UNIVERSIDAD POLITÉCNICA DE CARTAGENA

RESOLUCIÓN DE PROBLEMAS DE CLASIFICACIÓN CON DATOS INCOMPLETOS MEDIANTE REDES AUTOASOCIATIVAS PROFUNDAS

Tesis doctoral presentada por Adrián Sánchez Morales

Programa de Doctorado Tecnologías de la Información y Comunicaciones

Dirigida por el Dr. José Luis Sancho Gómez y el Dr. Aníbal R. Figueiras Vidal

Universidad Politécnica de Cartagena

DT-16

CONFORMIDAD DE SOLICITUD DEAUTORIZACIÓN DE DEPÓSITO DE TESIS DOCTORAL POR EL/LA DIRECTOR/A DE LA TESIS

D. José Luis Sancho Gómez y D. Aníbal R. Figueiras Vidal, Directores de la Tesis doctoral "Resolución de problemas de clasificación con datos incompletos mediante redes autoasociativas profundas".

INFORMA:

Que la referida Tesis Doctoral, ha sido realizada por D. Adrián Sánchez Morales dentro del Programa de Doctorado Tecnologías de la Información y Comunicaciones, dando mi conformidad para que sea presentada ante el Comité de Dirección de la Escuela Internacional de Doctorado para ser autorizado su depósito.

☒ Informe positivo sobre el plan de investigación y documento de actividades del doctorando/a emitido por el Director/ Tutor (**RAPI**).

La rama de conocimiento en la que esta tesis ha sido desarrollada es:

- ☐ Ciencias
- ☐ Ciencias Sociales y Jurídicas
- ☒ Ingeniería y Arquitectura

En Cartagena, a 19 de noviembre de 2019

EL/LA DIRECTOR DE LA TESIS

[redacted]

Fdo.: José Luis Sancho Gómez

EL/LA DIRECTOR DE LA TESIS

[redacted]

Fdo.: Aníbal R. Figueiras Vidal

COMITÉ DE DIRECCIÓN ESCUELA INTERNACIONAL DE DOCTORADO

Universidad Politécnica de Cartagena

DT-17

CONFORMIDAD DE DEPÓSITO DE TESIS DOCTORAL POR LA COMISIÓN ACADÉMICA DEL PROGRAMA

D/Dª. Jorge Larrey Ruiz, Presidente/a de la Comisión Académica del Programa Tecnologías de la Información y Comunicaciones.

INFORMA:

Que la Tesis Doctoral titulada, "Resolución de problemas de clasificación con datos incompletos mediante redes autoasociativas profundas", ha sido realizada, dentro del mencionado Programa de Doctorado, por D. Adrián Sánchez Morales, bajo la dirección y supervisión del Dr. D. José Luis Sancho Gómez y del Dr. D. Aníbal R. Figueiras Vidal.

En reunión de la Comisión Académica, visto que en la misma se acreditan los indicios de calidad correspondientes y la autorización del Director/a de la misma, se acordó dar la conformidad, con la finalidad de que sea autorizado su depósito por el Comité de Dirección de la Escuela Internacional de Doctorado.

☒ Evaluación positiva del plan de investigación y documento de actividades por el Presidente de la Comisión Académica del programa (**RAPI**).

La Rama de conocimiento por la que esta tesis ha sido desarrollada es:

- ☐ Ciencias
- ☐ Ciencias Sociales y Jurídicas
- ☒ Ingeniería y Arquitectura

En Cartagena, a 15 de noviembre de 2019

EL PRESIDENTE DE LA COMISIÓN ACADÉMICA

JORGE| LARREY| RUIZ

Firmado digitalmente por JORGE|LARREY|RUIZ
Fecha: 2019.11.15 10:23:50 +01'00'

Fdo: Jorge Larrey Ruiz

COMITÉ DE DIRECCIÓN ESCUELA INTERNACIONAL DE DOCTORADO

A mis padres y a Cristina.

Resumen

Hoy en día, prácticamente todas las aplicaciones en la industria explotan su información histórica para tomar decisiones y de esta forma realizar predicciones, optimizar procesos o simplemente monitorizar activos. Las técnicas de procesado de datos han sido ampliamente estudiadas durante los últimos años debido, entre otras cosas, al crecimiento de aplicaciones basadas en inteligencia artificial. Además, la presencia de valores desconocidos en un conjunto de datos es uno de los problemas más comunes en estas aplicaciones reales. Ésta es una de las razones por las que en la literatura se han propuesto muchas técnicas basadas en aprendizaje máquina que abordan esta tarea.

En la primera parte de este trabajo, se explota la gran capacidad de representación de los *Stacked Denoising Autoencoders* (SDAE) para obtener un nuevo método de imputación basado en dos ideas diferentes: borrado y compensación. El primer método ha demostrado mejorar los resultados en imputación borrando artificialmente algunas características y usándolas como etiquetas en el entrenamiento de la red. Sin embargo, aunque el borrado es realmente eficiente, puede causar un desbalanceo enrte la distribución de los datos de entrenamiento y test. Para solucionar esto, se propone un método de compensación basado en una ligera modificación de la función de error a optimizar. Se realizan experimentos sobre varios conjuntos de datos y se demuestra que el borrado y la compensación no sólo suponen mejoras en imputación en comparación con otras técnicas clásicas, sino también en clasificación.

Después, se propone proporcionar más información a un clasificador SDAE para mejorar su rendimiento. Más específicamente, se usa la salida de un clasificador auxiliar para extender la entrada de estas máqui-

nas, y llevar un entrenamiento capa a capa considerando la reconstrucción de la entrada y las etiquetas al mismo tiempo usando una combinación convexa. Esta red es llamada *Complete MSDAE* (CMSDAE). Se realizan también experimentos para apoyar la efectividad del modelo, demostrando que las máquinas resultantes ofrecen mejores resultados que los métodos estándares en todos los casos, así como reducen la sensibilidad del diseño de parámetros.

Finalmente, una vez demostrado que los mencionados clasificadores CMSDAEs ofrecen unos resultados de clasificación que son mejores que los de los propios MSDAEs, se ha investigado si los CMSDAEs pueden mejorar los mecanismos de imputación de los mismos. En la parte final de este trabajo, se consideran dos métodos diferentes de imputación con CMSDAEs. La primera resulta ser un método directo en el que la salida del CMSDAE es simplemente la etiqueta del conjunto. El segundo mecanismo surge a partir de la presencia de las etiquetas en el vector de salida y usa la técnica ampliamente conocida de aprendizaje multitarea (MTL), incluyendo las observaciones como tarea secundaria. Así, los resultados experimentales demuestran que estas estructuras CMSDAE incrementan la calidad de los valores imputados, en particular, en las versiones MTL.

Abstract

Nowadays, almost every industry application exploits the information of historical data to get useful insights and thus make predictions, optimize processes or simply monitorize assets. Data processing techniques have been widely studied for the last years due to the growth of artificial intelligence applications. Furthermore, missing values in a data set is one of the most common difficulties in these real applications. That is one of the reasons why many different techniques based on machine learning have been proposed in the literature to face this problem.

In the first part of this work, the great representation capability of the Stacked Denoising Auto-Encoders (SDAE) is used to obtain a new method of imputating missing values based on two ideas: deletion and compensation. This method has demonstrated to improve imputation performance by artificially deleting values in the input features and using them as targets in the training process. Nevertheless, although the deletion of samples is really efficient, it may cause an imbalance between the distributions of the training and the test sets. In order to solve this issue, a compensation mechanism is also proposed based on a slight modification of the error function to be optimized. Then, experiments over several datasets show that the deletion and compensation not only involve improvements in imputation but also in classification in comparison to other classical techniques.

Aterwards, we propose to provide more information to SDAE classifiers in order to increase their performance. Specifically, we use the output of an auxiliary classifier to extend the input to those machines, and carry out the layer-by-layer auto-encoding training considering the

input recovering and the label errors by means of a convex combination. This network is called *Complete MSDAE* (CMSDAE). Extensive experiments support the effectiveness of this proposal, showing that the resulting machines offer better results than standard designs in all the cases, as well as a reduced sensitivity to the design parameters.

Finally, once demonstrated that the mentioned CMSDAE classifiers offer classification results that are better than those provided by MSDAEs, it has been investigated if CMSDAEs can improve the MSDAEs imputation processes. In the final part of this work, two types of imputation mechanisms with CMSDAEs are considered. The first is a direct procedure in which the CMSDAE output is just the target. The second mechanism is suggested by the presence of the targets in the vectors to be auto-encoded, and it uses the well known Multi-Task Learning (MTL) ideas, including the observations as a secondary task. Thus, experimental results show that these CMSDAE structures increase the quality of the missing value imputations, in particular, the MTL versions.

Agradecimientos

Escribo estas líneas, una vez acabada la redacción de la tesis, con un cúmulo de emociones que difícilmente puedo describir con mis limitadas capacidades de escritura. Sólo los más allegados saben que estos cinco años han sido los más difíciles de mi vida y, en muchos sentidos, los más gratificantes. Durante esta aventura han sido muchos los altibajos, pasando del entusiasmo a la frustración en cuestión de días. Hoy, escribo con el orgullo de saber que lo más difícil está hecho, con la ilusión de seguir cumpliendo metas y hasta con un poco de tristeza porque sé que, en el fondo, hasta lo voy a echar de menos.

Quiero agradecer a mis directores, el Dr. D. José Luis Sancho Gómez de la Universidad Politécnica de Cartagena y el Dr. D. Aníbal R. Figueiras Vidal de la Universidad Carlos III de Madrid, su apoyo y confianza durante estos años. A este último, por ser una fuente de sabiduría y un ejemplo de excelencia. Al primero, no sólo por esto, sino también por estar ahí siempre y mostrarme la parte optimista de las cosas.

Por otro lado, quiero agradecerle a Cristina, mi compañera de vida, su apoyo incondicional y su paciencia durante todos estos años. Finalmente, hacer mención especial a mis padres. Gracias por haberme enseñado todo lo que sé de la vida y por haber dado todo para que yo hoy esté donde estoy.

A todos, gracias.

Adrián

Índice general

1 Introducción **1**

1.1 Introducción a los datos incompletos 1

1.2 Análisis de datos incompletos 3

1.2.1 Distribución de datos incompletos 3

1.2.2 Métodos para tratar datos incompletos 5

1.2.3 Objetivos teóricos de la imputación 6

1.3 Aprendizaje máquina y aprendizaje profundo 7

1.4 Redes autoasociativas profundas 12

2 Estado del Arte **17**

2.1 Imputación 18

2.2 Imputación mediante algoritmo EM 20

2.2.1 Marco formal de inferencia basada en muestras completas 21

2.2.2 Algoritmo EM 23

2.3 Técnicas de imputación estadística 25

2.3.1 Imputación a la media/mediana/moda 25

2.3.2 Método de imputación *Hot-deck* 26

2.3.3 Imputación por descomposición en valores singulares (SVD) 26

2.3.4 Imputación mediante regresión 27

2.3.5 Imputación múltiple 28

2.3.5.1 Imputación múltiple mediante ecuaciones encadenadas (MICE) 28

2.4 Técnicas de imputación basadas en aprendizaje máquina 29

2.4.1 Imputación mediante K vecinos más cercanos (KNN) . . . 30

2.4.2 Imputación mediante árboles de decisión 30
2.4.3 Imputación con un Perceptrón Multicapa (MLP) 32
2.4.4 Imputación basada en aprendizaje profundo 33
2.5 Aprendizaje multitarea (MTL) 35
2.5.1 Conceptos básicos 35
2.5.2 MTL en aprendizaje profundo 37
2.5.2.1 Trabajos recientes 37
2.6 Discusión 39

3 Mejora de una imputación con redes profundas 41
3.1 Imputación mediante SDAEs 42
3.1.1 Algoritmo de Retropropagación 43
3.1.2 Pre-imputación 45
3.1.3 Borrado 46
3.1.4 Compensación 47
3.2 Clasificación de patrones incompletos 50
3.3 Experimentos 51
3.3.1 Conjuntos de datos 51
3.3.2 Métodos de pre-imputación 54
3.3.3 Resultados en problemas de imputación 54
3.3.4 Resultados de clasificación 57
3.4 Discusión 57

4 SDAEs modificados completos 63
4.1 CMSDAEs para clasificación 64
4.1.1 Metodología 65
4.1.2 Experimentos 68
4.1.2.1 Bases de datos 68
4.1.2.2 Máquinas 69
4.1.2.3 Entrenamiento 70
4.1.2.4 Resultados 71
4.1.3 Discusión 71
4.2 CMSDAEs para clasificación de patrones incompletos 74
4.2.1 Metodología 75

4.2.2 Experimentos 78
4.2.3 Discusión de resultados 79

5 Conclusiones y líneas futuras **81**
5.1 Contribuciones 81
5.2 Líneas futuras 83

A Publicaciones **87**
A.1 Artículos de revista 87
A.2 Contribuciones a congresos 88

Bibliografía **89**

No existen preguntas sin respuesta,
sólo preguntas mal formuladas.

Matrix

CAPÍTULO

1

Introducción

1.1 Introducción a los datos incompletos

A pesar de su corta edad, el siglo XXI ya es denominado como la "era de la información" debido a la velocidad de vértigo a la que se han creado y desarrollado las tecnologías digitales de la información y las comunicaciones. Estos avances y, sobre todo la rápida evolución de internet, han permitido la aparición de conceptos hasta hace no mucho tiempo inimaginables tales como el Internet de las Cosas (*Internet of Things*, IoT).

El IoT supone la interconexión de dispositivos y objetos en general a través de una red, bien privada o el propio Internet, de manera que puedan interaccionar entre todos. Las aplicaciones a las que está orientado el IoT son casi infinitas ya que prácticamente se puede implementar en cualquier objeto o dispositivo. Estas aplicaciones van desde control del tráfico en las ciudades pasando por la geolocalización del ganado perteneciente a una ganadería hasta el control de las fechas de caducidad de los alimentos que se encuentran dentro del frigorífico de cualquier hogar.

Esta gran innovación tecnológica ha hecho que el campo del procesado de datos sea enormemente explotado en los últimos años. Además, la cantidad de información que es diariamente producida ha provocado que sea necesario el desarrollo de

nuevos mecanismos para obtener información útil a partir de los datos. Todo ello supone una revolución aún mayor que la que causó la invención de la imprenta de manos de Gutenberg allá por 1450.

Disponer de tanta información permite la optimización de procesos en cuanto a tiempos y costes así como trabajar de una forma mucho más ordenada y sistematizada. Sin embargo, para poder emplear dicha información de forma beneficiosa es necesario recolectar y tratar correctamente el nivel más bajo de la misma, los datos.

Considerados el recurso estratégico y comercial por excelencia del siglo vigente, los datos han de ser correctamente inspeccionados, limpiados y transformados para que, en conjunto, se pueda resaltar la información útil y servir de apoyo en la toma de decisiones. Toma una gran importancia en este contexto el reconocimiento de patrones [1, 2], un campo del Aprendizaje Máquina (*Machine Learning*, ML) que consiste en la detección de patrones de las señales de entrada, obtenidos a partir de los procesos de segmentación y extracción de características. El punto esencial del reconocimiento de patrones es la regresión o clasificación: se quiere estimar o clasificar una señal dependiendo de sus características.

Generalmente, la extracción de información útil a partir de los datos se realiza mediante el empleo de diferentes algoritmos matemáticos y durante la fase del análisis de datos, previa al desarrollo de los algoritmos de clasificación mencionados. Es en este momento donde se afronta el problema de los valores perdidos o datos incompletos, también llamado *missing values* en la literatura anglosajona. Básicamente, este problema consiste en la pérdida de ciertos valores en una o varias de las características del conjunto de datos procesado y supone uno de los principales escollos presente en todos los problemas reales.

Siguiendo con el ejemplo anterior, el uso de sensores en procesos industriales es uno de los ejemplos más claros de presencia de valores perdidos durante el proceso de adquisición de datos. Fallos eléctricos y/o mecánicos, desconexión de la alimentación eléctrica o interferencias son algunos de los motivos por los que se produce este inconveniente de valores perdidos. La fuente de la pérdida de información es importante ya que el conjunto tiene diferentes efectos sobre los métodos de clasificación. Por tanto, al tratarse de un problema ampliamente extendido, se verá más adelante cómo existen en la literatura numerosas formas de abordarlo.

1.2 Análisis de datos incompletos

Esta tesis doctoral se centra en el estudio de datos incompletos y en nuevos métodos capaces de clasificar patrones con valores desconocidos. Se considera por tanto en todo momento un conjunto de datos de la forma $\mathbf{X} = \{\mathbf{x}_n, \mathbf{t}_n\}, n = 1, ..., N$, donde $\mathbf{x}_n = \{x_{ni}\}_{i=1}^{i=d} \in \mathbb{R}^d$ es el patrón n-ésimo y $\mathbf{t}_n$ su etiqueta correspondiente. En general, $\mathbf{X}$ estará compuesto por patrones completos e incompletos, es decir, $\mathbf{X} = \{\mathbf{x}_n^C, \mathbf{x}_n^I\}$ con $\mathbf{x}_n^C$ como patrones completos y $\mathbf{x}_n^I$ como patrones incompletos del conjunto. Además, es necesario incluir una matriz indicadora de valores perdidos M con la forma

$$m_{nd} = \begin{cases} 1, & x_{ni}\ es\ conocido, \\ 0, & x_{ni}\ es\ ausente, \end{cases} \tag{1.1}$$

donde i $(1 \leq i \leq d)$ indica el componente del vector.

De esta forma, un problema de clasificación, teniendo en cuenta el conjunto de datos incompleto, queda definido de la forma:

$$\mathbf{Z} = \{\mathbf{X}, \mathbf{M}, \mathbf{T}\} = \{\mathbf{x}_n, \mathbf{m}_n, \mathbf{t}_n\}_{n=1}^{N} \tag{1.2}$$

A pesar de las numerosas opciones que existen para el tratamiento de valores perdidos, se tiende a infravalorar el efecto de no adoptar ninguna acción sobre ellos y este hecho conlleva graves consecuencias como la aparición de sesgos inaceptables. Por este motivo, para saber cómo hacer frente a este problema, es necesario conocer antes cómo se genera esta pérdida de datos.

1.2.1 Distribución de datos incompletos

Tal y como se presenta en [3], no todos los valores perdidos muestran el mismo comportamiento. La variable binaria $\mathbf{M}$ debe analizarse como un fenómeno estocástico [4], por lo que debe considerarse como variable aleatoria con distribución de probabilidad conjunta, la cual da cuenta del porcentaje de omisión existente y de su relación con las observaciones completas. Las tipologías de datos faltantes propuestas por Rubin en [3] es ampliamente utilizada en la literatura.

Si se define $\mathbf{X} = \{\mathbf{X}^C, \mathbf{X}^I\}$ donde $\mathbf{X}^C$ y $\mathbf{X}^I$ son los conjuntos de valores completos e incompletos respectivamente, se define la primera categoría MAR (*Missing*

at Random) cuando los datos omitidos se generan de forma completamente aleatoria. Es decir, en este caso la distribución de los valores observados no depende del patrón de comportamiento de los valores desconocidos, $P(\mathbf{M}/\mathbf{X}) = P(\mathbf{M}/\mathbf{X}^C)$.

Bajo esta suposición, la probabilidad de que un valor presente missing depende solamente de la información disponible, es decir, los valores perdidos tienen dependencia de otras variables del conjunto de datos, generalmente variables independientes. Es frecuente estudiar este mecanismo como una regresión logística, donde la variable de salida es 1 para los casos observados y 0 para los valores perdidos. Los valores perdidos MAR pueden ser excluidos siempre y cuando la regresión controle todas las variables que afectan la probabilidad de pérdida.

Un ejemplo de ello podría ser una encuesta realizada a unos alumnos, en la que aquellos que superan cierta nota reciben preguntas adicionales. En este caso, algunos tendrán valores desconocidos en dichas preguntas, mientras que otros no, dependiendo de las primeras variables.

Otra opción es que los datos perdidos sigan una distribución completamente aleatoria o MCAR (*Missing Completely at Random*). En este caso, los datos perdidos no dependen de los observados: $P(\mathbf{X}/\mathbf{X}^C) = P(\mathbf{M})$. Así, una variable pertenece a esta categoría si la probabilidad de pérdida es la misma para todas las variables. En este caso, los valores perdidos no tienen dependencia alguna de sí mismos ni de otros atributos. Por otro lado, si todos los valores perdidos del conjunto de datos son MCAR pueden ser descartados.

Dicho de otra manera, las observaciones con datos perdidos son una muestra aleatoria del conjunto de observaciones. Un ejemplo de este tipo de valores perdidos podría ser un tubo que se rompe por accidente en un análisis de sangre a una serie de pacientes.

Finalmente, a un proceso que no es ni MAR ni MCAR se le denomina MNAR (*Missing not at Random*). MNAR significa que la falta de respuesta no puede ser ignorada en el proceso de construcción del estimador ni al analizar las relaciones de causalidad entre variables. Dentro de esta categoría se incluyen aquellos valores perdidos que en sí son considerados los causantes de su propia pérdida. El caso de valores NMAR debe modelarse explícitamente o, de lo contrario, se debe aceptar algún sesgo en sus inferencias. Un ejemplo de este caso sería un dato faltante en una encuesta debido a una pregunta que no ha sido respondida intencionadamente.

Además de la clasificación anterior, en líneas generales los valores perdidos pueden ser clasificados como ignorables y no ignorables. Los valores perdidos ignorables son aquellos que pueden ser excluidos del conjunto de datos de entrada y, posteriormente, proceder a su clasificación sin que se modifique la distribución de los datos. Por el contrario, los valores perdidos no ignorables han de ser tratados de cara a la clasificación puesto que su efecto es significativo en la inferencia de los datos.

Es importante tener en cuenta que la distribución de datos faltantes es significativa en algunos de los métodos que abordan el problema. De hecho, en la mayoría de las técnicas de imputación clásicas se asume de forma implícita o explícita la hipótesis de ignorabilidad. Esta hipótesis afirma que el mecanismo de datos incompletos es ignorable si se verifica la hipótesis MAR y los parámetros de la distribución son distinguibles. Se recomienda ir a los trabajos [5, 6] para más información sobre esta hipótesis. A continuación se presentan las principales maneras de abordar el problema de los datos perdidos.

1.2.2 Métodos para tratar datos incompletos

Como se ha comentado, la falta de datos en un conjunto reduce la representatividad de la muestra y puede distorsionar las inferencias sobre la población. A lo largo de la literatura, las soluciones habituales a la hora de afrontar un problema con valores perdidos pueden clasificarse como se presenta a continuación. En general, son bastante intuitivas y algunas de ellas funcionan bien en casos con cantidades pequeñas de datos perdidos.

- **Análisis de casos completos.** Un primer enfoque consiste en entrenar los clasificadores sólo con muestras completas. Como su nombre indica, se descartan los casos que presentan al menos un valor perdido y, por tanto, se realiza la inferencia sobre los valores completos. Con esto, se reduce la muestra a estudiar por lo que, dependiendo de la cantidad de información perdida, el mecanismo de datos faltantes y la relación entre casos completos e incompletos, las consecuencias pueden ser diferentes. En general, la pérdida de información relevante producirá un sesgo y falta de precisión en la inferencia final. Por contra, destaca por su simplicidad y el hecho de que todos los

estadísticos se calculan utilizando el mismo tamaño muestral, lo que permite su comparación [7].

- **Análisis de casos disponibles.** Métodos que toman toda la información disponible, evitando la distorsión que resulta tener posibles valores imputados erróneos y tomados como observados. Con este tipo de técnicas sólo se pierde la información no observada del conjunto. Así, generalmente se trabaja con diferentes tamaños muestrales e incluso se combinan en el cálculo de un mismo estadístico. Como se indica en [5], de este modo es posible que se obtengan correlaciones fuera del intervalo $[-1, 1]$ o matrices de correlaciones no definidas positivas, condición requerida en diversas técnicas multivariantes.

- **Imputación.** Los métodos de imputación pretenden solucionar el problema de los datos faltantes sustituyendo los mismos por valores estimados a partir de la información suministrada por la muestra. Con esto se consigue una matriz de datos completos y solucionan algunos de los problemas mencionados en los casos anteriores. Éste es el procedimiento más extendido para el tratamiento de valores perdidos. El motivo de su uso se debe a que la mayoría de herramientas de toma de decisiones no pueden utilizar datos incompletos directamente para la clasificación. Además, se produce una mejora durante la clasificación puesto que la imputación extrae información adicional (los propios valores imputados). Aunque en los siguientes capítulos se entrará en más detalles sobre los métodos de imputación, es importante destacar aquí que el tipo de técnica utilizada será clave a la hora de obtener mejores o peores resultados en la inferencia final.

1.2.3 Objetivos teóricos de la imputación

Independientemente de la presencia o no de datos incompletos, se acepta que el objetivo del análisis estadístico es generar inferencia válida [4]. No se trata únicamente de obtener estimadores insesgados y de mínima varianza, ni tampoco ajustar modelos para sustituir de cualquier forma la información faltante. Por ejemplo, la sustitución de información faltante a partir de promedios puede ser adecuada para

lograr predicciones más precisas, pero esta práctica tiene implicaciones negativas en la varianza del estimador e introduce distorsiones en el patrón de correlación de los datos [8].

La imputación debe considerarse parte del proceso de investigación con el propósito de obtener conclusiones sustentadas en evidencia empírica sólida. Así, generalmente no debe evaluarse la bondad de un método de imputación sólo por su capacidad de completar información y probar hipótesis. Algunos métodos de evaluación de un modelo se presentan en [9, 10] y, en general, están relacionados con el error cuadrático medio y no con el sesgo del estimador.

Sin embargo, esto es algo que debe tenerse en cuenta a la hora de crear el estimador. Si se hace una buena imputación, éste será cercano al verdadero valor de los datos perdidos, minimizando el sesgo y la varianza. En muchas ocasiones, especialmente en las técnicas clásicas, es común establecer supuestos acerca de las causas que generaron las omisiones, contrastando la factibilidad de las hipótesis con el comportamiento observado de los datos. Además, es importante tener claro el objetivo final a resolver para la elección del método de imputación. Esto es así porque, si la variable incompleta resulta de gran importancia a la hora de resolver la tarea principal –clasificación o regresión–, pequeñas diferencias en los datos imputados pueden conllevar cambios significativos en los resultados finales. Así, por ejemplo, supongamos una encuesta con la que se pretende saber el nivel de vida de algunas familias y existe un dato con valores faltantes que es el ingreso per cápita de la familia. De esta forma, si se usa un método de imputación para rellenarlo, es importante saber que cualquier variación del valor imputado será relevante en la estimación final.

1.3 Aprendizaje máquina y aprendizaje profundo

La Inteligencia Artificial (IA) se define como la simulación de procesos de inteligencia humana (aprendizaje, razonamiento y autocorrección) por parte de máquinas, generalmente sistemas informáticos. Este término, acuñado por John McCarthy en el año 1956, ha experimentado un gran auge en estos últimos años debido, entre otras cosas, a los grandes volúmenes de datos que se manejan en la actualidad. A

día de hoy, la IA abarca desde la automatización de procesos hasta la robótica. También está presente en prácticamente todos los ámbitos de la vida cotidiana aunque, generalmente, pasa desapercibida: asistentes personales virtuales, navegadores, interpretación de pruebas médicas, etc. Como tal, la IA es un campo general dentro del que se engloba el aprendizaje máquina (*Machine Learning*, ML), también conocido como aprendizaje automático, y el aprendizaje profundo (*Deep Learning*, DL), que a su vez es un subcampo del anterior (Figura 1.1).

Los primeros programas de IA, hasta finales de los 80, se basaban en conjuntos de reglas rígidas creadas previamente por los programadores, es decir, órdenes que ejecutar. Este hecho reducía los campos de aplicación de la IA, limitando su uso a la resolución de problemas lógicos y bien definidos, quedando obsoletos para tareas tales como la clasificación de imágenes, la traducción de lenguaje natural o el reconocimiento de voz. Ante la necesidad de resolver estos problemas más complejos surgió el denominado Aprendizaje Máquina.

El término aprendizaje máquina se puede definir como el subcampo de la IA que tiene por objetivo el desarrollo de técnicas que permitan que los ordenadores aprendan. En base a lo anterior, se trata de un proceso de inducción de conocimiento, puesto que el objetivo es generalizar comportamientos a partir de datos ejemplos o experiencia previa.

El aprendizaje automático se basa en el uso de algoritmos complejos que posibilitan el análisis de grandes cantidades de datos, el establecimiento de patrones y la realización de predicciones en base a los datos recogidos con anterioridad. La amplia gama de aplicaciones que tiene el aprendizaje automático incluye la detección de fraude en el uso de tarjetas de crédito, el diagnóstico médico y el reconocimiento del habla o del lenguaje escrito, entre otros.

A pesar de la gran capacidad de resolución que tienen este tipo de algoritmos, muchos de ellos tienen una capacidad de aprendizaje finita, independientemente de la cantidad de datos que adquieran. Es por ello que surgen los sistemas de aprendizaje profundo, los cuales permiten mejorar su rendimiento cuando se procesa un mayor número de datos y, consecuentemente, la máquina adquiere más experiencia. En todo caso, existe una clasificación de los algoritmos en función de su aprendizaje:

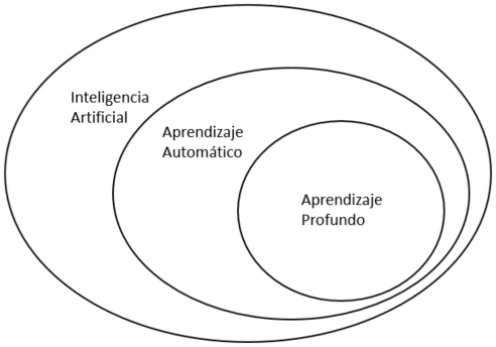


Figura 1.1: Aprendizaje Profundo como subcampo de la Inteligencia Artificial.

- **Aprendizaje supervisado.** En este caso, se entrena al algoritmo otorgándole las observaciones, denominadas características, y las respuestas, denominadas etiquetas. Esto se realiza con el fin de que el algoritmo combine las características con sus etiquetas y pueda realizar predicciones. Existen, a su vez, algoritmos de regresión o clasificación, si se enseña al algoritmo a obtener un número específico o predecir una clase, respectivamente.

- **Aprendizaje no supervisado.** A diferencia del anterior, aquí sólo se le otorgan las características al modelo, sin proporcionarle ninguna etiqueta. Su función es la agrupación, por lo que el algoritmo debería catalogar por similitud y poder crear grupos, sin tener la capacidad de definir cómo es cada individualidad de cada uno de los integrantes del grupo.

- **Aprendizaje por refuerzo.** En este caso, el algoritmo aprende a optimizar un proceso de decisión de la siguiente forma: si el resultado de esa decisión es beneficioso, el agente aprende automáticamente a repetir esa decisión en el futuro, mientras que si el resultado fuera perjudicial evitará volver a tomar la misma decisión.

Aunque el aprendizaje profundo ha tenido muchas definiciones a lo largo de su historia [11], se podría definir como “un tipo de técnicas de aprendizaje máquina que explotan muchas capas de procesamiento no lineal para extracción y transformación de características supervisadas o no supervisadas, así como análisis y clasificación de patrones”. En otras palabras, se puede decir que es una rama del ML que

emplea arquitecturas computacionales con trasformaciones no lineales múltiples e iterativas. Este tipo de algoritmos están basados en las redes neuronales profundas, tales como las redes convolucionales o las redes profundas de creencia. Desde 2006, el Aprendizaje Profundo ha emergido como un área del ML y se ha aplicado en infinidad de campos [11], pero quizá la visión por computador y el procesado del lenguaje natural son los más destacados.

Las redes neuronales artificiales son un modelo computacional inspirado en el concepto biológico de las redes neuronales de los organismos vivos. Consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida. Su principal objetivo es encontrar la mejor combinación de unos parámetros dados y ajustarlos a unos datos de entrada para predecir cierto resultado. Encontrar la combinación que mejor se ajusta es lo que se denomina entrenar la red.

La unidad fundamental que compone una red neuronal es el perceptrón simple, concepto introducido por Frank Rosenblatt [12]. Un perceptrón es un algoritmo de reconocimiento de patrones basado en una red de aprendizaje de dos capas, que utiliza adición y sustracción simples (Figura 1.2). En otras palabras, un elemento compuesto por varias entradas con un cierto peso cada una de ellas, denominados pesos sinápticos, y una salida. Esta salida depende de una función de activación que suele ser de tipo escalón. Su funcionamiento consiste básicamente en calcular la suma de las entradas por cada peso y, si ese resultado es mayor que un determinado número, la salida del perceptrón será uno. En caso contrario, será cero.

Cabe destacar que el perceptrón simple sólo tiene una neurona, por lo que está limitado a resolver problemas de clasificación de patrones con sólo dos clases y separables linealmente. Para realizar la clasificación de más de dos clases y resolver problemas más complejos, es necesario expandir el perceptrón e incluir más de una neurona, tal y como se muestra en la Figura 1.3. Es lo que se denomina Perceptrón Multicapa (*Multilayer Perceptron* o MLP en inglés).

Básicamente, el MLP es una red neuronal artificial de tipo *feedforward*[1] compuesta por múltiples capas de perceptrones. Consiste en, al menos, tres capas de

[1]Este tipo de redes son las que no tienen ciclos, es decir, la información fluye en una sola dirección, entrada-salida.

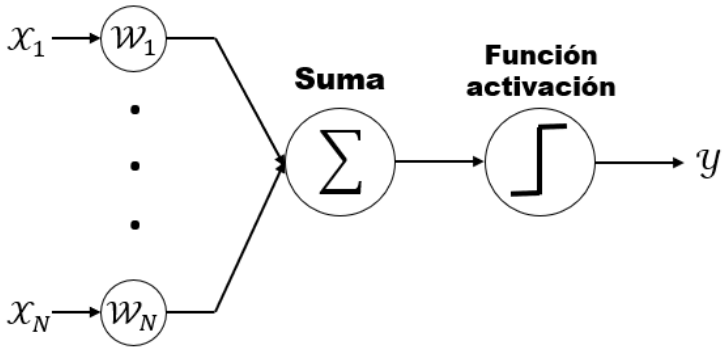


Figura 1.2: Perceptrón Simple. Varias entradas X son combinadas con unos pesos sinápticos W y pasados por una función de activación tipo escalón para obtener una salida Y.

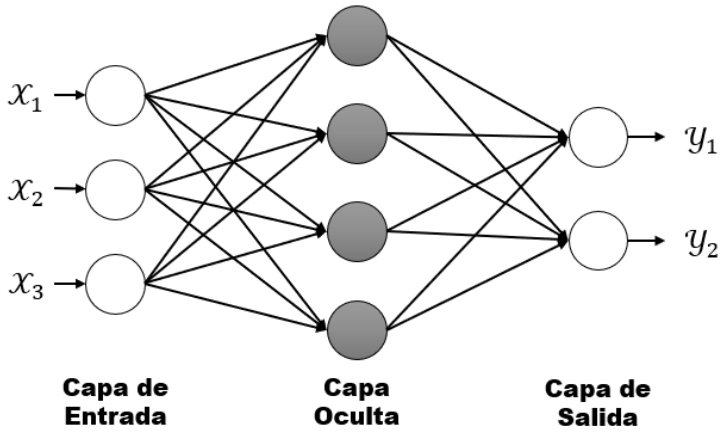


Figura 1.3: Perceptrón Multicapa. El MLP es un tipo de red neuronal artificial formada por múltiples capas de perceptrones. Consiste en, al menos, tres capas de neuronas: entrada, oculta y salida. Excepto en la entrada, el resto de neuronas usan funciones de activación no lineal, que hacen a esta red más robusta ante problemas complejos.

nodos: entrada, oculta y capa de salida. Excepto en la entrada, donde no se produce procesamiento, cada nodo es una neurona que usa una función de activación no lineal. Así, sus múltiples capas y activaciones no lineales distinguen el MLP del perceptrón simple y lo hacen más robusto ante problemas complejos, permitiendo distinguir datos que no son linealmente separables [13].

El MLP ha sido ampliamente estudiado por su capacidad de resolver problemas que no son linealmente separables. Son considerados como aproximadores universales [13], de modo que pueden ser usados para crear modelos matemáticos de análisis de regresión o clasificación. Se hicieron especialmente populares en los años 80 siendo utilizados en aplicaciones como procesado de imágenes o reconocimiento de voz, y tras un periodo de desinterés, volvieron a ser objeto de estudio con el auge del aprendizaje profundo.

1.4 Redes autoasociativas profundas

En el campo de las redes neuronales se ha estudiado ampliamente que la composición de varias capas no lineales es clave para modelar de forma eficiente las relaciones entre variables, y de esta forma conseguir una mejor capacidad de generalización en tareas complejas [14]. Esta línea de investigación está motivada parcialmente por el conocimiento de algunas partes del cerebro humano tales como el cortex visual. Básicamente, un MLP presenta grandes limitaciones a la hora de construir más de una o dos capas ocultas (como el desvanecimiento del gradiente), a excepción de las ampliamente conocidas Redes Convolucionales [15].

Es por esto que, tras una época en la que se perdió el interés por el aprendizaje profundo, aparecieron estudios que presentaban un nuevo paradigma de entrenamiento en las redes profundas [16, 17, 18]. Además, han surgido muchas técnicas alternativas desde que se presentó el trabajo de Redes Profundas de Creencia (*Deep Belief Networks*, DBN) de Hinton [16, 17]. Sin embargo, todos se basan en el mismo principio:

- Entrenar una red profunda directamente para optimizar de forma supervisada sólo el objetivo de interés mediante un método de gradiente, inicializando sus

parámetros aleatoriamente, no funciona bien. Funciona mucho mejor inicializar los parámetros usando un criterio local no supervisado para (pre)entrenar cada capa, con el objetivo de aprender a producir una representación útil a partir de la salida de la capa anterior. Así, partiendo de esto, un entrenamiento basado en gradiente de forma supervisada produce mucho mejores resultados en términos de generalización.

Este tipo de redes, entrenadas con esta metodología, han demostrado evitar soluciones pobres por efecto de estancamiento de gradiente en un óptimo local generado por incializaciones aleatorias. Esto se consigue principalmente por el entrenamiento no supervisado que guía el aprendizaje de cada capa.

Una red autoasociativa (*Autoencoder*, AE) es un tipo de red neuronal artificial entrenada de forma no supervisada para aprender una copia de su entrada a su salida. El objetivo de esta red es aprender una representación (codificación) eficiente de los datos de entrada [19], normalmente para tareas de reducción de dimensionalidad. Existen algunas variantes de esta arquitectura básica que son entrenadas con el objetivo de extraer propiedades de los datos de entrada mediante el aprendizaje de las representaciones intermedias [20], tales como el *Denoising Autoencoder* (DAE).

La arquitectura más básica de un AE es igual que la de un MLP –con una capa de entrada, una oculta y una de salida–, con la diferencia de que la capa de salida tiene las mismas neuronas que la de entrada. Se entrena para minimizar el error entre entrada y salida, en lugar de predecir unos valores objetivo dadas unas entradas. Así, un AE está formado por:

- **Codificador.** Comprime la entrada de la red en un espacio de variables latentes. Así, se mapean los vectores de entrada en una representación oculta de la forma:

$$\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1.3}$$

  donde $\mathbf{x}$ es el vector de entrada, $\theta = \{\mathbf{W}, \mathbf{b}\}$ los parámetros de entrenamiento y $s()$ una función de activación.

- **Decodificador.** Reconstruye la entrada a partir de los datos provenientes del codificador, es decir, del espacio de dimensión latente. Así, la representación oculta **y** se mapea de nuevo para reconstruir la entrada de la forma:

$$\mathbf{z} = g_\phi(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}') \tag{1.4}$$

  donde $\phi = \{\mathbf{W}', \mathbf{b}'\}$ son los parámetros y la función $s()$ puede ser lineal simplificando la ecuación anterior.

Aunque los AEs simples son útiles en varias aplicaciones, sólo el criterio de reconstrucción es incapaz de garantizar la extracción de características útiles, ya que puede conllevar la solución obvia de identidad. Una estrategia para evitar este fenómeno es restringir la representación, es decir, crear redes compresivas. Además, existe otra opción, presentada en [14], que consiste en cambiar el criterio de reconstrucción para limpiar una entrada parcialmente corrupta, *denoising* en inglés. De esta forma, se puede definir una buena representación como "aquella que puede ser obtenida robustamente a partir de una entrada ruidosa y que será útil para recuperar la entrada limpia". Con esto, se entiende que la representación obtenida en la capa oculta será más estable ante modificaciones en los datos, y al mismo tiempo se obtendrán características que capturen mejor la distribución de entrada.

Según el criterio comentado, se puede definir un DAE como un AE entrenado para reconstruir una versión limpia de una entrada ruidosa. Esto se consigue añadiendo ruido a **x**, obteniéndose $\widetilde{\mathbf{x}}$ e $\widetilde{\mathbf{y}} = f_\theta(\widetilde{\mathbf{x}}) = s(\mathbf{W}\widetilde{\mathbf{x}} + \mathbf{b})$, y entrenando los parámetros θ y ϕ para minimizar el error de reconstrucción y hacer que la salida se aproxime a la versión limpia de la entrada, es decir, el error entre $\widetilde{\mathbf{z}} = g_\phi(\widetilde{\mathbf{y}})$ y **x**. Por tanto, la diferencia es que ahora la salida se obtiene mediante un mapeo determinístico de la entrada corrupta. Esto conlleva el aprendizaje y extracción de características más útiles para el *denoising* y evita la función identidad.

Los DAEs se pueden apilar para formar una red profunda llamada *Stacked Denoising Autoencoder* (SDAE). Para ello, se va generando la red profunda mediante un entrenamiento capa a capa, alimentando la entrada de cada DAE con la representación latente (capa oculta) de la capa anterior. Así, cada capa es entrenada como un DAE minimizando el error de reconstrucción de su entrada ruidosa, siendo ésta

la representación de la entrada en la última capa oculta de la red profunda. Básicamente, teniendo las primeras k capas entrenadas, se puede obtener la capa $k+1$ mediante la representación latente de la capa anterior.

Una vez que todas las capas ocultas se han pre-entrenado, la red profunda se refina en una segunda etapa de entrenamiento. En este caso, se realiza un refinamiento supervisado donde la red es entrenada para realizar la tarea a resolver –clasificación o regresión– con las etiquetas correspondientes. Para esto, primero es necesario añadir una capa de salida C y toda la red se entrena por gradiente como si fuera un MLP (Figura 1.4). Es importante destacar que la entrada ruidosa se usa sólo para el entrenamiento inicial de cada capa oculta de modo que éstas aprendan características útiles. Tras esto, el refinamiento se realiza con la entrada original.

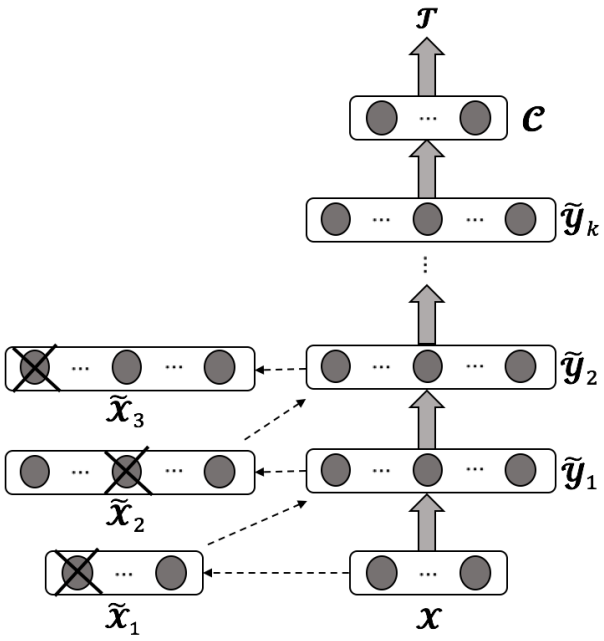


Figura 1.4: Stacked Denoising Autoencoder. Varios DAEs se apilan formando una red profunda mediante un entrenamiento capa a capa no supervisado. Cada DAE intermedio se alimenta con la representación latente de la capa anterior. Finalmente, se añade una capa de salida C y toda la red profunda se refina de forma supervisada usando las etiquetas **t** y la entrada original **x**.

El pasado puede doler pero, tal y como yo lo veo, puedes: o huir de él o aprender.

El Rey León

CAPÍTULO

2

Estado del Arte

Con el auge reciente del análisis de datos, queda aún más clara la necesidad de técnicas que manejen conjuntos de datos con valores perdidos, también conocido como problema de valores desconocidos, datos faltantes o datos incompletos. Cuando uno se enfrenta a este problema en el que, por lo general, se manejan grandes volúmenes de datos, el número de valores perdidos puede llegar a ser realmente significativo. De entre las técnicas existentes para solucionar este problema la imputación es la más adecuada, especialmente cuando no se puede permitir la pérdida de información. Esta tarea se vuelve incluso más importante para la clasificación de patrones incompletos, donde la falta de conocimiento puede conllevar resultados desastrosos cuando la tasa de valores perdidos es alta.

Es importante mencionar que la estimación de datos incompletos ha sido un campo de estudio a lo largo de los años típico de la estadística aplicada. Esto se puede comprobar en un gran número de estudios [21, 22, 23, 24]. Sin embargo, desde inicios de siglo, ha emergido un nuevo paradigma a la hora de tratar valores perdidos y el aprendizaje máquina ha empezado a ocupar un papel relevante en este tipo de problemas. En este escenario, el modelo de aprendizaje automático se entrena con las muestras de los valores observados, siendo usado después para predecir las muestras que contienen valores desconocidos. Es importante mencionar que en los capítulos posteriores se presentarán modelos capaces de abordar tareas

de clasificación con patrones incompletos, en algunos casos sin hacer distinción entre tarea de imputación y clasificación. Sin embargo, en la literatura la mayoría de los métodos se centran en la imputación como tarea independiente.

El término ”multitarea” es normalmente entendido como algo negativo –una persona que intenta hacer varias cosas al mismo tiempo y no tiene éxito en ninguna. Pero si pensamos en nuestros sentidos, la cosa cambia. Si nos fijamos en nuestro sistema de visión, somos capaces de realizar varias tareas a la vez con un simple vistazo: identificar objetos y segmentarlos, entender el estado anímico de las personas o incluso cómo van vestidos. El oído es igual. Somos capaces de conocer el sexo de una persona al mismo tiempo que entendemos el mensaje que está lanzando. Se cree en la capacidad de las redes neuronales artificiales de llegar a realizar este tipo de tareas.

En este capítulo, se presentarán los conceptos básicos de la imputación, se hará una revisión de los métodos más comunes a la hora de resolver este problema, haciendo una distinción entre técnicas estadísticas y de aprendizaje máquina, y finalmente se expondrán los conceptos básicos del aprendizaje multitarea, introduciendo algunos de los trabajos recientes sobre aprendizaje profundo.

2.1 Imputación

Existen tres problemas principales que la presencia de datos incompletos puede causar: introducción significativa de sesgo, dificultad a la hora de manejar y analizar los datos, y reducción de la eficiencia del modelo [25]. La técnica más eficiente a la hora de manejar los datos incompletos es la imputación. Con ella, se pretenden evitar los problemas mencionados sobre los resultados finales de clasificación.

Como se define en el campo de la estadística, los métodos de imputación son aquellos que tratan de rellenar los valores perdidos de un conjunto de datos mediante su estimación. El estudio sistemático y la formalización de este problema desde un punto de vista probabilístico no se inicia hasta mediados de los setenta, destacando principalmente el trabajo de Rubin [3]. Estas técnicas están basadas en la relación existente entre atributos, ya que en la mayoría de los casos éstos no son independientes entre sí. Además, en la mayoría de los estudios, la tarea de imputación es un método aislado de la tarea de clasificación, que se realiza en un paso

posterior. Aunque este punto siempre se ha considerado como una ventaja, se cree que son necesarios más métodos que tengan en cuenta la relación entre imputación y clasificación. De ahí que la imputación orientada a la clasificación parezca más útil que la imputación independiente [26].

Existe una amplia variedad de técnicas de imputación en la literatura con diferentes niveles de complejidad. Aunque la clasificación de los métodos de imputación puede variar según el trabajo, en este capítulo se diferenciarán según su naturaleza:

1. *Métodos basados en máxima verosimilitud*. Esta parte presenta las bases de los métodos basados en funciones de verosimilitud, bajo los cuales subyace un modelo probabilístico. En concreto, se presenta la teoría del algoritmo EM (Expectation-Maximization) junto con el marco formal de estos métodos.

2. *Métodos basados en estimadores estadísticos*. Se repasan los métodos estadísticos más frecuentes a la hora de estimar datos incompletos. Son:

   - Imputación mediante el estimador media/mediana/moda.
   - Método *Hot-deck*.
   - Imputación por Descomposición en Valores Singulares (SVD).
   - Imputación mediante regresión.
   - Imputación múltiple.

3. *Métodos de imputación mediante técnicas de aprendizaje máquina*. Estos métodos de imputación usan algoritmos de aprendizaje máquina adaptados a la estimación de valores incompletos. Se emplea para ello el conocimiento proporcionado por las muestras completas del conjunto. Se presentan las técnicas:

   - K vecinos más cercanos (*K Nearest Neighbors*, KNN).
   - Árboles de decisión.
   - Perceptrón Multicapa (*MultiLayer Perceptron*, MLP).

4. *Métodos de imputación con técnicas de aprendizaje profundo*. Aquí se presentarán las técnicas desarrolladas en los últimos años para estimar valores desconocidos usando redes neuronales profundas.

Aunque cada método mencionado tiene su funcionamiento específico, todos poseen algo en común. Básicamente, si se tiene un conjunto de datos de la forma $\mathbf{X} = \{\mathbf{x}_n, \mathbf{t}_n\}, n = 1, ..., N$, donde $\mathbf{x}_n$ es el patrón n-ésimo, y $\mathbf{x}_n = \{x_n^C, x_n^I\}$ con x_n^C como patrones completos y x_n^I como patrones incompletos del conjunto, el objetivo de todo método de imputación es construir un estimador de la información incompleta $f()$ a partir de la observada, es decir, $x_n^{Imp} = f(x_n^C)$. Así, se consigue un nuevo conjunto de entrenamiento de la forma $\widehat{\mathbf{X}} = \{\widehat{\mathbf{x}}_n, \mathbf{t}_n\}, n = 1, ..., N$ y $\widehat{\mathbf{x}}_n = \{x_n^C, x_n^{Imp}\}$ con el que, en el caso en que sea necesario, se resuelve el problema principal de clasificación. Así, como se ha comentado, se puede decir que la imputación no está orientada a la clasificación sino completamente independiente.

2.2 Imputación mediante algoritmo EM

En [3], se propuso un marco conceptual para el análisis de datos perdidos sustentado en métodos de inferencia estadística. Posteriormente, la aparición del algoritmo "Expectation Maximization" (EM) permitió generar estimadores robustos a partir de la aplicación del método de máxima verosimilitud (*Maximum Likelihood*, ML) [27], en donde las observaciones faltantes se asumen como variables aleatorias y los datos imputados se generan sin necesidad de ajustar modelos.

El algoritmo EM es un algoritmo iterativo que trata de encontrar estimadores de máxima verosimilitud de parámetros en problemas con muestras no observables. Este algoritmo fue desarrollado en 1977 por Dempster, Laird y Rubin en [27]. La idea del método es bastante sencilla. En primer lugar, se sustituyen los datos incompletos por una estimación de los mismos. Con el conjunto completo, se estiman los parámetros del modelo, con los cuales se vuelven a estimar los valores desconocidos. Así, sucesivamente, se consigue la convergencia iteración a iteración.

En esta sección, se introduce el método EM para imputación, comentando en primer lugar un marco formal de inferencia basada en muestras completas. Es importante destacar que, para encontrar todos los detalles de este método, será necesa-

rio ir a los trabajos de Ghahramani y Jordan [28, 29], Little y Rubin [5], McLachlan y Krishnan [30] y Wu [31].

2.2.1 Marco formal de inferencia basada en muestras completas

Consideremos, siguiendo el desarrollo de [32], un vector aleatorio X *k*-dimensional que genera los datos y un vector M, también *k*-dimensional, formado por variables aleatorias binarias tomando valores 0 o 1 para indicar valor observado o no observado, respectivamente. Se considera mecanismo de no respuesta a la distribución de probabilidad de M, que tal y como se ha comentado en el capítulo anterior, podrá ser tipo MAR, MCAR o NMAR. Así, por ejemplo, se puede definir el mecanismo tipo MAR como $P(M|X_{obs}, X_{per}, \xi) = P(M|X_{obs}, \xi)$, siendo ξ un vector de parámetros desconocidos del mecanismo de no respuesta.

Por tanto, se puede dividir X de la forma $X = (X_{obs}, X_{per})$, donde X_{obs}, X_{per} denotan la parte observada y no observada, respectivamente. Así, se define la distribución de probabilidad de X como $P(X;\theta)$, siendo θ el vector de parámetros desconocidos.

Se pueden usar una gran variedad de métodos de inferencia para interpretar $P(X;\theta)$ como una función de verosimilitud que resume la evidencia de θ sobre los datos cuando se tiene la muestra completa de X. Sin embargo, con la presencia de datos incompletos, se tiene:

$$P(X_{obs};\theta) = \int P(X;\theta)dX_{per} \tag{2.1}$$

Partiendo de la parte observada, para hacer inferencia sobre θ, se debe comprobar que (2.1) es una función de verosimilitud adecuada. Para que así sea, Rubin presenta en [3] las condiciones necesarias, determinando que es suficiente con verificar la hipótesis MAR.

Por tanto, se debe definir un modelo para X, $P(X;\theta)$ y para la no respuesta $P(M|X_{obs}, X_{per}, \xi)$. Se puede obtener la distribución conjunta $P(X, M;\theta, \xi)$ mediante $P(M|X_{obs}, X_{per}, \xi)P(X;\theta)$. Así, basándose en la parte conocida, se puede expresar la verosimilitud:

$$P(X_{obs}, M;\theta, \xi) = \int P(X, M;\theta, \xi)dX_{per} = \int P(M|X_{obs}, X_{per}, \xi)P(X;\theta)dX_{per} \tag{2.2}$$

Esta ecuación queda, bajo la hipótesis MAR, como:

$$P(X_{obs}, M; \theta, \xi) = P(M|X_{obs}, \xi) \int P(X; \theta) dX_{per} = P(M|X_{obs}, \xi) P(X_{obs}; \theta) \tag{2.3}$$

Como se puede apreciar, se obtienen dos partes, una relativa a θ y otra a ξ. Si estos dos vectores son distinguibles, es decir, proporcionan poca información el uno del otro, se puede afirmar que las inferencias sobre θ no se ven afectadas por $P(M|X_{obs}, \xi)$, por lo que la función de verosimilitud L de θ queda $L(\theta|X_{obs}) \alpha P(X_{obs}; \theta)$. Así, bajo ignorabilidad se puede hacer inferencia sobre el vector de parámetros θ de la distribución X a partir de la verosimilitud $L(\theta|X_{obs})$.

Por otro lado, desde una perspectiva bayesiana, todas las inferencias se basan en la distribución de probabilidad a posteriori de los parámetros desconocidos, que puede escribirse utilizando el Teorema de Bayes en la forma:

$$P(\theta, \xi|M, X) = \frac{P(M, X_{obs}|\theta, \xi)\phi(\theta, \xi)}{\int\int P(M, X_{obs}|\theta, \xi)\phi(\theta, \xi) d\theta d\xi} \tag{2.4}$$

donde ϕ representa la distribución a priori de (θ, ξ). Bajo la comentada hipótesis MAR, se puede sustituir (2.3) en (2.4), obteniendo que $P(\theta, \xi|M, X_{obs})$ es proporcional a $P(M|X_{obs}, \theta) P(X_{obs}|\theta)\phi(\theta, \xi)$. Si además θ y ξ son distinguibles, la distribución marginal a posteriori de θ queda

$$\begin{aligned} P(\theta|X_{obs}, M) = & \int P(\theta, \xi|M, X_{obs}) d\xi \alpha \; P(X_{obs}|\theta)\phi_\theta(\theta) \\ & \int P(M|X_{obs}, \xi)\phi_\xi(\xi) d\xi \alpha \; L(\theta|X_{obs})\phi_\theta(\theta) \end{aligned} \tag{2.5}$$

Por tanto, se concluye que, bajo la hipótesis de ignorabilidad, toda la información sobre θ se recoge en la distribución a posteriori que ignora el mecanismo de no respuesta observado, $P(\theta|X_{obs}) \; \alpha \; L(\theta|X_{obs})\phi_\theta(\theta)$. La necesidad de especificar una distribución a priori para los parámetros puede resultar algo subjetivo desde un punto de vista bayesiano. Sin embargo, como se destaca en [33], conforme aumenta el tamaño muestral la verosimilitud domina a la distribución a priori, y las respuestas bayesianas y verosímiles tienden a converger.

2.2.2 Algoritmo EM

Se define ahora una muestra incompleta $X = (X_{obs}, X_{per})$ del conjunto $X \sim P(X;\theta)$ del que se desea obtener el EMV de θ. Básicamete, el algoritmo trata de aprovechar los datos conocidos para reformular el problema y resolverlo con la información conocida. Así, se puede definir $P(X;\theta)$ como:

$$P(X;\theta) = P(X_{obs};\theta)P(X_{per}|X_{obs},\theta) \tag{2.6}$$

En lugar de usar la función de verosimilitud, suele usarse su logaritmo, que se conoce como log-verosimilitud. De esta forma, se puede deducir:

$$logL(\theta|X_{obs}) = logL(\theta|X) - logP(X_{per}|X_{obs},\theta) \tag{2.7}$$

En presencia de datos desconocidos, el objetivo es estimar θ mediante la maximización de $logL(\theta|X_{obs})$ con respecto a θ dada X_{obs}. Como se demuestra a continuación, el algoritmo EM relaciona el EMV de θ a partir de $logL(\theta|X_{obs})$ con el EMV de θ a partir de $logL(\theta|X)$. Tomando esperanzas respecto a $P(X_{per}|X_{obs},\theta)$ a ambos lados de (2.7) y dado un estimador $\theta^{(t)}$ de θ, se tiene:

$$logL(\theta|X_{obs}) = Q(\theta;\theta^{(t)}) - H(\theta;\theta^{(t)}), \tag{2.8}$$

donde

$$Q(\theta;\theta^{(t)}) = \int logL(\theta|X)P(X_{per}|X_{obs},\theta^{(t)})dX_{per} \tag{2.9}$$

y

$$H(\theta;\theta^{(t)}) = \int logP(X_{per}|X_{obs},\theta)P(X_{per}|X_{obs},\theta^{(t)})dX_{per} \tag{2.10}$$

El paso E (*Expectation*) del algoritmo EM calcula $Q(\theta;\theta^{(t)})$, reemplazando los valores perdidos, o una función de ellos, por su esperanza condicionada dados X_{obs} y $\theta^{(t)}$. El paso M (*Maximization*) simplemente determina el EMV $\theta^{(t+1)}$ que maximiza $Q(\theta;\theta^{(t)})$ como si no hubiera datos perdidos. Los pasos E y M se repiten alternativamente generando una sucesión de estimadores $\{\theta^{(t)}\}$. La diferencia en el valor de la log-verosimilitud $logL(\theta|X_{obs})$ en dos iteraciones sucesivas viene dada por

$$\begin{aligned} logL(\theta^{(t+1)}|X_{obs}) - logL(\theta|X_{obs}) = {} & Q(\theta^{(t+1)};\theta^{(t)}) - Q(\theta^{(t)};\theta^{(t)}) \\ & + H(\theta^{(t)};\theta^{(t)}) - H(\theta^{(t+1)};\theta^{(t)}) \end{aligned} \tag{2.11}$$

En este algoritmo el estimador $\{\theta^{(t)}\}$ se escoge de forma que $Q(\theta^{(t+1)};\theta^{(t)}) \geq Q(\theta^{(t)};\theta^{(t)})$ y $H(\theta^{(t)};\theta^{(t)}) \geq H(\theta^{(t+1)};\theta^{(t)})$, se tiene que $logL(\theta|X_{obs})$ va incrementándose en cada iteración, convergiendo hacia el EMV de θ. El proceso se para cuando la diferencia entre dos iteraciones sucesivas es suficientemente pequeña.

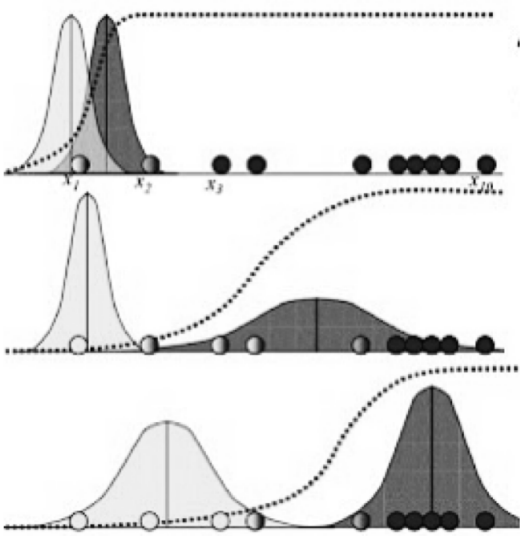


Figura 2.1: Algoritmo EM. Alterna pasos de esperanza (paso E), donde se computa la esperanza de la verosimilitud mediante la inclusión de variables latentes como si fueran observables, y un paso de maximización (paso M), donde se computan estimadores de máxima verosimilitud de los parámetros mediante la maximización de la verosimilitud esperada del paso E. Los parámetros que se encuentran en el paso M se usan para comenzar el paso E siguiente, y así el proceso se repite hasta que el modelo se estabiliza.

Como se explica en [32], es sencillo emplear el algoritmo EM como método de imputación. Una vez que se ha producido la convergencia, basta con dar un nuevo paso E y obtener las esperanzas matemáticas de los valores no observados condicionadas a los valores observados dado el EMV del vector de parámetros θ. Este método es similar al Estimador de Máxima Versosimilitud (EMV) ya que ambos son capaces de ajustar los parámetros de los datos. Sin embargo, EMV acumula todos los datos primero y luego los usa para construir el modelo más probable, mientras que el algoritmo EM hace una suposición de los parámetros primero –

contabilizando los valores perdidos– y después ajusta el modelo para aproximar la suposición a los valores observados. En la Figura 2.1 se muestra el proceso.

2.3 Técnicas de imputación estadística

Como se ha comentado, existen muchas formas de abordar el problema de datos perdidos. Aunque frecuentemente se descartan las muestras con valores desconocidos, los métodos estadísticos son en la práctica los más utilizados para llevar a cabo la tarea de imputación. Probablemente, esto sea así porque estas técnicas han sido las más estudiadas en la literatura, o simplemente porque son las más sencillas de implementar en la mayoría de los casos.

Existen dos formas de imputación estadística: simple o múltiple. Cuando sólo hay unos pocos valores desconocidos en el conjunto de datos, las técnicas simples son suficientes para obtener buenos resultados. Además, se requiere un cómputo considerablemente menor ya que proporcionan sólo un número específico para cada valor perdido del conjunto de datos. Además, las técnicas simples pueden ser divididas entre las que usan información únicamente de la variable incompleta y las que tienen en cuenta otras variables.

Por otro lado, cuando uno se enfrenta a una cantidad más significativa de valores perdidos, los métodos simples pueden no conseguir resultados aceptables. Los métodos de imputación múltiple resuelven este problema mediante el uso de modelos de simulación iterativos que aportan un conjunto de posibles valores para cada valor desconocido.

En general, el proceso de imputación requiere analizar la información disponible y buscar la respuesta (o respuestas) más probable de entre todas las posibles. A continuación, se describen algunas de las técnicas de imputación estadística más conocidas, poniendo foco en algunos métodos que se emplean en los estudios posteriores de esta tesis.

2.3.1 Imputación a la media/mediana/moda

En muchas ocasiones, una opción simple de imputación, aunque poco satisfactoria, es el uso de un parámetro “central” de la variable a imputar. Estos parámetros

suelen ser la media, la mediana o la moda, ya que no suponen un impacto en la distribución de los datos. Por contra, además de su rendimiento pobre, tienen otras desventajas tales como que imputar a la media reduce la varianza en el conjunto de datos. Así, por ejemplo, la media sería buena opción si los datos se distribuyen en una normal. En cambio, si se tiene una variable categórica, la moda es el parámetro apropiado.

2.3.2 Método de imputación *Hot-deck*

Durante la década de los setenta, la imputación de datos significaba identificar y sustituir los registros sin información. En este contexto, los procedimientos *Hot-deck*, en sus distintas formas, se aplicaban profusamente para suplir información en censos y encuestas, y los métodos de ajuste por promedios y *Cold-desk* eran frecuentemente utilizados.

Es quizá el método más sencillo junto con la opción de rellenar todos los valores desconocidos con cero. El método *Hot-deck* consiste en seleccionar el valor para el dato desconocido aleatoriamente de entre los casos que son similares al incompleto, llamados donantes, basándose en algunas variables definidas por el usuario. El conjunto de casos donantes se le conoce como ”deck”. El caso más simple sería seleccionar aleatoriamente el valor perdido de entre todas las muestras completas.

Este método es muy antiguo y popular, y suele usarse frecuentemente por su sencillez. Es adecuado en situaciones donde los valores perdidos no se encuentran de forma aleatoria en el conjunto de datos, es decir, para el patrón MAR. El método *Hot-deck* no distorsiona la distribución de los datos de entrada, al menos parcialmente. Sin embargo, afecta a las correlaciones entre variables. Esto puede evitarse con versiones más sofisticadas del algoritmo.

2.3.3 Imputación por descomposición en valores singulares (SVD)

Este método fue presentado en el trabajo [34]. En él, se aproximan los valores desconocidos mediante vectores propios, los cuales se ajustan a través de ciertas combinaciones lineales. Estos patrones, idénticos a la descomposición en componentes principales, pueden ser seleccionados de forma que se mantiene la mayoría

de la varianza de los datos. Está demostrado que sólo son necesarios unos pocos vectores significativos para hacerlo.

Este método se basa en la Descomposición de Valores Singulares (SVD) del conjunto de datos. Dada la descomposición $A = U \sum V^T$, se recurre a los vectores propios más significativos de V^T para estimar los valores perdidos linealmente. Aunque se ha demostrado que la correcta elección de varios vectores propios significativos es suficiente para describir los datos con un pequeño error, la fracción exacta debe ser determinada empíricamente. Una vez seleccionados los k vectores más significativos de V^T, para estimar un valor desconocido j en un dato i primero se debe hacer regresión sobre el dato i contra los k vectores propios, y luego usar los coeficientes de la regresión para reconstruir j como una combinación lineal de los k vectores. El j-ésimo valor del dato i y el j-ésimo valor de los k vectores propios no se usan para determinar estos coeficientes de regresión. Es importante mencionar que el método SVD no funciona con matrices incompletas, por lo que primero se rellenan los valores desconocidos con otros métodos.

2.3.4 Imputación mediante regresión

Este método consiste en rellenar los valores perdidos con el resultado de un algoritmo de regresión [35]. Por tanto, es necesario entrenar un algoritmo para predecir la característica con datos faltantes, donde el resto de características se usarán como entrenamiento y los valores conocidos de ésta como valores objetivo. Así, los elementos desconocidos se imputarán como valores predichos por el algoritmo.

Un ejemplo de este algoritmo puede ser el implementado por una regresión lineal, aunque en la práctica no se recurre a ella ya que se requiere una relación lineal entre variables. Por tanto, es común el uso de métodos no lineales que puedan modelar el comportamiento no lineal de las variables. Generalmente, estos modelos pueden ser empleados tanto para predecir el valor faltante como para explicar la relación de la característica incompleta con las demás.

La principal desventaja de este método es que se debe repetir para cada una de las características con valores perdidos. Esto puede ser un problema cuando varias características del conjunto son incompletas, ya que se necesitan varios algoritmos

de regresión y la información disponible puede ser escasa para entrenarlos individualmente.

2.3.5 Imputación múltiple

Unos años más tarde de la aparición de métodos estadísticos simples, Rubin [6] introdujo el concepto de Imputación Múltiple (IM), sustentado en la premisa de que cada dato faltante debe ser reemplazado a partir de $m > 1$ simulaciones. La aplicación de esta técnica se facilitó con los avances computacionales y el desarrollo de los métodos bayesianos de simulación que aparecieron hacia finales de los ochenta [36]. A partir de este momento, se hizo un uso intensivo de estos algoritmos.

La imputación múltiple se basa en obtener un conjunto m de estimaciones para cada valor perdido, obteniendo así m conjuntos posibles completos. Estos conjuntos se combinan mediante reglas sencillas de forma que se obtiene una estimación final y un intervalo de confianza asociado a la estimación. Un esquema de las etapas de este método se puede ver en la Figura 2.2. Como se ha comentado, la idea se propuso por primera vez en [6], y permite minimizar el sesgo causado por la pérdida de datos bajo MCAR, MAR e incluso NMAR [36, 37].

Al igual que para la imputación simple, existe un gran número de métodos de imputación múltiple. Una ventaja de estos métodos sobre los demás es su flexibilidad, y además pueden ser usados en una gran variedad de escenarios distintos. Sin embargo, su principal desventaja reside en el hecho de que requiere un mayor tiempo computacional para generar el vector de valores imputados y para analizar posteriormente los datos [38].

2.3.5.1 Imputación múltiple mediante ecuaciones encadenadas (MICE)

En los últimos años, el método "Multiple Imputation by Chained Equations" (MICE) ha emergido como el más efectivo de entre los algoritmos de imputación múltiple. La mayoría de los métodos de imputación múltiple asumen un modelo de distribución normal conjunta para todas las variables. En conjuntos grandes, con cientos de variables de diferentes tipos, esto raramente se cumple. El método MICE es una alternativa. De hecho, éste se ha aplicado con conjuntos de miles de observaciones y cientos de variables. Básicamente, en el procedimiento MICE se ejecuta una

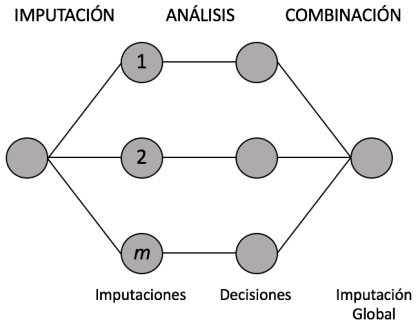


Figura 2.2: Proceso de imputación múltiple. Primero, se imputan los valores desconocidos m veces con un proceso de imputación simple, obteniendo así m conjuntos completos. Después, se analiza cada unos de estos conjuntos. Finalmente, los m resultados se consolidan en uno solo mediante la media, varianza, y el intervalo de confianza de la variable en cuestión.

serie de modelos de regresión en los que cada variable con datos faltantes se modela condicionalmente a las otras variables con datos completos. Esto significa que cada variable puede ser modelada acorde a su distribución con, por ejemplo, regresión logística para variables binarias o regresión lineal para variables continuas. Se pueden encontrar todos los detalles sobre este método en [39].

2.4 Técnicas de imputación basadas en aprendizaje máquina

Es importante entender que no existe una manera única de abordar un problema de imputación. Esto es, existen diferentes aproximaciones y es complejo obtener una solución general para diferentes problemas (análisis de series temporales, regresión, clasificación, etc.). La imputación mediante técnicas de aprendizaje máquina se basa en el diseño de un modelo predictivo que, partiendo de la información completa del conjunto de datos, se entrena para predecir los valores desconocidos.

Existen muchos algoritmos de aprendizaje máquina para realizar tareas de imputación. Aquí revisaremos tres que se consideran importantes y representativos: imputación mediante K vecinos más cercanos (KNN), mediante árboles de decisión, y mediante un Perceptrón Multicapa (MLP). Es importante mencionar que,

para todos estos métodos, la tarea de imputación debe realizarse previamente a la resolución de la tarea principal, ya sea esta de clasificación o de regresión.

2.4.1 Imputación mediante K vecinos más cercanos (KNN)

El algoritmo de los K vecinos más cercanos (*K Nearest Neighbors*, KNN) se basa en asociar una muestra con sus K más cercanas según un espacio multi-dimensional. Se puede usar para datos continuos, discretos, ordinales y categóricos, lo que implica que es especialmente útil para resolver problemas de todos los tipos de valores perdidos. La imputación mediante KNN es quizás la más utilizada en la práctica, a pesar de su gran coste computacional [40, 41, 42, 43]. Esto es así porque requiere almacenar todo el conjunto de entrenamiento en memoria y, aunque existen búsquedas aceleradas, la versión más básica necesita recorrerlo completamente para cada estimación.

La idea detrás del método KNN para imputar valores perdidos es que cualquier punto puede ser aproximado por los valores de los puntos que se encuentran más cerca de él. Un ejemplo de ello se muestra en la Figura 2.3. Así, la métrica de distancia entre puntos se obtiene mediante sus variables conocidas. Una vez seleccionados los vecinos más cercanos, se estima un valor para sustituir el dato incompleto a partir de los datos conocidos de la característica de interés. Si se trata de una característica continua, se suele usar la estimación a la media, mientras que se emplea la moda para las variables discretas.

Pensemos en un conjunto de datos sobre personas con tres variables: género, ganancias y nivel de depresión (esta última con valores perdidos). Se asume que las personas que tienen el mismo género y ganancias similares, tendrán el mismo nivel de depresión. Así, para un valor de depresión desconocido, se mirará el género, las ganancias, se buscarán los K vecinos más cercanos según la métrica aplicada sobre estas dos características y de ahí se obtiene su nivel de depresión.

2.4.2 Imputación mediante árboles de decisión

La idea de usar árboles de decisión para imputar valores perdidos se presentó en [44]. Sin embargo, fue en [45] donde sirvieron para identificar y construir clases de imputación. Creel y Krotki [46] fueron más allá y recurrieron a los nodos de

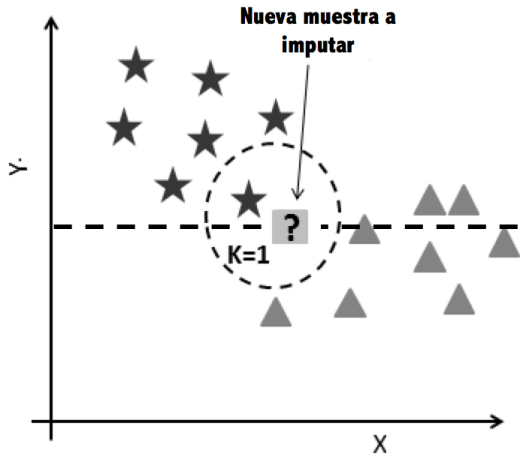


Figura 2.3: Imputación mediante K vecinos más cercanos. Básicamente, para la imputación del valor desconocido de la nueva muestra, se seleccionan los K vectores más cercanos en base a una métrica. El nuevo valor se imputa usando las características conocidas de estos vectores.

un árbol de decisión para definir clases de imputación y después aplicar diferentes métodos sobre ellas. Tras estos estudios que asientan las bases del método, muchos más se han centrado en cómo aplicar árboles de decisión para la imputación de valores perdidos [47, 48].

Los algoritmos más populares de árboles de decisión son C4.5 [49], CART [50], o CHAID [51]. Estos algoritmos, aunque se diferencian en detalles, están basados en la estrategia divide y vencerás. Esto significa que los algoritmos dividen un conjunto de datos de forma recurrente en subconjuntos que son cada vez más homogéneos con según un criterio.

Los algoritmos mencionados aplican diferentes medidas de impureza, como la entropía de información, como criterio para determinar la división óptima de subconjuntos. Esta medida aumenta con el número de objetos distintos en el conjunto, lo que en este contexto significa que pertenecen a distintas clases [52]. El árbol de decisión calcula el criterio (como el de diferencia de entropía) para cada posible división y elige la que obtiene mejor resultado en base a ese criterio. El algoritmo deja de calcular divisiones cuando se consigue un conjunto con sólo objetos de la misma clase. En un árbol, un nodo representa un subconjunto, una rama representa

una división, y los nodos finales se denominan hojas [53].

Una vez creado el árbol, se puede utilizar para imputar los valores desconocidos. Para esto existen dos posibilidades: por mayoría [54, 55, 56] o por probabilidad [47]. Si se hace por mayoría, la clase predicha para una muestra corresponde a la más común de su subconjunto final. Esto equivale a una imputación a la moda dentro de su subconjunto. Por otro lado, si se hace por probabilidad, la predicción se realiza aleatoriamente con una probabilidad equivalente a la frecuencia de aparición de cada muestra en el subconjunto final. Esto es equivalente a una imputación *Hot-deck* aleatoria en el subconjunto. Según la literatura, no queda claro cuál de los dos métodos consigue mejores resultados. Respecto a la calidad de imputación, una estimación estocástica debe producir mejores resultados [5]. Sin embargo, en [57] se demuestra cómo la imputación por mayoría ayuda a obtener mejores resultados en la tarea de clasificación final.

2.4.3 Imputación con un Perceptrón Multicapa (MLP)

Como se ha expuesto, existe un gran número de estudios estadísticos enfocados al análisis de datos faltantes, basándose en el empleo de la media, la moda, la mediana, la regresión o la imputación múltiple. Consideradas globalmente [58], se trata de técnicas que alivian el problema de falta de información, si bien no se pueden considerar soluciones óptimas por dos motivos: (1) pueden provocar un importante sesgo en las posteriores estimaciones realizadas sobre los datos completos, especialmente de varianzas y covarianzas, y (2) requieren el cumplimiento de determinados supuestos sobre la distribución de los datos [5].

Se ha visto en la Sección 1 que las redes neuronales artificiales (RNAs) son sistemas formados por unidades de procesamiento que intentan imitar el funcionamiento de las neuronas naturales y que, organizadas en paralelo, emulan la estructura cerebral. Así, son capaces de reconocer patrones y resolver tareas complejas casi siempre con mejores resultados que los métodos estadísticos anteriormente mencionados. Las RNAs también han sido utilizadas en el campo de los datos incompletos. Se ha demostrado que un MLP es un aproximador universal, es decir, es capaz de modelar cualquier función continua si sus parámetros se escogen de forma correcta [59]. Sin embargo, una de sus limitaciones es que no extrapola bien,

es decir, si la red se entrena mal o de manera insuficiente, las salidas pueden ser imprecisas.

En [60] ya se propuso un método de imputación mediante un MLP donde se explotaba la información de las características conocidas para predecir el valor de la característica con valores perdidos. En este trabajo ya se tiene en cuenta el entrenamiento de la red según la tipología de *missing* del conjunto de entrada. Tras esto, muchos trabajos se han centrado en esta técnica de imputación [61, 62, 63]. Básicamente, se entrena un MLP como un modelo de regresión para predecir los valores desconocidos a partir de los patrones completos. Así, dadas d características de entrada, si suponemos la existencia de un sólo atributo incompleto, su imputación se consigue mediante un MLP que aprende a reconstruirlo a partir de los $(d-1)$ atributos restantes.

Esta técnica de imputación, aunque puede ser una buena opción, no suele utilizarse en la práctica. Esto es así por sus desventajas cuando el conjunto de datos presenta varias características incompletas, algo que ocurre frecuentemente. La principal es la necesidad de entrenar una red neuronal independiente para la imputación de cada una de ellas. Además, se necesita una cantidad considerable de patrones completos a la entrada para que la máquina sea capaz de obtener buenos resultados. Esto conlleva un elevado coste computacional y, al mismo tiempo, resultados pobres en los casos donde el porcentaje de *missing* es alto.

2.4.4 Imputación basada en aprendizaje profundo

A partir de 2006, las redes neuronales profundas ganan popularidad porque se descubre que su entrenamiento puede ser mejorado apilando sus diferentes capas ocultas mediante el entrenamiento no supervisado de varias redes simples, seguido de un refinamiento global de la red profunda. Con el auge de este tipo de técnicas, en los últimos años han surgido nuevas estrategias de imputación de valores perdidos, e incluso actualmente se trata de un campo nuevo que sigue bajo estudio.

Además de los trabajos presentados en esta tesis, centrados en el uso de SDAEs, el uso de modelos generativos profundos ha sido el más extendido recientemente para realizar la tarea de imputación. Estos métodos tienen la gran ventaja de que

pueden ser mucho más precisos, pero al igual que con el MLP pueden ser muy costosos computacionalmente. Un modelo generativo es una técnica de aprendizaje no supervisado muy potente capaz de aprender casi cualquier distribución. Es por esto que su uso ha crecido enormemente en los últimos años. Todos los tipos de modelos generativos aprenden la distribución de los datos de entrada y sirven para generar nuevos puntos con algunas variaciones. Ya que a veces no se puede aprender la distribución exacta de los datos, se usan redes neuronales para aprender una función que se aproxime lo máximo posible. Así, las redes generativas más comunes son los Autoencoders Variacionales (*Variational Autoencoders*, VAEs) [64, 65] y las Redes Adversas Generativas (*Generative Adversarial Networks*, GANs) [66, 67]. Estas redes han surgido recientemente, junto con los SDAEs, como técnicas de aprendizaje profundo orientadas a tareas de imputación.

Sin entrar en demasiado detalle sobre los VAEs porque no es el objetivo de este trabajo, la idea básica consiste en aprender una representación comprimida de los datos de entrada mediante un conjunto de variables latentes [64, 65]. Se asume que estas variables sirven para generar el conjunto de datos de entrada y son capaces de almacenar información útil sobre el tipo de salida que el modelo necesita generar. Como se dice, esta técnica ha sido ampliamente utilizada para imputar desconocidos del conjunto de entrada [68, 69, 70].

Las GANs tienen un objetivo similar [66, 67], aprender a mapear muestras de la distribución del conjunto de entrada a un espacio latente donde los datos presentan una distribución conocida (por ejemplo, una gausiana). Este método está basado en la teoría de juegos y se entrena con el objetivo de encontrar un equilibrio entre dos redes llamadas generador y discriminante. Así, ha resultado ser una buena opción para imputar como se desprende de los resultados obtenidos en [71, 72, 73, 74].

Además de las técnicas mencionadas, es importante destacar la gran capacidad de los SDAEs para tareas de imputación. Estos modelos, como veremos en los siguientes capítulos de la tesis, poseen una gran capacidad de reconstrucción y es por ello que han sido estudiados en estos últimos años en diferentes trabajos de imputación [75, 76, 77]. Así, los estudios presentados en esta tesis forman parte de esta nueva corriente de investigación, que se cree que aún tiene un largo recorrido en la clasificación de patrones incompletos.

2.5 Aprendizaje multitarea (MTL)

Nuestro sistema nervioso está preparado para captar un gran número de estímulos y realizar varias tareas con la misma información de entrada. En la mayoría de los problemas de aprendizaje máquina, el modelo se entrena para realizar una sola tarea. Aunque de esta forma se suelen conseguir buenos resultados, se pierde información proveniente de los puntos en común entre tareas. Es decir, al resolver varias tareas al mismo tiempo se descubren patrones que la máquina no puede ver al entrenar por separado, y se ha demostrado que de esta forma el modelo es capaz de generalizar mejor [78]. Esta aproximación es conocida como aprendizaje multitarea (*Multi-Task Learning*, MTL).

Esta técnica ha sido empleada en una gran variedad de aplicaciones del aprendizaje máquina, desde procesado de lenguaje natural [79], hasta reconocimiento de voz [80] y procesado de imagen [81]. Básicamente, en este campo, se puede decir que se está haciendo MTL cuando se entrena una máquina para optimizar más de una función de error. Caruana [82] resume el objetivo del MTL como: "el MTL mejora la generalización del modelo mediante la explotación de información contenida en tareas relacionadas". Este trabajo explora las capacidades del MTL sobre redes neuronales profundas, por lo que esta sección se centra en el estudio del MTL sobre estas redes. Sin embargo, es importante comentar que la teoría presentada aquí aplica perfectamente en el caso de redes superficiales.

2.5.1 Conceptos básicos

Se puede ver el aprendizaje multitarea desde diferentes perspectivas. Biológicamente, es un intento de emular el aprendizaje de los humanos. Por ejemplo, si pensamos en nuestro sistema de vision, mirando una escena somos capaces de resolver varias tareas al mismo tiempo: detectar objetos, segmentarlos, pero también estimar el estado de ánimo de las personas, su país o incluso el tiempo que hace mirando su vestimenta. Desde la perspectiva del aprendizaje automático, se puede ver como una forma de transferencia inductiva. Recordemos que un problema de reconocimiento de patrones consiste en crear un modelo que mapee un vector de características de entrada a una variable objetivo. Se entiende por aprendizaje

inductivo al proceso de inferencia que construye el modelo. Así, la transferencia inductiva puede ayudar a mejorar el modelo mediante un sesgo inductivo que ayuda a seleccionar unas hipótesis sobre otras, es decir, añade restricciones en el espacio de soluciones durante la etapa de optimización de los parámetros y hace que la máquina prefiera unas soluciones sobre otras. Esto, generalmente, ayuda a mejorar la capacidad de generalización del modelo.

Aunque este concepto del sesgo inductivo es la idea intrínseca del MTL, existen varios motivos que hace que esta técnica funcione bien en muchas ocasiones. Si se consideran dos tareas relacionadas A y B, los beneficios del MTL sobre el entrenamiento de redes neuronales artificiales son:

- **Aumento de datos implícito.** MTL incrementa el tamaño de la muestra de entrenamiento del modelo. Toda tarea tiene un ruido asociado en los datos, y cuando un modelo se entrena para realizar A, se quiere que éste ignore el ruido y generalice bien. Diferentes tareas tienen diferentes patrones de ruido. Así, al aprender A y B al mismo tiempo, el modelo aprende una mejor representación conjunta de las tareas y generaliza mejor.

- **Atención.** En cualquier problema multidimensional, el modelo debe diferenciar entre características relevantes e irrelevantes. MTL puede ayudar al modelo a poner atención sobre las que realmente son importantes para resolver la tarea.

- **Espionaje.** Algunas características G son más fáciles de aprender para una tarea B que para otra A. Esto puede ser debido a que A interactúa con estas características de una forma más compleja, o incluso porque otras características impiden al modelo aprender G al ser entrenado para resolver A. Con MTL, el modelo puede aprender G a través de B. La manera más sencilla es mediante *hints* [83], es decir, entrenando directamente el modelo para predecir las características más importantes.

- **Representaciones sesgadas.** MTL introduce un sesgo al modelo para preferir unas representaciones de los datos que puedan ser útiles para otras tareas. Esto ayuda al modelo a generalizar en situaciones donde hay nuevas tareas de naturaleza similar [84].

- **Regularización.** Finalmente, MTL actúa como regularizador introduciendo un sesgo en el aprendizaje. Así, se reduce el riesgo de sobreajuste y complejidad del modelo.

2.5.2 MTL en aprendizaje profundo

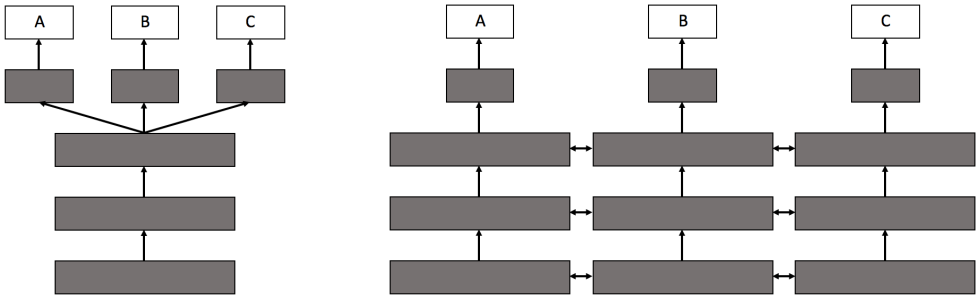
Las dos formas más comunes de incluir MTL en redes neuronales profundas se basan en la manera de compartir los parámetros de las capas ocultas:

- **Intercambio fuerte de parámetros.** Ésta es la aproximación más común en redes neuronales [82]. Básicamente, todas las tareas comparten las mismas capas ocultas de la red, pero cada una de ellas posee capas específicas de salida (Figura 2.4). Así, se reduce en gran medida el sobreajuste. De hecho, en [85] se demuestra que esta arquitectura reduce el sobreajuste en un orden de N –donde N es el número de tareas– comparado con la arquitectura monotarea. Es sencillo de entender, conforme el modelo aprende más tareas, las capas ocultas deben aprender una representación de todas ellas al mismo tiempo.

- **Intercambio suave de parámetros.** En este caso, cada tarea posee su propio modelo con sus propios parámetros. Lo que se hace es regular la distancia entre los parámetros de los modelos para ajustarlos a ser lo más parecidos posibles. Las restricciones aplicadas en este tipo de entrenamiento se inspiran en técnicas de regularización usadas en otros modelos. Un esquema ejemplo se representa en la Figura 2.4.

2.5.2.1 Trabajos recientes

Aunque muchos trabajos recientes de aprendizaje profundo han usado MTL –explícita o implícitamente– como parte de sus modelos, todos ellos utilizan una de las dos aproximaciones que se han presentado. Por contra, sólo unos pocos artículos se han centrado en el desarrollo de nuevos métodos de aprendizaje multitarea sobre redes neuronales profundas. Algunos de los trabajos más destacados son:



(a) (b)

Figura 2.4: Aprendizaje multitarea en redes neuronales profundas. (a) Intercambio fuerte de parámetros. En esta arquitectura, todas las tareas comparten las mismas capas ocultas de la red, pero cada una de ellas posee capas específicas de salida. (b) Intercambio suave de parámetros. Aquí, cada tarea posee su propio modelo con sus propios parámetros, regulándose la distancia entre ellos mediante resctricciones.

- **Redes relacionales profundas.** En MTL para procesado de imagen con redes convolucionales, las aproximaciones normalmente comparten las capas convolucionales y poseen unas capas específicas para cada tarea. En [86] se muestra un método para mejorar estos modelos, basado en colocar unas matrices antes de las capas de salida que aprenden la relación entre las diferentes tareas, similar a algunos modelos bayesianos. Esta arquitectura ha demostrado funcionar bien en problemas típicos de visión por computador.

- **Características compartidas adaptativas.** En [87] se presenta una aproximación que comienza por una red pequeña que se va ampliando dinámicamente durante el entrenamiento utilizando un criterio que promueve la agrupación de tareas similares. Este método tiene la desventaja de que puede no conseguir un modelo globalmente óptimo, terminando con una rama independiente para cada una de las tareas.

- **Ponderación de pérdidas con incertidumbre.** En lugar de una estructura compartida, [88] propone una arquitectura profunda en la que se combinan diferentes funciones de error según la incertidumbre de cada tarea. Después se ajusta el peso relativo de cada una de las tareas en la función de coste en base a la maximización de una probabilidad gausiana.

Además de los mencionados, existen más estudios que podrían destacarse en el ámbito de las redes neuronales profundas con aprendizaje multitarea. Se puede encontrar más información al respecto en [78].

2.6 Discusión

En este capítulo se han presentado las diferentes técnicas estudiadas en la literatura a la hora de imputar valores perdidos, agrupándolas en grandes bloques. En primer lugar, se ha revisado el método clásico de imputación con el algoritmo EM. Tras una revisión de las técnicas estadísticas clásicas, se han introducido los modelos basados en aprendizaje máquina. Finalmente, se ha acabado con una breve revisión de técnicas actuales de aprendizaje profundo.

Como se ha comentado, en esta tesis se presenta un nuevo modelo de imputación de valores perdidos basado en SDAEs, así como la extensión del mismo para la clasificación de patrones incompletos. Así, aunque existen ya otros trabajos que explotan las capacidades de los SDAEs para imputar, los presentados aquí aportan conceptos innovadores a la hora de entrenar estas redes y conseguir una mejora significativa de los resultados.

El capítulo termina con la revisión de los conceptos básicos del aprendizaje multitarea, haciendo un breve resumen de los trabajos recientes centrados en aprendizaje profundo. Aunque estos estudios son prometedores, las arquitecturas tradicionales de MTL siguen predominando en la mayoría de los trabajos. A pesar de todo, este tipo de aprendizaje está ganando protagonismo con el auge de las redes neuronales profundas, y conviene seguir explorando su verdadero potencial.

Hazlo o no lo hagas, pero no lo intentes.

Star Wars: El imperio contraataca

CAPÍTULO

3

Mejora de una imputación con redes profundas

El procesado de datos está siendo un campo muy estudiado en los últimos años. Además, se están llevando a cabo grandes esfuerzos para desarrollar mecanismos de Aprendizaje Profundo que sean capaces de extraer información útil de enormes cantidades de datos. Es por esto que se cree en la necesidad de aprovechar las capacidades de las redes profundas para luchar contra el problema de los valores perdidos anteriormente presentado.

Como se ha visto antes, no son muchos los trabajos que se pueden encontrar en la literatura sobre imputación de valores perdidos con máquinas profundas. En este capítulo se presenta un método de imputación mediante máquinas SDAE que se entrenan considerando dos conceptos nuevos que ayudan a mejorar las prestaciones de la red: el borrado y la compensación. Se trata del trabajo correspondiente a [89] y [90]. En la primera publicación, se presenta el método original y el concepto de borrado, mientras que la segunda profundiza en este concepto y estudia la eficiencia de la compensación entre los conjuntos de entrenamiento y test, desequilibrados por el efecto del borrado. Esta compensación resulta ser muy relevante para mejorar la calidad de imputaciones realizadas. Además, se propone un método

que se beneficia de las ventajas del borrado y de la compensación para aumentar las prestaciones en clasificación de una red profunda tipo SDAE.

Este capítulo está organizado de la siguiente forma. En la siguiente sección se presenta el método propuesto para la imputación de valores perdidos con redes profundas. Después, se verá cómo aplicar este método para la clasificación de patrones incompletos. Los conceptos de borrado y compensación se utilizarán en ambos casos. Finalmente, se presentarán los resultados de todos los escenarios.

3.1 Imputación mediante SDAEs

Según los estudios previos sobre técnicas de imputación, la mayoría de ellas se basan en el uso de las muestras completas del conjunto de entrada para la obtención de la información desconocida. Sin embargo, estas técnicas no tienen en cuenta la información conocida de las muestras incompletas, algo que puede ser clave para la imputación.

En esta sección se presenta un método que considera los valores perdidos a partir de toda la información disponible en los datos con el fin de resolver la tarea principal de clasificación. El método recurre a tres elementos: pre-imputación, borrado y compensación. Al mismo tiempo, se sustenta en el uso de la máquinas profundas, para obtener ventaja de su alta capacidad de representación.

En particular, en este trabajo se emplean máquinas Stacked Denoising Autoencoders (SDAE). Como se ha presentado, los SDAEs son una extensión profunda de los DAE y, junto a las Deep Belief Networks (DBN), son parte del ampliamente conocido aprendizaje de características, un conjunto de técnicas caracterizadas por su capacidad de obtener diferentes representaciones jerárquicas de las características de los datos. Básicamente, un SDAE es una máquina profunda que se construye mediante la apilación de DAE entrenados secuencialmente. Se trata de redes muy utilizadas por su elevada capacidad de representación y por su robustez al ruido que puedan incluir las entradas.

Cuando un conjunto de datos presenta en algunas de sus muestras valores sin asignar (valores perdidos), es habitual inicializarlos con valores numéricos que puedan ser candidatos próximos a los valores reales. De hecho, se intenta evitar la distorsión de la distribución de los datos de entrada. En este estudio, estos valores

pre-imputados pueden considerarse versiones ruidosas de los valores reales. En esta idea se fundamenta el empleo de los SDAE para obtener un método eficiente de imputación.

En este apartado, el diseño y entrenamiento de los SDAEs se lleva a cabo según [91]. Además, se emplean DAEs expansivos para ir construyendo las diferentes capas ocultas de la red. Tras comprobar su buen funcionamiento sobre diferentes conjuntos de datos, el factor de expansión de cada DAE se ha establecido en un 75 % de la dimensión de la entrada al mismo. Así, primero se entrena un DAE mediante el algoritmo que se verá a continuación. Una vez entrenado, se conserva sólo la capa oculta de este primer DAE (de dimensión $h_1 = 0{,}75 \times D$, siendo D la dimensión de los datos de entrada), se fijan sus pesos y se entrena otro DAE que codifica las h_1 salidas ocultas en $h_2 = 0{,}75 \times h_1$ nuevos valores que, posteriormente, se decodifican a la dimensión original de los datos. Y se añade una tercera capa oculta de dimensión $h_3 = 0{,}75 \times h_2$ procediendo de igual manera que en la anterior. Como se indicó anteriormente, la entrada de cada DAE se corrompe con ruido para aumentar la robustez de la red completa. Finalmente, se añade una capa de salida lineal y se refina toda la red para la reconstrucción del conjunto original de datos (Figura 3.1).

3.1.1 Algoritmo de Retropropagación

El algoritmo de aprendizaje comúnmente utilizado en el entrenamiento de este tipo de redes es el de retropropagación o Back-Propagation (BP) [92]. Se trata de un procedimiento de descenso por gradiente aplicado sobre el error cuadrático. Este método se basa en un ciclo de propagación-adaptación en dos fases, una propagación hacia delante y otra hacia atrás. En la primera (forward), se presenta un patrón a la entrada y éste se propaga por toda la red hasta generar una salida. En la fase de propagación hacia atrás (backward), se obtiene un error entre esta salida y la original, el cual se propaga hacia todas las neuronas ocultas y hace que la red se ajuste hasta minimizar el error total. Además del BP estándar, existen diversas alternativas para optimizar los pesos de la red [93, 94, 95]: algoritmo de gradiente conjugado, gradiente conjugado escalado, métodos quasi-Newton, algoritmo Levenberg-Marquardt o descenso por gradiente estocástico.

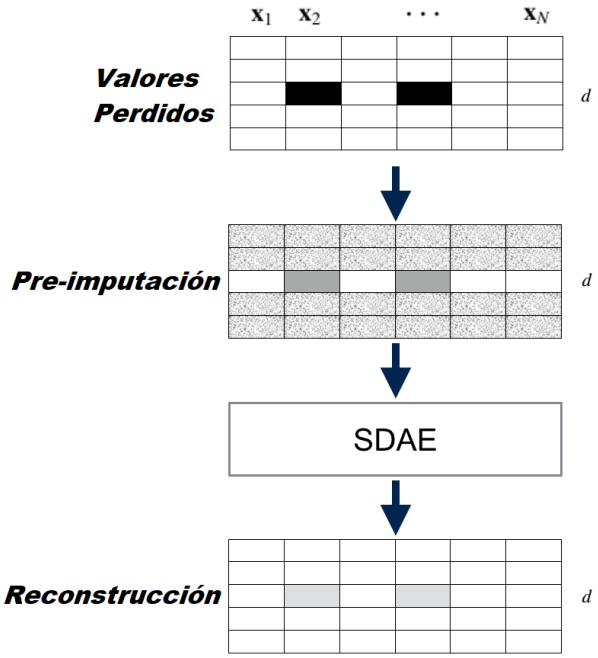


Figura 3.1: Imputación con un Stacked Denoising Autoencoder. En esta imagen se muestra el proceso de cómo se utiliza un SDAE para imputar valores perdidos partiendo de una versión pre-imputada de los mismos. Los valores en negro muestran los valores desconocidos del conjunto original. En este caso, sólo se consideran *missing* en una característica d. En la etapa de pre-imputación, los valores grises oscuros representan los obtenidos por el método de pre-imputación y los corrompidos con ruido están representados por las muestras punteadas. Finalmente, las muestras grises claras de la última fase indican los valores reconstruidos.

De los comentados, uno de los métodos más utilizados en el entrenamiento de DAEs es el Descenso por Gradiente Estocástico (SGD), que permite llegar a una solución mediante una aproximación estocástica de la función de error. Con el objetivo de hacer uso de toda la información disponible en los datos durante el entrenamiento, se propone una ligera modificación del SGD [89]. De esta forma, considerando un conjunto de datos $\{\mathbf{x}_n\}_{n=1}^{N}, n = 1, ..., N$, donde $\mathbf{x}_n$ es el patrón n-ésimo, se define otro conjunto de vectores $\{\mathbf{m}_n\}_{n=1}^{N}, n = 1, ..., N$ de la misma dimensión para determinar la presencia o ausencia de valores perdidos según la siguiente expresión:

$$m_{nd} = \begin{cases} 1, & if\ x_{nd}\ es\ un\ valor\ conocido \\ 0, & if\ x_{nd}\ es\ desconocido \end{cases} \tag{3.1}$$

donde d $(1 \leq d \leq D)$ denota las componentes. Así, los DAEs usados para la construcción del SDAE se entrenan para minimizar la siguiente función de Suma de Errores Cuadráticos (SSE)

$$E = \sum_{n=1}^{N} \parallel (\mathbf{z}_n - \mathbf{x}_n) \odot \mathbf{m}_n \parallel^2 \tag{3.2}$$

donde $\mathbf{z}_n$ es la salida del SDAE y $\odot$ representa el producto de Hadamard. De esta forma, todos los datos disponibles se emplean en el entrenamiento de la red, ya que se manejan todos los valores de las muestras completas y todos los valores observados de las muestras con valores perdidos.

Durante el entrenamiento de un DAE, para cada muestra de entrada $\mathbf{x}_n$, todos los pesos de la capa oculta se actualizarán, mientras que no ocurre lo mismo con los pesos de salida: sólo se actualizarán aquellos conectados con nodos de salida con valores en $\mathbf{m}_n$ iguales a uno.

Como se ha comentado anteriormente, el método propuesto de imputación con SDAE se hace uso de tres técnicas que se explican a continuación: la pre-imputación, el borrado y la compensación.

3.1.2 Pre-imputación

La pre-imputación consiste en asignar valores numéricos a las características de las muestras que presentan valores perdidos. Una vez escogido e implementado el

método de pre-imputación, los valores pre-imputados pueden considerarse valores ruidosos respecto a los reales.

Por tanto, el primer paso del método de imputación será una pre-imputación para rellenar los valores desconocidos. Después, se usará el conjunto pre-imputado como conjunto de entrenamiento de un SDAE para imputación y se comprobará si los valores a la salida de la red profunda se acercan más a los reales que los obtenidos en la pre-imputación.

Existen muchos métodos de imputación en la literatura que pueden emplearse para pre-imputar [5, 36]. El método de pre-imputación más simple consiste en rellenar los valores perdidos con ceros. Sin embargo, parece más adecuado realizar una pre-imputación con la que se obtengan valores más próximos a los verdaderos para mejorar la calidad de la imputación final. En este trabajo, para evaluar las prestaciones de la imputación con SDAE, se emplearán varios métodos de diversa complejidad y naturaleza. En concreto, se implementarán MLPs para imputación, el método de Descomposición en Valores Singulares (SVD) y MICE.

3.1.3 Borrado

Una vez realizada la pre-imputación de los valores perdidos, estos valores pre-imputados, junto con los valores observados, se emplean como entradas al SDAE durante el entrenamiento, empleándose como salidas deseadas sólo los valores conocidos según se muestra en la fórmula (3.2). Emplear como salidas deseadas los valores pre-imputados tiene el inconveniente de que el entrenamiento se puede sesgar hacia resultados sub-óptimos.

Se cree que el funcionamiento del método sería mejor si se ayuda explícitamente a la red a aprender la reconstrucción de los valores desconocidos. Esto se puede conseguir añadiendo artificialmente valores perdidos. Suponiendo, sin perder generalidad, que sólo la característica *d* presenta valores perdidos con una tasa μ en tanto por uno, es decir ($0 \leq \mu \leq 1$), ésta se puede incrementar artificialmente en un valor ε con $0 \leq \varepsilon \leq 1 - \mu$. De esta manera, $(\mu + \varepsilon)N$ representa la tasa efectiva de valores perdidos respecto al número total de muestras N. Así, la red se entrena con N muestras de entrada que presentan $(\mu + \varepsilon)N$ valores perdidos en la característica *d*, pero usando como salidas deseadas N muestras que presentan

μN valores desconocidos en la misma característica. Mientras μ viene dado por el problema, ε es un parámetro del entrenamiento que habrá que asignar mediante validación cruzada. Es de esperar que los nuevos valores perdidos introducidos a las entradas, cuyos valores reales se emplean como objetivos, sirvan para aprender mejor la reconstrucción de la característica d.

En la Figura 3.2 se muestra el proceso de borrado. En ella, los μN valores perdidos se representan en color negro y los εN valores borrados están tachados. Los datos grises representan estas muestras después de la pre-imputación y antes del proceso de aprendizaje profundo, donde las más oscuras indican que sus valores originales son conocidos y pueden usarse durante el entrenamiento de la red. A la salida de la red, se han reconstruido todos estos valores, tanto los perdidos como los borrados.

Esta idea de borrado fue introducida por primera vez en el trabajo [89] y más tarde analizada y detallada en [90], donde se completa con el concepto de compensación, que se introduce a continuación.

3.1.4 Compensación

A pesar de las ventajas que el borrado puede suponer para la imputación de valores perdidos, existe un inconveniente debido al desequilibrio que el borrado introduce entre los conjuntos de entrenamiento y test. Con el borrado de valores en la característica *d*, se pasa de una tasa de valores perdidos μ y valores completos $1-\mu$, a una tasa $\mu+\varepsilon$ de valores perdidos y $1-(\mu+\varepsilon)$ de valores completos. Sin embargo, no ocurre lo mismo en el conjunto de test, que sigue teniendo una tasa μ de valores perdidos y $1-\mu$ de muestras completas. En este escenario, si el valor de ε es pequeño este desequilibrio no es grave, pero si no lo es, se producirá una degradación de las prestaciones, que se podrá enmendar mediante una compensación apropiada. Así, merece la pena analizar la compensación exacta y proponer así un método efectivo y práctico de compensación.

Antes de llevar a cabo la tarea de borrado sobre el conjunto de datos, la función de error SSE se puede expresar como:

$$E_1 = E_C + E_I \tag{3.3}$$

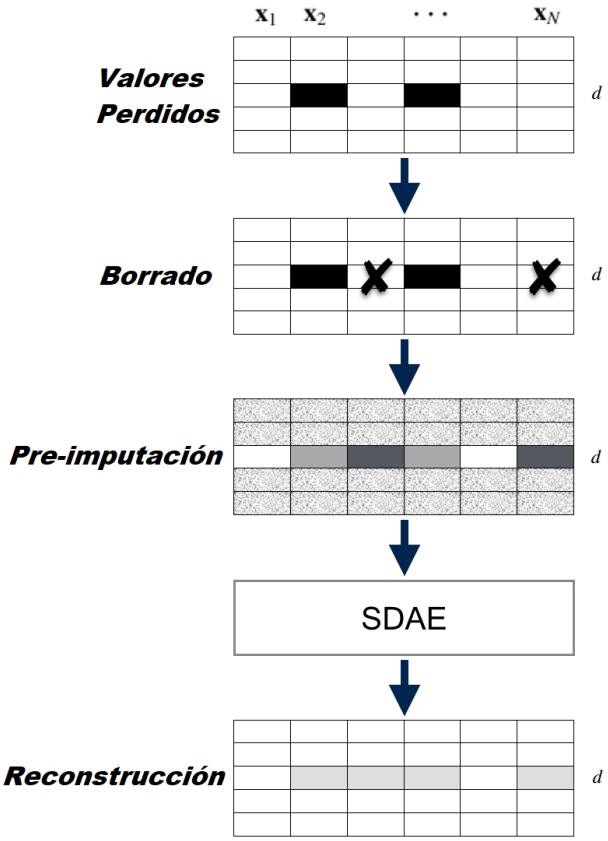


Figura 3.2: Proceso de borrado. Los valores perdidos en la característica d se muestran en negro y el borrado como una cruz. Los colores gris claro y oscuro en la etapa de pre-imputación se usan para mostrar los valores perdidos y borrados, respectivamente. Los valores conocidos de las muestras borradas se emplearán como objetivo durante el entrenamiento. Finalmente, las muestras grises más claras a la salida representan la reconstrucción por la red profunda.

donde E_C y E_I se corresponden con el SSE de las muestras completas e incompletas, respectivamente. Esta expresión puede ser reescrita en términos de las tasas μ y ε de la forma

$$E_1 = (1-\mu)N\overline{E}_C + \mu N\overline{E}_I \tag{3.4}$$

donde $\overline{E}_C$ y $\overline{E}_I$ son el Error Cuadrático Medio (MSE) de las muestras completas e incompletas, respectivamente. El primer término representa la contribución del error cuadrático de las $(1-\mu)N$ muestras completas, y el segundo término representa la contribución de las μN muestras con valores perdidos.

Después del proceso de borrado, la distribución de los datos de entrenamiento cambia mientras que la del conjunto de test permanece inalterada. Al mismo tiempo, los términos del SSE cambiarán debido a que las tasas de *missing* han sido modificadas, quedando de la forma

$$E_2 = E'_C + E'_I \tag{3.5}$$

donde los dos términos de E incorporan los valores adicionales de muestras desconocidas εN según la siguiente expresión:

$$E_2 = (1-(\mu+\varepsilon))N\overline{E'}_C + (\mu+\varepsilon)N\overline{E'}_I \tag{3.6}$$

donde $\overline{E'}_C$ y $\overline{E'}_I$ son el MSE de las muestras completas e incompletas después del borrado, respectivamente.

Con el objetivo de tratar ambos conjuntos de entrenamiento y test de la misma forma, compensando la diferencia introducida en sus distribuciones, se propone un mecanismo de equilibrado. Para obtener un equilibrado exacto después del borrado, el entrenamiento debe ser llevado a cabo con una tasa efectiva de muestras completas $1-\mu$ y muestras incompletas μ, es decir, como en (3.4).

Los experimentos de este trabajo mostraron que el equilibrado exacto de los conjuntos de datos no es siempre la mejor opción, por lo que se exploran un rango de posibles valores para el parámetro en cuestión. Así, dados unos valores de μ y ε, se propone la siguiente función convexa a optimizar:

$$E = \alpha E'_C + (1-\alpha)E'_I \tag{3.7}$$

donde α es el parámetro de equilibrado, que irá entre $\alpha = 0$ (situación donde sólo se optimiza el error de las muestras completas) hasta $\alpha = 1$ (sólo los errores de las

muestras incompletas). Se debe puntualizar que para $\alpha = 0{,}5$ ambos errores están equilibrados, es decir, (3.7) es equivalente a (3.5). El valor óptimo de α se obtendrá mediante validación cruzada teniendo en cuenta la precisión de la red sobre el conjunto de validación sin borrado. El objetivo es encontrar los pesos óptimos de la red profunda que obtienen mejores resultados sobre un conjunto de validación similar al de test que, recordemos, tendrá la tasa de valores perdidos original.

Después del proceso de borrado, el número de muestras con valores perdidos se incrementa y el número de muestras completas decrece. Para compensar este cambio, se espera obtener un valor óptimo de α mayor que 0.5 que ponga mayor atención en los errores de las muestras completas (que ha disminuido) que en los de las incompletas (ha aumentado). Como se verá en la Sección 3.3, los resultados apoyarán la utilidad de este procedimiento.

3.2 Clasificación de patrones incompletos

Hasta ahora la presentación del método se ha centrado en la imputación de valores perdidos. Sin embargo, los SDAEs sirven frecuentemente para resolver problemas de clasificación y de regresión. Se ha demostrado que el potencial de un SDAE reside en el pre-entrenamiento no supervisado de los DAEs que se apilan para crear la red profunda. Este pre-entrenamiento se utiliza para inicializar los pesos de las capas ocultas de la red, resultando de gran ayuda para solventar el problema de optimización. Además, la adición de ruido a la entrada hace que el modelo sea más robusto a distorsiones.

El entrenamiento de un SDAE para clasificación tiene dos etapas principales: una primera en la que se diseña y entrena un conjunto de DAEs de forma no supervisada tal y como se ha descrito en la Section 3.1, y una segunda etapa orientada a la resolución del problema mediante el entrenamiento por gradiente (DGS) de un clasificador colocado tras la última capa oculta. Este entrenamiento permite diferentes alternativas: se pueden fijar los pesos de las capas ocultas del SDAE y entrenar sólo el clasificador (en este caso, es aconsejable realizar al final un refinamiento global de la red) o se puede, tras el pre-entrenamiento, entrenar conjuntamente el clasificador y los pesos ocultos.

En esta parte de la tesis se analizan las prestaciones de un SDAE para resolver problemas de clasificación con valores perdidos en alguna de sus características. Se pretende demostrar que, gracias a la gran capacidad de representación de los SDAEs junto con la incorporación del borrado y la compensación cuando hay valores perdidos, se pueden mejorar las prestaciones de dichas máquinas en tareas de clasificación.

3.3 Experimentos

Una vez presentados los métodos de imputación y clasificación de patrones incompletos, en esta sección se analiza el impacto del borrado y la compensación. Primero, se analizan los efectos de estas dos técnicas sobre una imputación. Para ello, se comparan varios procedimientos de imputación frecuentemente usados en la práctica con un SDAE con borrado y compensación. Es decir, primero se imputan los valores desconocidos mediante los métodos mencionados en el apartado 3.1.2. Así, es sencillo calcular las mejoras obtenidas por las técnicas que se proponen en este apartado de la tesis. Finalmente, se evalúan las técnicas de borrado y compensación con un SDAE a la hora de clasificar patrones con valores perdidos.

3.3.1 Conjuntos de datos

Para llevar a cabo la comparación con otras del estado del arte, se seleccionan seis conjuntos de datos de diferentes tamaños y características. Los detalles de estos se pueden encontrar resumidos en la Tabla 3.1. Es importante mencionar que sólo se utilizan conjuntos de datos completos, con el objetivo de insertar *missing* artificial y poder verificar el correcto funcionamiento de los algoritmos propuestos. Estos valores artificialmente perdidos se insertan aleatoriamente en una sola característica del conjunto, excepto en el conjunto AREM, donde se seleccionan cuatro características aleatoriamente con el objetivo de ampliar el alcance del estudio y evitar la pérdida de generalidad. Como se ha visto en capítulos anteriores, existen diferentes tipos de *missing*, lo que puede conllevar diferentes comportamientos de los algoritmos. En este caso, se implementa el tipo más estricto, MCAR, donde

los valores desconocidos se insertan completamente de forma aleatoria y no tienen relación con los valores observados.

Cuadro 3.1: Conjuntos de datos. Principales características.

	Muestras		
	Total	Clases	Características
MAGIC Gamma Telescope (M)	19020	2	11
Pima Indians Diabetes (P)	768	2	8
Twonorm (T)	7400	2	20
Sensorless Drive Diagnosis (S)	58509	11	48
Gas Sensor Array Drift (G)	13910	6	128
AREM (A)	42240	7	6

Todos los conjuntos de datos utilizados en esta parte de la tesis son públicos y se encuentran en el repositorio UCI [96], excepto *Twonorm* que está disponible en [97]. A continuación, se presenta una breve descripción de cada uno de ellos:

- **Magic Gamma Telescope.** Básicamente, este conjunto de datos consiste en una simulación de partículas de energía gamma usando una técnica de imagen con un telescopio Cherenkov. La información disponible consiste en pulsos creados por los llamados fotones de Cherenkov dispuestos sobre un plano, la cámara. Dependiendo de la energía del rayo gamma, se recolectan un total de unos pocos cientos a unos 10000 fotones de Cherenkov, en patrones (llamados imágenes de la lluvia), lo que permite discriminar estadísticamente los causados por gammas primarios de las imágenes de hadrones causadas en la atmófera superior.

- **Pima Indians Diabetes.** Conjunto original del Instituto Nacional de Diabetes, Digestivo y Enfermedades Renales de EEUU. El objetivo de este conjunto es la predicción de la diabetes basado en ciertas medidas de un diagnóstico: número de embarazos del paciente, su índice de masa corporal, edad, etc.

- **Sensorless Drive Diagnosis.** Conjunto formado por la extracción de señales eléctricas de un motor. Tiene algunas señales intactas y algunas defectuosas,

resultando en 11 clases diferentes bajo distintas condiciones. Cada condición ha sido medida varias veces bajo 12 situaciones distintas de operación, es decir, diferentes velocidades, momentos de carga y fuerzas de carga.

- **Gas Sensor Array Drift.** Conjunto creado durante 36 meses en una plataforma de suministro de gas de la Universidad de California en San Diego. Consta de 13910 medidas obtenidas con 16 sensores químicos en diferentes simulaciones de 6 gases a varios niveles de concentración. El objetivo de este conjunto es utilizar técnicas de aprendizaje automático para estudiar la concentración de los diferentes sensores.

- **Activity Recognition system based on Multisensor data fusion (AREM).** Este problema está preparado para predecir el tipo de actividad llevada a cabo por el usuario. El conjunto se genera a partir de una red de sensores que monitorizan temporalmente los movimientos de un usuario. El sistema de sensores se situa en el pecho y rodillas del usuario. Por cada actividad, se tienen 15 secuencias temporales de entrada por lo que el conjunto está formado por 480 secuencias y un total de 42240 instancias.

- **Twonorm.** Una implementación del ejemplo de Leo Breiman [98]. Cada clase es creada a partir de una distribución normal con varianza unitaria. La clase 1 tiene media $(a, a, ...a)$ y la clase 2 $(-a, -a, ...-a)$, donde $a = 2/\sqrt{20}$. Los trabajos de Breiman indican que el error de clasificación teórico esperado es de 2.3 %.

Durante el entrenamiento de las redes, cada conjunto de datos se divide en entrenamiento (80 %) y test (20 %). Como se ha mencionado en la sección 3.1, todas las redes se diseñan para tener 3 capas ocultas expansivas con una tasa de expansión del 75 % de la capa anterior. Además, se implementa validación cruzada de 5-*folds* para obtener los valores óptimos de los hiperparámetros. Así, con el objetivo de obtener resultados de las técnicas estudiadas sobre diferentes escenarios, se implementan distintos porcentajes de muestras desconocidas *missing* μ y borradas ε. En particular, se prueban todas las combinaciones para $\mu = \{0{,}1, 0{,}2, 0{,}3\}$ y $\varepsilon = \{0{,}25, 0{,}5, 0{,}75\}$.

3.3.2 Métodos de pre-imputación

Como se ha descrito en el Capítulo 2, existe una gran variedad de mecanismos de imputación de valores perdidos en la literatura. De entre esta variedad, tres de ellos se seleccionan en este trabajo con el objetivo de cubrir una muestra representativa de la literatura. Estas técnicas serán usados para pre-imputar los valores desconocidos del conjunto de entrenamiento. Por este motivo, nos referimos a ellos como métodos de pre-imputación. De nuevo, para evitar la pérdida de generalidad del estudio, se implementarán tres técnicas de pre-imputación con diferentes capacidades y niveles de complejidad. Las técnicas son: perceptrón multicapa (MLP), descomposición en valores singulares (SVD) e imputación múltiple mediante ecuaciones encadenadas (MICE). De entre éstos, el último método se trata de una técnica de imputación múltiple que, como se ha comentado, forma parte de una de las técnicas avanzadas imputación. Se estudia si las técnicas basadas en borrado y compensación son capaces de competir o incluso mejorar en los escenarios que se plantean a continuación.

3.3.3 Resultados en problemas de imputación

En las Tablas 3.2 y 3.3 se muestran los resultados en imputación para cada conjunto de datos y método. Cada valor se corresponde con el error entre valores imputados y reales según (3.2) y (3.7), respectivamente. Se presentan los errores obtenidos mediante los SDAEs diseñados y entrenados, así como los métodos de pre-imputación directamente aplicados sobre el conjunto con valores perdidos. Se presentan tres técnicas distintas basadas en redes profundas: un SDAE sin borrado ni compensación, un SDAE sólo con borrado, y un SDAE con borrado y compensación. Los valores óptimos para ε y α en cada uno de los escenarios se presentan entre paréntesis, el mejor resultado para cada grupo de pre-imputación en cursiva y el mejor resultado global para cada percentaje de *missing*, en negrita. Es importante mencionar que dos resultados se consideran estadísticamente diferentes si $\phi_1 - \phi_2 \geq \frac{\sigma_1+\sigma_2}{2}$, donde ϕ representa la media y σ la desviación estándar del resultado.

Cuadro 3.2: Resultados de imputación. Se presenta el error descrito en (3.2) y (3.7) para métodos sin y con compensación, respetivamente. En las columnas se muestran los diferentes porcentajes de valores perdidos μ y en las filas los diferentes procedimientos según la nomenclatura: primera inicial del conjunto de datos - *método de pre-imputación* - procedimiento profundo *(tasa de borrado, tasa de compensación)*. Por cada conjunto, el mejor resultado de cada familia de métodos de pre-imputación se muestra en cursiva y el mejor resultado global para un percentaje de *missing* en negrita.

Procedimiento \ μ	0.1	0.2	0.3
M-MLP	0.5 ± 0.05	0.4 ± 0.05	0.4 ± 0.09
M-MLP-SDAE	0.2 ± 0.06	0.4 ± 0.02	0.2 ± 0.06
M-MLP-SDAE (ε)	0.1 ± 0.02 (0.75)	0.2 ± 0.06 (0.75)	**0.1 ± 0.02 (0.5)**
M-MLP-SDAE (ε, α)	**0.05 ± 0.01 (0.75, 0.6)**	**0.1 ± 0.07 (0.75, 0.3)**	**0.08 ± 0.006 (0.75, 0.7)**
M-SVD	0.7 ± 0.05	0.6 ± 0.05	0.7 ± 0.04
M-SVD-SDAE	0.3 ± 0.03	0.5 ± 0.06	0.3 ± 0.02
M-SVD-SDAE (ε)	0.15 ± 0.01 (0.5)	0.1 ± 0.01 (0.5)	0.25 ± 0.03 (0.75)
M-SVD-SDAE (ε, α)	**0.06 ± 0.01 (0.75, 0.2)**	**0.08 ± 0.01 (0.5, 0.7)**	0.11 ± 0.01 (0.75, 0.5)
M-$MICE$	**0.04 ± 0.004**	**0.08 ± 0.005**	**0.1 ± 0.01**
M-$MICE$-SDAE	**0.05 ± 0.008**	0.15 ± 0.01	**0.1 ± 0.02**
M-$MICE$-SDAE (ε)	**0.04 ± 0.005 (0.5)**	**0.1 ± 0.005 (0.5)**	**0.08 ± 0.03 (0.75)**
M-$MICE$-SDAE (ε, α)	**0.05 ± 0.005 (0.75, 0.2)**	**0.09 ± 0.01 (0.5, 0.7)**	**0.1 ± 0.01 (0.75, 0.5)**
P-MLP	0.5 ± 0.02	0.5 ± 0.01	0.6 ± 0.02
P-MLP-SDAE	0.2 ± 0.05	0.7 ± 0.02	0.6 ± 0.08
P-MLP-SDAE (ε)	0.1 ± 0.01 (0.5)	0.2 ± 0.02 (0.5)	0.09 ± 0.002 (0.5)
P-MLP-SDAE (ε, α)	**0.03 ± 0.005 (0.75, 0.7)**	*0.08 ± 0.01 (0.75, 0.6)*	**0.05 ± 0.006 (0.5, 0.8)**
P-SVD	0.65 ± 0.02	0.7 ± 0.008	0.7 ± 0.02
P-SVD-SDAE	0.4 ± 0.04	0.35 ± 0.04	0.4 ± 0.004
P-SVD-SDAE (ε)	0.2 ± 0.01 (0.75)	0.15 ± 0.02 (0.5)	0.25 ± 0.01 (0.75)
P-SVD-SDAE (ε, α)	*0.05 ± 0.008 (0.75, 0.3)*	**0.07 ± 0.01 (0.75, 0.7)**	**0.06 ± 0.004 (0.75, 0.5)**
P-$MICE$	**0.02 ± 0.008**	0.07 ± 0.004	**0.05 ± 0.008**
P-$MICE$-SDAE	0.04 ± 0.005	0.07 ± 0.005	0.08 ± 0.01
P-$MICE$-SDAE (ε)	**0.02 ± 0.005 (0.75)**	**$0.05 \pm 0.$ (0.5)**	**0.06 ± 0.004 (0.75)**
P-$MICE$-SDAE (ε, α)	**0.02 ± 0.004 (0.75, 0.7)**	**0.06 ± 0.004 (0.5, 0.4)**	**0.05 ± 0.005 (0.75, 0.7)**
T-MLP	0.9 ± 0.01	0.9 ± 0.05	0.8 ± 0.03
T-MLP-SDAE	0.6 ± 0.09	0.8 ± 0.1	0.5 ± 0.05
T-MLP-SDAE (ε)	0.08 ± 0.01 (0.5)	0.1 ± 0.002 (0.5)	0.08 ± 0.01 (0.75)
T-MLP-SDAE (ε, α)	**0.05 ± 0.006 (0.75, 0.6)**	**0.04 ± 0.004 (0.75, 0.7)**	**0.05 ± 0.008 (0.75, 0.7)**
T-SVD	0.75 ± 0.03	0.8 ± 0.05	0.8 ± 0.05
T-SVD-SDAE	0.5 ± 0.02	0.2 ± 0.04	0.4 ± 0.05
T-SVD-SDAE (ε)	*0.14 ± 0.02 (0.5)*	0.2 ± 0.03 (0.75)	0.2 ± 0.02 (0.75)
T-SVD-SDAE (ε, α)	*0.12 ± 0.01 (0.5, 0.2)*	**0.05 ± 0.01 (0.75, 0.5)**	*0.12 ± 0.01 (0.75, 0.4)*
T-$MICE$	0.04 ± 0.01	**0.04 ± 0.001**	0.07 ± 0.005
T-$MICE$-SDAE	0.07 ± 0.006	0.07 ± 0.003	0.08 ± 0.004
T-$MICE$-SDAE (ε)	0.07 ± 0.002 (0.5)	0.05 ± 0.005 (0.5)	**0.05 ± 0.003 (0.5)**
T-$MICE$-SDAE (ε, α)	**0.05 ± 0.003 (0.5, 0.8)**	**0.04 ± 0.002 (0.5, 0.7)**	**0.05 ± 0.01 (0.75, 0.6)**

Cuadro 3.3: Continuación de resultados de imputación. Se presenta el error descrito en (3.2) y (3.7) para métodos sin y con compensación, respetivamente. En las columnas se muestran los diferentes porcentajes de valores perdidos μ y en las filas los diferentes procedimientos según la nomenclatura: primera inicial del conjunto de datos - *método de pre-imputación* - procedimiento profundo *(tasa de borrado, tasa de compensación)*. Por cada conjunto, el mejor resultado de cada familia de métodos de pre-imputación se muestra en cursiva y el mejor resultado global para un percentaje de *missing* en negrita.

Procedimiento \ μ	0.1	0.2	0.3
S-MLP	2.65 ± 0.08	2.82 ± 0.1	2.89 ± 0.07
S-MLP-SDAE	2.36 ± 0.12	2.53 ± 0. 07	2.45 ± 0.08
S-MLP-SDAE (ε)	1.73 ± 0.07 (0.75)	1.9 ± 0.08 (0.5)	1.85 ± 0.02 (0.5)
S-MLP-SDAE (ε, α)	*1.24 ± 0.08 (0.5, 0.4)*	*1.71 ± 0.05 (0.75, 0.6)*	*1.56 ± 0.04 (0.5, 0.6)*
S-SVD	2.45 ± 0.08	2.57 ± 0.05	2.6 ± 0.05
S-SVD-SDAE	2.23 ± 0.1	2.34 ± 0.08	2.4 ± 0.05
S-SVD-SDAE (ε)	1.84 ± 0.06 (0.5)	*1.88 ± 0.02 (0.5)*	2.12 ± 0.02 (0.75)
S-SVD-SDAE (ε, α)	*1.57 ± 0.05 (0.75, 0.6)*	*1.94 ± 0.1 (0.5, 0.8)*	*1.93 ± 0.07 (0.75, 0.7)*
S-$MICE$	**0.82 ± 0.04**	**0.81 ± 0.05**	**0.92 ± 0.03**
S-$MICE$-SDAE	1.47 ± 0.1	1.24 ± 0.02	1.36 ± 0.05
S-$MICE$-SDAE (ε)	1.2 ± 0.06 (0.5)	**0.97 ± 0.1 (0.5)**	1.15 ± 0.06 (0.75)
S-$MICE$-SDAE (ε, α)	**0.91 ± 0.08 (0.75, 0.7)**	**0.85 ± 0.08 (0.75, 0.6)**	**0.87 ± 0.04 (0.5, 0.6)**
G-MLP	2.65 ± 0.05	2.85 ± 0.04	3.12 ± 0.03
G-MLP-SDAE	2.34 ± 0.07	2.54 ± 0.06	2.74 ± 0.06
G-MLP-SDAE (ε)	1.78 ± 0.05 (0.5)	*1.74 ± 0.07 (0.75)*	2.54 ± 0.04 (0.75)
G-MLP-SDAE (ε, α)	*1.56 ± 0.08 (0.75, 0.7)*	*1.82 ± 0.08 (0.75, 0.6)*	*2.37 ± 0.03 (0.75, 0.6)*
G-SVD	2.3 ± 0.04	2.64 ± 0.02	2.57 ± 0.03
G-SVD-SDAE	2.45 ± 0.05	2.51 ± 0.05	2.72 ± 0.1
G-SVD-SDAE (ε)	2.06 ± 0.03 (0.5)	*2.32 ± 0.05 (0.5)*	*2.35 ± 0.06 (0.5)*
G-SVD-SDAE (ε, α)	*1.83 ± 0.06 (0.5, 0.7)*	*2.25 ± 0.04 (0.75, 0.6)*	*2.32 ± 0.05 (0.75, 0.6)*
G-$MICE$	**1.12 ± 0.02**	**1.1 ± 0.02**	**0.92 ± 0.04**
G-$MICE$-SDAE	1.46 ± 0.04	1.34 ± 0.05	1.54 ± 0.06
G-$MICE$-SDAE (ε)	1.23 ± 0.06 (0.5)	1.19 ± 0.04 (0.5)	1.17 ± 0.08 (0.75)
G-$MICE$-SDAE (ε, α)	1.2 ± 0.04 (0.5, 0.5)	**0.92 ± 0.07 (0.75, 0.6)**	**0.95 ± 0.05 (0.75, 0.6)**
A-MLP	0.018 ± 0.002	0.021 ± 0.002	0.021 ± 0.002
A-MLP-SDAE	0.017 ± 0.001	0.019 ± 0.002	0.020 ± 0.002
A-MLP-SDAE (ε)	0.015 ± 0.001 (0.5)	0.019 ± 0.001 (0.25)	**0.017 ± 0.001 (0.5)**
A-MLP-SDAE (ε, α)	**0.012 ± 0.001 (0.5, 0.6)**	*0.018 ± 0.002 (0.5, 0.7)*	**0.017 ± 0.002 (0.5, 0.8)**
A-SVD	0.07 ± 0.001	0.05 ± 0.002	0.1 ± 0.002
A-SVD-SDAE	0.04 ± 0.001	0.03 ± 0.001	0.05 ± 0.002
A-SVD-SDAE (ε)	*0.016 ± 0.001 (0.5)*	*0.018 ± 0.001 (0.75)*	0.02 ± 0.002 (0.5)
A-SVD-SDAE (ε, α)	*0.015 ± 0.002 (0.75, 0.7)*	*0.019 ± 0.001 (0.75, 0.6)*	0.02 ± 0.001 (0.75, 0.6)
A-$MICE$	0.016 ± 0.002	0.019 ± 0.001	0.018 ± 0.001
A-$MICE$-SDAE	0.016 ± 0.001	0.019 ± 0.002	0.017 ± 0.002
A-$MICE$-SDAE (ε)	**0.013 ± 0.002 (0.75)**	**0.016 ± 0.001 (0.5)**	**0.015 ± 0.001 (0.5)**
A-$MICE$-SDAE (ε, α)	**0.012 ± 0.001 (0.75, 0.7)**	**0.016 ± 0.001 (0.5, 0.7)**	**0.016 ± 0.001 (0.5, 0.6)**

3.3.4 Resultados de clasificación

En las Tablas 3.4 y 3.5 se muestran los resultados de clasificación. Para cada conjunto y cada método, los resultados se muestran en términos de precisión de clasificación. Todos ellos se obtienen con 50 repeticiones independientes. Es importante mencionar que tanto la presentación como la comparación de resultados se realizan de la misma forma que en los resultados de imputación, y que los resultados de pre-imputación se obtienen mediante un MLP.

Por cada experimento, las redes profundas se diseñan usando un clasificador lineal a la salida del SDAE. Esto es así porque, en este caso, se obtienen resultados similares a otros métodos no lineales tales como un MLP. En este sentido, se demuestra cómo el SDAE es capaz de llevar el problema a un espacio donde las clases son linealmente separables [99].

3.4 Discusión

Si se analizan los resultados de imputación presentados en la Sección 3.3.3, se puede concluir:

- Los resultados de las pre-imputaciones son siempre mejorados por alguna de las soluciones basadas en SDAEs, incluso para el mejor método de pre-imputación.
- Cuanto mejor es el método de pre-imputación, mejores resultados de imputación se consiguen con la red profunda.
- Las redes SDAE con borrado siempre mejoran el resultado de las mismas sin borrado.

Cuadro 3.4: Precisión de clasificacion para cada experimento. Los diferentes porcentajes de valores perdidos aplicados al conjunto de entrada se muestran en las columnas, mientras que los distintos métodos se presentan en las filas con la siguiente nomenclatura: primera inicial del conjunto de datos - *método de pre-imputación* - procedimiento profundo *(tasa de borrado, tasa de compensación)*. Para cada conjunto, el mejor resultado de la familia de pre-imputaciones se muestra en cursiva, mientras el mejor resultado global para cada valor de μ se presenta en negrita.

Procedimiento \ μ	0.1	0.2	0.3
M-MLP	89.3 ± 1.6	87.3 ± 0.7	88.6 ± 0.4
M-MLP-SDAE	89.8 ± 0.7	89.7 ± 0.2	87.4 ± 0.2
M-MLP-SDAE (ε)	**92 ± 0.2 (0.75)**	**91.6 ± 0.3 (0.5)**	*90.9 ± 0.5 (0.75)*
M-MLP-SDAE (ε, α)	**92.3 ± 0.2 (0.5, 0.6)**	**91.3 ± 0.2 (0.75, 0.6)**	*90.8 ± 0.4 (0.75, 0.7)*
M-SVD	88.5 ± 0.4	87.1 ± 0.2	88.2 ± 0.2
M-SVD-SDAE	90.1 ± 0.6	89.4 ± 0.5	87.5 ± 0.6
M-SVD-SDAE (ε)	91.2 ± 0.3 (0.5)	**91.3 ± 0.2 (0.5)**	88.4 ± 0.4 (0.75)
M-SVD-SDAE (ε, α)	**91.9 ± 0.2 (0.5, 0.5)**	**91.5 ± 0.4 (0.5, 0.7)**	**91.3 ± 0.1 (0.75, 0.4)**
M-$MICE$	89.7 ± 0.6	90.3 ± 0.2	89.4 ± 0.2
M-$MICE$-SDAE	89.6 ± 0.8	**90.8 ± 0.4**	89.3 ± 0.5
M-$MICE$-SDAE (ε)	91.2 ± 0.4 (0.5)	**90.4 ± 0.5 (0.75)**	90.2 ± 0.3 (0.75)
M-$MICE$-SDAE (ε, α)	**92.3 ± 0.6 (0.75, 0.7)**	**91.2 ± 0.4 (0.75, 0.5)**	**91.4 ± 0.5 (0.75, 0.6)**
P-MLP	72.4 ± 1.7	70.3 ± 0.8	74.6 ± 1.3
P-MLP-SDAE	77.3 ± 2.8	72.8 ± 3.4	75 ± 1.2
P-MLP-SDAE (ε)	76.7 ± 1.6 (0.5)	**78 ± 1.3 (0.5)**	**79.8 ± 2.0 (0.5)**
P-MLP-SDAE (ε, α)	**80.8 ± 1.2 (0.75, 0.7)**	**79.4 ± 1.5 (0.75, 0.3)**	**80.5 ± 1.1 (0.5, 0.7)**
P-SVD	74.6 ± 0.4	74.3 ± 0.8	73.2 ± 0.4
P-SVD-SDAE	75.3 ± 0.6	76.2 ± 0.5	75.3 ± 0.7
P-SVD-SDAE (ε)	78.6 ± 0.4 (0.25)	**79.5 ± 0.4 (0.75)**	78.1 ± 0.2 (0.75)
P-SVD-SDAE (ε, α)	**79.5 ± 0.7 (0.5, 0.5)**	78.3 ± 0.2 (0.75, 0.6)	**79.2 ± 0.5 (0.75, 0.7)**
P-$MICE$	77.4 ± 0.7	77.8 ± 0.3	77.5 ± 0.4
P-$MICE$-SDAE	78.3 ± 0.2	78.1 ± 0.4	78.1 ± 0.5
P-$MICE$-SDAE (ε)	79.5 ± 0.3 (0.5)	**79.4 ± 0.4 (0.25)**	78.7 ± 0.1 (0.75)
P-$MICE$-SDAE (ε, α)	**80.7 ± 0.4 (0.5, 0.3)**	**79.5 ± 0.2 (0.5, 0.7)**	**79.3 ± 0.3 (0.75, 0.6)**
T-MLP	91.8 ± 1.3	91.2 ± 0.6	90 ± 1.4
T-MLP-SDAE	92.2 ± 0.2	95.5 ± 0.2	94.7 ± 1.4
T-MLP-SDAE (ε)	**98.3 ± 0.6 (0.5)**	**97.6 ± 0.8 (0.5)**	**97.6 ± 0.4 (0.75)**
T-MLP-SDAE (ε, α)	**97.8 ± 0.5 (0.75, 0.6)**	**97.1 ± 0.2 (0.75, 0.6)**	**98.1 ± 0.2 (0.75, 0.6)**
T-SVD	90.3 ± 0.4	90.4 ± 0.2	89.4 ± 0.5
T-SVD-SDAE	93.5 ± 0.5	92.7 ± 0.4	91.2 ± 0.4
T-SVD-SDAE (ε)	97.1 ± 0.3 (0.5)	96.3 ± 0.3 (0.75)	96.5 ± 0.7 (0.75)
T-SVD-SDAE (ε, α)	**97.9 ± 0.4 (0.5, 0.2)**	**97.4 ± 0.6 (0.75, 0.7)**	**97.8 ± 0.4 (0.75, 0.8)**
T-$MICE$	96.3 ± 0.2	95.8 ± 0.3	95.4 ± 0.2
T-$MICE$-SDAE	**97.5 ± 0.4**	96.2 ± 0.4	96.4 ± 0.5
T-$MICE$-SDAE (ε)	**98.2 ± 0.3 (0.5)**	**97.2 ± 0.2 (0.75)**	**97.3 ± 0.2 (0.75)**
T-$MICE$-SDAE (ε, α)	**97.9 ± 0.3 (0.5, 0.5)**	**97.7 ± 0.3 (0.75, 0.6)**	**98.3 ± 0.3 (0.75, 0.7)**

Cuadro 3.5: Continuación de los resultados de clasificación. Los diferentes porcentajes de valores perdidos aplicados al conjunto de entrada se muestran en las columnas, mientras que los distintos métodos se presentan en las filas con la siguiente nomenclatura: primera inicial del conjunto de datos - *método de pre-imputación* - procedimiento profundo *(tasa de borrado, tasa de compensación)*. Para cada conjunto, el mejor resultado de la familia de pre-imputaciones se muestra en cursiva, mientras el mejor resultado global para cada valor de μ se presenta en negrita.

Procedimiento \ μ	0.1	0.2	0.3
S-MLP	94.5 ± 0.3	93.2 ± 0.3	94.3 ± 0.5
S-MLP-SDAE	95.6 ± 0.6	94.5 ± 0.3	95.2 ± 0.3
S-MLP-SDAE (ε)	97.3 ± 0.3 (0.75)	*97.7 ± 0.6 (0.75)*	97.4 ± 0.2 (0.75)
S-MLP-SDAE (ε, α)	**98.7 ± 0.4 (0.5, 0.8)**	*97.3 ± 0.4 (0.75, 0.7)*	**98.2 ± 0.6 (0.75, 0.6)**
S-SVD	95.3 ± 0.4	95.3 ± 0.4	94.8 ± 0.4
S-SVD-SDAE	95.7 ± 0.6	94.6 ± 0.5	96.3 ± 0.5
S-SVD-SDAE (ε)	97.7 ± 0.4 (0.75)	97.5 ± 0.4 (0.75)	97.4 ± 0.3 (0.75)
S-SVD-SDAE (ε, α)	*97.9 ± 0.4 (0.75, 0.6)*	**98.4 ± 0.6 (0.75, 0.7)**	**98.4 ± 0.2 (0.5, 0.6)**
S-$MICE$	97.8 ± 0.1	97.6 ± 0.2	97.7 ± 0.4
S-$MICE$-SDAE	98.2 ± 0.2	**98.4 ± 0.4**	98.2 ± 0.1
S-$MICE$-SDAE (ε)	**99.1 ± 0.3 (0.5)**	**99.1 ± 0.2 (0.75)**	**98.7 ± 0.3 (0.75)**
S-$MICE$-SDAE (ε, α)	**99.2 ± 0.2 (0.5, 0.6)**	**98.6 ± 0.4 (0.75, 0.6)**	**98.9 ± 0.2 (0.75, 0.6)**
G-MLP	96.4 ± 0.4	95.7 ± 0.4	95.0 ± 0.4
G-MLP-SDAE	97.6 ± 0.4	97.1 ± 0.2	96.8 ± 0.4
G-MLP-SDAE (ε)	**98.6 ± 0.5 (0.75)**	97.7 ± 0.3 (0.75)	98.1 ± 0.2 (0.75)
G-MLP-SDAE (ε, α)	**98.8 ± 0.4 (0.75, 0.5)**	*98.4 ± 0.2 (0.5, 0.4)*	**98.8 ± 0.4 (0.75, 0.4)**
G-SVD	97.1 ± 0.2	97.3 ± 0.4	96.8 ± 0.3
G-SVD-SDAE	97.5 ± 0.2	98.1 ± 0.2	96.6 ± 0.6
G-SVD-SDAE (ε)	*98.2 ± 0.2 (0.5)*	**98.7 ± 0.2 (0.75)**	97.6 ± 0.2 (0.75)
G-SVD-SDAE (ε, α)	*98.6 ± 0.1 (0.5, 0.4)*	**98.6 ± 0.3 (0.75, 0.6)**	**98.7 ± 0.4 (0.75, 0.6)**
G-$MICE$	98.2 ± 0.2	97.8 ± 0.3	98.0 ± 0.2
G-$MICE$-SDAE	98.5 ± 0.1	98.2 ± 0.4	98.6 ± 0.2
G-$MICE$-SDAE (ε)	**99.1 ± 0.2 (0.5)**	98.6 ± 0.2 (0.75)	**99.2 ± 0.1 (0.75)**
G-$MICE$-SDAE (ε, α)	**99.4 ± 0.2 (0.5, 0.6)**	**99.1 ± 0.1 (0.75, 0.8)**	**99.1 ± 0.1 (0.75, 0.6)**
A-MLP	86.7 ± 0.4	84.7 ± 0.2	86.8 ± 0.1
A-MLP-SDAE	87.1 ± 0.2	85.2 ± 0.2	87.3 ± 0.3
A-MLP-SDAE (ε)	87.5 ± 0.2 (0.25)	*87.4 ± 0.3 (0.5)*	88.7 ± 0.1 (0.5)
A-MLP-SDAE (ε, α)	*89.2 ± 0.1 (0.75, 0.6)*	*87.6 ± 0.1 (0.25, 0.5)*	*89.2 ± 0.2 (0.75, 0.6)*
A-SVD	88.3 ± 0.1	87.3 ± 0.2	87.1 ± 0.2
A-SVD-SDAE	88.6 ± 0.2	88.1 ± 0.2	87.6 ± 0.2
A-SVD-SDAE (ε)	88.1 ± 0.2 (0.5)	*89.5 ± 0.1 (0.5)*	*88.7 ± 0.2 (0.5)*
A-SVD-SDAE (ε, α)	**91.1 ± 0.2 (0.75, 0.7)**	*89.4 ± 0.3 (0.5, 0.7)*	*88.9 ± 0.3 (0.75, 0.6)*
A-$MICE$	89.2 ± 0.3	88.4 ± 0.1	88.8 ± 0.3
A-$MICE$-SDAE	90.1 ± 0.3	88.5 ± 0.3	89.9 ± 0.2
A-$MICE$-SDAE (ε)	**90.7 ± 0.4 (0.75)**	**90.1 ± 0.2 (0.5)**	**90.3 ± 0.1 (0.5)**
A-$MICE$-SDAE (ε, α)	**91.3 ± 0.2 (0.75, 0.7)**	**90.4 ± 0.3 (0.75, 0.4)**	**90.1 ± 0. 3(0.5, 0.7)**

Cuadro 3.6: Efectos de la compensación en tareas de clasificación. Los números muestran las ocasiones en las que los resultados con compensación y borrado son peores, iguales o mejores a aquéllos obtenidos sólo con borrado. Como se aprecia, sólo se empeora el resultado en una ocasión ($2\,\%$), se mejoran en 24 casos ($45\,\%$) y se consideran estadísticamente iguales en 29 casos ($53\,\%$).

Conjunto de datos	*Peor*	*Igual*	*Mejor*
MAGIC Gamma Telescope (M)	0	5	4
Pima Indians Diabetes (P)	1	3	5
Two Norms (T)	0	6	3
Sensorless Drive Diagnosis (S)	0	4	5
Gas Sensor Array Drift (G)	0	5	4
AREM (A)	0	6	3

- Si además se aplica compensación, se mejora el resultado en la mayoría de las ocasiones. Ésta es una prueba de que la compensación es útil.
- Según los valores óptimos de ε, se puede concluir que los valores altos producen mejores resultados. Esto permite concluir que el borrado es útil para aprender una mejor reconstrucción de la entrada.
- La mayoría de los valores óptimos de α son ligeramente superiores a $0{,}5$, de modo que dar más importancia a las muestras completas supone mejores resultados de imputación.
- En el caso del conjunto AREM, cuando el *missing* se inserta en diferentes características, los resultados son buenos. Esto significa que el método generaliza bien ante diferentes escenarios de valores perdidos, que no es sorprendente debido a la alta capacidad de representación de los AE junto con el mecanismo de compensación introducido.

Si, por otro lado, se analizan los resultados en clasificación de la Sección 3.3.4:

- El borrado (con o sin compensación) obtiene mejores resultados en la mayoría de los casos, es decir, no sólo mejora los resultados de imputación sino que también ayuda a la clasificación del SDAE.

- En la Tabla 3.6 se presenta el efecto de la compensación, mostrando el número de veces que mejora, empeora o iguala los resultados con borrado. Como se aprecia, estos resultados empeoran sólo en una ocasión (2 % del total), mejoran en 24 (45 %) y se consideran estadísticamente iguales en los otros 29 escenarios (53 %).

- Como se ha visto, aunque la compensación produce una mejora de los resultados de imputación, en algunos casos ésta no implica una mejora significativa en los resultados de clasificación. Esto significa que, aunque los valores de salida están más cerca de los reales, esta mejora no es lo suficientemente potente como para suponer una diferenciación de las clases.

- Finalmente, destacar que los resultados de clasificación también se mejoran con el borrado y la compensación en AREM, el conjunto con más de una característica con valores perdidos.

Lo que hacemos en la vida tiene su eco en la eternidad.

Gladiator

CAPÍTULO

4

SDAEs modificados completos

A finales del Siglo XX, sólo existía un tipo de red neuronal profunda que merecía toda la atención, la red convolucional (CNN) [15, 100, 101]. Este tipo de red no sufre la dificultad de desvanecimiento del gradiente al aplicar Retropropagación (BP) [102, 103]. Esto es así porque no tienen todas las conexiones en sus capas ocultas, sino que la mayoría de ellas son simples filtros aplicados sobre las entradas. A pesar de que las CNNs se han empleado en un gran número de trabajos [104, 105, 106, 107], esta característica limita su aplicación a problemas de procesado de voz, sonido, texto o imagen. Por tanto, tras su descubrimiento se continuó con la búsqueda de nuevos algoritmos.

Es entonces cuando surgieron nuevas aproximaciones. Primero, aquellas basadas en máquinas de Boltzman multicapa [16, 108], y unos años después los Autoencoders apilados [14]. Este tipo de redes, entrenadas en dos etapas, obvian los problemas del algoritmo BP, y se enmarcan en el conocido como aprendizaje de representación [109], y poseen ciertas características que les hacen ser de gran utilidad. En general, son capaces de desenmarañar las muestras de entrada en los subespacios intermedios [99], y además extraen características de las mismas en

sus capas profundas que son realmente útiles [110] para, por ejemplo, reducción de dimensionalidad.

En este capítulo se presenta una nueva arquitectura llamada SDAE Modificado Completo (CMSDAE) que incluye las variables objetivo en el entrenamiento capa a capa de un SDAE Modificado [91]. En primer lugar, se estudiará cómo esta arquitectura resulta ser una buena opción en problemas de clasificación. Después, se comprueba el funcionamiento de esta versión desde un punto de vista monotarea y se presenta una modificación que explota las capacidades del aprendizaje multitarea para la clasificación de patrones incompletos. El capítulo se cierra con unas conclusiones.

4.1 CMSDAEs para clasificación

Como se ha comentado en capítulos anteriores, esta tesis está centrada en el uso de SDAEs [14] para imputar y para clasificar patrones incompletos. En concreto, en este capítulo se usa la versión modificada presentada en [91]. La razón de usar esta versión es que puede ser adaptada al principio de diseño de las redes profundas apiladas (DSN) [111, 112]. Una DSN se construye capa a capa, inyectando a cada una de ellas las salidas de la anterior y usando como variables objetivos las observaciones originales. Así, la versión modificada del SDAE, que a partir de ahora llamaremos MSDAE, es más apropiada para los objetivos de este trabajo porque se entrena para reconstruir en cada capa la salida no ruidosa de la entrada global de la red, no la salida de la capa anterior como en los SDAEs convencionales.

En esta sección se sentarán las bases teóricas de los mencionados CMSDAEs, estudiando su comportamiento en problemas de clasificación genéricos. Primero, se presenta el proceso llevado a cabo para construir un CMSDAE, basado en la introducción de las variables objetivo en el entrenamiento capa a capa de un SDAE clásico. Esta arquitectura se representará gráficamente y se discutirá su rendimiento. Como punto a destacar, el desarrollo de un CMSDAE parte de un MSDAE, considerado más apropiado por la razón mencionada.

4.1.1 Metodología

Cuando se trabaja con un conjunto de datos de la forma $\{\underline{x}_n, \underline{t}_n\}, n \in \{1, ..., N\}$, donde $\underline{x}_n$ es una muestra y $\underline{t}_n$ su etiqueta, explotar toda la información disponible en las muestras $[\underline{x}_n^T, \underline{t}_n^T]^T$ supone incluir la información de la etiqueta $\underline{t}_n$ en el entrenamiento de sus capas ocultas de dimensión superior y por tanto en la extracción de características ocultas. Pero esta posibilidad produce dificultades a la hora de entrenar la red en el modo operación, cuando las muestras aún no están etiquetadas. Existe entonces una mejor opción, aplicada en DSNs [111, 112] y otros contextos como imputación de valores perdidos [89, 90]: a la entrada del CMSDAE, se usa una estimación de las etiquetas verdaderas $\widetilde{\underline{t}}_n$ obtenida mediante una máquina auxiliar. De esta forma, se completan las muestras de entrenamiento. El esquema propuesto se presenta en la Figura 4.1. Como se puede apreciar, las etiquetas reales del conjunto de datos se usan como objetivo durante el entrenamiento de cada una de las capas de la red junto con el resto de las muestras.

El mismo procedimiento se sigue en modo operación. No existe ningún problema, ya que las etiquetas no son necesarias para la clasificación de muestras desconocidas: son asimiladas por los pesos del CMSDAE. Por tanto, las capas ocultas de la red representarán sucesivamente características de mayor dimensión que además incluyen información de la clase a la que cada muestra pertenece. Así, las representaciones obtenidas en las capas ocultas para cada muestra de entrada, están guiadas por la etiqueta de la misma. En otras palabras, estas representaciones están forzadas por el aprendizaje conjunto de la muestra y la etiqueta, de modo que existe una transferencia de conocimiento entre ellas.

Acorde con el procedimiento descrito, un CMSDAE mantiene las ventajas de representación de un MSDAE –como su capacidad de desenmarañamiento– e incluye información de la tarea de clasificación a resolver mediante el clasificador incluido a la salida del CMSDAE –añadiendo un último paso de clasificación y refinando el entrenamiento para reducir el coste de clasificación–. Gracias a este incremento de información, se espera una mejora significativa de los resultados finales.

Obviamente, no existe a priori una pista que indique la importancia relativa que deben tener tanto la reconstrucción de la muestra como la de la etiqueta. Es

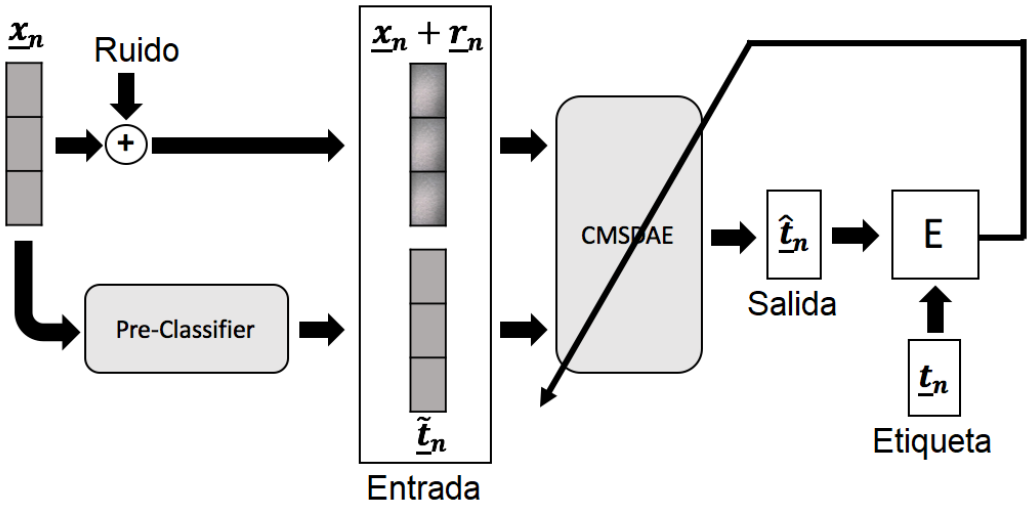


Figura 4.1: Esquema de entrenamiento de un SDAE Modificado Completo. Las muestras originales se pasan por un clasificador auxiliar, con el que se obtiene una estimación de la variable objetivo. Ésta, junto con la versión ruidosa de la muestra, constituyen la entrada de la red, que se entrena para clasificar las etiquetas originales.

fácil pensar que dependerá del problema a resolver. Por tanto, la solución más adecuada será realizar una combinación convexa de los errores correspondientes a ambas componentes durante el entrenamiento de cada una de las capas ocultas. El párametro de la combinación convexa puede establecerse mediante el proceso convencional de validación cruzada.

No es necesario decir que, cuando se trata de problemas de clasificación multi-clase, se aplica un esquema estándar de activaciones softmax para las salidas de la red con el objetivo de conseguir resultados consistentes.

La Figura 4.2 representa el clasificador CMSDAE propuesto. Éste incluye indicaciones simbólicas del proceso de entrenamiento mediante líneas discontinuas. $\underline{r}_n$ representan las componentes ruidosas añadidas a las muestras de entrada[1] $\underline{x}_n$. Las etiquetas usadas para las capas de representación $(l_1, ..., l_L)$ son $[\underline{x}_n, \underline{t}_n]^T$. Después de entrenar, los pesos de entrada de cada capa oculta se mantienen congelados, y el proceso se repite para la capa siguiente. La capa final (o, en general, la máquina

[1] Ya que $\{\widetilde{\underline{t}}_n\}$ son proporcionados por una máquina auxiliar, no es estrictamente necesario añadir ruido a estos valores.

de clasificación final) se entrena para resolver el problema de clasificación, y los pesos anteriormente congelados se refinan mediante BP.

Finalmente, es importante mencionar que 1) la principal diferencia entre un CMSDAE y un SDAE es la inclusión de un clasificador auxiliar a la entrada, y 2) la dimensión de $\underline{t}_n$ es normalmente pequeña (mucho más pequeña en la mayoría de los casos prácticos) comparado con $\underline{x}_n$. Esto significa que el coste computacional de un CMSDAE, aunque ligeramente superior, es muy similar al de un SDAE.

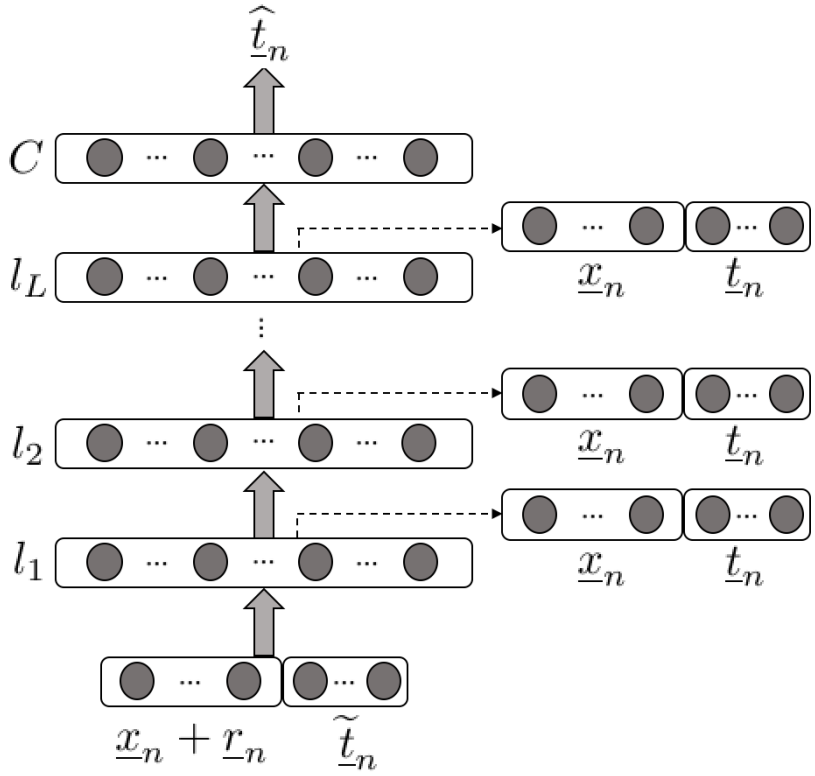


Figura 4.2: Arquitectura de un CMSDAE y su entrenamiento. La entrada ruidosa $\underline{x}_n + \underline{r}_n$ y la salida de la máquina auxiliar $\widetilde{\underline{t}}_n$ (estimando la etiqueta $\underline{t}_n$) se inserta en la entrada, y cada capa oculta de representación $(l_1, ..., l_L)$ se entrena para reconstruir la entrada libre de ruido $\underline{x}_n$ y la etiqueta $\underline{t}_n$ mediante una fórmula de combinación convexa. El clasificador C sirve para llevar a cabo la clasificación final según $\widehat{\underline{t}}_n$.

4.1.2 Experimentos

4.1.2.1 Bases de datos

Se seleccionan 6 conjuntos de datos con diferentes tamaños, dimensiones (número de características) y clases, con el objetivo de testar el procedimiento propuesto ante diferentes escenarios. Estas bases de datos son:

- **Activity Recognition system based on Multisensor data fusion (AREM).** Este problema está preparado para predecir el tipo de actividad llevada a cabo por el usuario. Se han dado algunos detalles del conjunto en el Capítulo 3 [96].

- **Activity Recognition from Single Chest-Mounted Accelerometer (CHEST).** Conjunto formado por datos provenientes de un acelerómetro descalibrado colocado en el pecho de 15 personas, recogidos durante la realización de 7 actividades distintas. El objetivo de los datos es la detección y autenticación de personas usando los patrones de movimiento [96].

- **Dataset for Sensorless Drive Diagnosis (DRIVE).** Conjunto formado por la extracción de señales eléctricas de un motor. El motor posee componentes intactos y defectuosas. Esto conlleva 11 clases diferentes en distintas condiciones. Cada clase ha sido medida varias veces en 12 situaciones de operación distintas, es decir, con diferentes velocidades, momentos de carga o fuerzas de carga. Las señales reales son medidas con una sonda y un osciloscopio [96].

- **Rectangles (RECT).** Conjunto generado con el propósito de evaluar empíricamente arquitecturas profundas en investigación. Consiste en una serie de imágenes en blanco y negro con rectángulos de diferentes dimensiones. El objetivo es enseñar a la máquina a distinguir entre rectángulos anchos y altos [113].

- **Skin Segmentation (SKIN).** Conjunto obtenido mediante un muestreo aleatorio de valores RGB a partir de imágenes de rostros de varios grupos de

edades (jóvenes, mediana edad y mayores), grupos raciales (blancos, asiáticos y de color), y géneros obtenidos de las bases de datos FERET y PAL [96].

- **Sloan Digital Sky Survey RD14 (SKY).** Los datos consisten en un gran número de observaciones espaciales, obtenidas de la encuesta *Sloan Digital Sky* que ofrece datos públicos. Cada observación se describe mediante un gran número de características –que contienen datos fotométricos y espectrales– y una clase que indica si se trata de una estrella, una galaxia o un cuásar [114].

La Tabla 4.1 muestra las principales características de las bases. Estas bases de datos se dividen aleatoriamente en 70 %-30 % de entrenamiento y test, respectivamente. Los valores de las características se escalan en el intervalo [0,1] con el objetivo de usar costes de entropía cruzada, así como valores de clasificación de salida.

Cuadro 4.1: Bases de datos y sus características.

	Muestras	Características	Clases
AREM	42240	6	7
CHEST	1926896	3	7
DRIVE	58509	49	11
RECT	62000	784	2
SKIN	245057	3	2
SKY	10000	17	3

4.1.2.2 **Máquinas**

Obviamente, el clasificador auxiliar que proporciona las etiquetas de entrada al CMSDAE tendrá una influencia importante sobre los resultados de clasificación finales. Teniendo en cuenta esto, se considerarán dos clasificadores auxiliares distintos:

- Un MLP, que puede ser visto como una máquina simple de clasificación.

- Un MSDAE convencional, que es una red profunda y, adicionalmente, tiene la misma estructura que la red final CMSDAE.

Se usarán funciones sigmoidales en todas las neuronas de activación, así como para las salidas de los problemas binarios. Para problemas multiclase, se usan activaciones softmax en todas las salidas. Con respecto a las dimensiones de las capas ocultas, se aplica un factor de expansión $F \in \{1{,}25, 1{,}75, 2{,}5\}$ con respecto a la dimensión de la entrada correspondiente (número de características en el caso de la máquina auxiliar, y número de características más la etiqueta para el CMSDAE). Este factor de expansión F se determina por medio de una validación cruzada convencional 5-fold y 50 simulaciones. Es importante mencionar que el CMSDAE se entrena con un número constante de unidades ocultas en todas sus capas. Los clasificadores finales MSDAE y CMSDAE tienen la misma dimensión.

4.1.2.3 **Entrenamiento**

Los MLPs se entrenan por BP como las redes anteriores de este trabajo. Para los MSDAEs y los CMSDAEs, se añade ruido blanco gausiano a los valores de las muestras de entrada. Este ruido tiene media nula y una varianza que es un porcentaje υ de la varianza de cada característica de entrada. Se exploran valores de υ entre $\{10\,\%, 20\,\%, 30\,\%, 40\,\%\}$ y se selecciona el mejor de ellos mediante validación cruzada, como se ha indicado en la subsección anterior.

Finalmente, el entrenamiento de cada capa de un CMSDAE se lleva a cabo minimizando

$$E = (1 - \lambda)E_{\underline{x}} + \lambda E_{\underline{t}} \tag{4.1}$$

donde $E_{\underline{x}}$ es el error de entropía cruzada de la reconstrucción de $\underline{x}$, y $E_{\underline{t}}$ el error de entropía cruzada para las etiquetas. Así,

$$E_{\underline{x}} = -\sum \underline{x} \log(\underline{x}') \tag{4.2}$$

$$E_{\underline{t}} = -\sum \underline{t} \log(\underline{t}') \tag{4.3}$$

donde $\underline{x}'$ y $\underline{t}'$ son las predicciones para las capas de representación. El parámetro λ también se determina mediante validación cruzada, con $\lambda \in \{0{,}1, 0{,}2, ..., 0{,}9\}$.

4.1.2.4 **Resultados**

La Tabla 4.2 presenta los resultados de los experimentos llevados a cabo con las 6 bases de datos presentadas anteriormente, en forma de % media $\pm$ desviación estándar de las decisiones correctas (precisión) para 50 simulaciones y los valores óptimos de los parámetros de entrenamiento (F,υ,λ) obtenidos mediante validación cruzada.

Los mejores resultados en media se indican en negrita, cuando su media es al menos su desviación estándar mayor a los otros resultados.

En los casos en los que los valores óptimos de los parámetros toman un extremo ($F = 2{,}50$ para SKY con un MLP y CHEST con un MSDAE y $\lambda = 0{,}9$ para SKY con un MLP-CMSDAE), los valores explorados se expanden para asegurarnos de que son óptimos.

Cuadro 4.2: Resultados presentados en % media $\pm$ desviación estándar para los diferentes diseños. Los valores óptimos de los parámetros están incluídos. Los mejores resultados se representan en negrita según el criterio mencionado en el texto principal.

Conjunto	MLP (F)	MSDAE (F, υ)	MLP-CMSDAE (F, υ, λ)	MSDAE-CMSDAE (F, υ, λ)
AREM	65.9 $\pm$ 0.7 (1.75)	61.5 $\pm$ 0.5 (1.75, 0.3)	89.1 $\pm$ 0.2 (1.75, 0.2, 0.3)	**89.7 $\pm$ 0.1 (1.75, 0.2, 0.3)**
CHEST	36.5 $\pm$ 0.5 (1.75)	37.9 $\pm$ 0.2 (2.50, 0.2)	57.7 $\pm$ 0.3 (1.75, 0.3, 0.5)	**58.8 $\pm$ 0.4 (1.75, 0.2, 0.5)**
DRIVE	85.5 $\pm$ 0.2 (1.75)	87.8 $\pm$ 0.4 (1.75, 0.2)	94.7 $\pm$ 0.5 (1.75, 0.3, 0.2)	**97.6 $\pm$ 0.2 (1.75, 0.3, 0.5)**
RECT	73.8 $\pm$ 0.1 (1.75)	74.6 $\pm$ 0.4 (1.75, 0.2)	75.2 $\pm$ 0.4 (1.75, 0.2, 0.4)	**76.8 $\pm$ 0.1 (1.75, 0.2, 0.3)**
SKIN	95.3 $\pm$ 0.1 (1.75)	97.7 $\pm$ 0.2 (1.75, 0.3)	**99.7 $\pm$ 0.03 (1.75, 0.2, 0.4)**	**99.8 $\pm$ 0.03 (1.75, 0.3, 0.6)**
SKY	81.4 $\pm$ 0.6 (2.50)	83.4 $\pm$ 0.2 (1.75, 0.2)	89.2 $\pm$ 0.2 (1.75, 0.2, 0.9)	**90.8 $\pm$ 0.7 (1.75, 0.2, 0.8)**

4.1.3 **Discusión**

Los resultados de la Tabla 4.2 proporcionan una clara visión de los beneficios conseguidos al usar CMSDAEs. Ambos clasificadores, MLP-SDAE y MSDAE-CMSDAE, ofrecen precisiones que son claramente mejores a las obtenidas por los

clasificadores MLPs o MSDAEs. En particular, aplicando la guía de un MSDAE se reducen los malos resultados –con la única excepción de un empate en SKIN–, como era de esperar, porque la red tiene la misma estructura (y es mejor en todos los casos menos SKIN).

Es importante comprobar cuánta atención se debe poner a la reconstrucción de las etiquetas, es decir, en el término $E_{\underline{t}}$. En general, se puede decir que sus valores son bajos o moderados con respecto a $(1-\lambda)E_{\underline{x}}$. La Tabla 4.3 sirve para demostrar este hecho para las últimas (tercera) capa de la mejor familia de clasificadores, MSDAE-CMSDAE, para los seis conjuntos. La única excepción a esta regla aparece para SKIN. La posible conclusión es que la atención que debe ponerse en la reconstrucción de las etiquetas depende del problema.

Cuadro 4.3: Valores medios de $(1-\lambda)E_{\underline{x}}$ y $\lambda E_{\underline{t}}$ en la tercera capa del clasificador MSDAE.CMSDAE.

Conjunto	$(1-\lambda)E_{\underline{x}}$	$\lambda E_{\underline{t}}$
AREM	1.894	0.038
CHEST	0.085	0.0045
DRIVE	1.424	0.75
RECT	0.413	0.147
SKIN	0.0368	0.324
SKY	1.442	0.165

Es importante destacar que en varios casos las precisiones medias obtenidas en las capas ocultas de los clasificadores CMSDAE son incluso mayores que las de la salida final. Esto es particularmente intenso para la base de datos CHEST: la primera, segunda y tercera capa del clasificador MSDAE-CMSDAE generan salidas con precisiones de 63 %, 64 % y 65 %, respectivamente, que es superior a la precisión final del 58.8 %. Esto es una consecuencia de nuestra selección del número de capas, así como de usar el mismo valor de λ en todas ellas. Obviamente, diseños más flexibles reducirán estas inconsistencias, pero una nueva conclusión aparece: podría ser interesante introducir etiquetas al diseñar DNNs convencionales.

La sensibilidad de los modelos ante la variación de los parámetros de entrenamiento es un tema siempre importante. Un análisis detallado del funcionamiento de

los métodos propuestos sobre diferentes valores de validación cruzada indica que su sensibilidad es baja. Se mostrarán aquí sólo un par de casos que demuestran esta afirmación, con el objetivo de no perder el foco: RECT y SKIN, para los que la ventaja del clasificador MSDAE-CMSDAE con respecto a un MLP o un MSDAE es moderada.

La Tabla 4.4 muestra el rendimiento del clasificador MSDAE-CMSDAE para RECT cuando F, υ y λ toman diferentes valores en la validación cruzada, así como los inmediatamente superiores e inferiores. Cuando se varía uno de ellos, el resto se mantiene intacto.

Cuadro 4.4: Rendimiento medio $\pm$ desviación estándar ($\%$) del clasificador MSDAE-CMSDAE cuando cada parámetro de entrenamiento toma valores inmediatamente superiores e inferiores a los óptimos, para el conjunto RECT. Los valores de los parámetros se representan entre paréntesis.

F	74.5 ± 0.3 (1.25)	76.8 ± 0.1 (1.75)	76.4 ± 0.2 (2.5)
υ	76.2 ± 0.2 (0.1)	76.8 ± 0.1 (0.2)	76.0 ± 0.4 (0.3)
λ	76.2 ± 0.1 (0.2)	76.8 ± 0.1 (0.3)	75.9 ± 0.4 (0.4)

Es fácil ver que los resultados cambian levemente al modificar los parámetros, excepto para $F = 1{,}25$, donde aparece un gran cambio con el del tamaño de la capa oculta. Podría intentar resolverse esta degradación haciendo un barrido con valores más cercanos al óptimo.

La Tabla 4.5 representa la misma idea. En este caso, la sensibilidad es incluso menor. Además, la ventaja con respecto a un MLP y un MSDAE se mantiene a pesar de los cambios.

Para cerrar esta discusión, se debe destacar que existe la posibilidad de mejorar los resultados de los clasificadores CMSDAE aplicando técnicas complementarias, tales como diversidad [115] –en particular, la conmutación de etiquetas [116]–,

Cuadro 4.5: Rendimiento medio $\pm$ desviación estándar (%) del clasificador MSDAE-CMSDAE cuando cada parámetro de entrenamiento toma valores inmediatamente superiores e inferiores a los óptimos, para el conjunto SKIN. Los valores de los parámetros se representan entre paréntesis.

F	98.2 ± 0.2 (1.25)	99.8 ± 0.03 (1.75)	99.1 ± 0.1 (2.5)
υ	99.2 ± 0.03 (0.2)	99.8 ± 0.03 (0.3)	98.9 ± 0.03 (0.4)
λ	99.4 ± 0.03 (0.5)	99.8 ± 0.03 (0.6)	98.9 ± 0.02 (0.7)

preénfasis –siguiendo las ideas propuestas en [117, 118]–, aumento de datos [119], y procesos similares. La eficiencia de introducir este tipo de técnicas se muestra claramente en [91, 120] en contextos similares.

4.2 CMSDAEs para clasificación de patrones incompletos

Como se ha comentado a lo largo de la tesis, los SDAEs han sido ampliamente utilizados porque, además de proporcionar robustez a los DAEs, poseen una gran capacidad de representación [14, 109]. Muchos trabajos han aprovechado estas ventajas. Un ejemplo de ello es [121], un artículo donde un conjunto de SDAEs se prueban para clasificar dígitos manuscritos, y [122], donde los SDAEs son usados para la predicción del riesgo de mortalidad sobre un conjunto de datos clínicos desequilibrado.

Los SDAEs también han sido usados en problemas con datos incompletos. En general, la imputación es la técnica preferida para tratar este tipo de problemas [123, 124]. La mayoría de métodos de imputación usan sólo la información conocida, extraída de los valores observados de los datos de entrada. Sin embargo, existe más información que normalmente no es tenida en cuenta, los valores objetivo.

El aprendizaje multitarea (MTL) es una variante del aprendizaje máquina donde se aprenden varias tareas al mismo tiempo, considerando una de ellas como principal y las otras como secundarias [90]. Este aprendizaje simultáneo produce una transferencia de información que mejora el resultado de la tarea principal. Desde

que Caruana demostró la eficacia del MTL en múltiples problemas [90], algunos trabajos se han centrado en el uso del MTL en problemas incompletos [76, 82].

En esta sección, la arquitectura presentada anteriormente se evalua como método para resolver problemas de clasificación con patrones incompletos. Recordemos que un CMSDAE es una técnica que incorpora las etiquetas como parte de la entrada en el entrenamiento de un MSDAE, y proporciona muy buenos resultados en problemas de clasificación [91]. Aquí, además de la formulación necesaria para el uso de CMSDAEs sobre imputación, se presenta una variación basada en aprendizaje MTL que obtiene mejores resultados. Así, tras una descripción del método, se presentan algunos experimentos ilustrativos sobre los mismos conjuntos de datos de la sección anterior. Finalmente, se discuten los resultados.

4.2.1 Metodología

El CMSDAE se ha presentado anteriormente como un método que mejora los resultados de un clasificador SDAE [91]. En esta aproximación, la entrada se expande con una estimación de las etiquetas proporcionada por una máquina auxiliar. De esta forma, el SDAE recibe información adicional que ayuda a la extracción de características de alto nivel y, por tanto, produce una mejora de los resultados.

Aquí, el CMSDAE se adapta para manejar datos incompletos. Se evalúan dos esquemas diferentes. El primero es un esquema monotarea con una única tarea de clasificación, y el segundo es un esquema multitarea (MTL), en el que se aprenden de forma simultánea las tareas de clasificación (principal) y la de reconstrucción de las observaciones (secundaria). El caso MTL se muestra en la Figura 4.3 donde, comparado con el CMSDAE original, se inserta un nuevo bloque de pre-imputación para completar los datos desconocidos del conjunto de entrada. Después, se entrena un clasificador auxiliar para realizar una primera estimación de las etiquetas que, junto con una versión ruidosa de las observaciones, constituyen la entrada de la red CMSDAE. Esta red es finalmente entrenada para minimizar el error entre su salida y las etiquetas.

Dado un conjunto de datos $\{\underline{x}_n, \underline{t}_n\}, n \in \{1, ..., N\}$, donde $\underline{x}_n$ representa una muestra que puede contener valores perdidos y $\underline{t}_n$ su etiqueta, el vector de entrada al CMSDAE es el vector formado por $[\hat{\underline{x}}_n, \hat{\underline{t}}_n]$, siendo $\hat{\underline{x}}_n$ una versión ruidosa de

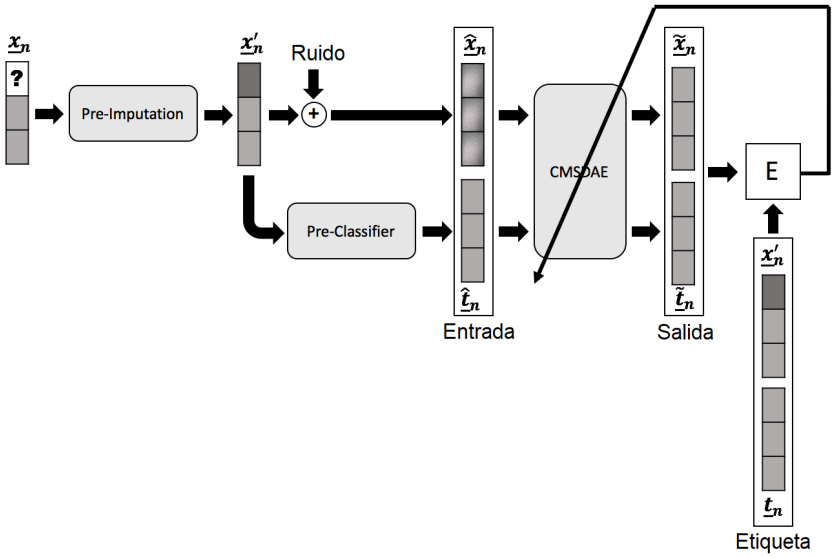


Figura 4.3: Esquema de un CMSDAE Multitarea y su entrenamiento. Dada una muestra de entrada $\underline{x}_n$ con valores desconocidos, se aplica un método de pre-imputación para completarla. Una versión ruidosa de $\underline{x}'_n$ junto con una estimación de las etiquetas obtenida mediante un clasificador auxiliar constituyen la entrada del CMSDAE. Finalmente, el problema de clasificación se resuelve mediante el esquema MT CMSDAE

la entrada completa $\underline{x}'_n$ —obtenida mediante un bloque de pre-imputación— y $\hat{\underline{t}}_n$ la estimación de las etiquetas dada por un clasificador auxiliar. Así, la salida del CMSDAE Multitarea (MT-CMSDAE) es el vector $[\tilde{\underline{x}}_n, \tilde{\underline{t}}_n]$, que necesita ajustarse al objetivo $[\underline{x}'_n, \underline{t}_n]$. La salida del esquema Monotarea (ST-CMSDAE) sería $\tilde{\underline{t}}_n$.

Desde una perspectiva monotarea (ST-CMSDAE), sólo se aprenden las etiquetas de clasificación, mientras que en el caso de la arquitectura MT-CMSDAE se aprenden simultáneamente las características $\underline{x}'_n$ y las etiquetas $\underline{t}_n$. En esta aproximación conjunta, el problema de clasificación se considera como la tarea principal y el aprendizaje de las características de entrada como la secundaria. El proceso de aprendizaje producirá una transferencia de conocimiento entre ambas tareas que será clave para obtener mejores resultados finales.

En el estudio de esta sección se usan como métodos de pre-imputación tanto imputación a ceros (Z) como MICE, presentados previamente [125, 126]. Estos dos métodos se han seleccionado a causa de su diferente complejidad y eficiencia, de tal forma que se pueda comprobar si los resultados de la pre-imputación afectan a los resultados finales de la técnica propuesta. Además, se usa un MSDAE como máquina auxiliar de clasificación ya que, como se ha visto en la sección anterior, ha resultado ser la mejor opción [90].

El entrenamiento de ambas arquitecturas CMSDAE se hace mediante un doble equilibrado de los errores. Así, el error de clasificación $E_{\underline{t}}$ y la tarea de estimación $E_{\underline{x}}$ se equilibran mediante una doble combinación convexa controlada por el parámetro λ de la forma que se muestra a continuación:

$$E = \lambda E_{\underline{t}} + (1 - \lambda) E_{\underline{x}} \tag{4.4}$$

Al mismo tiempo, se aplica una segunda combinación convexa al error producido por las muestras completas ($E^C_{\underline{x}}$) e incompletas ($E^I_{\underline{x}}$). En este caso, se emplea otro parámetro α de la forma:

$$E_{\underline{x}} = \alpha E^C_{\underline{x}} + (1 - \alpha) E^I_{\underline{x}} \tag{4.5}$$

En este trabajo, se usa la función de error Entropía Cruzada para todas las tareas ($E_{\underline{t}}$, $E^C_{\underline{x}}$ y $E^I_{\underline{x}}$).

4.2.2 Experimentos

En esta sección se evalúa el rendimiento de las arquitecturas presentadas sobre seis conjuntos de datos. Se usan las mismas bases de datos que en la sección anterior: AREM, CHEST, DRIVE, RECT, SKIN y SKY. Por recordar sus principales características, se presenta la Tabla 4.6, pero se pueden encontrar más detalles sobre ellos en [96, 113]. Es importante mencionar que sólo se han considerado conjuntos de datos completos con el objetivo de insertar valores perdidos de forma artificial.

Cuadro 4.6: Conjuntos de datos y sus características.

	Muestras	Características	Clases
AREM	42240	6	7
DRIVE	58509	49	11
SKY	10000	17	3
CHEST	1926896	3	7
SKIN	245057	3	2
RECT	62000	784	2

Para comparar el rendimiento de las arquitecturas, tanto MSDAE, ST-CMSDAE como MT-CMSDAE se implementan con tres capas ocultas y factor de expansión de $F \in \{1{,}25, 1{,}75, 2{,}5\}$, seleccionando el valor óptimo de F mediante CV. Se usarán neuronas sigmoidales tanto en las capas ocultas como en las capas de salida binarias, mientras que para la clasificación se emplean activaciones softmax. Además, para el entrenamiento, se divide el conjunto de datos en 70 % de entrenamiento y 30 % de test, escalando todas las características de entrada en el intervalo $[0, 1]$.

Para entrenar las redes, se añade un ruido blanco gausiano con media cero y varianza igual a un porcentaje v de la varianza de cada característica del conjunto de entrada. Además, en este caso se eliminan valores en un 20 % de las muestras y un 50 % de las características, seleccionadas de forma aleatoria. Se explorarán valores de v entre $\{5\,\%, 10\,\%, 20\,\%, 30\,\%\}$ y, con el objetivo de comprobar el funcionamiento del método en diferentes escenarios, se explorarán valores de $\{0{,}1, 0{,}2, 0{,}3, 0{,}4, 0{,}5, 0{,}6, 0{,}7, 0{,}8, 0{,}9\}$ para el parámetro α y valores de

$\{0{,}1, 0{,}2, 0{,}3, 0{,}4, 0{,}5, 0{,}6, 0{,}7, 0{,}8, 0{,}9\}$ para λ. De nuevo, todos los parámetros se ajustan mediante 5-*fold* CV con 50 iteraciones.

La Tabla 4.7 muestra los resultados obtenidos con los diferentes métodos sobre los seis conjuntos de datos distintos. La precisión de cada método (porcentaje de decisiones correctas) se expresa en términos de %media $\pm$desviación estándar de las 50 iteraciones, donde los parámetros comentados (F, υ, λ, α) han sido ajustados mediante CV. Para cada conjunto de datos, el mejor resultado de una familia de pre-imputaciones se muestra en cursiva, mientras que el mejor resultado global se muestra en negrita.

4.2.3 Discusión de resultados

Tal y como se puede ver en la Tabla 4.7, las arquitecturas basadas en un CMSDAE son capaces de mejorar los resultados de los MSDAEs en todos los casos. En general, los métodos propuestos muestran mejora, especialmente el MT-CMSDAE, que es el mejor en 5 de los 6 casos, proporcionando en algunos de ellos mejoras realmente significativas. Un ejemplo de ello son los casos de DRIVE y SKY, donde la mejora de precisión del MT-CMSDAE supera en más del 10 % a la del MSDAE.

Como conclusión, se destaca que las capas ocultas desempeñan un papel muy importante. Excluyendo el caso de SKIN con pre-imputación MICA, donde el mejor factor de expansión F es el máximo $(2{,}5)$, en el resto de casos el valor óptimo es $1{,}75$, que es relativamente alto.

Además, los valores óptimos de α están generalmente por encima de 0.5. Esto significa que la reconstrucción de las etiquetas es claramente más útil para obtener mejores resultados. El hecho de que α no tome valores extremos indica que es importante tanto la reconstrucción de los valores completos como incompletos.

Finalmente, se puede apreciar que a mejor método de pre-imputación, mejor resultado final, como era de esperar.

Cuadro 4.7: Precisión (media±desviación estándar) en % para los diferentes diseños validados mediante CV. Se incluye información de los parámetros obtenidos mediante CV. Las técnicas de pre-imputación se asocian a los nombres de las bases de datos y, para cada conjunto, el mejor resultado de una familia de pre-imputaciones se muestra en cursiva y el mejor resultado global en negrita.

Procedimiento	MSDAE (F)	ST-CMSDAE $(F, \upsilon, \lambda, \alpha)$	MT-CMSDAE $(F, \upsilon, \lambda, \alpha)$
AREM-Z	61.5 ± 0.2 (1.75)	66.7 ± 0.1 (1.75, 0.1, 0.4, 0.5)	*67.3 ± 0.2 (1.75, 0.1, 0.5, 0.6)*
AREM-MICE	62.3 ± 0.3 (1.75)	67.4 ± 0.1 (1.75, 0.1, 0.7, 0.7)	**67.7 ± 0.1 (1.75, 0.2, 0.6, 0.8)**
DRIVE-Z	85.1 ± 0.1 (1.75)	86.5 ± 0.1 (1.75, 0.1, 0.8, 0.7)	*90.7 ± 0.2 (1.75, 0.1, 0.8, 0.6)*
DRIVE-MICE	86.3 ± 0.2 (1.75)	96.2 ± 0.2 (1.75, 0.1, 0.7, 0.5)	**97.7 ± 0.1 (1.75, 0.1, 0.6, 0.8)**
SKY-Z	77.2 ± 0.3 (2.5)	85.6 ± 0.2 (1.75, 0.2, 0.7, 0.7)	*87.8 ± 0.2 (1.75, 0.2, 0.6, 0.7)*
SKY-MICE	82.6 ± 0.1 (2.5)	86.5 ± 0.2 (1.75, 0.1, 0.3, 0.4)	**88.4 ± 0.2 (1.75, 0.1, 0.4, 0.5)**
CHEST-Z	34.9 ± 0.1 (1.75)	54.5 ± 0.2 (1.75, 0.2, 0.5, 0.7)	**56.4 ± 0.2 (1.75, 0.2, 0.8, 0.6)**
CHEST-MICE	35.7 ± 0.2 (1.75)	53.8 ± 0.2 (1.75, 0.2, 0.7, 0.6)	*55.3 ± 0.4 (1.75, 0.1, 0.8, 0.4)*
SKIN-Z	97.4 ± 0.2 (1.75)	**99.3 ± 0.1 (1.75, 0.1, 0.5, 0.8)**	98.4 ± 0.3 (1.75, 0.1, 0.6, 0.7)
SKIN-MICE	97.6 ± 0.1 (1.75)	**99.4 ± 0.2 (2.5, 0.1, 0.6, 0.7)**	98.7 ± 0.3 (2.5, 0.1, 0.5, 0.6)
RECT-Z	72.9 ± 0.4 (1.75)	73.8 ± 0.2 (1.75, 0.1, 0.5, 0.3)	*74.3 ± 0.1 (1.75, 0.2, 0.7, 0.8)*
RECT-MICE	74.3 ± 0.1 (1.75)	70.6 ± 0.1 (1.75, 0.1, 0.5, 0.6)	**74.8 ± 0.2 (1.75, 0.1, 0.6, 0.7)**

Ahora puedo decirte que tomé la decisión correcta, sin embargo, no hay un día que pase sin arrepentirme de no haber tomado una opción diferente.

Seven

CAPÍTULO 5

Conclusiones y líneas futuras

La presencia de valores perdidos en un conjunto de datos es uno de los problemas más comunes cuando uno se enfrenta a aplicaciones reales de Aprendizaje Máquina. Se han desarrollado diferentes técnicas para abordar este problema, siendo la imputación la más usada en la práctica. La motivación de esta tesis doctoral surge a partir de la idea de explotar la gran capacidad de representación que poseen los SDAEs para comprobar su funcionamiento en tareas de imputación y, posteriormente, clasificación de patrones incompletos.

En esta sección se mostrarán las conclusiones de la tesis, haciendo hincapié en las principales contribuciones de la misma –relativas a los artículos científicos del Apéndice A– y comentando las líneas futuras de investigación que emanan de este trabajo.

5.1 Contribuciones

En esta tesis doctoral, tras una breve introducción y un repaso a los métodos de imputación más comunes de la literatura, se han presentado los principales trabajos llevados a cabo durante los años de investigación, que contienen las siguientes contribuciones:

1. En primer lugar, se ha introducido una técnica capaz de mejorar la imputación de valores perdidos a través del borrado de algunas muestras conocidas. Aunque esta técnica ha demostrado ser eficiente, puede causar un desequilibrio entre las distribuciones de entrenamiento y test. Para solucionar esto, se ha propuesto un mecanismo de compensación basado en una ligera modificación de la función de error a optimizar. Es importante destacar que, aunque algunos valores de entrada se borran en este proceso, sus correspondientes valores reales se usan como etiquetas durante el entrenamiento. Esto ayuda a la máquina a aprender la reconstrucción de las características incompletas de una forma más eficiente, mejorando los resultados de imputación finales.

2. Se ha demostrado que los mecanismos de borrado y compensación también son útiles a la hora de clasificar patrones incompletos. Se ha visto que un SDAE seguido de un clasificador lineal produce mejores resultados cuando se introduce borrado durante el entrenamiento. Sin embargo, en algunas ocasiones en que la mejor imputación incluye compensación (junto con borrado), no significa que también mejore el resultado de clasificación. A partir de los resultados obtenidos sobre diferentes conjuntos de datos, se observa que las mejoras sólo ocurren en aproximadamente un 45 % de los casos. En el resto, aunque el método propuesto conlleva mejores resultados de imputación, no se traslada a una mejor clasificación. Esto podría ser debido a que la mejora no es lo suficientemente significativa para que la red aprenda a distinguir mejor la clase de la muestra incompleta.

3. En el Capítulo 4 se ha propuesto una idea innovadora basada en expandir la entrada a un clasificador SDAE Modificado (MSDAE) con las salidas de una máquina auxiliar, con el propósito de mejorar el rendimiento mediante el uso de información adicional a la hora de hacer la extracción de características intermedia. El entrenamiento capa a capa de la red minimiza la combinación convexa entre el error correspondiente a la reconstrucción de la entrada y el de las predicciones. El parámetro que regula este aprendizaje entre los errores se escoge mediante Validación Cruzada (CV). Como era de esperar, estas

arquitecturas que hemos llamado clasificadores SDAEs Modificados Completos (CMSDAE), consiguen mejores resultados de clasificación en todos los experimentos que se han llevado a cabo.

4. El primer entrenamiento de un SDAE tiene el efecto de desenmarañamiento de los subespacios de representación, lugar donde se encuentran las diferentes muestras de las clases. Este tipo de desenmarañamiento se basa en las características intrínsecas de las muestras. Obviamente, un entrenamiento directo de una red profunda también ofrece este efecto –la capa final es un clasificador lineal–, pero guiado por la tarea de clasificación. En el caso de un CMSDAE, el desenmarañamiento es guiado tanto por las muestras de entrenamiento como por las de clasificación. En esta tesis también se ha realizado un estudio comparativo de los tres tipos de procesos, revelando aspectos interesante sobre la forma en que funcionan las redes profundas.

5. Finalmente, se ha explorado si las versiones Completas de los MSDAEs (CMSDAEs) son capaces de mejorar los resultados de los MSDAEs en tareas de imputación. Para ello, se han estudiado dos mecanismos diferentes: el propio CMSDAE directamente –como arquitectura monotarea (ST-CMSDAE)–, que aprovecha la información de las etiquetas a la entrada para el entrenamiento capa a capa; y una versión multitarea del mismo (MT-CMSDAE), en el que las observaciones juegan un papel muy importante, ya que son aprendidas como tarea secundaria. Los resultados experimentales muestran que la imputación mediante CMSDAEs nunca es peor a la de los SDAEs, y además es mejor en la mayoría de los casos. En particular, la versión multitarea con CMSDAEs (MT-CMSDAE) proporciona estadísticamente mejores resultados en 5 de los 6 casos estudiados. Este hecho hace a estas estructuras realmente interesantes para trabajar con problemas de clasificación de datos incompletos.

5.2 Líneas futuras

A continuación se comentan posibles extensiones de los estudios desarrollados en este trabajo.

- Acorde a los resultados de imputación obtenidos con las técnicas de borrado y compensación, cabe una posible mejora mediante una ponderación de los errores correspondientes a las muestras completas, borradas e incompletas.

- Como línea abierta de mejora en el rendimiento de un CMSDAE, se propone estudiar diferentes arquitecturas y combinaciones de parámetros de combinación convexa para las capas ocultas. Además, podrían aplicarse técnicas complementarias como diversidad, pre-énfasis y aumento de datos, que serían efectivas para incrementar el rendimiento de los clasificadores profundos.

- Finalmente, sería interesante estudiar cómo introducir las ideas de los ST-CMSDAEs y MT-CMSDAEs en el entrenamiento de otras redes profundas. Resulta especialmente interesante el concepto de incluir etiquetas a la entrada de una red con el objetivo de incrementar el rendimiento de la misma.

Además de las posibles mejoras mencionadas, centradas en los desarrollos presentados en los capítulos anteriores, quedan abiertas algunas preguntas, que podrían ser líneas cercanas a los SDAEs a explotar en futuros trabajos, y que se resumen a continuación.

- Como quiera que en los CMSDAEs se imponen repetidamente como salidas las etiquetas reales del conjunto de entrada durante el entrenamiento capa a capa, ¿puede conseguirse que se hagan resistentes al desequilibrio (*Imbalance*, IB) mediante algún "trick"?

- ¿Qué pasa si en el diseño de (cualesquiera de) los clasificadores SDAE se utiliza en cada paso (i.e., capa a capa) una red neuronal profunda en lugar de una combinación lineal? Si se usa un MLP, tendría que entrenarse una arquitectura con dos capas ocultas. Naturalmente, sólo se deben "congelar" los pesos de entrada, pero no se sesgarán las características profundas que se obtengan por la exigencia de que permitan una clasificación lineal; lo que puede suponer mejoras sensibles en todos los aspectos.

- Obviamente, la técnica del punto anterior también podría emplearse para la construcción iterativa de una red profunda convencional. No es útil si se hace

capa a capa, es decir, pidiendo que cada capa oculta proporcione una estimación del "*target*" mediante una combinación lineal. Pero si se introduce un MLP con más de una capa oculta no ocurre eso. Así se podrían "compactar" los diseños Deep NN: reducir su número de capas y mantener altas prestaciones.

APÉNDICE

A

Publicaciones

A.1 Artículos de revista

- Adrián Sánchez-Morales, José-Luis Sancho-Gómez & Aníbal-R. Figueiras-Vidal. Complete autoencoders for classification with missing values. *Neural Computing and Applications*, en proceso de revisión.

- Adrián Sánchez-Morales, José-Luis Sancho-Gómez & Aníbal-R. Figueiras-Vidal. Exploiting label information to improve auto-encoding based classifiers. *Neurocomputing*, volúmen 370, páginas 104–108, 2019.

- Juan-Antonio Martínez-García, José-Luis Sancho-Gómez, Adrián Sánchez-Morales & Aníbal-R. Figueiras-Vidal. Designing non-linear minimax and related discriminants by disjoint tangent configurations applied to RBF networks. *Neurocomputing*, en proceso de revisión.

- Adrián Sánchez-Morales, José-Luis Sancho-Gómez, Juan-Antonio Martínez-García & Aníbal-R. Figueiras-Vidal. Improving deep learning performance with missing values via deletion and compensation. *Neural Computing and Applications*, páginas 1–12, 2019.

A.2 Contribuciones a congresos

- Adrián Sánchez-Morales, José-Luis Sancho-Gómez & Aníbal-R. Figueiras-Vidal. Values Deletion to Improve Deep Imputation Processes. *International Work-Conference on the Interplay Between Natural and Artificial Computation: Biomedical Applications Based on Natural and Artificial Computing, IWINAC 2017*, La Coruña, España, páginas 240–246, 2017.

- Rosa-María Menchón-Lara, José-Luis Sancho-Gómez, Adrián Sánchez-Morales, Álvar Legaz-Aparicio, Juan Morales-Sánchez, Rafael Verdú-Monedero & Jorge Larrey-Ruiz. Using machine learning techniques for the automatic detection of arterial wall layers in carotid ultrasounds. *Ambient Intelligence - Software and Applications*, Cham, Alemania, páginas 193–201, 2015.

Bibliografía

[1] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley, New York, USA, 2001. 2

[2] D. Paulus and J. Hornegger. *Applied Pattern Recognition (2nd Edition)*. Vieweg, 1998. 2

[3] Donald B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, December 1976. 3, 18, 20, 21

[4] F. Medina and F. Galván. *Imputación de datos: teoría y práctica*. Serie Estudios Estadísticos y Prospectivos Nº 54. Santiago de Chile, CEPAL, 2007. 3, 6

[5] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data*. Wiley, 1986. 5, 6, 21, 32, 46

[6] D. B. Rubin. *Multiple imputation for nonresponse in surveys*. Wiley Series in Probability and Statistics, 1987. 5, 28

[7] J. G. Gómez-García, J. P. Albaladejo, and J. A. Martín-Fernández. Métodos de inferencia estadística con datos faltantes: estudio de simulación sobre los efectos en las estimaciones. 2006. 6

[8] J. L. Schafer and J. W. Graham. Missing data: our view of the state of the art. *Psychol Methods*, 7:147 – 177, 2002. 7

[9] J. Neyman. Outline of a theory of statistical estimation based on the classical thoery of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 236:333 – 380, 1937. 7

[10] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289 – 337, 1933. 7

[11] L. Deng and D. Yu. *Deep learning: methods and applications*. Now Publishers Inc., Hanover, MA, USA, 2014. 9, 10

[12] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958. 10

[13] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989. 12

[14] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J. Machine Learning Res.*, 11:3371–3408, 2010. 12, 14, 63, 64, 74

[15] Y. LeCun, B. Boser, J .S. Denker, D. Henderson, R. E. Howard, and W. Hubbard. Back-propagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989. 12, 63

[16] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief networks. *Neural Computation*, 18:1527–1554, 2006. 12, 63

[17] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Sicience*, 313:504–507, 2006. 12

[18] Y. Bengio, P. Lamblin, D. Popovic, and H. Larochelle. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 153–160, 2007. 12

[19] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE*, 37:233–243, 1991. 13

[20] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 13

[21] P. D. Allison. *Missing Data*. Sage University Papers Series on Quantitative Applications in the Social Sciences, Thousan Oaks, California, USA, 2001. 17

[22] Y. Dodge. *Analysis of experiments with missing data*. New York: Wiley, 1985. 17

[23] J. W. Graham. Missing data analysis: making it work in the real world. *Annual Review Psychology*, 60:549 – 576, 2008. 17

[24] A. R. T. Donders, G. J.M.G. van der Heijden, T. Stijnen, and K. G.M. Moons. Review: a gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, 59:1087 – 1091, 2006. 17

[25] J. Barnard and X. L. Meng. Applications of multiple imputation in medical studies: from AIDS to NHANES. *Statistical Methods in Medical Research*, 8:17 – 36, 1999. 18

[26] P. J. García-Laencina, J. L. Sancho-Gómez, A. R. Figueiras-Vidal, and M. Verleysen. K nearest neighbours based on mutual information for incomplete data classification. In *16th European Symposium on Artificial Neural Networks (ESANN2008), Brujas (Bélgica)*, pages 37–42, 2008. 19

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1 – 38, 1977. 20

[28] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via em approach. In *Advances in Neural Information Processing Systems - NIPS 6*, pages 120–127, 1993. 21

[29] Z. Ghahramani and M. I. Jordan. *Learning from incomplete data*. Technical Report AIM-1509, MIT, Cambridge, MA, USA, 1994. 21

[30] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. New York: Wiley, 1997. 21

[31] C. F. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11:95 – 103, 1983. 21

[32] J. G. Gómez-García, J. P. Albaladejo, and J. A. Martín-Fernández. Métodos de inferencia estadística con datos faltantes: estudio de simulación sobre los efectos en las estimaciones. 2006. 21, 24

[33] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall, 1995. 22

[34] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David, David Botstein, and Russ B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17:520–525, 2001. 26

[35] S. F. Buck. A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *Journal of the Royal Statistical Society, Series B*, 22:302 – 306, 1960. 27

[36] J. L. Schafer. *Analysis of incomplete multivariate data*. Chapman and Hall, 1997. 28, 46

[37] T. Baguley and M. Andrews. Handling missing data. *Modern Statistical Methods for HCI*, pages 57 – 82, 2016. 28

[38] P. Schmitt, J. Mandel, and M. Guedj. A comparison of six methods for missing data imputation. *J Biomet Biostat*, 6:224, 2015. 28

[39] Melissa J. Azur, Elizabeth A. Stuart, Constantine Frangakis, and Philip J. Leaf. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20:40–49, 2011. 29

[40] G. E. Batista and M. C. Monard. A study of K-nearest neighbour as an imputation method. In *Second International Conference on Hybrid Intelligent Systems*, volume 87, pages 251–260, 2002. 30

[41] G. E. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17:519–533, 2003. 30

[42] J. Chen and J. Shao. Nearest neighbour imputation for survey data. *Journal of Official Statistics*, 16:113–131, 2000. 30

[43] P. Jonsson and C. Wohlin. An evaluation of k-nearest neighbour imputation using Likert data. In *Proc. 10th International Symposium on Software Metrics*, pages 108–118, 2004. 30

[44] G. Kalton and D. Kasprzyk. Imputing for missing survey responses. In *Proc. of the Section on Survey Research Methods, American Statistical Association*, pages 22–31, 1982. 30

[45] G. Kalton. Compensating for missing survey data. In *Ann Arbor: Survey Research Center, University of Michigan*, 1983. 30

[46] D.V. Creel and K. Krotki. Creating imputation classes using classification tree methodology. In *Proc. of the Section on Survey Research Methods, American Statistical Association*, pages 2884–2887, 2006. 30

[47] T. Rockel, D. W. Joenssen, and U. Bankhofer. Decision trees for the imputation of categorical data. *Archives of Data Science, Series A (Online First)*, 2:2363–9881, 2017. 31, 32

[48] H. C. Valdiviezo and S. V. Aelst. Tree-based prediction on incomplete data using imputation or surrogate decisions. *Information Sciences*, 311:163–181, 2015. 31

[49] S. L. Salzberg. C4.5: programs for machine learning. *Machine Learning*, 16:235–240, 1994. 31

[50] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. Chapman and Hall, 1984. 31

[51] G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society*, 29:119–127, 1980. 31

[52] L. Rokach and O. Maimon. *Data mining with decision trees: theory and applications*. Series in Machine Perception and Artificial Intelligence: Volume 69, 2007. 31

[53] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan Kaufmann, 2011. 32

[54] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 32

[55] O. Ortega-Lobo and M. Numao. Ordered estimation of missing values. In *PAKDD*, 1999. 32

[56] B. Twala. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23:373–405, 2009. 32

[57] J. R. Quinlan. The effect of noise on concept learning. In *Machine Learning, An Artificial Intelligence Approach Volume II*, pages 149–166, 1986. 32

[58] J. B. Navarro-Pastor and J. M. Losilla-Vidal. Análisis de datos faltantes mediante redes neuronales artificiales. *Psicothema*, 12:503–510, 2000. 32

[59] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989. 32

[60] P. Vamplew and A. Adams. Missing values in a backpropagation neural net. In *Proc. of the 3rd Australian Conference on Neural Networks - ACNN*, pages 3–5, 1992. 33

[61] E. L. Silva-Ramírez, R. Pino-Mejías, M. López-Coello, and M. D. Cubiles de-la Vega. Missing value imputation on missing completely at random data using multilayer perceptrons. *Neural Networks*, 24:121–129, 2011. 33

[62] S. R. Amer. Neural network imputation in complex survey design. *International Journal of Electrical, Computer and Systems Engineering*, 3:52–57, 2009. 33

[63] E. L. Silva-Ramírez, R. Pino-Mejías, and M. López-Coello. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing*, 29:65–74, 2015. 33

[64] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems 29*, pages 2352–2360. Curran Associates, Inc., 2016. 34

[65] C. Doersch. Tutorial on variational autoencoders. 2016. 34

[66] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 34

[67] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015. 34

[68] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera. Handling incomplete heterogeneous data using VAEs. 2018. 34

[69] V. Fortuin, G. Ratsch, and S. Mandt. Multivariate time series imputation with variational autoencoders. 2019. 34

[70] J. T. McCoy, S. Kroon, and L. Auret. Variational autoencoders for missing data imputation with applications to a simulated milling circuit. *IFAC-PapersOnLine*, 51:141–146, 2018. 34

[71] J. Yoon, J. Jordon, and M. van-der Schaar. GAIN: missing data imputation using generative adversarial nets. 2018. 34

[72] C. Shang, A. Palmer, J. Sun, K. S. Chen, J. Lu, and J. Bi. VIGAN: missing view imputation with generative adversarial networks. 2017. 34

[73] Y. Luo, X. Cai, Y. ZHANG, J. Xu, and Y. Xiaojie. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems 31*, pages 1596–1607. Curran Associates, Inc., 2018. 34

[74] R. D. Camino, C. A. Hammerschmidt, and R. State. Improving missing data imputation with deep generative models. 2019. 34

[75] B. K. Beaulieu-Jones and J. H. Moore. Missing data imputation in the electronic health record using deeply learned autoencoders. *Pac Symp Biocomput.*, 22:207–218, 2017. 34

[76] P. Vamplew and A. Adams. MIDA: multiple imputation using denoising autoencoders. In *Advances in Knowledge Discovery and Data Mining*, pages 260–272, 2018. 34, 75

[77] A. Fonseca-Costa, M. Seoane-Santos, J. Pompeu-Soares, and P. Henriques-Abreu. Missing data imputation via denoising autoencoders: the untold story. In *Advances in Intelligent Data Anlysis XVII*, pages 87–98, 2018. 34

[78] S. Ruder. An overview of multi-task learning in deep neural networks, 2017. 35, 39

[79] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, 2008. 35

[80] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, page 8599?8603, 2013. 35

[81] R. Girshick. Fast r-cnn. In *2015 IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1440–1448, 2015. 35

[82] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997. 35, 37, 75

[83] Y. S. Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 6:192–198, 1990. 36

[84] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000. 36

[85] J. Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, Jul 1997. 37

[86] M. Long, Z. Cao, J. Wang, and P. S. Yu. Learning multiple tasks with multilinear relationship networks, 2015. 38

[87] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification, 2016. 38

[88] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2017. 38

[89] A. Sánchez-Morales, J. L. Sancho-Gómez, and A. R. Figueiras-Vidal. Values deletion to improve deep imputation processes. In *Int. Work-Conf. on the Interplay Between Natural and Artificial Computation, IWINAC 2017, Coruna, Spain*, pages 240–246, 2017. 41, 45, 47, 65

[90] A. Sánchez-Morales, J. L. Sancho-Gómez, J. A. Martínez-García, and A. R. Figueiras-Vidal. Improving deep learning performance with missing values

via deletion and compensation. *Neural Computing and Applications*, pages 1–12, 2019. 41, 47, 65, 74, 75, 77

[91] R. F. Alvear-Sandoval and A. R. Figueiras-Vidal. On building ensembles of stacked denoising auto-encoding classifiers and their further improvement. *Information Fusion*, 39:41 – 52, 2018. 43, 64, 74, 75

[92] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by backpropagating errors. *Nature*, 323(6088):533–536, 1986. 43

[93] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995. 43

[94] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006. 43

[95] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993. 43

[96] M. Lichman. UCI machine learning repository, 2013. 52, 68, 69, 78

[97] Delve: Data for evaluating learning in valid experiments. 52

[98] L. Breiman. Bias, variance, and arcing classifiers. *Technical report 460*, April 1996. 53

[99] P. P. Brahma, D. Wu, and Y. She. Why deep learning works: A manifold disentanglement perspective. *IEEE Trans. Neural Networks and Learning Systems*, 27:1997–2008, 2016. 57, 63

[100] Y. LeCun. A learning scheme for asymmetric threshold networks. In *Proceedings of Cognitiva 85*, pages 599–604, Paris (France), 1985. 63

[101] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998. 63

[102] P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral science*. PhD thesis, Harvard University, 1974. 63

[103] D. E. Rumelhart and J. L. McClelland. Learning internal representations by error propagation. In *D. E. Rumelhart and J. L. McClelland, eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambrigde, MA, MIT Press*, volume 1, pages 318–362, 1986. 63

[104] A. Bhandare, M. Bhide, P. Gokhale, and R. Chandavarkar. Applications of convolutional neural networks. *International Journal of Computer Science and Information Technologies*, 7(5):2206–2215, 2016. 63

[105] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017. 63

[106] J. Yu, C. Zhu, J. Zhang, Q. Huang, and D. Tao. Spatial pyramid-enhanced netvlad with weighted triplet loss for place recognition. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2019. 63

[107] O. Jun and L. Yujian. Vector-kernel convolutional neural networks. *Neurocomputing*, 330:253–258, 2019. 63

[108] M. A. Carreira-Perpiñán and G. E. Hinton. On contrastive divergence learning. In *Proc. 10th. Int. Workshop on Artificial Intelligence and Statistics, Barbados: Soc. Artificial Intelligence and Statistics*, pages 33–40, 2005. 63

[109] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35:1798–1828, 2013. 63, 74

[110] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *J. of Big Data*, 2:1–9, 2015. 64

[111] L. Deng and D. Yu. Deep convex net: A scalable architecture for speech pattern recognition. In *Proc. Interspeech 2011, Florence (Italy)*, pages 2285–2288, 2011. 64, 65

[112] B. Hutchinson, L. Deng, and D. Yu. A deep architecture with bilinear modeling of hidden representations: Applications to phonetic recognition. In *Proc. IEEE Int. Conf. Acoustic, Speech and Signal Proc., New York (NY)*, pages 4805–4808, 2012. 64, 65

[113] Rectangles data. 68, 78

[114] Sloan digital sky survey RD14. 69

[115] Lior Rokach. *Pattern Classification Using Ensemble Methods*. World Scientific, 2009. 73

[116] L. Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40:229–242, 2000. 73

[117] V. Gómez-Verdejo, M. Ortega-Moral, J. Arenas-García, and A. R. Figueiras-Vidal. Boosting by weighting critical and erroneous samples. *Neurocomputing*, 69:679–685, 2006. 74

[118] V. Gómez-Verdejo, J. Arenas-García, and A. R. Figueiras-Vidal. A dynamically adjusted mixed emphasis method for building boosting ensembles. *IEEE Trans. Neural Networks*, 19:3–17, 2008. 74

[119] S. Tabik, D. Peralta, A. Herrera-Poyatos, and F. Herrera. A snapshot of image pre-processing for convolutional neural networks: Case study of MNIST. *Int. J. of Computational Intelligence Systems*, 10:555–568, 2017. 74

[120] R. F. Alvear-Sandoval, J. L. Sancho-Gómez, and A. R. Figueiras-Vidal. On improving CNNs performance: The case of MNIST. *Information Fusion*, 52:106–109, 2019. 74

[121] C. C. Tan and C. Eswaran. Reconstruction and recognition of face and digit images using autoencoders. *Neural Computing and Applications*, 19:1069–1079, 2010. 74

[122] A. H. Hadjahmadi and M. M. Homayounpour. Robust feature extraction and uncertainty estimation based on attractor dynamics in cyclic deep denoising autoencoder. *Neural Computing and Applications*, pages 1–14, 2018. 74

[123] Zakhriya Alhassan, David Budgen, Riyad Alshammari, Tahani Daghstani, A. Stephen McGough, and Noura Al Moubayed. Stacked denoising auto-encoders for mortality risk prediction using imbalanced clinical data. *Proc. 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL*, pages 541–546, 2018. 74

[124] A. Sánchez-Morales, J. L. Sancho-Gómez, and A. R. Figueiras-Vidal. Exploiting label information to improve auto-encoding based classifiers. *Neurocomputing. Accepted, 2019*. 74

[125] T. W. Raghunathan, J. M. Lepkowksi, J. Van Hoewyk, and P. Solenbeger. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, 27:85–95, 2001. 77

[126] S. Van Buuren. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, 16:219–242, 2007. 77

Declaration

I herewith declare that I have produced this work without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This work has not previously been presented in identical or similar form to any examination board.

The dissertation work was conducted from 2015 to 2019 under the supervision of Dr. D. José Luis Sancho Gómez at the Universidad Politécnica of Cartagena and Dr. D. Aníbal R. Figueiras Vidal at the Universidad Carlos III of Madrid.

Madrid,

This dissertation was finished writing in Madrid on 10 de diciembre de 2019