticas, con lo que la neurona tendrá comportamiento probabilístico [24].

Este modelo de neurona se inspira en la operación de la neurona biológica, en el sentido de integrar una serie de entradas y proporcionar cierta respuesta, que se propaga por el axón.

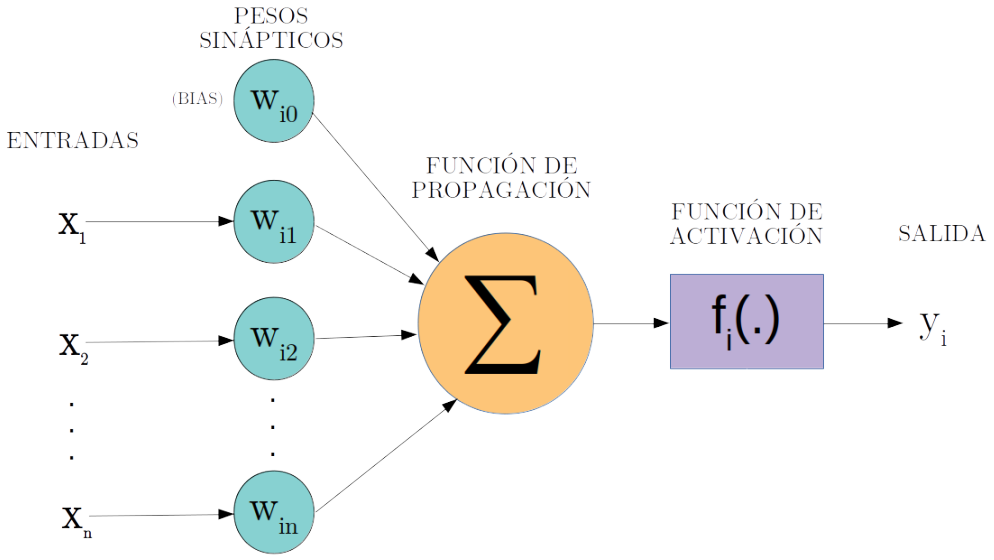


Figura 1.2: Modelo de neurona artificial.

La respuesta de las neuronas biológicas es de tipo no lineal, característica que se intenta emular en los sistemas artificiales [14, 22, 23, 24]. La formulación de la neurona artificial como un dispositivo no lineal constituye una de sus características más destacables y suele emplearse en problemas no lineales, donde no es sencillo abordar las posibles soluciones a partir de técnicas convencionales.

Arquitectura de una red neuronal

La arquitectura es la topología, estructura o patrón de conexiones de una red neuronal, determina el comportamiento de la red [14, 22, 23, 24]. Las conexiones sinápticas son direccionales, con lo que la información puede propagarse en un único sentido (desde la presináptica a la postsináptica). Las neuronas se suelen agrupar en unidades estructurales que se denominan capas. Las neuronas pertenecientes a una capa pueden agruparse, a su vez, formando grupos neuronales o *clusters*. Dentro de una capa, si no se da este tipo de agrupamiento, las neuronas suelen ser del mismo tipo. El conjunto de una o más capas constituye la red neuronal [14, 22, 23, 24].

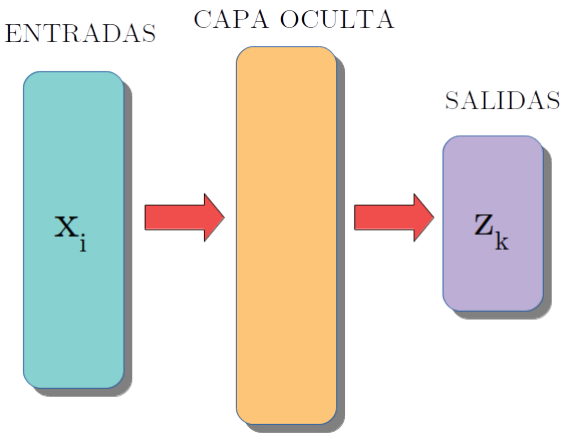


Figura 1.3: Modelo de red neuronal con tres capas: entradas, capa oculta y salidas.

En cuanto a los tipos de capas, existen tres tipos, como se muestra en la figura 1.3:

- Capa de entrada. Está compuesta por neuronas que reciben datos o señales procedentes del entorno.
- Capa oculta. No está conectada directamente con el entorno, sino a través de las capas de entrada y salida o de otras capas ocultas. Este tipo de capas dota a la red de grados de libertad adicionales, gracias a los cuales puede encontrar representaciones internas correspondientes a determinados características del entorno.
- Capa de salida. Las neuronas proporcionan la respuesta de la red neuronal.

En cuanto a las conexiones, pueden ser excitativas o inhibitorias, como se mencionó previamente. Además, atendiendo al modo en que se conectan, se dan los siguientes tipos [22, 23]:

- Conexiones entre capas, que se dan entre neuronas de distintas capas.
- Conexiones en la misma capa o laterales, que se dan entre neuronas pertenecientes a la misma capa.
- Conexiones realimentadas, que tienen un sentido contrario al de entrada-salida.

Clasificación de redes neuronales

Las posibles clasificaciones [22, 23, 24] de las redes neuronales atienden a una serie de parámetros como el número de capas, el flujo de información, el tipo de entrenamiento, etc. Atendiendo al número de capas se pueden clasificar las redes en monocapa o multicapa. Atendiendo al flujo de datos [24], entonces se puede hablar

de redes unidireccionales o *feedforward* donde la información circula en un único sentido y redes recurrentes (*feedback*) o realimentadas donde la información puede circular en cualquier sentido [22, 23].

También se pueden clasificar según el tipo de entrenamiento [23] como autoasociativas o heteroasociativas. La segunda, ante un determinado patrón de entrada responde con un cierto patrón de salida. En cambio la primera se asocia un patrón consigo mismo, un ejemplo sería una red de tipo Hopfield (o modelo de Hopfield) [28, 29, 24, 30, 23], que consiste básicamente en una única capa de neuronas, donde cada una se conecta con todas las demás.

Clásicamente se distinguen dos modos de operación en los sistemas neuronales [22], el modo de recuerdo o ejecución y el modo de aprendizaje o entrenamiento. Este último es de especial interés ya que determina que las redes neuronales son sistemas entrenables, capaces de realizar un determinado tipo de procesamiento aprendiéndolo a partir de un conjunto de patrones de aprendizaje o ejemplos.

Fase de aprendizaje

Se puede definir el aprendizaje como el proceso por el que se produce un ajuste de los parámetros libres de la red a partir de un proceso externo a la red [22, 23]. El tipo de aprendizaje vendrá determinado por la forma en la que dichos parámetros son ajustados. En la mayor parte de las ocasiones, estos parámetros suelen ser un conjunto de pesos sinápticos que una vez ajustados permitirían a la red realizar el procesamiento deseado.

Al comienzo del diseño de un sistema neuronal artificial, se parte de un cierto modelo de neurona y de una determinada arquitectura de red, estableciendo los pesos sinápticos inicialmente con un valor nulo nulo o aleatorio. Para que la red resulte operativa es necesario ajustar los parámetros de la red, en otras palabras entrenar la red, lo que constituye el modo de aprendizaje. El entrenamiento o aprendizaje se puede llevar a cabo en dos niveles. El más convencional [22] es el de modelado de las sinapsis, que consiste en modificar los pesos sinápticos siguiendo una cierta regla de aprendizaje, construida a partir de la optimización de una función de error o coste, que mide la eficacia actual de la operación de la red. Denominando $w_{ij}(t)$ al peso que conecta la neurona presináptica j con la postsináptica i en la iteración i, el algoritmo de aprendizaje, en función de las señales que en el instante t llegan procedentes del entorno, proporcionará el valor $\delta w_{ij}(t)$ que resulta ser el valor de la actualización de dicho peso, que quedará de la siguiente forma:

$$w_{ij}(t+1) = w_{ij}(t) + \delta w_{ij}(t) \tag{1.4}$$

El proceso de aprendizaje es usualmente iterativo, actualizándose los pesos (de la manera descrita), continuamente, hasta que la red alcance el rendimiento deseado o hasta un número máximo de iteraciones [22, 23].

Con respecto al otro nivel [22], algunos modelos neuronales incluyen la modificación de la arquitectura durante el proceso de entrenamiento por medio de la creación o destrucción de neuronas.

Como se ha visto anteriormente, los dos tipos básicos de aprendizaje son el **supervisado** [18, 19, 20, 23] y el **no supervisado** [23, 21, 17, 16], cuya distinción viene, en origen, del campo de reconocimiento de patrones [15, 17].

Aprendizaje supervisado En este tipo de aprendizaje se presenta a la red un conjunto de patrones, junto con la salida deseada (objetivo), e iterativamente se ajustan sus pesos hasta que la salida tiende a ser la salida esperada, utilizando para ello información detallada del error que comete en cada paso. De este modo, la red es capaz de estimar relaciones entrada/salida sin necesidad de proponer una cierta forma funcional de salida. Ejemplos serían el perceptrón [18, 15], el perceptrón multicapa [24, 15], etc.

Aprendizaje no supervisado En este aprendizaje, también llamado autoorganizado, se presenta a la red un gran número de patrones sin adjuntar la salida esperada. La red, por medio de la regla de aprendizaje, estima la densidad de probabilidad, a partir de la que se podrán identificar regularidades en el conjunto de entrada, extraer rasgos propios, agrupar patrones según similitud (*clustering*). La red, a partir de un proceso de autoorganización, proporcionará cierto resultado, el cual dependerá de las relaciones de similitud existentes entre los patrones de entrada. La principal aplicación de estos modelos es la realización de agrupamiento de patrones, análisis exploratorio, y visualización de minería de datos (*Data Mining*). El más popular de los modelos de este tipo de aprendizaje es el de los mapas autoorganizados de Kohonen [31, 32, 33].

Otros tipos de aprendizaje Al comienzo de esta sección ya se comentó el aprendizaje reforzado, pero también hay uno híbrido donde coexisten el aprendizaje supervisado y el autoorganizado [22, 14].

Capacidad de generalización de una red neuronal

Uno de los aspectos fundamentales de los sistemas neuronales (y dentro de estos últimos, las redes neuronales) es la capacidad de generalizar a partir de ejemplos [22, 23]. Por generalización se entiende la capacidad que tiene una red de proveer de una respuesta correcta ante patrones de un conjunto de ejemplos que no hayan sido utilizados en el entrenamiento. Existe un problema a este respecto: el problema de la memorización frente a generalización [22]. Una red neuronal correctamente entrenada será capaz de generalizar, lo que significa que ha aprendido adecuadamente el *mapping*, no solo los ejemplos concretos presentados, sino que sea capaz de responder correctamente ante patrones no vistos con anterioridad [22, 14]. En caso de que únicamente diese respuesta adecuada a patrones vistos anteriormente, y no a los nuevos que se le presenten, la red habría memorizado las peculiaridades de los patrones vistos y solo respondería correctamente a estos y no tendría la capacidad de generalización.

Muchos de los algoritmos de aprendizaje se basan en métodos numéricos iterativos que tratan de minimizar una función de coste [22, 23] o de error, lo que puede dar lugar en ocasiones a problemas en la convergencia del algoritmo. La convergencia del algoritmo es una manera de comprobar si una arquitectura, y su regla de aprendizaje, es capaz de resolver un problema, pues el error cometido que se mide durante el proceso describe la precisión del ajuste realizado [22].

En el proceso de entrenamiento se distingue entre el nivel de error alcanzado al final de la fase de aprendizaje y el error que la red entrenada comete ante patrones no utilizados con anterioridad, lo cual permite una generalización de la red. Se busca que en vez de un error pequeño en el entrenamiento haya una buena generalización [22].

Validación cruzada En un proceso de entrenamiento de la red se debe considerar, por una parte, un error de aprendizaje y por otra un error de generalización. El primero se suele calcular como el error cuadrático medio de los resultados proporcionados por la red para el conjunto de patrones de aprendizaje [22]. Se puede reducir el valor de este error empleando más iteraciones si se cuenta con una red suficientemente grande. El error de generalización se puede medir empleando un conjunto representativo de patrones diferentes a lo utilizados en el entrenamiento [22], este conjunto se denomina de prueba. Entonces, se puede entrenar una red neuronal empleando un conjunto de aprendizaje, y comprobar la eficiencia real del entrenamiento, o error de generalización, mediante un conjunto de prueba (*test*). El valor de este error de generalización (ecuación 1.5) es la diferencia entre el error en *aprendizaje* cometido en el entrenamiento y el error *test*, cometido sobre un ejemplo escogido aleatoriamente:

$$Err_{generalizacin} = Err_{test} - Err_{aprendizaje} \tag{1.5}$$

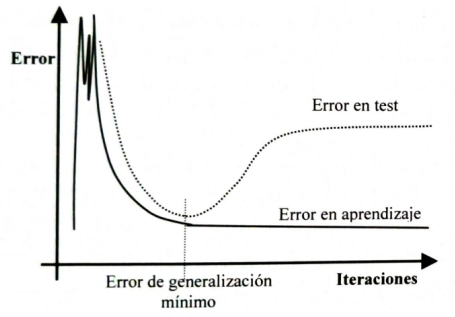


Figura 1.4: Evolución del error de aprendizaje y del error de generalización en un caso ideal. Fuente:[22].

Un hecho experimental es que si se entrena una red hasta alcanzar un error de aprendizaje muy pequeño, la eficacia real del sistema para generalizar podría ser baja [22]. Al principio la red se adapta progresivamente al conjunto de aprendizaje,

acomodándose al problema y mejorando la generalización. Sin embargo, en un momento dado, el sistema se ajusta demasiado a las particularidades de los patrones empleados en el entrenamiento, aprendiendo incluso el ruido presente en ellos, por lo que crece el error que comete ante diferentes patrones a los empleados en el entrenamiento (el error de generalización). En este momento la red no ajusta correctamente el *mapping*, sino que está memorizando los patrones del conjunto de entrenamiento, lo que se denomina sobreaprendizaje o sobreajuste [22]. Para evitar esta situación, lo ideal sería entrenar la red hasta un punto óptimo en el que el error de generalización sea mínimo (figura 1.4). Al procedimiento que consiste en entrenar y, además, validar a la vez para detenerse en el punto óptimo del error de generalización se denomina **validación cruzada** [22].

La clave está en que las redes neuronales son estimadores no lineales muy eficientes y potentes, como se detalla más adelante, capaces de modelar situaciones complejas. En herramientas lineales, la complejidad del modelo viene dada por el número de parámetros libres que se debe ajustar, mientras que en redes neuronales la complejidad del modelo depende tanto del número de parámetros como de su valor actual [34]. Una red neuronal tiene tal capacidad de modelado que puede aprender casi cualquier cosa, llegando a incurrir fácilmente en el sobreaprendizaje [22].

Modelos de redes neuronales

Modelo McCulloch-Pitts En 1943 Warren McCulloch y Walter Pitts publicaron un artículo [35] que sería la primera piedra para el posterior desarrollo de las redes neuronales. Estudiaron desde una perspectiva computacional el cerebro e intuyeron una alternativa a la resolución de problemas. Fueron los primeros en describir un modelo computacional para una red neuronal que realizaba una serie de suposiciones sobre el funcionamiento de las neuronas en una red neuronal:

- Neuronas binarias. La salida de las neuronas artificiales es binaria.
- Umbral de activación: Un número determinado de sinapsis, mayor que uno, debe activarse para la que neurona llegue a activarse.
- Propagación de señales: La única demora que experimentan las señales de activación en su propagación se produce en las sinapsis.
- La actividad de una sinapsis inhibidora evita la excitación de la neurona.
- La estructura de la red de interconexión no cambia con el tiempo. (Suposición incorrecta).

Hebb [27] sugirió que el fortalecimiento de las conexiones entre las neuronas del cerebro es lo que da lugar al aprendizaje, lo que sitúa a este a un nivel sináptico. Aunque propuesto por Hebb, fue desarrollado independientemente por Hayek [36].

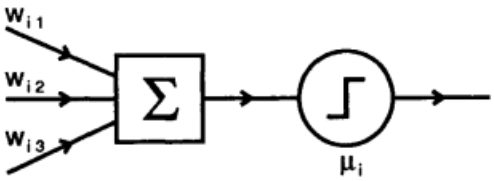


Figura 1.5: Modelo de neurona artificial tipo McCulloch-Pitts. Fuente:[23].

Perceptrón Rosemblatt, inspirado por el modelo de aprendizaje hebbiano, introduce a finales de los cincuenta del pasado siglo el modelo de Perceptrón simple [18, 15, 23]. La estructura del perceptrón se inspira en las primeras etapas de procesamiento de los sistemas sensoriales de los animales, en los cuales la información va atravesando sucesivas capas de neuronas, que realizan un procesamiento progresivamente de más alto nivel.

El perceptrón simple es un modelo unidireccional, compuesto por dos capas de neuronas: una de entrada y otra de salida. Las neuronas de entrada no realizan ningún tipo de cómputo, únicamente envían la información a las neuronas de salida. La función de activación de las neuronas de salida es de tipo escalón. La operación de un perceptrón simple puede escribirse como:

$$y_i = H\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right) \tag{1.6}$$

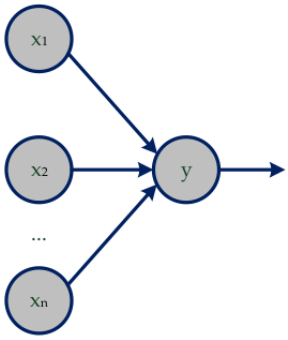


Figura 1.6: Arquitectura de la red neuronal más sencilla, el perceptrón. Fuente:[14].

El perceptrón puede utilizarse como clasificador y también como representación de funciones booleanas, ya que sus neuronas son del tipo McCulloch-Pitts. La importancia histórica del perceptrón radica en su carácter de dispositivo entrenable, pues el algoritmo de aprendizaje del modelo permite determinar automáticamente los pesos sinápticos que clasifican un conjunto de patrones a partir de un conjunto de ejemplos etiquetados. Este algoritmo entra dentro de los algoritmos de corrección de errores, que ajustan los pesos en proporción a la diferencia existente entre la salida de la red y la salida deseada, con el objetivo de minimizar el error actual de la red.

Rosemblatt distinguía entre perceptrones simples con dos capas de neuronas (una de entrada y otra de salida) y perceptrones multicapa con tres o más capas. Generalizó el algoritmo de entrenamiento de Farley y Clark [37], que solo servía para redes de dos capas, de forma que pudiese aplicarse a redes multicapa.

El modelo tiene muchas limitaciones, como indicaron Minsky y Papert en 1969 [20], lo que resultó decisivo para que durante unos años se desterrase la idea del uso de redes neuronales.

Perceptrón multicapa (MLP) Al añadir capas intermedias (ocultas) a un perceptrón simple, se obtiene el perceptrón multicapa ***MLP*** (*Multi-Layer Perceptron*) [23, 15]. Para esta arquitectura se suele emplear el algoritmo ***Backpropagation*** (retropropagación de errores) para el entrenamiento. Este algoritmo fue introducido por Werbos [38] pero difundido por el grupo **PDP** [25, 39] como una técnica útil para la resolución de problemas complejos.

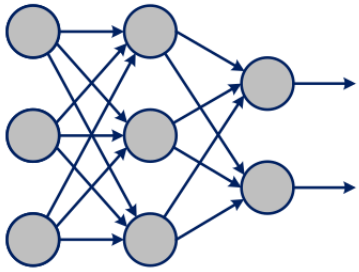


Figura 1.7: Arquitectura de una red multicapa con una única capa oculta. Fuente:[14].

Si se considera una red multicapa con la siguiente configuración: una capa de entrada, una capa oculta y una capa de salida, se denomina x_i a las entradas de la red, y_j a las salidas de la capa oculta, y z_k a las de la capa de salida; t_k serán las salidas objetivo, o *target*. Por otro lado w_{ij} son los pesos sinápticos de las neuronas de la capa oculta, θ_j sus umbrales, w'_{kj} los pesos de la capa de salida y θ'_k sus umbrales. La operación de un perceptrón multicapa con una capa oculta y neuronas de salida lineal se expresa matemáticamente de la siguiente manera:

$$z_k = \sum_j w'_{kj} y_j - \theta'_k = \sum_j w'_{kj} f(\sum_i w'_{ji} x_i - \theta_j) - \theta'_k \tag{1.7}$$

La función de activación f suele ser de tipo sigmoideo (una sigmoide como la función logística o la tangente hiperbólica, ver apéndice A). Esta es la arquitectura más común, sobre todo en soluciones de clasificación, aunque existen numerosas variantes, como incluir neuronas no lineales en la capa de salida, introducir más capas ocultas, emplear otras funciones de activación, limitar el número de conexiones entre neurona y capa siguiente, introducir dependencias temporales o arquitecturas recurrentes.

Algoritmo de aprendizaje Backpropagation (BP) Una solución al problema de entrenar los nodos de las capas ocultas de las arquitecturas multicapa la proporciona el algoritmo de retropropagación de errores o ***backpropagation*** [25, 39, 40, 23].

Si se considera un preceptrón multicapa de tres capas, con las entradas, salidas, pesos y umbrales de las neuronas previamente definidos, para un patrón de entrada x^μ, con $\mu = 1, ..., p$, la operación global de esta arquitectura se expresa de la siguiente:

$$z_k^\mu = g\left(\sum_j w'_{kj} y_j^\mu - \theta'_k\right) = g\left(\sum_j w'_{kj} f\left(\sum_i w_{ji} x_i^\mu - \theta_j\right) - \theta'_k\right) \tag{1.8}$$

Donde g es la función de activación de las neuronas de salida y f es la de las neuronas de la capa oculta. Ambas pueden sigmoideas, aunque a menudo la función de las neuronas de salida se considera la identidad.

Ahora partimos del error cuadrático medio para la diferencia entre le valor esperado t^μ y la salida:

$$E\left(w_{ji}, \theta_j, w'_{kj}, \theta'_k\right) = \frac{1}{2}\sum_\mu \sum_k \left[t_k^\mu - g\left(\sum_j w'_{kj} y_j^\mu - \theta'_k\right)\right]^2 \tag{1.9}$$

La minimización se lleva a cabo mediante descenso por el gradiente, donde la variación del error con respecto a la dirección de los pesos de la capa de salida y los pesos de la capa oculta son respectivamente:

$$\delta w'_{kj} = -\epsilon \frac{\partial E}{\partial w'_{kj}} \tag{1.10a}$$

$$\delta w_{ji} = -\epsilon \frac{\partial E}{\partial w_{ji}} \tag{1.10b}$$

Las expresiones de actualización de los pesos se obtienen derivando la anterior expresión y teniendo en cuenta las dependencias funcionales y aplicando la regla de la cadena:

$$\delta w'_{kj} = \epsilon \sum_\mu \Delta'^\mu_k y_j^\mu \tag{1.11a}$$

$$\delta w_{ji} = \epsilon \sum_\mu \Delta_j^\mu x_j^\mu \tag{1.11b}$$

Teniendo en cuenta que:

$$\Delta'^\mu_k = \left[t_k^\mu - g\left(\sum_j w'_{kj} y_j^\mu - \theta'_k\right)\right] \frac{\partial}{\partial w'_{kj}} \left[g\left(\sum_j w'_{kj} y_j^\mu - \theta'_k\right)\right] \tag{1.12a}$$

$$\Delta_j^\mu = \left(\sum_k \Delta'^\mu_k w'_{kj}\right) \frac{\partial}{\partial w_{ji}} \left[f\left(\sum_i w_{ji} x_i^\mu - \theta_j\right)\right] \tag{1.12b}$$

La actualización de los umbrales (*bias* o θ) se realiza haciendo uso de las expresiones anteriores, considerando que el umbral es un caso particular de peso sináptico, cuya entrada es una constante (-1).

En estas expresiones está implícito el concepto de propagación hacia atrás de errores, que da nombre al algoritmo. En primer lugar se calcula la expresión Δ'^{μ}_{k}, que se denomina señal de error, por ser proporcional al error de la salida actual de la red, con el que se calcula la actualización δ'_{kj} de los pesos de la capa de salida. A continuación se propagan hacia atrás los errores Δ'^{μ}_{k} a través de las sinapsis, proporcionando así las señales de error Δ^{μ}_{j}, correspondientes a las sinapsis de la capa oculta; con estas últimas se calcula la actualización de los pesos δ_{ji} de las sinapsis de las neuronas de la capa oculta. El algoritmo puede extenderse fácilmente a arquitecturas con más de una capa oculta siguiendo el mismo esquema.

El procedimiento a seguir, de manera esquemática, para entrenar un perceptrón multicapa mediante el algoritmo ***backpropagation*** sería [22]:

1. Establecer aleatoriamente los pesos y umbrales iniciales.

2. Para cada patrón μ del conjunto de aprendizaje se realizan las siguientes tareas:

   a) Llevar a cabo una fase de ejecución para obtener la respuesta de la red ante el patrón en posición μ

   b) Calcular las señales de error asociadas Δ'^{μ}_{k} y Δ^{μ}_{j} según las ecuaciones (1.12).

   c) Calcular el incremento parcial de los pesos y umbrales debidos a cada patrón μ

3. Calcular el incremento total (para todos lo patrones) actual de los pesos $\delta w'_{kj}$ y δw_{ji} según las ecuaciones (1.11). Proceder de igual forma para los umbrales.

4. Actualizar pesos y umbrales.

5. Calcular el error actual y volver al paso segundo si el resultado no es satisfactorio.

Se debe comenzar con pesos iniciales aleatorios (normalmente números pequeños), ya que si se parte de pesos y umbrales nulos, el aprendizaje no progresará, puesto que las salidas de las neuronas y el incremento en los pesos será nulo.

En este esquema, se lleva a cabo una fase de ejecución para todos y cada uno de los patrones del conjunto de entrenamiento, se calcula la variación en los pesos debida a cada patrón, se acumula dicha variación, y solamente entonces se procede a la actualización de los pesos. Esta manera de proceder se suele denominar aprendizaje por lotes (*batch*). Otra posibilidad consiste en actualizar los pesos tras la presentación de cada patrón μ, esquema que para este caso se denomina aprendizaje en serie u (*online*). Este último estima mejor el gradiente, permite emplear ritmos de entrenamiento mayores y suele ser más rápido; se suele emplear en problemas en los que se dispone de extensos conjuntos de patrones de aprendizaje. El orden

en la presentación de los patrones en el modo *online* debe ser aleatorio, puesto que si siempre se sigue el mismo orden, el entrenamiento estaría predispuesto hacia el último patrón del conjunto de entrenamiento, cuya actualización, por ser la última, siempre predominaría sobre las anteriores. Además, esta aleatoriedad permite escapar de mínimos locales, lo que es una ventaja a la hora de alcanzar mínimos más profundos para el error.

El algoritmo ***backpropagation*** constituye un método de gran generalidad, lo que puede presentar ventajas e inconvenientes [22]. Entre las ventajas se puede destacar:

- Se puede aplicar a multitud de problemas diferente.
- No necesita mucho tiempo de desarrollo, aunque si se quiere una solución excelente, habrá que invertir un poco más de tiempo.
- No requiere muchos recursos computacionales.

Entre los inconvenientes destaca:

- Lentitud de convergencia
- Se puede incurrir en sobreaprendizaje o sobreajuste, fenómeno relacionado con la capacidad de generalización de la red a partir de ejemplos presentados.
- No garantiza alcanzar el mínimo global de la función de erro, tan solo un mínimo local.

Aproximador universal de funciones

A pesar de las limitaciones del perceptrón [20], el adaptarlo hasta llegar a la arquitectura multicapa [23, 15] ha supuesto una ampliación del radio de acción de este tipo de sistemas, lo que lleva a aplicarlo a numerosos problemas donde se comprobó experimentalmente que era capaz de representar complejos *mappings* y de abordar problemas de clasificación de gran envergadura, de manera eficaz y relativamente sencilla [22, 23].

Ha habido un proceso en el que se ha ido demostrando la aplicabilidad de los sistemas neuronales para la representación de ciertas funciones, como en el caso de las booleanas [41]. Se han ido, con el paso del tiempo, demostrando que se pueden representar cualquier función arbitraria [42]. A finales de los años 80, diversos grupos propusieron teoremas que demostraban matemáticamente que un perceptron multicapa (***MLP***) convencional con un a capa oculta constituía un aproximador universal de funciones [43, 44]. Estos teoremas proporcionan una base teórica al campo de las redes neuronales, al incidir un aspecto como la aproximación funcional y un modelo, el ***MLP***, centrales en la teoría de las redes neuronales. La desventaja de estos teoremas es que no dan indicaciones sobre el número de neuronas de la capa oculta, que es un gran problema a la hora de diseñar una red neuronal.

1.2. Matemáticas en la educación secundaria

La competencia en educación en España está en las Comunidades Autónomas, en concreto, en la Comunidad Autónoma de Galicia viene recogida esta competencia en el artículo 31 del Título Segundo de su estatuto de Autonomía. El aprendizaje de las matemáticas en el sistema educativo viene reglado por un marco general a nivel estatal donde se detallan los contenidos mínimos y por las distintas legislaciones a nivel autonómico donde se especifican las competencias clave y los aprendizajes y las diferentes reglas para la comunidad (materias troncales, optativas, número de suspensos en la promoción, etc). Actualmente, el marco general vigente está en la *LOMCE*, Ley Orgánica 8/2013, del 9 de diciembre, para la mejora de la calidad educativa, y a nivel de la comunidad autónoma de Galicia por el decreto 86/2015, del 25 de junio, por el que se establece el currículo de la educación secundaria obligatoria y de bachillerato en la comunidad.

En estas leyes, las matemáticas son consideradas una asignatura troncal obligatoria e incluida en todos los cursos dividiéndose en dos modalidades en los últimos cursos, 3º y 4º:

- Matemáticas orientadas a las enseñanzas académicas (o, también llamadas, matemáticas académicas).
- Matemáticas orientadas a las enseñanzas aplicadas (o, también llamadas, matemáticas aplicadas).

La diferencia entre ambas asignaturas es principalmente de contenidos, siendo las "aplicadas" una materia menos densa, ya que está orientada a alumnos/as que enfocan su siguiente paso a nivel educativo en las enseñanzas profesionales (FP básica, grados medios de Formación Profesional) y, en algunos casos, alumnos/as con dificultades académicas. Para estos últimos, en los primeros cursos se recogen una serie de medidas para favorecer el que puedan superar la asignatura. Estas medidas también se pueden adoptar en los cursos superiores, y son las denominadas "Medidas de atención a la diversidad".

1.2.1. Medidas de atención a la diversidad

Son una serie de actuaciones que adoptan los equipos docentes y el departamento de orientación en los centros, normalmente a principio de curso, aunque a lo largo de este se pueden modificar o incluir según se detecten nuevas necesidades, o se supriman si ya no son necesarias o se han conseguido los objetivos. Son individualizadas y concretas para cada asignatura, si bien, pueden ser adoptadas por varios alumnos/as a la vez. Hay tres tipos fundamentales de actuaciones, que en concreto para la materia de Matemáticas son:

1. Refuerzo significativo en el aula. El departamento de matemáticas y el de orientación del colegio dan dos actuaciones en este caso, principalmente adaptar los exámenes a un grado de complejidad menor y, si se requiere, disponer

de más tiempo en exámenes. Este caso se reserva para una problemática leve de comprensión de las matemáticas, que tenga la signatura pendiente del curso anterior, o que tenga un bajón de rendimiento con el que pueda no superar la asignatura o el curso. Los contenidos son los mismos que el resto de alumnos sin refuerzo.

2. Refuerzo fuera del aula. Se parte de las medidas del refuerzo dentro del aula y en una hora de la asignatura de matemáticas o de otra asignatura de la cual los alumnos estén exentos, se da esa hora con un profesor/a de refuerzo o con un miembro del departamento de orientación del centro. Este caso de refuerzo se reserva para alumnos con problemática más severa que el anterior pero que puedan llevar con cierta normalidad los contenidos del curso.

3. Adaptación curricular. Este es un caso excepcional, es una medida de carácter individual y extraordinaria, en el que se adaptan los contenidos del curso al nivel de conocimiento que tenga el alumno/a. Esta adaptación no puede recortar el nivel en más de dos cursos y ha de pasar por la inspección educativa, que la ha de revisar y aprobar. Además se propone para que salga fuera del aula como en el caso anterior. Los exámenes estarán adaptados a esos contenidos. Al tener que pasar por la inspección, como se ha mencionado, esta medida tiene carácter permanente durante el curso y debe tomarse al principio de este, y salvo casos extraordinarios se toma una vez empezado.

1.2.2. Valoración del rendimiento

A la hora de valorar la nota final del alumnado, se han de tener en cuenta, fundamentalmente, una serie de parámetros, que influirán de manera directa algunos y otros indirectamente; estos parámetros serán:

- Notas de exámenes. Cada evaluación se calcula una media de estas lo que da una nota con la que se evalúa esa evaluación. La media de la nota final de las tres evaluaciones de las que consta el curso es la nota final. Esta puede modificarse ligeramente (redondeo) si se atienden a alguno de los otros parámetro.

- Atención a la diversidad. Como se mencionó antes, la dificultad en los exámenes se reduce y en algunos casos se puede impulsar la nota por medio de tareas al margen de los exámenes que pueden ayudar a mejorar la calificación final del alumnado.

- Comportamiento, esfuerzo. A diferencia de los anteriores parámetros, este es subjetivo, y mide el grado de compromiso del/la alumno/a con la asignatura, si se esfuerza, si el comportamiento en clase es el adecuado. Normalmente se intentan ponderar con un 10 % de la nota final, además de hacer que, en caso de faltar unas centésimas, se redondee hacia el valor siguiente (que puede ser la diferencia entre el aprobado y el suspenso.)

De todos estos datos, el estudio solo recoge notas finales de dos evaluaciones, medidas de atención a la diversidad y nota final numérica.

Capítulo 2

Objetivo

Como se comentó al principio de la introducción, muchos de los trabajos que se mencionan en la bibliografía se enmarcan en estudios preuniversitarios y universitarios, hay pocos registros sobre trabajos que se centren en la etapa secundaria (12 a 16 años) y en concreto en la asignatura de matemáticas (obligatoria y de importancia capital en cualquier sistema educativo), además de que muchos de los trabajos previos se basan en datos extraídos de las plataformas digitales de algunas universidades. La novedad de este trabajo se centra en los datos, recogidos en un entorno educativo real, un colegio en Galicia, durante todo el curso 2018/19, que corresponden a medias de exámenes parciales (presenciales) y, a diferencia de los otros trabajos, medidas de actuación ante dificultades de aprendizaje consensuadas en reuniones docentes por profesores y orientadores.

El principal objetivo del trabajo es es buscar una predicción el rendimiento académico a partir de los datos recabados y comprobar si las medidas de atención a la diversidad pueden refinar la precisión de los resultados que se obtengan. También se buscará una predicción de estas medidas de apoyo a alumnado con dificultades. Muchas veces es complicado atender las necesidades educativas con los *ratios* de alumnado por clase, y el tener una herramienta objetiva puede ayudar a detectar necesidades que en el día a día no se podrían detectar.

Para todas estas consideraciones se intentará responder a las siguientes preguntas:

1. ¿Se puede clasificar el aprobado/suspenso de los alumnos? (Predecir aprobados/suspensos).

2. ¿Se puede clasificar la necesidad de refuerzo de los alumnos? (Predecir el tipo de refuerzo).

3. ¿Se puede predecir la nota final de cada alumno/a?

Además hay otras cuestiones que pueden ser de interés y que complementarían a las anteriores:

- ¿Qué cantidad de entradas mínima de datos se necesitan? ¿Cuanta más información, más precisión?

- ¿Se podría contestar conjuntamente la primera y la segunda de las preguntas anteriores? (Salida múltiple).

Para contestar estas preguntas se procederá con el diseño de una serie de redes neuronales (del tipo multicapa, *feedforward* y algoritmo ***backpropagation***) para modelar una solución para los datos recabados. Un objetivo sería encontrar que ese diseño sea el adecuado y que responda a las cuestiones previas. Para completar dicho diseño, además de buscar número óptimo de unidades de la capa de entrada y salida, relacionadas con las preguntas arriba mencionadas, se buscará encontrar un número adecuado para la cantidad de unidades de la capa oculta. Además, se buscará comprobar si añadir una segunda capa oculta mejoría las predicciones.

Capítulo 3

Metodología

En esta sección se detallará cómo se modelará el procedimiento para conseguir los objetivos propuestos y resolver el problema asociado: se describirá el diseño de la red/redes neuronales que se emplearán, las herramientas computacionales empleadas en el diseño y ejecución de la red, los tipos de datos empleados, etc.

3.1. Datos

Los datos recogidos pertenecen al curso 2018/2019, a la asignatura de Matemáticas de 1º y 2º de ESO, de Matemáticas orientada a las enseñanzas académicas de 3º y 4º de ESO y Matemáticas orientadas a las enseñanzas aplicadas de 3º y 4º de ESO. Estos constituyen la muestra con la que se realizarán los cálculos del trabajo.

Se distribuyen los grupos de registros/alumnos de la siguiente manera:

- 1º de ESO: 2 grupos de 24 y 23 registros
- 2º de ESO: 2 grupos de 29 y 30 registros
- 3º de ESO: 3 grupos de 24, 12 y 11 registros
- 4º de ESO: 3 grupos de 12, 17 y 14 registros

En total son 10 grupos y 191 registros que corresponden a 3 docentes.

Los datos recogidos en el transcurso del trabajo y que fueron empleados para obtener los resultados son:

- Datos para las entradas:
    1. Identificador del grupo, que encuadra al alumno en un grupo y curso específico.
    2. Nota numérica de la 1º evaluación.
    3. Nota numérica de la 2º evaluación.
    4. Refuerzo de la 1º evaluación.

5. Refuerzo de la 2º evaluación.

- Datos para las salidas

1. Nota final numérica del curso.
2. Aprobado/suspenso.
3. Refuerzo en la 3º evaluación o refuerzo final.

Aunque se han recogidos otros datos, no son determinantes para el propósito del trabajo.

La explicación, más detallada, de los datos mencionados y el tipo de valor que se le asigna a cada uno de ellos se muestra a continuación:

- Identificador de grupo, al que se le asigna un valor numérico entero (número natural) entre 1 y 10.
- Notas numéricas de evaluaciones, son valores numéricos decimales positivos entre 0 y 10.
- Refuerzos, son valores numéricos que siguen las siguientes indicaciones:

  0 si el registro no tiene ningún tipo de refuerzo.

  1 si el registro tiene refuerzo significativo dentro del aula.

  2 si el registro tiene refuerzo significativo fuera del aula.

- Aprobado/suspenso, valor numérico que corresponde a:

  0 si el registro está suspenso.

  1 si el registro está aprobado.

No se han tenido en cuenta los datos correspondientes a adaptaciones curriculares, ya que son medidas fijas y prácticamente inmutables, que no suponían más de dos o tres registros; ni datos de alumnos de incorporación tardía o que hayan dejado el colegio durante el curso correspondiente.

3.2. Diseño de la Red Neuronal

Para modelar la red neuronal se procederá a emplear una del tipo multicapa *feedforward* entrenada con el algoritmo ***backpropagation***.

3.2.1. Topología

Debido al número bajo de registros de la muestra, solo se considerará una red neuronal de 3 capas, esto es, con una única capa oculta, ya que seguramente sea suficiente para obtener un resultado satisfactorio. En la figura (1.3) se muestra un modelo de capas sencillo a modo de ejemplo.

Los distintos diseños se realizarán en base a las 3 capas de la propia red como parámetros. Según se varíen estos parámetros tendremos una configuración distinta de red neurona. Para responder a las distintas preguntas formuladas en los objetivos se buscarán distintas configuraciones para la capa de entrada y salida, además de buscar una configuración óptima para la capa oculta en combinación con las otras dos. A continuación se detalla cómo variarán estos parámetros de diseño para obtener las distintas configuraciones empleadas en el trabajo.

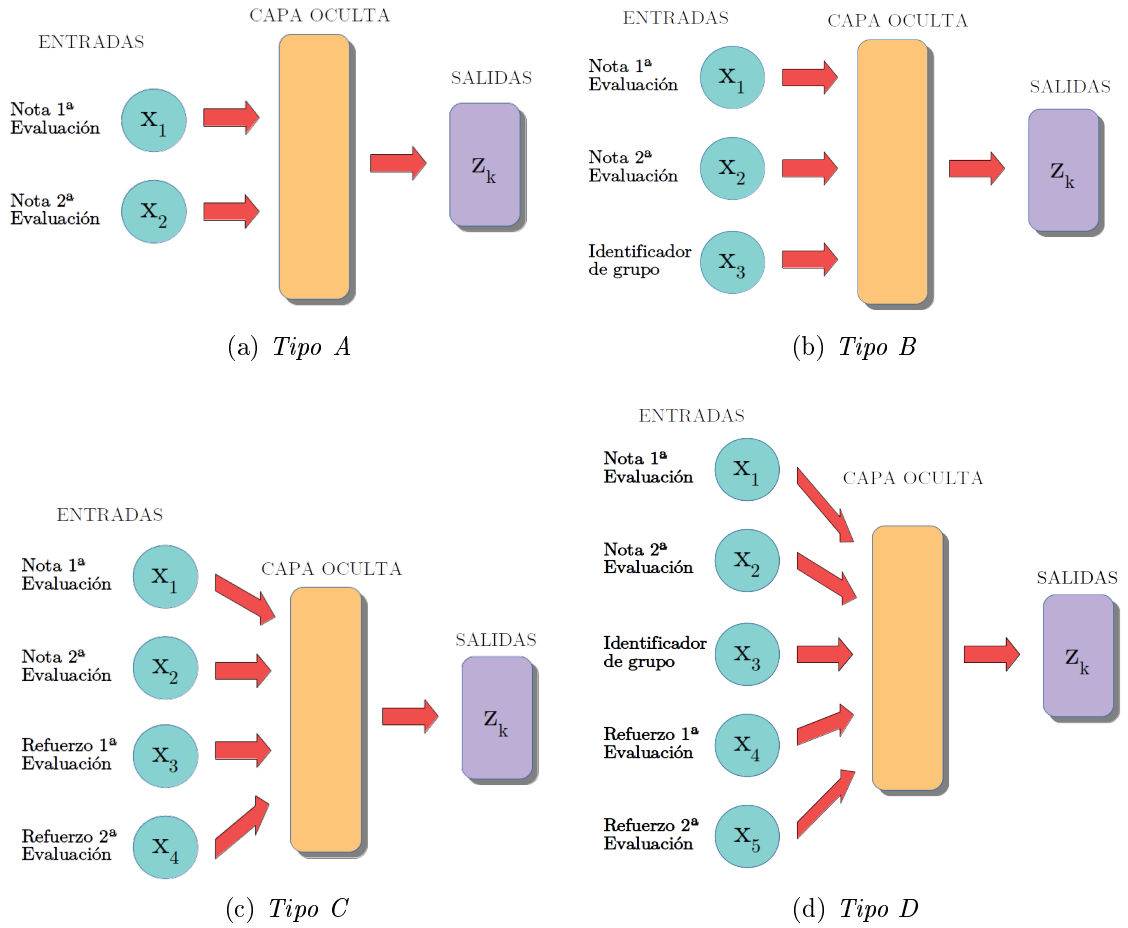
Primer parámetro: unidades de entrada

La capa de entrada corresponde con el nivel de información que dependiendo del número de unidades de entrada tendrá un nivel mayor o menor, con lo que se busca responder si este hecho ayuda a mejorar la predicción. Para ello se tendrán en cuenta redes con los siguientes grupos (tipos de capa) de entradas donde se varía el número y tipo de estas en base a tres tipos de datos (notas, refuerzos e identificador de grupo). En la figura 3.1 se observa un modelo descriptivo de los distintos tipos de entradas que se muestran a continuación:

- Nota de 1ª evaluación y nota de 2ª evaluación, que se denominará como capa *Tipo A*.
- Nota de 1ª evaluación, nota de 2ª evaluación e identificador de grupo, que se denominará como capa *Tipo B*.
- Nota de 1ª evaluación, nota de 2ª evaluación, refuerzo de 1ª evaluación y refuerzo de 2ª evaluación, que se denominará como capa *Tipo C*.
- Nota de 1ª evaluación, nota de 2ª evaluación, refuerzo de 1ª evaluación, refuerzo de 2ª evaluación e identificador de grupo, que se denominará como capa *Tipo D*.

Segundo parámetro: unidades de capa oculta

También hay que tener en cuenta el número de neuronas de la capa oculta. En este caso se variará a partir de un número mínimo, que será 1, y aumentando en pasos de una unidad se llegará hasta un valor máximo. Como no hay una receta universal, sino recomendaciones, se ha tenido en cuenta que el número máximo de neuronas no debería superar el doble del número de unidades de entradas: en este caso, el número de entradas máximo empleado (3.1) es 5, con lo que se tomará como número máximo de neuronas de la capa oculta el valor de 9.



(a) *Tipo A* (b) *Tipo B*

(c) *Tipo C* (d) *Tipo D*

Figura 3.1: Modelos de redes neuronales de tres capas según los tipos de entradas.

También se incluirá una comparativa de los resultados añadiendo una segunda capa oculta a la red. Para este caso el modelo de red será como se muestra en la figura (3.2), introduciendo otro parámetro análogo, las unidades de esta segunda capa oculta, las cuales se variarán de la misma manera que para la primera capa oculta en combinación con esta.

Al ser una muestra pequeña, y que en un ordenador de recursos limitados no tarda más de unos pocos segundos en realizar todos los cálculos de una simulación de una red neuronal, se optó por realizar, además de las variaciones de los tipos de entradas y salidas, considerar también el variar el número de neuronas de la capa oculta entre el número mínimo, 1, y el máximo antes mencionado. El total de simulaciones/redes neuronales fue de 143, lo que no supuso más de 2 minutos de tiempo absoluto de cálculo (no de procesador). El incluir la segunda capa oculta incrementó el tiempo de cálculo considerablemente, pero siguió siendo asumible considerar todas las posibles configuraciones.

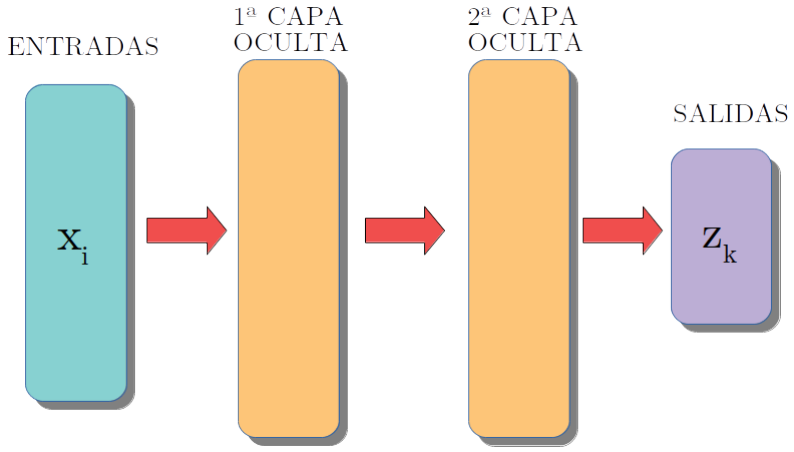


Figura 3.2: Modelo de red neuronal con cuatro capas: entradas, una primera capa oculta, una segunda capa oculta y salidas.

Tercer parámetro: unidades de capa de salida

Se plantearán dos tipos de capa de salida, las cuales, a su vez, determinarán los tipos con los que se clasificarán los resultados en función de la topología de la red, clasificación que coincidiría con las cuestiones de los objetivos:

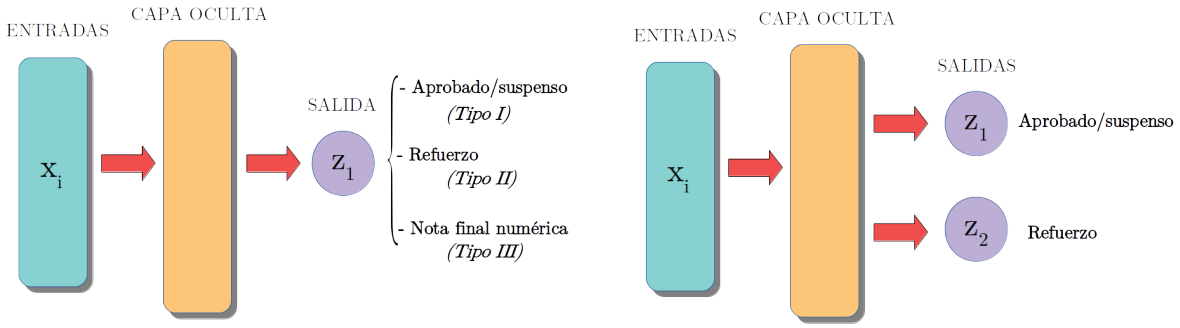
- Salida única:
  1. Aprobado/suspenso, que se denominará como red *Tipo I*.
  2. Refuerzo (final) de la 3ª evaluación, que se denominará como red *Tipo II*.
  3. Nota final numérica, que se denominará como red *Tipo III*.
- Salida múltiple:
  1. Aprobado/suspenso final y refuerzo (final) de la 3ª evaluación (combinación de dos de los tipos anteriores), que se denominará como red *Tipo IV*.

Los resultados se clasificarán según los tipos de capa de salida, los tres primeros tipos coinciden con las cuestiones expuestas en los objetivos. El último tipo se encuadra en las preguntas complementarias, junto con los tipos de capa de entrada.

3.3. Herramientas de cálculo/simulación

Se empleó el lenguaje de programación **Python**[45], que incluye un ecosistema de librería muy amplio, muchas de las cuales están orientadas al cálculo y al aprendizaje automático. Dentro de las librerías incluidas en el trabajo están:

- **Numpy** [46, 47] para cálculos numéricos intermedios en los que se suelen ver envueltas matrices (*arrays*).



(a) *Tipos I, II, III* (b) *Tipo IV*

Figura 3.3: Modelos de redes neuronales de tres capas según los tipos de salidas.

- **Pandas** [48, 47] para cargar y manejar datos desde archivos *csv* (*comma-separated values*), simplifica el trabajo con matrices enormemente.
- **Matplotlib** [49] para realización de gráficos a partir de los datos iniciales y resultados obtenidos.
- **Scikit-Learn** [50, 51, 47], específica para *machine learning*, dentro de la que se han utilizado los siguientes métodos específicos para redes neuronales y métricas en general de los métodos empleados:
  - `MLPClassifier`
  - `MLPRegressor`
  - `MultiOutputClassifier`
  - `train_test_split`
  - `score`
  - `mean_squared_error`
  - `mean_absolute_error`
  - `r2_score`

Aunque en foros de expertos, se recomienda dejar **Scikit-Learn** para propósitos educativos y de aprendizaje, para este trabajo debería ser suficiente, ya que emplear librerías como **TensorFlow** de Google estaría enmarcado en propósitos comerciales o de una muestra de datos poco manejable.

Los datos se han separado en dos conjuntos, de entrenamiento y de prueba, por medio de la función `train_test_split`, que lo hace aleatoriamente y los dispone en 4 conjuntos:

- Un conjunto de entradas de entrenamiento.
- Un conjunto de salidas de entrenamiento.

- Un conjunto de entradas de prueba.
- Un conjunto de salidas de prueba.

La función emplea una semilla (*seed*) y la indicación de un indice entre 0 y 1 que da el tanto por uno de lo que representaría el conjunto de entrenamiento dentro del conjunto inicial de datos.

Las funciones `MLPClassifier` y `MLPRegressor` son idénticas en uso, tienen una serie de parámetros, que a continuación se detallarán los más importantes, junto con los valores escogidos:

1. *`hidden_layer_sizes`*, aquí se escoge el número de elementos de la capa oculta. Dependiendo de si se utilizó una o dos capas ocultas, la manera de expresarlo sería para una capa oculta (n,) con n número de unidades de la capa, y para dos capas ocultas (n,m,) con n número de unidades para la primera capa oculta y m número de unidades para la segunda.

2. *`activation`*, función de activación, que entre las que tiene disponible esta librería se empleó la función logística (A) que daba buenos resultados en las pruebas previas. Del resto, en cálculos preliminares, la tangente hiperbólica (A) no dio buenas predicciones.

3. *`Solver`*, es el optimizador, donde se escogió "lbfgs", que pertenece a la familia de los métodos quasi-Newtonianos, que se parece a lo que hay en la literatura. El modo por defecto sería "adam".

4. *`alpha`*, penalización L2, valor por defecto 0.0001.

5. *`max_iter`*, número de iteraciones máxima, valor escogido 1000, que coincide con el valor por defecto.

6. *`learning_rate`*, *`learning_rate_init`*. Tasa de aprendizaje. Constante y 0.001 (el valor por defecto). En caso de no haber escogido constante, el valor de *`learning_rate_init`* se iría modificando en las sucesivas iteraciones.

La mayoría de parámetros se escogieron valores por defecto. `MLPClassifier` se emplearía para la clasificación y `MLPRegressor` para realizar un ajuste de datos. `MultiOutputClassifier` es para salidas múltiples, y se emplea conjuntamente con `MLPClassifier`. `MLPRegressor` no admite salida múltiple.

Capítulo 4

Resultados

En la presente sección se analizarán las predicciones obtenidas para los distintos tipos de redes. Para conseguirlos se ha empleado, como se comento en el capítulo previo, el lenguaje de programación **Python** y la librería **Scikit-Learn**. El funcionamiento de esta librería es bastante sencillo, se encarga prácticamente de todos los cálculos de ajuste de parámetros de la red además de las propias predicciones y en estimar su precisión. Se comenzó seleccionando los datos recabados a partir de un archivo **csv** (*comma-separated values*) con otra librería específica para tratar con amplias estructuras de datos (**Pandas**). A partir de aquí se escogieron dos conjuntos de datos, parecidos a matrices donde las columnas son los tipos de dato mencionados en la metodología y las filas son los registros de alumnos con los respectivos valores. El primero con todas la posibles entradas y el segundo con todas las posibles salidas. En este punto, para seleccionar el conjunto de datos de entrenamiento y de prueba se emplea una función específica de **Scikit-Learn** llamada `train_test_split` que separa los dos conjuntos de datos de las salidas y entradas en los siguientes cuatro:

- Un conjunto de entradas de entrenamiento.
- Un conjunto de salidas de entrenamiento.
- Un conjunto de entradas de prueba.
- Un conjunto de salidas de prueba.

Una vez separados los conjuntos de entrenamiento y prueba, a partir de estos se seleccionan las entradas específicas de cada apartado mencionado en la metodología con su/sus respectiva salida/s. En este punto es cuando se procede a entrenar la red neuronal, para ello se inicia una instancia de la función `MLPClassifier` o `MLPRegressor`, según el caso, con los parámetros siguientes:

1. *`hidden_layer_sizes`*, el número de elementos de la capa oculta (n,). Se fueron variando las configuraciones de este parámetro, n, a partir de 1 unidad hasta 9.
2. *`activation`*, función de activación, se empleó la función logística: 'logistic'.

3. *Solver*, es el optimizador, donde se escogió 'lbfgs'.
4. *alpha*, penalización L2, valor por defecto 0.0001.
5. *max_iter*, número de iteraciones máxima, valor escogido 1000.
6. *learning_rate*, *learning_rate_init*. Se escogió constante y 0.001 respectivamente.

Esto genera un clasificador (o regresor) donde se ajustarán los parámetros de la red con la función `fit` usando los conjuntos de entrenamiento. Cuando se considere una salida múltiple se emplea una instancia de `MultiOutputClassifier` asignándole como parámetro la instancia del clasificador anterior, que generará un clasificador nuevo que será el que se utilice a partir de este momento con su propia función `fit` para ajustar los datos de entrenamiento. En este paso es cuando se inicia la estructura de la red y se entrena. Una de las ventajas (no aprovechada en este trabajo) es que **Scikit-Learn** puede funcionar con valores numéricos y cadenas de caracteres para en los conjuntos de entrada y salidas para la clasificación.

Una vez realizado el ajuste del entrenamiento se emplea la función de la instancia del clasificador/regresor `predict` introduciendo el conjunto de entradas de prueba para obtener las predicciones para estos valores. Con esto obtenemos un *array* de datos de una columna en el caso de clasificador o regresión y dos columnas para el caso de la salida múltiple. Este conjunto se compara con el conjunto de salidas de prueba (los datos de la salida esperada) empleando las funciones de métrica que provee **Scikit-Learn** para comprobar los aciertos y errores en la estimación. Para los clasificadores se emplea la función `score` y para la parte de regresión las funciones `r2_score` y `mean_squared_error`.

A continuación se detallan los resultados obtenidos por las distintas simulaciones realizadas, se clasificarán estos datos según el tipo de salida de la red que se ha establecido en la metodología. Para el análisis de los resultados, se han calculado según el tipo de simulación: los porcentajes de acierto o precisión para la clasificación/predicción, y el coeficiente de regresión r^2 y el error cuadrático medio para la parte de regresión.

4.1. Predicción de aprobados/suspensos

En esta sección se comentarán los resultados obtenidos para las redes *Tipo I* donde la salida, como se comentó previamente en la metodología, corresponde con una única unidad con dos posibles valores: aprobado (1) y suspenso (0). Para estas simulaciones se consideran los cuatro tipos de capas de entrada definidos en el apartado correspondiente a la topología dentro de la Metodología, siempre partiendo de la base de las notas de las dos primeras evaluaciones.

Los resultados han dado valores mayoritariamente superiores al 90 % de precisión como se puede observar en la tabla 4.1, dándose además cierta homogeneidad entre los distintos tipos de entradas. Emplear las notas de las dos primeras evaluaciones,

capa *Tipo A*, por sí solas suponen un rendimiento más que aceptable y bueno, y añadir el refuerzo en las correspondientes evaluaciones, *Tipo C*, no supone diferencia significativa. El considerar el identificador del grupo aumenta la precisión de la predicción, dando mejores resultados el no incluir el refuerzo, *Tipo B*, aunque no es una mejora que se pueda definir como clara y, que ante una hipotética elección de qué tipo de red fuera ideal (de las cuatro), dé un argumento determinante para escoger una de los cuatro tipos. Aunque se puede afirmar que aumentar la información trae consigo una pequeña mejora del rendimiento.

número neuronas	Precisión			
capa oculta	Tipo A	Tipo B	Tipo C	Tipo D
1	0.9138	0.9138	0.9138	0.9483
2	0.9138	0.9138	0.9138	0.8966
3	0.8966	0.9483	0.8966	0.8621
4	0.9138	0.9483	0.9138	0.9483
5	0.9138	0.9138	0.8966	0.9138
6	0.8793	0.9310	0.9138	0.9310
7	0.9138	0.9483	0.8966	0.8793
8	0.9138	0.9310	0.9138	0.9310
9	0.8966	0.9655	0.8966	0.9483

Cuadro 4.1: Tabla con la precisión para las distintas simulaciones sobre las redes de salidas *tipo I*. La tabla se ha obtenido a partir de los resultados de las métricas calculadas con los aciertos y errores en el conjunto de prueba, comparado con la predicción hecha para los distintos tipos de entradas y según el número de unidades de la capa oculta.

Con respecto a los resultados atendiendo al número de unidades en la capa oculta, no se ve que el aumentar este número mejore el rendimiento. Este varía entre el 86 % y el 96 % según el número de unidades, ofreciendo una buena precisión de resultados y ser más uniforme al moverse en una horquilla del 10 %. Definitivamente, el aumentar la información hace mejorar el rendimiento de la red más que el aumentar o disminuir el número de unidades de la capa oculta. El variar el número de neuronas produce configuraciones más o menos favorables según el tipo de entradas.

Como nota destacable, decir que una configuración del *tipo B* (tres entradas: notas e identificador de grupo) al emplear 9 unidades aparece el mejor resultado en cuanto a predicción del aprobado/suspenso, cercano del 96 % (0.9655). En la figura (4.1) se puede comprobar cómo únicamente acumula dos errores en la parte donde las notas de las dos evaluaciones son bajas, acertando los casos extremos de los aprobados y el suspenso de la esquina inferior izquierda de la gráfica (notas de la primera y segunda evaluación bajas).

Como se puede ver en la figura 4.1 los únicos errores son para valores bajos de las notas de primera y segunda evaluación, además de predecir bien los aprobados claros. En cuanto al suspenso erróneo, seguramente la red "entienda" que en este caso, lo más probable es que al tener valores bajos de las dos primeras evaluaciones,

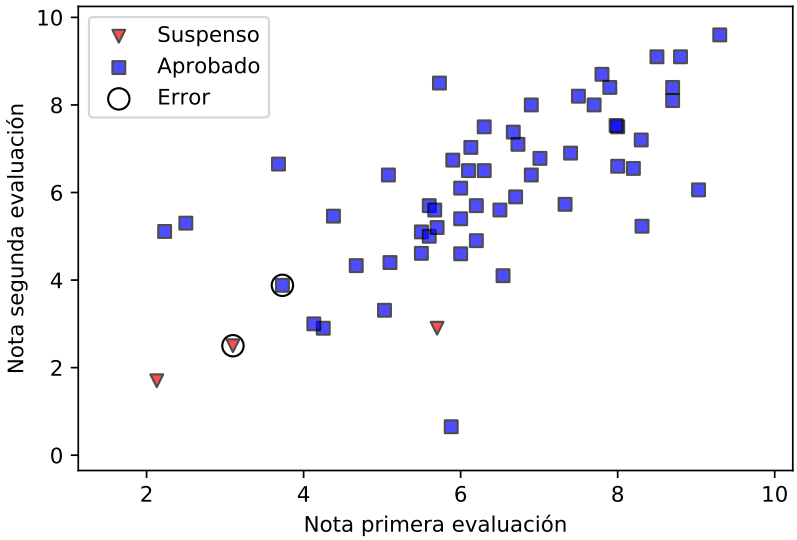


Figura 4.1: Predicción de aprobados/suspensos con una red *Tipo B* con tres entradas (dos correspondientes a notas y una a identificador de grupo), 9 unidades neuronales en la capa oculta y una salida que asigna un 0 a un suspenso y un 1 a un aprobado. Los parámetros con los que se realizaron el ajuste de los datos fueron los que vienen por defecto para un cálculo de este tipo con la librería **Scikit-Learn** salvo el optimizador que se escogió "lbfgs". Este gráfico se ha elaborado a partir de los resultados obtenidos con la función `predict` de `MLPClassifier` para el conjunto de entrada de prueba. Se representa el aprobado o el suspenso de un registro en función de las notas obtenidas en las dos primeras evaluaciones. Los errores, marcados con círculos, se han obtenido comparando el *array* obtenido con en clasificador y el *array* con los datos reales.

la nota final sea un suspenso. Parece que entiende que es una media y que pueda que le falte un valor que complemente a los de las dos notas.En el segundo caso, los valores de las notas son cercanos al aprobado en las dos evaluaciones y puede que este hecho "empuje" la nota final hacia el aprobado definitivo. Sin embargo, en la realidad fue todo lo contrario. Se puede especular con que, en el primer caso, el registro (alumno/a) logró aprobar en alguna convocatoria extraordinaria o en diferentes exámenes de recuperación. Mientras que en el segundo caso, se puede teorizar en la posible bajada de rendimiento del registro ya que con un pequeño esfuerzo podría haber superado la asignatura.

Si nos fijamos en todos los errores cometidos por todos los distintos diseños de red para la predicción de aprobados/suspensos, vemos que predice muy bien los casos extremos en los que un registro aprueba las dos evaluaciones, pero falla en muchos casos límite, en los que el registro suspende al menos una evaluación y que la media entre las notas de ambas evaluaciones está entre el aprobado y el suspenso. También parece, que el suspenso claro lo predice en todos las configuraciones. Como se ven en

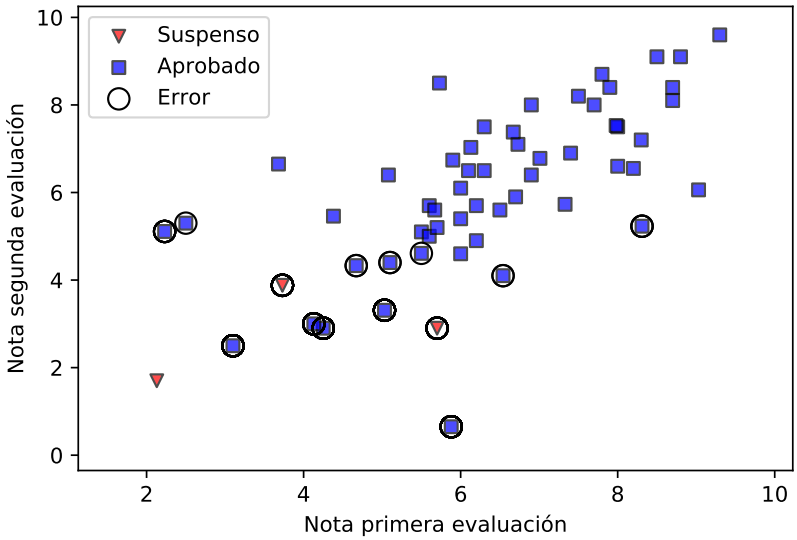


Figura 4.2: Errores acumulados en la predicción aprobados/suspensos del conjunto de prueba para el *Tipo I*. A diferencia de la anterior gráfica 4.1, aquí se muestran registros aprobados y suspensos finales y se marcan con un círculo los errores que se hayan cometido en las distintas configuraciones de red para este tipo.

la gráfica 4.2 los aprobados los cataloga perfectamente, pero falla en los valores límite del aprobado. En el caso de los suspensos tampoco los cataloga satisfactoriamente. Una de las razones puede ser que se tomaron medidas de refuerzo adecuadas para que los registros pudiesen aprobar finalmente.

4.2. Predicción del tipo de refuerzo

En esta sección se comentarán los resultados obtenidos para las redes *Tipo II* donde la salida, como se mencionó previamente en la metodología, corresponde con una única unidad con tres posibles valores. Nuevamente se consideraron cuatro tipos de capas de entrada, partiendo como base siempre de las notas de las dos primeras evaluaciones.

Se pueden observar, en la tabla 4.2, como hay variaciones que pueden llegar al 19 %, y como las notas por sí solas (*Tipo A*), independientemente del número de unidades neuronales en la capa oculta, no predicen bien el refuerzo final como era de esperar. Como se ha comentado, incluir el refuerzo de las dos primeras evaluaciones *Tipo C* ayuda a mejorar esa predicción como se puede observar en la figura 4.4 que únicamente comete un error asignando un refuerzo significativo (valor 1) a un registro que carece de él (valor 0). Es interesante esta asignación ya que aprueba una evaluación (la segunda) pero la media de las dos es baja, lo que la red interpreta como que el registro acabaría siéndole asignado un refuerzo, cosa que finalmente no ocurrió

número neuronas capa oculta	Precisión			
	Tipo A	Tipo B	Tipo C	Tipo D
1	0.7931	0.7931	0.9828	0.9828
2	0.8103	0.7759	0.8966	0.9828
3	0.7759	0.7931	0.9483	0.9655
4	0.7759	0.7586	0.8966	0.9138
5	0.7931	0.7759	0.8966	0.9310
6	0.7586	0.7931	0.9655	0.8966
7	0.7241	0.7586	0.9655	0.9310
8	0.7414	0.7931	0.8966	0.9138
9	0.7414	0.7241	0.8966	0.8966

Cuadro 4.2: Tabla con la precisión para las distintas simulaciones sobre las redes de salidas *tipo II*. La tabla se ha obtenido a partir de los resultados de las métricas calculadas con los aciertos y errores en el conjunto de prueba, comparado con la predicción hecha para los distintos tipos de entradas y según el número de unidades de la capa oculta.

y superó satisfactoriamente el curso (como se puede comprobar en la figura 4.1). En la figura 4.3 se ve que con otra configuración de red, esta última le asigna un refuerzo mayor (valor 2) a este mismo registro. Con respecto a la inclusión del identificador del grupo (*Tipos B* y *D*), aparentemente no afecta de manera destacada, salvo en alguna configuración específica.

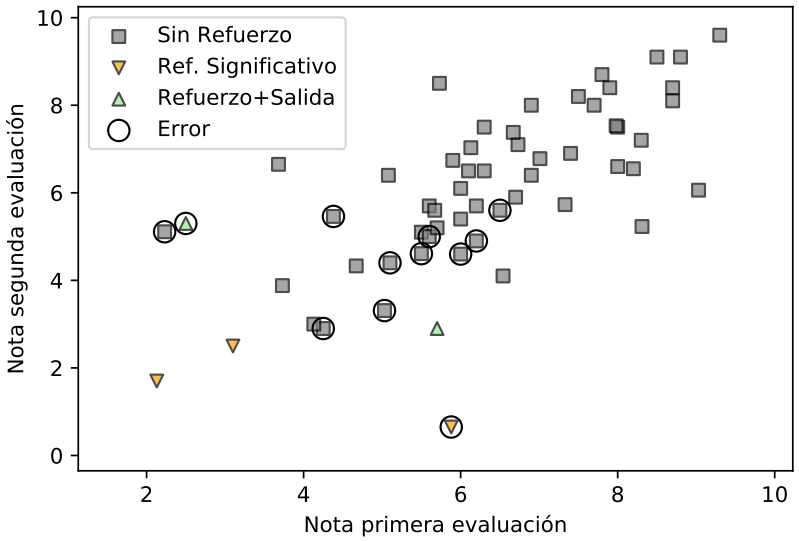


Figura 4.3: Predicción de refuerzo para una red *Tipo B* con tres entradas (dos correspondientes a notas y una al identificador del grupo), 6 unidades neuronales en la capa oculta y una salida, que asigna los distintos valores para los refuerzos.

A la hora de clasificar los refuerzos al final del curso, a diferencia de la predicción de las notas en el anterior apartado, se observan diferencias marcadas dependiendo de si en las entradas se incluye el refuerzo de las dos primeras evaluaciones. Considerando únicamente las notas, e incluso las notas y el identificador de grupo como se observa en la gráfica 4.3, algunos registros que logran aprobar gracias al refuerzo como medida de atención a la diversidad son clasificados sin la necesidad de un refuerzo al finalizar el curso, cuando esto se da únicamente en casos donde hay una mejoría en el rendimiento o el motivo por el que se adoptó la medida ha desaparecido, con lo que el equipo evaluador determina el fin de la medida. Lo más probable es que estos registros tuviesen establecido el refuerzo como una medida ante una dificultad de aprendizaje y que con dicho refuerzo se consigue que supere la asignatura. Al incluir el refuerzo de las dos primeras evaluaciones, la red podría captar los matices de asignar un refuerzo para un registro que, siempre hipotéticamente, carecía de este y el rendimiento fuese bajo. La consideración del identificador de grupo debería dar una ligera orientación a la red de que en algunos casos, en ciertos cursos, se agruparían los registros que presentan dificultades, como ocurre en 3º y 4º de ESO.

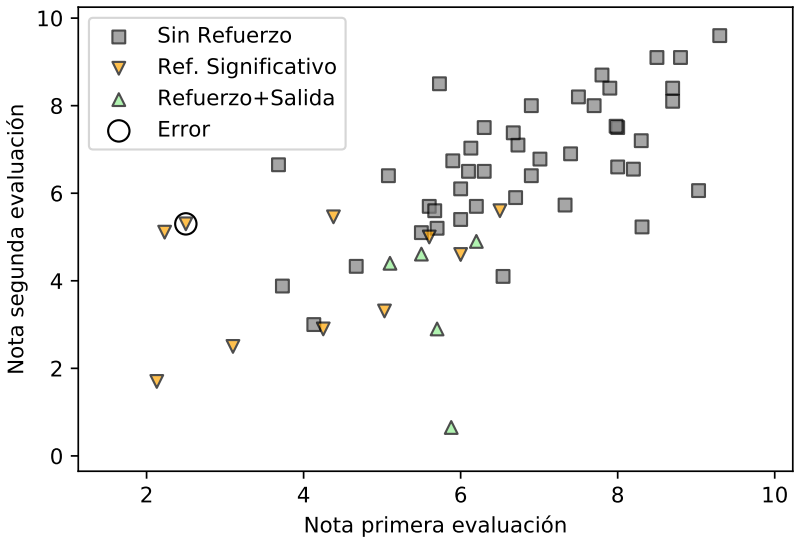


Figura 4.4: Predicción de refuerzo para una red *Tipo D* con cinco entradas (dos correspondientes a notas, una al identificador del grupo y dos a los refuerzos), 1 unidad neuronal en la capa oculta y una salida, que asigna los distintos valores para los refuerzo.

Atendiendo al número de unidades de capa oculta, no se ven grandes cambios, y no se ve claramente que el aumento o disminución, produzcan resultados destacables. Si bien, cabe destacar que los mejores resultados de las predicciones se dan para configuraciones con pocas unidades, 5 o menos unidades. Este es el caso del *Tipo A* y considerando el refuerzo, los *Tipos C* y *D*, siendo la configuración en este último

tipo de 1 y 2 neuronas, con un 98 % (0.9828) y de 1 neurona en *Tipo C*. En el *Tipo B* no se observan grandes diferencias. Parece que en este punto se puede deducir que más que mejora, lo que se producen son configuraciones más o menos favorables según el tipo de entradas.

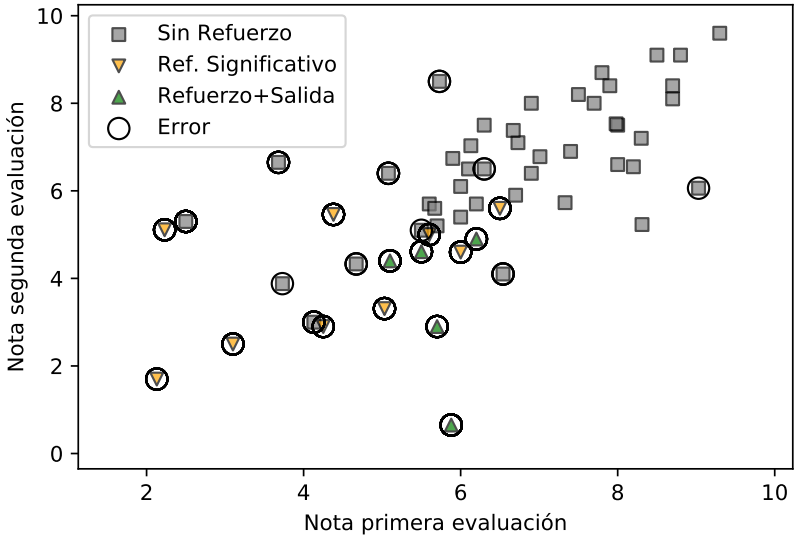


Figura 4.5: Errores acumulados en la predicción del refuerzo del conjunto de prueba para el *Tipo II*. A diferencia de las anteriores gráficas de esta sección, aquí se muestran los valores correspondientes al refuerzo final de los registros y se marcan con un círculo los errores que se hayan cometido en las distintas configuraciones de red para este tipo.

4.3. Predicción de aprobados/suspensos y tipo de refuerzo (salida múltiple)

En esta sección se comentarán los resultados correspondientes con el tipo de red *Tipo IV*, que conforma una salida múltiple, de dos neuronas (el aprobado/suspenso y el refuerzo), como se describió en la metodología. Como en los anteriores casos, también se comentarán los distintos tipos de entradas y la variación en el número de unidades de capa oculta. En este caso se incluyó `MultiOutputClassifier` conjuntamente con `MLPClassifier` para poder combinar las salidas de notas y refuerzo.

En la tabla 4.3 se puede observar, como ocurría en la anterior sección, que las notas por sí solas no bastan para obtener una buena predicción, tanto de nota final como de refuerzo final conjuntamente. Es necesario incluir los refuerzos de las dos primeras evaluaciones en las entradas para mejorar notablemente los resultados. En este caso no se da una mejoría amplia como en el caso anterior, sino que la diferencia no es tan marcada como en la predicción del refuerzo con una única salida. Además

número neuronas capa oculta	Precisión			
	Tipo A	Tipo B	Tipo C	Tipo D
1	0.7586	0.7586	0.8276	0.8966
2	0.7759	0.7759	0.8966	0.8276
3	0.7586	0.7586	0.8966	0.8621
4	0.7241	0.7241	0.8621	0.8793
5	0.7759	0.6724	0.8448	0.8621
6	0.7414	0.6897	0.8448	0.8793
7	0.7069	0.7414	0.8448	0.8966
8	0.7069	0.6724	0.8448	0.8621
9	0.6724	0.7586	0.8793	0.8621

Cuadro 4.3: Tabla con la precisión para las distintas simulaciones sobre las redes de salidas *tipo IV*. La tabla se ha obtenido a partir de los resultados de las métricas calculadas con los aciertos y errores en el conjunto de prueba, comparado con la predicción hecha para los distintos tipos de entradas y según el número de unidades de la capa oculta.

de incluir los refuerzos de las dos primeras evaluaciones, también tener en cuenta el identificador de grupo mejora la predicción en muchas configuraciones, aunque en otras empeora, produciéndose en ambos casos de manera leve.

Al usar una salida múltiple se empeora el rendimiento con respecto a las salidas simples. Como en el caso de la predicción del rendimiento, el incluir junto con las notas el identificador de grupo no mejora los resultados, en cambio introducir el refuerzo sí que lo hace. Mantiene la validez lo comentado en la anterior sección para el refuerzo: se hace necesario incluir el refuerzo (y en menor medida el identificador) para obtener unos resultados que mejoren la precisión. Cabe decir que el peso del refuerzo es más evidente, ya que con las notas de las evaluaciones únicamente se clasificaría bien el aprobado/suspenso y no el refuerzo, como quedó claro en la anterior sección (se puede observar este hecho en la figura 4.8).

Como se ve en la figura 4.6 la configuración de red predice bien los registros en los que se aprueban las dos evaluaciones como en el ejemplo de la primera sección, sin embargo, las notas bajas en la segunda evaluación que aprueban el curso no los predice bien. En la salida correspondiente al refuerzo, en la misma figura, ocurre un resultado análogo al de la figura 4.4 pero con una configuración de red diferente, el mismo análisis hecho en la sección de aquel resultado se puede hacer para este resultado. A mayores, se puede comentar que el factor limitante en la precisión de la salida múltiple es la predicción del refuerzo. En la figura 4.7 se observa que al mejorar la predicción de los aprobados/suspensos con respecto a la anterior figura no se consigue una mejora en la precisión, sino que el número de errores en la predicción de los refuerzos aumenta, en estos dos ejemplos la precisión de ambas configuraciones es la misma. Se puede ver más claro este hecho en la figura 4.8 donde consigue una buena predicción para aprobados y suspensos pero falla estrepitosamente con los refuerzos, no clasificando ninguno de los refuerzos con salida (fuera del aula).

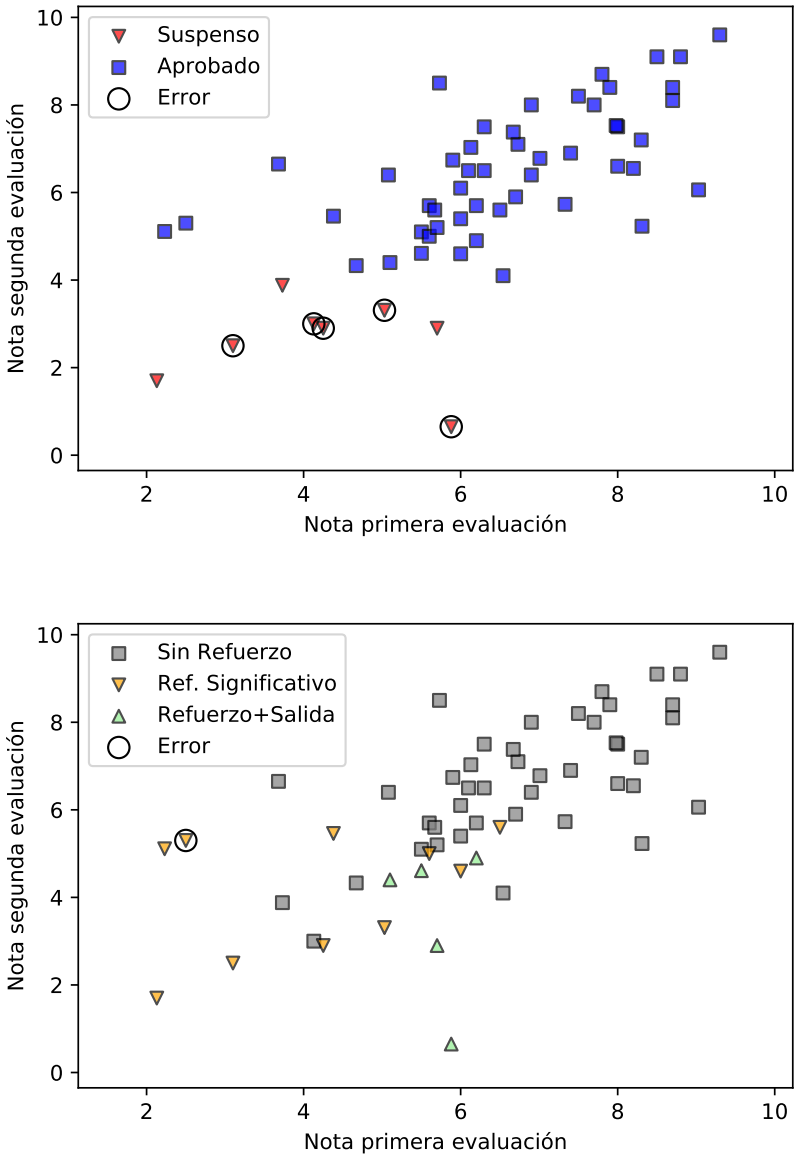


Figura 4.6: Predicción de aprobados/suspensos (arriba) y refuerzo (abajo) para salida múltiple con una red *Tipo C* con cuatro entradas (dos correspondientes a notas y dos a los refuerzos), 3 unidades neuronales en la capa oculta y dos salidas, que asignan aprobados/suspensos y refuerzos.

En cuanto al número de unidades en la capa oculta, como se puede ver en la tabla 4.3, no hay mejora evidente hacia el aumento o disminución de neuronas, salvo en el *tipo A* que la precisión mejora con valores de 5 o menos unidades. Para el *tipo C* se podría hacer el mismo análisis ya que para 2 y 3 unidades se dan los mejores resultados, pero también tiene el peor registro, de este tipo, con 1 unidad. Para 5 o más, los resultados parecen más homogéneos (iguales). El punto común que tienen

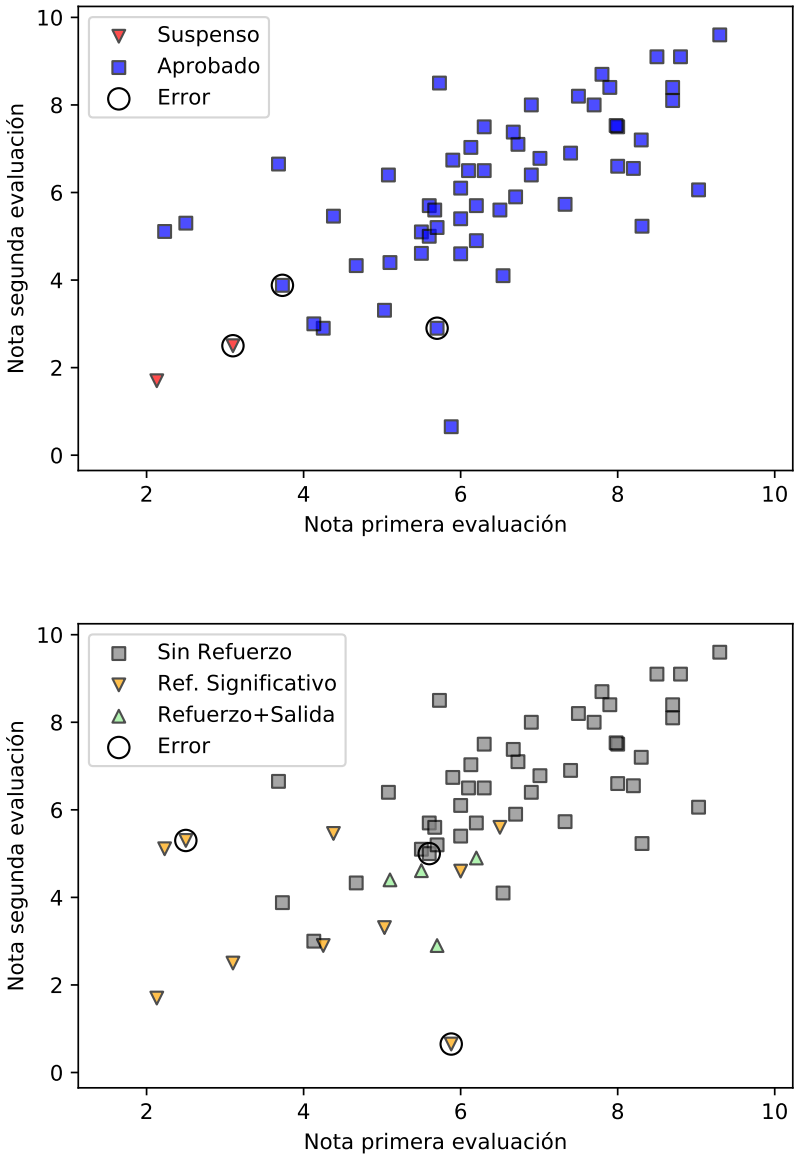


Figura 4.7: Predicción de aprobados/suspensos (arriba) y refuerzo (abajo) para salida múltiple con una red *Tipo D* con cinco entradas (dos correspondientes a notas, una al identificador de grupo y dos a los refuerzos), 7 unidades neuronales en la capa oculta y dos salidas, que asignan aprobados/suspensos y refuerzos.

estos dos tipos de entradas es la ausencia del identificador de grupo, que quizás actúe como factor homogeneizador y haga que no sea tan evidente la mejora al aumentar o disminuir el número de elementos de la capa oculta. Al igual que en el anterior caso, más que mejorar o empeorar, hay configuraciones más adecuadas según el tipo de entradas.

Si nos fijamos en los valores de la precisión en la tabla 4.3, los resultados varían

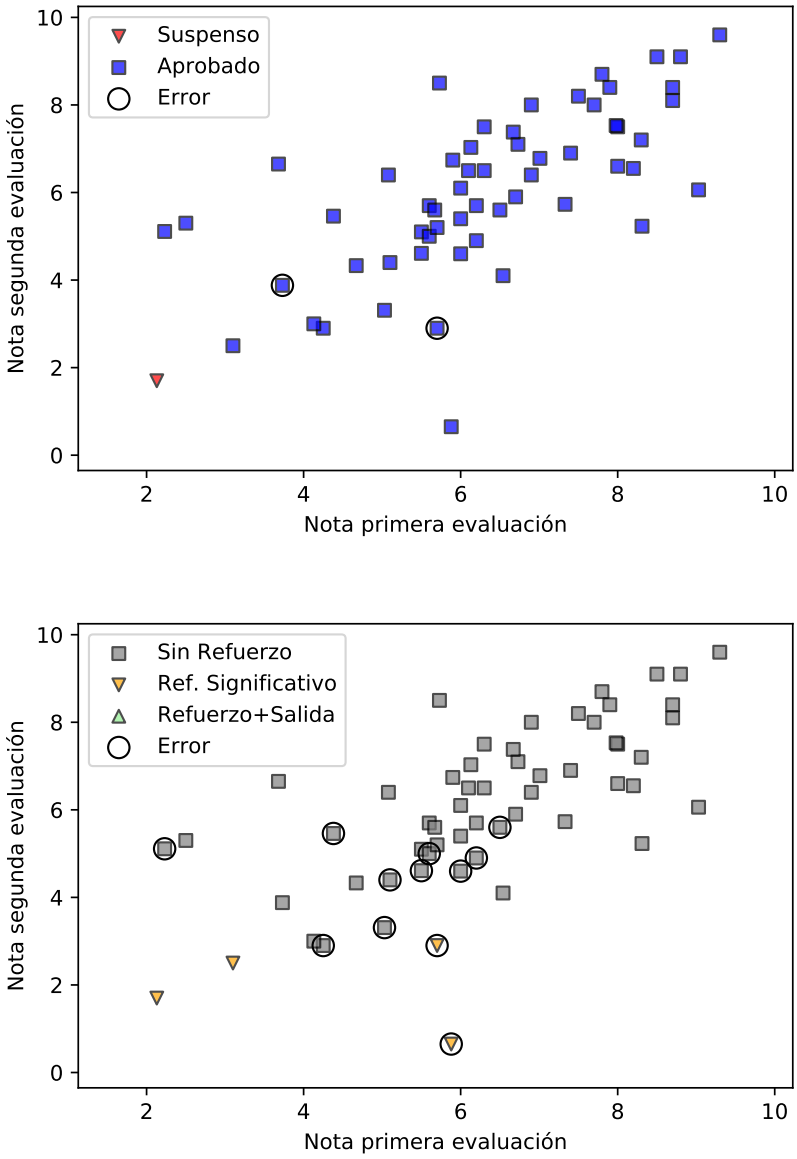


Figura 4.8: Predicción de aprobados/suspensos (arriba) y refuerzo (abajo) para salida múltiple con una red *Tipo B* con tres entradas (dos correspondientes a notas y una al identificador de grupo), 2 unidades neuronales en la capa oculta y dos salidas, que asignan aprobados/suspensos y refuerzos.

entre un 67 % y el 89 % dependiendo de el tipo de entradas y del número de unidades en la capa oculta, y son ligeramente inferiores que considerar por separado la clasificación de aprobados/suspensos y del refuerzo final.

Como se pudo ver en el caso del *Tipo I*, al comprobar la acumulación de errores en la gráfica 4.9 se muestra que predice bien los aprobados claros, como en el *tipo I*, pero falla en el caso de los suspensos 4.9. Los aprobados los cataloga perfectamente,

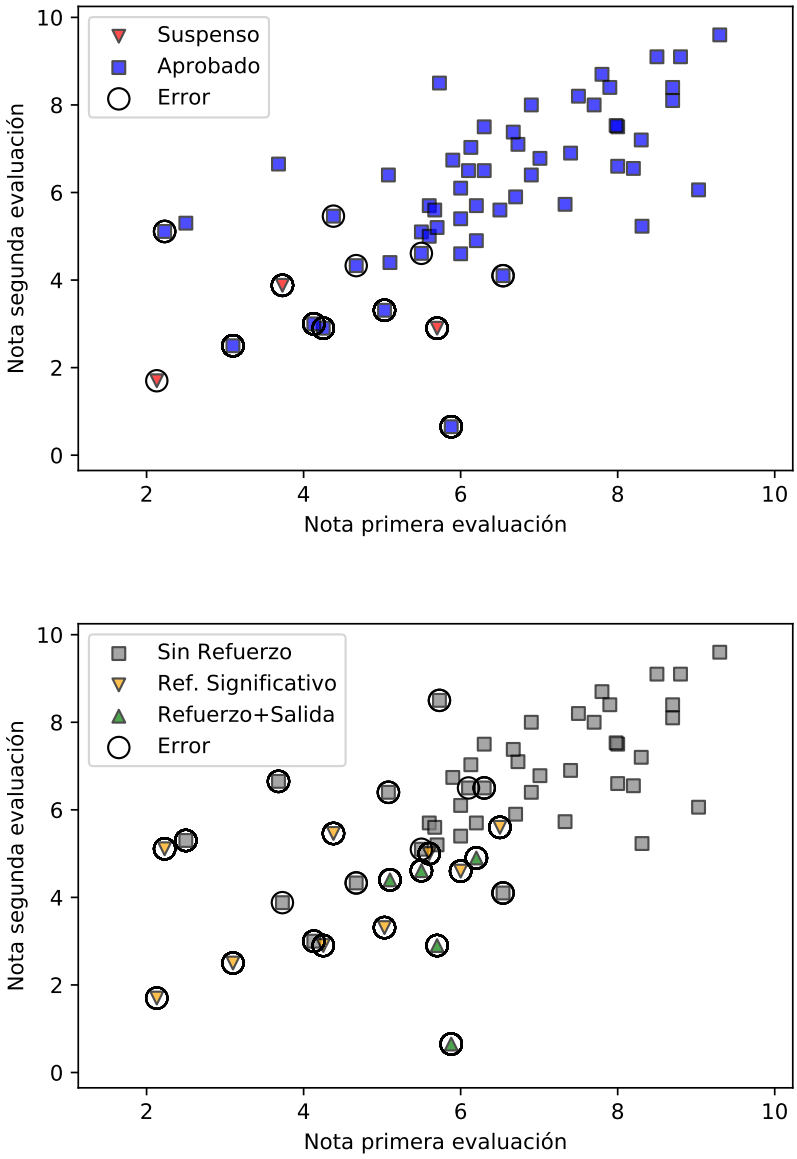


Figura 4.9: Errores acumulados en la predicción aprobados/suspensos (arriba) y refuerzo (abajo) del conjunto de prueba para el *Tipo IV*. A diferencia de las anteriores gráficas, aquí se muestran registros aprobados y suspensos finales, los tres tipos de refuerzo y se marcan con un círculo los errores que se hayan cometido en las distintas configuraciones de red para las salidas múltiples.

pero falla en los valores límite del aprobado y en los suspensos. Parece que este tipo de redes predicen bien los aprobados, pero por contra, con los suspensos no tiene la misma fiabilidad. En este caso, se podría pensar que no tiene mucha utilidad al margen de predecir el buen rendimiento académico, siendo de interés el catalogar bien las situaciones extremas dentro de los valores bajos de las notas de la primera y

segunda evaluación, pero sería mejor el considerar una configuración que produce dos buenas predicciones que dos configuraciones que producen muy buenas predicciones.

4.4. Predicción de nota final

Los resultados de esta sección se corresponden con las redes *Tipo III*, donde se consideran los cuatro tipos de entradas mencionados en la metodología y una única salida que especifica la nota final numérica. La obtención de estos resultados fue por medio de `MLPRegressor` con una salida de tipo numérica entre 0 y 10. Para la precisión de estos resultados, mostrada en la tabla 4.4, se recurrió a la función `r2_score`. Para la gráfica 4.10 se empleó la métrica `mean_squared_error` que calcula el error cuadrático medio cometido en los distintos resultados de esta sección.

número neuronas capa oculta	coeficiente r^2			
	Tipo A	Tipo B	Tipo C	Tipo D
1	0.7948	0.7947	0.7826	0.7851
2	0.8202	0.7980	0.7994	0.8138
3	0.7774	0.8351	0.7633	0.7778
4	0.7830	0.8207	0.7143	0.4524
5	0.7868	0.8289	0.7662	0.8844
6	0.8098	0.5833	0.7834	0.7814
7	0.7421	0.6760	0.7544	0.5177
8	0.7990	0.2048	0.0604	0.8157
9	0.7386	0.7057	0.2610	0.7139

Cuadro 4.4: Tabla con los coeficientes r^2 para las simulaciones de la red de salidas *Tipo III*. La tabla se ha obtenido a partir de los resultados de las métricas calculadas del ajuste de datos realizado sobre el conjunto de prueba para los distintos tipos de entradas y según el número de unidades de la capa oculta.

El ajuste de los datos para la predicción no es muy bueno ya que todos los coeficientes r^2 están por debajo de 0.9 como se puede observar en la tabla 4.4. En general incluir, tanto notas, refuerzo, como el identificador de grupo mejora la predicción, pero depende del número de unidades en la capa oculta. Una configuración de 5 neuronas es la que mejor resultado ha dado para las redes de 5 entradas (*Tipo D*), 0.88 con un error cuadrático medio de 0.26. También para la de *Tipo B*, pero en este caso le supera una configuración con 3 neuronas en la capa oculta (0.83 y 0.37). Para 2 entradas (*Tipo A*) los resultados son más homogéneos, siendo para 2 neuronas de capa oculta el mejor resultado (0.82 y 0.40). La de 4 de entrada (notas y refuerzo, o *Tipo C*) da también resultados homogéneos para configuraciones con una capa oculta menor de 8 neuronas.

Se puede ver en el gráfica (4.10) la representación del error cuadrático medio en función del número de neuronas de la capa oculta. Aumentar el número de unidades

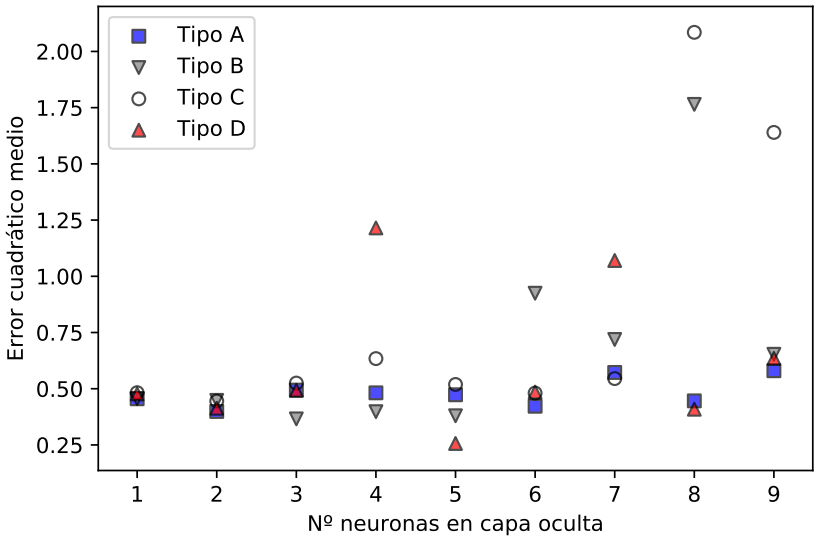


Figura 4.10: Error cuadrático medio en función del número de unidades neuronales en la capa oculta para cada tipo de red según el número de entradas para una red de salidas *Tipo III*. Para la obtención de estos valores se ha aplicado la función `mean_squared_error` sobre el conjunto de valores de la salida esperada de prueba y se comparó con la predicción que ofreció `MLPRegressor` al conjunto de entrada de prueba.

de la capa oculta no implica el aumentar el error, ya que con 4 unidades también da errores altos. En general, el *tipo A* es el que menores errores ofrece, dando menores variaciones en el coeficiente r^2, aunque no ofrece el mejor rendimiento como se comento, aunque está cerca.

Como se ve en la tabla 4.5 los valores numéricos de la nota final son parecidos, en algunos casos, a los predichos por las distintas configuraciones de red. Para la gran mayoría resulta un valor intermedio entre las notas de las dos primeras evaluaciones. Parece que la red "entiende" que la nota final sea una media de los valores de las notas de las dos primeras evaluaciones, dando en muchos caso valores intermedios o cercanos a estas. El incluir el refuerzo y el identificador afina la precisión en muchos casos.

Hay registros en que la predicción obtenida difiere del resultado real en casi un punto. En algunos casos, unos 5 en total (aunque 3 corresponden al mismo registro), se obtienen notas con un valor negativo, o por encima de 10, estos se dan para entradas del *tipo B, C* y *D* pero sobre todo *tipo C* y en configuraciones de capa oculta de 4, 6, 8 y 9 unidades, provocando que el coeficiente de ajuste r^2 se reduzca considerablemente, y el error cuadrático medio se dispare. Estos valores negativos se dan en registros donde las notas de la primera evaluación es mayor que la de la segunda, siendo en la primera mayor que 5 y en la segunda menor que 5. Además se da la circunstancia que los refuerzos de dos de los registros tienen un valor de

neuronas	id.	Nota 1ª	Nota 2ª	Nota	Predicción Nota final			
c. oculta	grupo	eval.	eval.	final	Tipo A	Tipo B	Tipo C	Tipo D
2	1	5.70	2.90	3.96	4.40	3.96	4.30	4.60
5	1	5.70	2.90	3.96	4.75	4.63	4.39	2.62
8	1	5.70	2.90	3.96	4.79	12.52	-0.34	3.47
2	2	8.50	9.10	9.10	8.64	8.87	8.73	8.92
5	2	8.50	9.10	9.10	8.90	8.93	9.01	8.97
8	2	8.50	9.10	9.10	8.95	8.95	9.07	9.02
2	3	6.90	6.40	6.60	6.76	6.66	6.63	6.56
5	3	6.90	6.40	6.60	6.48	6.48	6.43	6.48
8	3	6.90	6.40	6.60	6.45	6.46	6.37	6.38
2	4	2.50	5.30	5.10	4.57	4.20	4.27	4.74
5	4	2.50	5.30	5.10	4.81	5.05	4.66	5.01
8	4	2.50	5.30	5.10	4.95	5.09	6.79	5.14
2	5	8.20	6.55	6.67	7.44	7.32	7.29	7.11
5	5	8.20	6.55	6.67	7.08	7.35	6.98	7.10
8	5	8.20	6.55	6.67	7.28	7.17	7.32	7.39
2	6	4.67	4.33	4.67	4.58	4.61	4.69	4.98
5	6	4.67	4.33	4.67	4.96	5.02	5.04	5.04
8	6	4.67	4.33	4.67	4.99	5.20	5.02	5.10
2	7	2.13	1.70	1.88	1.61	2.10	1.37	1.64
5	7	2.13	1.70	1.88	1.75	1.69	0.96	1.72
8	7	2.13	1.70	1.88	1.67	1.34	1.60	1.64
2	8	3.73	3.88	3.33	3.83	3.99	4.12	4.60
5	8	3.73	3.88	3.33	4.60	4.63	4.60	4.36
8	8	3.73	3.88	3.33	4.61	4.53	4.43	4.41
2	9	5.50	4.61	5.17	5.18	4.74	5.22	5.02
5	9	5.50	4.61	5.17	5.27	5.07	5.36	5.17
8	9	5.50	4.61	5.17	5.20	5.18	5.27	5.05
2	10	2.23	5.11	5.33	4.41	3.81	4.17	4.51
5	10	2.23	5.11	5.33	2.78	4.67	4.74	4.92
8	10	2.23	5.11	5.33	4.33	4.81	4.50	4.07

Cuadro 4.5: Tabla con los valores de la nota final numérica para las simulaciones de la red de salidas *Tipo III*, para 10 registros específicos del conjunto de datos de prueba, para los valores de unidades de capa oculta correspondientes a 2, 5 y 8.

2, refuerzo junto con salida fuera del aula. Una explicación plausible, es que la red entienda que la nota final es el resultado de una progresión de las notas de las dos primeras evaluaciones, y como son valores más bajos los de la segunda evaluación, el correspondiente con la nota final debería ser negativo. Además en los registros con notas inferiores a 5 en alguna o las dos evaluaciones, es donde se ve una cierta variabilidad entre las predicciones, muchas no concuerdan con el promedio de las notas de las evaluaciones del resto de registros.

En referencia a los resultados según la configuración de la capa oculta, y al igual que en los anteriores tipos de salidas, no hay una tendencia o patrón que diga que un mayor o menor número de unidades mejore el ajuste de datos y reduzca el error, simplemente hay configuraciones más favorables según el tipo de entradas.

4.5. Resultados con dos capas ocultas

En esta sección se muestran los resultados para las anteriores secciones aumentando en una capa oculta la configuración de las redes empleadas. Al igual que en anteriores secciones se emplearán distintas entradas y se variará el número de neuronas de las capas ocultas. El único parámetro que se ha variado es *hidden_layer_sizes*, en el que se especifica (n, m,) con la inclusión de un segundo valor, m, en la configuración.

4.5.1. Predicción de aprobados/suspensos

La principal diferencia al incluir una segunda capa oculta es que aumenta la horquilla de la precisión ya que el valor más bajo pasa a ser en torno al 0.7931 (79 %) para una configuración del *Tipo D* y 1 unidad en la primera capa oculta y 2 en la segunda, como se puede observar en la tabla 4.6. El valor más preciso corresponde a 1.00 (100 %) para un *tipo D* al igual que el menos preciso. En este caso, este valor máximo de precisión se obtiene para una red con una primera capa oculta de 3 unidades y una segunda de 7. Como en el caso de una capa oculta, no hay una tendencia o patrón claros, más bien configuraciones más adecuadas según el tipo de entradas.

Ya de por sí, la predicción de aprobados/suspensos, daba buenos resultados con una única capa oculta. Al incluir la segunda capa no se obtiene una mejora importante, salvo ese único resultado de 100 % que se puede catalogar de anecdótico, ya que en el resto de simulaciones no se vuelve a obtener un valor igual, a lo sumo se obtiene un valor de 0.96.

4.5.2. Predicción del tipo de refuerzo

En este tipo de redes no hay una mejora aparente de los resultados, mostrados buena parte en la tabla 4.7, si bien aparecen un par de valores que reducen mucho la precisión, no pasa por ser algo anecdótico, ya que son únicamente dos valores dentro de un conjunto de simulaciones muy amplio. Como en el caso de una capa oculta, los mejores resultados se dan para las redes de *tipo C* y *D* lo que indica que un aumento en la información considerada mejora la predicción y en concreto no llega con incluir únicamente las notas de las dos evaluaciones, sino el introducir el refuerzo de dichas evaluaciones mejora ostensiblemente los resultados. Como en el caso de la predicción del refuerzo final con una capa oculta, hay configuraciones, según el número de unidades de las capas ocultas, mejores que otras para cada tipo de entradas.

Unidades capa oculta		Precisión			
capa oc.1	capa oc.2	Tipo A	Tipo B	Tipo C	Tipo D
1	2	0.9138	0.9138	0.9483	0.7931
1	8	0.9138	0.9655	0.9310	0.8793
2	1	0.9310	0.9138	0.9138	0.9655
2	5	0.8966	0.8793	0.8966	0.8966
3	4	0.8966	0.8966	0.9310	0.8793
3	6	0.8793	0.9310	0.9310	0.9138
4	7	0.8966	0.9138	0.9138	0.9655
4	8	0.8966	0.8966	0.9483	0.9655
5	3	0.9138	0.9655	0.9138	0.9310
5	4	0.9138	0.8276	0.9310	0.9310
6	1	0.9138	0.8276	0.9138	0.9310
6	2	0.8793	0.9310	0.8966	0.9310
7	2	0.8966	0.9655	0.9138	0.9310
7	7	0.8793	0.9655	0.9483	0.8793
8	1	0.8966	0.9310	0.8793	0.9138
8	9	0.9138	0.9138	0.9310	0.9483
9	5	0.8966	0.9310	0.8966	0.9310
9	9	0.8793	0.9655	0.8966	0.9310

Cuadro 4.6: Tabla con la precisión para las distintas simulaciones sobre las redes de salidas *tipo I* para dos capas ocultas. La tabla se ha obtenido a partir de los resultados de las métricas calculadas con los aciertos y errores en el conjunto de prueba para los distintos tipos de entradas y según el número de unidades de la capa oculta. En la tabla se expone una pequeña muestra de los resultados obtenidos.

4.5.3. Predicción de aprobados/suspensos y tipo de refuerzo (salida múltiple)

En este caso se da una mejora aparente como, se puede apreciar en la tabla 4.8, ya que con una capa oculta apenas se llegaba a una precisión del 89 % para los *tipos C* y *D*, en cambio al considerar una segunda capa oculta hay valores de la precisión que llegan al 93 % en estos mismos tipos, y en concreto un 94 % para configuraciones de unidades de las capas oculta (4, 7), (2, 8) y (9, 2) del *tipo D*.

Al igual que ocurría cuando se predecía en la misma salida los aprobados/suspensos con el refuerzo final, hay configuraciones, según el número de unidades de las capas ocultas, mejores según el tipo de entradas.

4.5.4. Predicción de nota final

Al aumentar las capas ocultas aparecen una serie de resultados extraños, como se observa en la tabla 4.9, en concreto tres valores del coeficiente r^2 negativos, dos para redes con una configuración de capas ocultas de (1, 2) y otro para una configuración (6, 3). En el caso de las entradas donde se considera el identificador de

Unidades capa oculta		Precisión			
capa oc.1	capa oc.2	Tipo A	Tipo B	Tipo C	Tipo D
1	1	0.7931	0.7931	0.9828	0.8793
1	9	0.7414	0.6897	0.9828	0.9828
2	2	0.7586	0.7586	0.9483	0.9828
2	6	0.7931	0.7759	0.8966	0.8276
3	3	0.7586	0.7931	0.9828	0.8793
3	4	0.7931	0.7759	0.8966	0.8793
4	5	0.7759	0.7759	0.9310	0.9483
4	7	0.7586	0.8103	0.9138	0.9310
5	2	0.8103	0.7759	0.9310	0.9138
5	8	0.7586	0.7586	0.9310	0.8966
6	3	0.7931	0.7241	0.9655	0.9310
6	9	0.7931	0.7414	0.8793	0.8966
7	3	0.8103	0.6552	0.9310	0.9138
7	4	0.7414	0.7069	0.9138	0.8966
8	4	0.6897	0.7241	0.8966	0.9310
8	5	0.7586	0.7414	0.9310	0.9483
9	6	0.7414	0.7586	0.9138	0.9483
9	7	0.7586	0.7586	0.8793	0.8966

Cuadro 4.7: Tabla con la precisión para las distintas simulaciones sobre las redes de salidas *tipo II* para dos capas ocultas. La tabla se ha obtenido a partir de los resultados de las métricas calculadas con los aciertos y errores en el conjunto de prueba para los distintos tipos de entradas y según el número de unidades de la capa oculta. En la tabla se expone una pequeña muestra de los resultados obtenidos.

grupo para la configuración de 1 unidad en la primera capa oculta y 2 en la segunda, se debió producir algún error a la hora del entrenamiento de esta configuración, ya que en todos los registros se obtuvo la misma predicción, de ahí que el ajuste hay sido pésimo. En la segunda configuración se dio para el *tipo C*, no se da el hecho anterior, y los valores predichos no distan mucho, salvo ciertas excepciones que lo hacen considerablemente. Las notas aclaratorias de la página web de la librería **Scikit-Learn** avisan que se puede dar que el coeficiente r^2 en la regresión presente valores negativos, sin dar más explicaciones.

Un dato importante es que los valores negativos se han reducido a uno con la incorporación de una segunda capa oculta. Además mejora ligeramente los ajustes en general en todas las configuraciones, si exceptuamos los valores negativos. Pero al igual que en los resultados previos para una capa oculta, no se aprecia una tendencia o patrón para la mejora o empeoramiento de los resultados según se aumente o disminuya el número de unidades de capa oculta, y como se ha dicho anteriormente, hay configuraciones más adecuadas que otras según el tipo de entradas.

Los errores cuadráticos medios cometidos al ajustar los datos con una configuración de dos capas ocultas se muestran en la figura 4.11. Se divide en cuatro

Unidades capa oculta		Precisión			
capa oc.1	capa oc.2	Tipo A	Tipo B	Tipo C	Tipo D
1	1	0.7586	0.7414	0.8276	0.8621
1	2	0.7586	0.7759	0.8276	0.8966
2	5	0.7069	0.7069	0.8103	0.8621
2	8	0.7069	0.7414	0.8793	0.9483
3	4	0.7414	0.7414	0.8621	0.8966
3	7	0.6897	0.7069	0.8448	0.9138
4	7	0.6897	0.7069	0.9138	0.9483
4	9	0.7414	0.6724	0.8621	0.8793
5	3	0.7759	0.7241	0.8966	0.8621
5	8	0.7241	0.7586	0.8793	0.8448
6	3	0.7586	0.7241	0.8448	0.8621
6	9	0.6897	0.7414	0.8793	0.9138
7	2	0.7241	0.7069	0.8621	0.8621
7	3	0.7241	0.7241	0.8793	0.8621
8	5	0.7069	0.7241	0.8621	0.9138
8	9	0.7069	0.7241	0.8276	0.8793
9	2	0.7069	0.7586	0.8448	0.9483
9	7	0.6897	0.6897	0.8966	0.8621

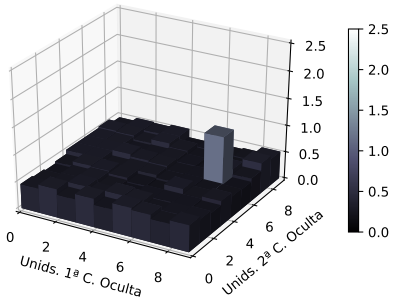
Cuadro 4.8: Tabla con la precisión para las distintas simulaciones sobre las redes de salidas *tipo IV* para dos capas ocultas. La tabla se ha obtenido a partir de los resultados de las métricas calculadas con los aciertos y errores en el conjunto de prueba para los distintos tipos de entradas y según el número de unidades de la capa oculta. En la tabla se expone una pequeña muestra de los resultados obtenidos.

subfiguras compuesta por un diagrama de barras tridimensional para cada uno de los tipos de entradas, en los que se muestra el valor del error en función de las unidades en cada capa oculta. El *tipo A* sigue dando valores razonablemente buenos en comparación con los errores de una única capa oculta, salvo para la configuración (8, 6) que supera ampliamente al resto, lo normal es que los valores estén entre 0.37 alto y 0.58 que es la variabilidad que se daba para una capa. En cuanto al *tipo B* la variabilidad es parecida, pero hay una serie de configuraciones que superan al resto: el pico que se ve en (1, 2) y su continuación en (2, 2) donde baja ligeramente. Si bien ofrecía errores bajos con configuraciones donde el número de unidades de la capa oculta eran bajas, con dos capas, si exceptuamos los dos valores extremos del análisis, no se ve este este comportamiento con una configuración de unidades de ambas capas bajas, ni con otras combinaciones. En el *tipo C* se ven dos picos altos, como en el anterior (aunque en *tipo B* estaban al lado), dándose fundamentalmente para configuraciones donde las unidades de capa oculta son valores altos en la primera capa y bajos para la segunda fundamentalmente. Si exceptuamos estos valores extremos, la variabilidad es parecida a los anteriores tipos. Comparando con el rendimiento con una capa, el *tipo C* daba los peores resultados con un número

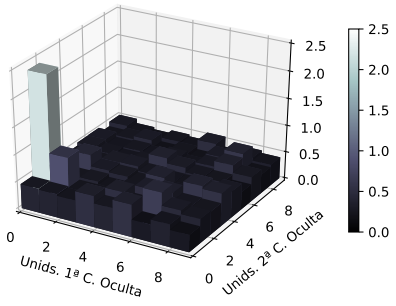
Unidades capa oculta		Coeficiente r^2			
capa oc.1	capa oc.2	Tipo A	Tipo B	Tipo C	Tipo D
1	2	0.8050	-0.0749	0.7648	-0.0749
1	7	0.7670	0.7666	0.7676	0.2182
2	1	0.7593	0.8018	0.8062	0.7827
2	7	0.7711	0.8085	0.7891	0.7814
3	1	0.8052	0.8148	0.7828	0.7639
3	5	0.8047	0.7622	0.6471	0.7999
4	5	0.7922	0.7583	0.7666	0.8162
4	6	0.7739	0.7947	0.7416	0.8076
5	3	0.8279	0.7990	0.7651	0.7886
5	9	0.8299	0.8214	0.7002	0.7928
6	3	0.8100	0.7449	-0.2346	0.8015
6	6	0.8149	0.7591	0.8001	0.7990
7	4	0.7646	0.8101	0.7920	0.4423
7	6	0.8081	0.7965	0.7540	0.8080
8	2	0.8006	0.7652	0.2430	0.8087
8	4	0.8066	0.7020	0.5951	0.8041
8	9	0.8136	0.7969	0.8203	0.7805
9	1	0.7751	0.8555	0.7872	0.8194
9	9	0.7695	0.8189	0.7738	0.8028

Cuadro 4.9: Tabla con el coeficiente de ajuste r^2 para las distintas simulaciones sobre las redes de salidas *tipo III* para dos capas ocultas. La tabla se ha obtenido a partir de los resultados de las métricas calculadas del ajuste de datos realizado sobre el conjunto de prueba para los distintos tipos de entradas y según el número de unidades de cada capa oculta. En la tabla se expone una pequeña muestra de los resultados obtenidos.

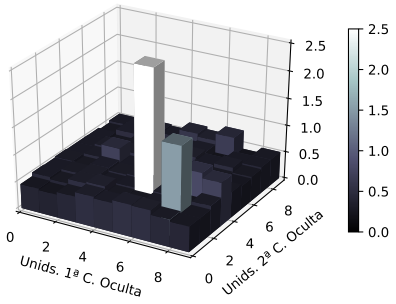
alto de unidades, 8 y 9 como se puede ver en 4.10, con dos capas parece empeorar con valores altos en la primera capa. Por último, el *tipo D*, que con una capa daba errores razonables salvo dos configuraciones, para dos capas le ocurre algo parecido, tiene tres configuraciones con un error muy grande, la que más configuraciones con error muy alto tiene, otras seis con un error más bajo, pero superior al resto, y la amplia mayoría que ofrece valores ligeramente más bajos que el *tipo A*. En general, se dan algunas configuraciones con errores desorbitados, pero el resto está bastante acotado. El incluir más información (identificador del curso y tipo de refuerzo) ayuda a mejorar, ligeramente, pero también aumenta el número de configuraciones con errores altos. Los errores, salvo configuraciones específicas, están acotados entre los valores 0.30 y 0.60.



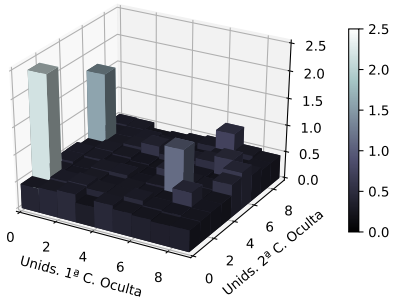
(a) *Tipo A*



(b) *Tipo B*



(c) *Tipo C*



(d) *Tipo D*

Figura 4.11: Error cuadrático medio calculado para el ajuste de regresión de los datos con una configuración de una salida (*Tipo III*) y dos capas ocultas para los distintos tipos de entradas. Los datos fueron extraídos a partir de la función `mean_squared_error` sobre el conjunto de valores de la salida esperada de prueba y se comparó con la predicción que ofreció `MLPRegressor` al conjunto de entrada de prueba. Además de con la altura de las barras, el valor del error está relacionada con con el color de estas: cuanto más oscuro, el valor del error es más bajo y cuanto más claro, más alto será el valor del error.

Capítulo 5

Conclusiones

Al comienzo del trabajo, uno de los objetivos era ver si se podía predecir el rendimiento académico de un grupo de alumnos de secundaria en un entorno educativo real, un colegio, a partir de datos recopilados *in situ*. Entre otros se recogieron notas medias de evaluaciones trimestrales, que corresponden a medias de exámenes presenciales, y un instrumento importante contemplado en la legislación educativa: las medidas de atención a la diversidad, que no incluyen los estudios mencionados en la introducción. En este caso se entiende diversidad como posibles dificultades de aprendizaje que pueda tener un alumno/a en una determinada asignatura, en este caso las Matemáticas de la etapa Secundaria. Conviene indicar la dificultad que supuso la recogida y codificación de estos datos, ya que, como dice el dicho popular "*cada maestrillo tiene su librillo*", cada docente tiene su manera de almacenar los datos referentes a las calificaciones de su alumnado.

Del análisis de los resultados se pueden extraer una serie de ideas interesantes. La primera es que se puede predecir de manera bastante precisa el rendimiento académico a partir de una serie de parámetros de entrada, como son las notas de dos evaluaciones, el tipo de refuerzo asignado (que da idea de las dificultades de aprendizaje del alumno/a) y el identificador del grupo. Es un conjunto muy pequeño de información ya que se podría tener en cuenta otros parámetros, en algunos casos subjetivos, como el comportamiento, la motivación o incluso la evaluación inicial del curso, que es una prueba que se realiza para comprobar el grado de asimilación de conceptos del curso anterior. En esencia, se buscaba un grupo, lo más reducido y representativo, de parámetros que reportasen buenos resultados, y en algunos casos, como en los aprobados/suspensos el conjunto mínimo de entradas dio resultados satisfactorios, en otros casos era necesario la inclusión de más parámetros para obtener resultados parecidos. La segunda idea es que el número de neuronas afecta al resultado, pero su variación no representa una tendencia clara, es decir, algunas configuraciones funcionaban mejor con un número bajo de unidades y en otras configuraciones los mejores resultados se obtenían con un número alto de unidades. La tercera idea es que la opción de la salida combinada no mejora las salidas individuales, más bien parece una combinación de ambas con un un nivel de predicción ligeramente inferior, siendo el factor que limitaba los resultados el refuerzo.

Los resultados obtenidos indican que se puede predecir/clasificar el aprobado de una manera bastante precisa, solo considerando las notas. Realmente, al ser una media, el peso de las primeras evaluaciones conjuntamente será superior al de la tercera (dos tercios a un tercio): aprobando las dos primeras lo lógico es que se apruebe la última, salvo descalabro. Incluir el identificador de grupo mejora la precisión, pero no sustancialmente. A la hora de predecir/clasificar el refuerzo, la precisión se reduce, en líneas generales. En este caso, incluir junto con las notas al refuerzo en la información de entrada no mejora los resultados, influye más el incluir el identificador de grupo.

Un resultado interesante, fue ver que cualquier diseño de red predice bien los casos extremos de aprobados y suspensos, visto en las figuras (4.2) y (4.9), parece que cualquier topología de red predice bien los aprobados cuando estos son claros (buenas notas), pero falla con los suspensos o cuando el registro tiene alguna evaluación suspensa. Si se divide la gráfica (4.2) donde se presentan las notas de primera y segunda evaluación en los ejes, en 4 regiones o cuadrantes, donde el límite entre regiones es la nota correspondiente al 5, se observa que el cuadrante superior derecho (los que han aprobado todas la evaluaciones) quedan bien clasificados como aprobados, concentrándose la mayoría de errores en el cuadrante inferior izquierdo, las distintas configuraciones de red captan que lo habitual es seguir cierta regularidad en las notas, por eso clasifica bien esta región, ya que como se comentó anteriormente, lo habitual es aprobar. Los casos límite donde hay una evaluación aprobada (los cuadrantes restantes) también acumulan errores pero en menor medida.

En la predicción del refuerzo no influyen las notas, que empeoran bastante la precisión, pero sí el incluir el refuerzo asignado previamente en las evaluaciones, incluso hay leve mejoría al incluir el identificador de grupo. Queda claro que las notas solas no dan indicios del tipo de refuerzo asignado desde principio de curso, hecho lógico, ya que el objetivo de un refuerzo siempre es que se supere la asignatura cuando hay una dificultad, ya sea de aprendizaje o bajada en el rendimiento, y su correcta asignación desde un principio provoca que el rendimiento mejore (al menos las notas). En este caso queda claro que cuanta más información se dé a la red, mejor será la predicción. En cuanto a los errores que cometieron los distintos tipos de redes para el refuerzo (4.5), se ve que, al igual que en el caso de los aprobados/suspensos, se acumulan en el cuadrante inferior izquierdo, donde están, los suspensos y los casos límite. También casos con una diferencia notable entre las notas de las evaluaciones fueron susceptibles de que la red cometiera un error.

El combinar las salidas de aprobados/suspensos y refuerzo final, como se comentó antes, no mejora la precisión obtenida por separado de estos dos tipos de salidas, los resultados se acercan más a los guarismos del refuerzo final como salida única, como se puede observar en las tablas 4.2 y 4.3. Al igual que con el refuerzo, el considerar un mayor número de unidades de entrada (dicho de otra forma, más información) incluyendo el refuerzo de las primeras evaluaciones y el identificador del curso mejora la predicción. En caso de tener una aplicación práctica, en una situación dentro del entorno educativo, sería este tipo de redes con dos salidas, las que se deberían emplear: los resultados no distan mucho de los obtenidos para las

salidas simples, y seguramente capte mejor la relación entre las notas y los refuerzos.

En cuanto a la predicción de notas numéricas, los valores obtenidos están un poco dispersos, aunque generalmente predice de manera adecuada la nota o se aproxima mucho; en otros casos extremos, por así llamarlos, ofrece resultados no compatibles con los estándares de puntuación aceptados, en concreto valores negativos. En algunos casos ofrece valores intermedios entre las notas de la primera y segunda evaluación, muy cercanos a la nota final, una posible explicación sería que la red neuronal "entiende" que es una media, y que hay un posible valor intermedio que propicia esa nota final. En el caso comentado de valores negativos, se da para valores muy diferentes de notas de primera y segunda evaluación, en concreto es mayor el valor de la primera evaluación, se podría explicar que ofrece este valor como resultado de ser una progresión de los valores de las notas de las dos primeras evaluaciones: si obtiene cada vez peores notas, al final se llega a obtener un valor negativo.

En cuanto al número de unidades en la capa oculta, no se ve un patrón o tendencia clara que indique que un mayor o menor número de neuronas mejore los resultados, simplemente hay configuraciones más adecuadas que otras según los tipos de las entradas y salidas. Incluir una segunda capa oculta no mejora los resultados en la predicción para los *tipos I* y *II* pero sí lo hace en el *tipo IV* (salida múltiple), en este caso quizás tenga sentido el admitir que si complicamos el modelo se obtenga un mejor resultado ya que una segunda capa oculta debería recoger más detalles de los parámetros de entrada y encontrar una predicción más precisa.

En líneas generales el tipo de resultados obtenidos en este trabajo se podrían extender a otras asignaturas que tengan los mismos criterios de refuerzo, como pueden ser las lenguas (castellano y las propias de las autonomías). Otras asignaturas, como las correspondientes al área de ciencias y también las lenguas extranjeras, podrían ser adecuadas para extender esta metodología, siempre y cuando hubiese una modificación a la hora de considerar el refuerzo, ya que en estos casos no se tendrían que dar todos los tipos incluidos en el trabajo.

Bibliografía

[1] Chun-Teck Lye, Lik-Neo Ng, Mohd Hassan, Wei Goh, Check-Yee Law, and Noradzilah Ismail. Predicting pre-university student ś mathematics achievement. *Procedia - Social and Behavioral Sciences*, 8:299–306, 12 2010.

[2] Pauziah M. Arsad, Norlida Buniyamin, and Jamalul-Lail Ab Manan. A neural network students' performance prediction model (nnsppm). In *2013 IEEE International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, pages 1–5, 2013.

[3] Anal Acharya and Devadatta Sinha. Early prediction of students performance using machine learning techniques. *International Journal of Computer Applications*, 107:37–43, 2014.

[4] Eng T. Lau, Li Sun, and Qingping Yang. Modelling, prediction and classification of student academic performance using artificial neural networks. *SN Applied Sciences volume*, 1, 2019.

[5] Sotiris Kotsiantis, Christos Pierrakeas, and P. Pintelas. Predicting students' performance in distance learning using machine learning techniques. *Applied Artificial Intelligence*, May–June 2004:411–426, 08 2010.

[6] Thai-Nghe Nguyen, Paul Janecek, and Peter Haddawy. A comparative analysis of techniques for predicting academic performance. In *2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, pages T2G–7, 11 2007.

[7] Cristobal Romero and Sebastián Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33:135–146, 2007.

[8] Félix Castro Espinoza, Alfredo Vellido, and Àngela Nebot. *Applying Data Mining Techniques to e-Learning Problems*, volume 62, pages 183–221. 07 2007.

[9] Surjeet Kumar, Brijesh Bharadwaj, and Saurabh Pal. Data mining applications: A comparative study for predicting student ś performance. *International Journal of Innovative Technology and Creative Engineering*, 1:13–19, 02 2012.

[10] Paulo Cortez and Alice Silva. Using data mining to predict secondary school student performance. *EUROSIS*, pages 5–12, 01 2008.

[11] Nasir Bhanpuri, Everaldo Aguiar, Himabindu Lakkaraju, David Miller, Ben Yuhas, and Kecia Addison. Who, when, and why: A machine learning approach to prioritizing students at risk of not graduating high school on time. 03 2015.

[12] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[13] Thomas Mitchell. *Machine Learning.* McGraw-Hill Series in Computer Science. McGraw-Hill Education, 1997.

[14] Fernando Berzal. *Redes Neuronales & Deep Learning.* Edición independiente, 2018.

[15] Christopher Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[16] Christopher Bishop. *Pattern Recognition and Machine Learning.* Springer, 2004.

[17] Richard O. Duda, David G. Stork, and Peter E.Hart. *Pattern Classification.* Wiley, 2001.

[18] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[19] Thomas M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, 1965.

[20] Marvin Minsky and Seymour A. Papert. *Perceptrons.* MIT Press, Cambridge, 1969.

[21] Terrence Sejnowski. *Unsupervised Learning, Foundations of Neural Computation.* MIT Press, 1999.

[22] Bonifacio Martín del Brio. *Redes Neuronales y Sistemas Borrosos.* Ra-Ma, 2006.

[23] John A. Hertz, Anders S. Krogh, and Richard G. Palmer. *Introduction To The Theory Of Neural Computation.* Santa Fe Institute Series. Westview Press, CRC, 1991.

[24] Berndt Müller, Joachim Reinhardt, and Michael T. Strickland. *Neural networks : an introduction.* Springer, 1995.

[25] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1 : Foundations*, volume 1. Bradford Books, 1986.

[26] James L. McClelland and David E. Rumelhart. *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 2. Psychological and Biological Models*, volume 2. Bradford Books, 1986.

[27] Donald Hebb. *The Organization of Behavior.* John Wiley and Sons, 1949.

[28] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[29] John J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092, 1984.

[30] David Amit. *Modeling Brain Function: The World of Attractor Neural Networks.* Cambridge University Press, 1992.

[31] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biologicla Cybernetics*, 43:59–69, 1982.

[32] Teuvo Kohonen. Analysis of a simple self-organizing process. *Biological Cybernetics volume*, 44:135–140, 1982.

[33] Teuvo Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Information Sciences (IS 8).* Srpinger-Verlag, 1989.

[34] José C. Principe, Neil R. Euliano, and W. Curt Lefebvre. *Neural and Adaptive Systems: Fundamentals through Simulations.* Wiley, 2000.

[35] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[36] Friedrich A. Hayek. *Sensory order: an inquiry into the foundations of theoretical psychology.* University of Chicago Press, 1952.

[37] William A. Clark and Belmont G. Farley. Generalization of pattern recognition in a self-organizing system. In *Proceedings of the March 1-3, 1955, Western Joint Computer Conference*, AFIPS '55 (Western), page 86–91, New York, NY, USA, 1955. Association for Computing Machinery.

[38] Paul Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Thesis (Ph. D.). Appl. Math. Harvard University.* PhD thesis, Harvard University, 01 1974.

[39] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[40] Robert Hecht-Nielsen. *Neurocomputing.* Addison-Wesley Publishing Company, 1990.

[41] John S. Denker, Daniel B. Schwartz, Ben S. Wittner, Sara A. Solla, Richard E. Howard, Lawrence D. Jackel, and John J. Hopfield. Large automatic learning, rule extraction, and generalization. *Complex Systems*, 5:877–922, 1987.

[42] Richard Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4:4–22, 1987.

[43] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[44] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[45] Guido van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[46] Travis E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[47] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[48] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[49] Jonh D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[50] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Aalexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences

from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Apéndice A

Funciones de activación

En este anexo se comentarán las funciones de activación más comunes [14]. Los modelos de neuronas utilizados en redes neuronales artificiales combinan sus entradas usando pesos que modelan sus conexiones sinápticas y, a continuación, le aplican a la entrada neta de la neurona una función de activación o transferencia. La entrada neta de la neurona recoge el nivel de estímulo que la neurona recibe de sus entradas y es esta función la que determina cuál es la salida de la neurona. Las funciones de activación se pueden clasificar en dos grandes grupos:

- Funciones de activación discretas. Solo puede tomar un conjunto finito de valores. Normalmente, se utilizan dos valores, por lo que se habla de neuronas binarias cuando la salida es 0 o 1, mientras que se habla de neuronas bipolares cuando su salida puede ser -1 o $+1$.
- Funciones de activación continuas. La salida de la neurona puede tomar cualquier valor dentro de un intervalo. Generalmente, el rango de ese intervalo está limitado, o bien al intervalo $[0, 1]$ o bien al intervalo $[-1, 1]$. Dentro de las funciones de activación continuas, se pueden utilizar tanto funciones lineales como funciones no lineales. Estas últimas son las más utilizada, ya que dotan a una red neuronal multicapa de su capacidad de aproximador universal.

A continuación se enumeran algunas funciones de activación comunes.

Función de activación lineal

Es la función de activación más sencilla para una neurona artificial:

$$y = f(z) = z \tag{A.1}$$

La característica principal de las neuronas lineales es que se se conectan varias capas de neuronas lineales en serie, el resultado final será equivalente a una única capa de neuronas lineales.

Desde un punto de vista formal, las capas de neuronas lineales actúan como simples clasificadores lineales, por lo que no resultan adecuadas para resolver problemas complejos por sí mismas.

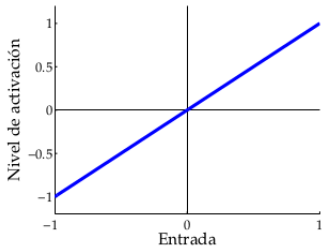


Figura A.1: Función de activación lineal. Fuente: [14].

Función escalón

Fue la primera función no lineal que se utilizó para modelar neuronas artificiales. Es también conocida como función umbral, limitador estricto o función de Heaviside. Es la función usada en el modelo neuronal de McCulloch-Pitts:

$$y = f(z) = \begin{cases} 1 & \text{Si } z \geq 0 \\ \\ 0 & \text{Si } z < 0 \end{cases} \tag{A.2}$$

Es una función de activación binaria, de la que se puede extraer una variante bipolar, la función signo:

$$y = f(z) = \begin{cases} 1 & \text{Si } z \geq 0 \\ \\ -1 & \text{Si } z < 0 \end{cases} \tag{A.3}$$

Siendo esta última, simétrica con respecto al origen.

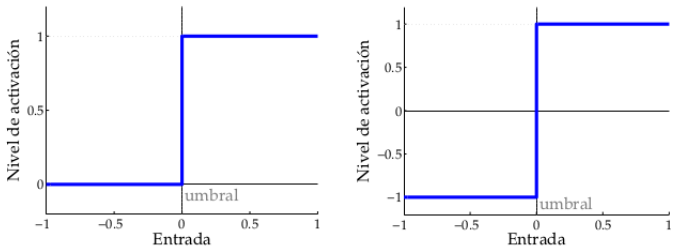


Figura A.2: Función de activación escalón. Fuente: [14].

Función lineal con saturación

Es una función lineal de activación a trozos. En su versión binaria:

$$y = f(z) = \begin{cases} 1 & \text{Si } z > \frac{1}{2} \\ \\ z + \frac{1}{2} & \text{Si } -\frac{1}{2} \geq z \geq \frac{1}{2} \\ \\ 0 & \text{Si } z < -\frac{1}{2} \end{cases} \tag{A.4}$$

Se puede derivar una función simétrica:

$$y = f(z) = \begin{cases} 1 & \text{Si } z > 1 \\ z & \text{Si } -1 \geq z \geq 1 \\ -1 & \text{Si } z < -1 \end{cases} \tag{A.5}$$

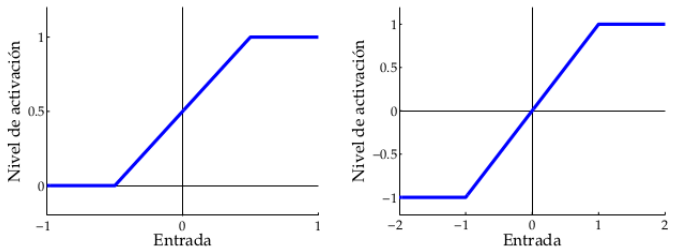


Figura A.3: Función de activación lineal con saturación. Fuente: [14].

Función de activación sigmoidal

Las funciones sigmoidales satisfacen requisitos como ser no lineal, estrictamente creciente, continua y derivable, lo que las hace especialmente útiles en redes neuronales que se entrenan usando el algoritmo ***backpropagation***. Como se comento en la introducción, dados los errores observados en la capa de salida, propaga estos hacia atrás en la red para ir ajustando sus parámetros internos. La forma en que se ajustan esos parámetros depende de la derivada de la función de activación de las neuronas. Existen distintas funciones sigmoidales, pero todas tienen en común su forma de "S". Algunas tienen propiedades matemáticas que las hacen especialmente interesantes para sus uso en redes neuronales artificiales. Algunos ejemplos de las más empleadas se muestran a continuación.

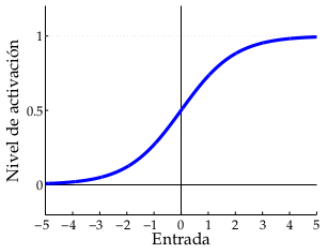


Figura A.4: Función de activación sigmoide, función logística. Fuente: [14].

Función logística Es una sigmoide binaria con rango en $[0, 1]$, su expresión:

$$y = f(z) = \frac{1}{1 + e^{-z}} \tag{A.6}$$

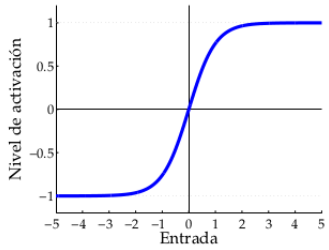


Figura A.5: Función de activación sigmoide, tangente hiperbólica. Fuente: [14].

Tangente hiperbólica Se emplea en trigonometría esférica y se define como:

$$y = f(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{A.7}$$

La tangente hiperbólica, según algunos autores, funciona mejor que la función logística al parecerse a la función identidad para valores de z cercanos a 0. El comportamiento de las neuronas con una función de activación como esta, es siimilar a las de las neuronas lineales, lo que puede facilitar el entrenamiento de la red.

Función de activación lineal rectificada

Las neuronas que emplean esta función son también llamadas *RELU* (*REctified Linear Units*). Estas unidades utilizan una función de activación lineal rectificada, que se define de la siguiente forma:

$$y = f(z) = \begin{cases} z & \text{Si } z \geq 0 \\ \\ 0 & \text{Si } z < 0 \end{cases} \tag{A.8}$$

En general, las unidades lineales rectificadas tienen una ventaja con respecto a las unidades sigmoidales cuando forman parte de una red que se entrena con algoritmos basados en ***backpropagation***. Las unidades sigmoidales se saturan a partir de cierto valor, lo que hace que la derivada de su función de activación sea prácticamente nula. En el entrenamiento basado en el gradiente descendente con ***backpropagation***, se utiliza el valor de esa derivada como parte del cálculo del gradiente del error. Ese gradiente es el que determina cómo modificar los pesos de la red. Cuando la derivada de la función de activación es virtualmente nula, la magnitud de las actualizaciones de los pesos será muy pequeña y el entrenamiento de la red avanzará muy lentamente. La saturación prematura de las neuronas sigmoidales puede ocasionar serios problemas en el entrenamiento de una red neuronal. No obstante, si se utiliza una función de coste o pérdida que compense esa saturación, sí se pueden seguir utilizando neuronas sigmoidales. También hay ocasiones en las que el uso de unidades *RELU* se descarta, al no estar su rango acotado, como sucede en redes recurrentes (para evitar una realimentación positiva, que haría que la red quedase fuera de control) o en modelos estocásticos (en los que las funciones de activación se emplean para determinar probabilidades, obviamente acotadas).

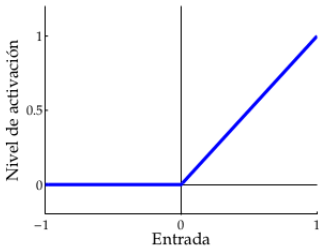


Figura A.6: Función de activación lineal rectificada. Fuente: [14].

Funciones de base radial

En ocasiones, puede resultar de interés que una neurona compare su vector de entradas con su vector de pesos y proporcione una salida en función de la similitud entre ambos. Es el caso de las funciones de base radial (*RBF* en inglés).

Una forma de medir la similitud entre el vector de entradas y el vector de pesos es calcular su distancia euclídea, también conocida como norma L^2 de la diferencia entre ambos vectores. Esto es lo que hacen las funciones de base radial de tipo gaussiano:

$$y = f(w, x) = e^{-\frac{1}{\sigma^2}||w-x||^2} \tag{A.9}$$

La salida de esa función es cero para la mayor parte de los valores de x, por lo que una red con elementos de este tipo puede ser difícil de entrenar utilizando técnicas basadas en el gradiente descendente.

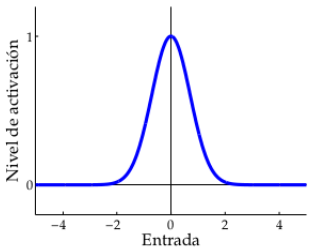


Figura A.7: Función de activación RBF de tipo gaussiano. Fuente: [14].

Apéndice B

Datos

A continuación se ofrecen un recuento sobre número de aprobados, suspensos y refuerzos dentro de los datos recogidos:

- Número total de registros: 191
- Número de refuerzos:
  - Sin refuerzo: 144
  - Refuerzo significativo: 32
  - Refuerzo fuera del aula: 15
- Número de aprobados: 169. De los cuales:
  - Aprobados sin refuerzo: 137
  - Aprobados con refuerzo: 32. De los cuales:
    - Refuerzo significativo: 23
    - Refuerzo fuera del aula: 9
- Número de suspensos: 22. De los cuales:
  - Suspensos sin refuerzo: 7
  - Suspensos con refuerzo: 15. De los cuales:
    - Refuerzo significativo: 9
    - Refuerzo fuera del aula: 6

Apéndice C

Código

A continuación se presenta un extracto del código empleado en el trabajo. Consiste en el ajuste de datos por medio del clasificador `MLPClassifier` en combinación con `MultiOutputClassifier` para una salida múltiple de aprobados/suspensos y tipo de refuerzo.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split as tts
from sklearn.metrics import score
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neural_network import MLPClassifier as mlpc
from sklearn.neural_network import MLPRegressor as mlpr
from sklearn.multioutput import MultiOutputClassifier as mlt

datos = pd.read_csv("archivo_datos.csv") # Carga de datos

# Eleccion de las entradas y salidas
X = datos.iloc[:,[5,6,9,10,11]].values    # Entradas
Y = datos.iloc[:,[14,15]].values          # Salidas

X_train, X_test, Y_train, Y_test = tts(X, Y, test_size=0.3,
                                       random_state=1)
n # unidades capa oculta
mlp = mlpc(hidden_layer_sizes=(n,),activation='logistic',
        solver='lbfgs',alpha=0.0001, max_iter=1000,
        learning_rate='constant',learning_rate_init=0.001)
mo = mlt(mlp)
mo.fit(X=X_train, y=Y_train)          # Ajuste parametros red
y_pred = mo.predict(X_test)           # Ajuste datos de prueba

# Precision del ajuste
print('Precision: %.4f' %score(Y_test,y_pred))
```

Primero se importan las librería necesarias, y se cargan los datos a partir de un archivo con la función `reac_csv` de la librería **Pandas**. Se eligen las entradas y salidas a partir de columnas específicas de la estructura de datos resultante. A partir de la función `train_test_split` se separa aleatoriamente el conjunto inicial de datos en cuatro:

- El conjunto de entradas de entrenamiento X_train
- El conjunto de salidas de entrenamiento Y_train
- El conjunto de entradas de prueba X_test
- El conjunto de salidas de prueba Y_test

El parámetro *test_size* indica que el conjunto de entrenamiento corresponde con el 30 % del conjunto inicial de datos. A continuación se inicializa un objeto `MLPClassifier` con los parámetros especificados en la metodología (y un número de unidades de capa oculta no especificado en este fragmento), además, se inicializa otro objeto `MultiOutputClassifier` (ya que el fragmento de código corresponde a una salida múltiple) con el anterior objeto como parámetro. A partir de este se pueden entrenar la red con las entradas y salidas de entrenamiento y una vez acabado se ajustan los datos de prueba y se comprueba la precisión de estos resultados con la función `score`.