IES SERRA PERENXISA

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Tema 2.

Utilización de modelos de Inteligencia Artificial.

Índice

2.1 Introducción.

Desde el comienzo del desarrollo de la inteligencia artificial, prácticamente simultáneo a la construcción de los primeros ordenadores, se ha debatido acerca de qué es la inteligencia artificial y de si realmente una máquina podría ser capaz de pensar. El punto clave de este debate se centra en saber si el hecho de crear hardware y software que se comportan de un modo que parece inteligente, supone un avance en la dirección que permita desarrollar máquinas que sean capaces de pensar por sí mismas. Así pues, este debate se centra en reflexionar acerca de si es posible considerar que lo que hoy en día hacen las máquinas que presentan comportamientos en apariencia inteligentes, se puede estimar que también es pensar al modo en que lo hacemos los humanos.

A lo largo de los años se han sucedido definiciones más o menos completas del concepto de inteligencia artificial. Es importante destacar que no existe unas de una única definición válida para este concepto, sino que hay muchas que aportan matices interesantes sobre lo que realmente es esta disciplina:

- Por ejemplo, Bellman (1978) la definió como **la automatización de actividades que normalmente se asocian con el pensamiento humano**, como por ejemplo la toma de decisiones, la resolución de problemas, el aprendizaje, etc.

- Aunque muy simple, también es de gran interés la definición que propusieron Rich y Knight (1991) para quienes es **el estudio de cómo hacer que las computadoras realicen tareas que, por el momento, llevan cabo mejor las personas.**

El matemático británico Alan Turing (1912-1954) es considerado uno de los padres de las Ciencias de la Computación. Entre sus muchas contribuciones, cabe destacar los trabajos que realizó durante la Segunda Guerra Mundial para interpretar los mensajes enviados por las máquinas de cifrado Enigma (Figura 2.1). También fue este matemático el que formuló la prueba que lleva su nombre. Según este investigador, un ordenador que fuese capaz de pasar la prueba de Turing se podría considerar inteligente.

Figura 2.1 Máquina para el cifrado de mensajes Enigma.

La **prueba de Turing o Test de Turing** consiste en hacer que una persona hable a través de una pantalla y un teclado simultáneamente con un grupo de individuos entre los

que se esconde un ordenador. A la vista de las respuestas que recibe de cada uno de ellos esta persona debería ser capaz de saber quién es el ordenador y quiénes son los seres humanos. Si el ordenador no es descubierto, se podría considerar que ha pasado la prueba de Turing. Dado que la conversación se lleva a cabo en forma de texto a través de un teclado y un monitor, no es necesario que la máquina sea capaz de transformar el texto en voz, aunque esto hoy en día ya es posible. Esta prueba no evalúa conocimientos, dado que un ser humano no lo sabe todo, sino que lo que mide es la capacidad de una máquina de conversar como lo haría un ser humano.

Para saber más...
En una conferencia impartida por Alan Turing en la London Mathematical Society el 20 de febrero de 1947, este dijo: Se considera que los ordenadores solo son capaces de llevar a cabo aquellos procesos para los que son programados. Esto es cierto, en el sentido de que, en la actualidad, si hacen algo distinto a lo que les ha sido programado, se trata de un error. Es además también verdad que la motivación fundamental de la construcción de estas máquinas es la de usarlas a modo de esclavos, asignándoles únicamente trabajos rutinarios tales que la máquina sea capaz de entender lo que tiene que hacer en todo momento. Hasta ahora, los ordenadores se han empleado únicamente de esta forma, pero eso no quiere decir que esto tenga que ser siempre así.
Lo que supone que en aquel momento Turing manifestaba la posibilidad de que en un futuro se desarrollasen máquinas dotadas de cierto grado de inteligencia.

Ejemplo 1
Ya han pasado más de 70 años desde que se enunciara la prueba de Turing y hay máquinas que han conseguido superarla.

En una prueba realizada en Londres el 7 de junio de 2014, un programa desarrollado por los programadores rusos Vladimir Veselov y Sergey Ulasen y el ucraniano Eugene Demchenko fue capaz de superar la prueba de Turing. Este programa es un bot conversacional que adopta la personalidad de un niño ucraniano de 13 años llamado Eugene Goostman. Dicho bot logró engañar al 33 % de los jueces quienes, tras una conversación de 5 minutos, pensaron que, efectivamente, se trataba de un niño ucranian.

Aunque este bot conversacional, según la organización del evento, pasó la prueba de Turing, ha recibido críticas de los especialistas. Entre las críticas que se le han formulado, está que al simular que se trata de un niño ucraniano de 13 años que se comunica en inglés con sus interlocutores, esto hace que ciertas limitaciones del propio programa conversacional puedan enmascararse por las dificultades que tendría un hablante extranjero de dicha edad.

En la actualidad existen multitud de aplicaciones que hacen uso de esta tecnología en múltiples sectores.

Para recordar...
A la hora de formular un problema para garantizar que un sistema de resolución de problemas sea capaz de alcanzar la solución, el medio ha de ser observable, finito y determinista.

2.2 Requisitos básicos de un sistema de resolución de problemas.

Aunque el concepto de **sistema de resolución de problemas** resulte muy general, todo sistema de esta naturaleza debe de ser capaz de evolucionar hasta **alcanzar** una de las **posibles soluciones** del problema.

Los sistemas de resolución de problemas tienen como objetivo alcanzar alguno de los estados en los que el problema se puede considerar resuelto. Así, **para que un sistema de resolución de problemas sea de utilidad**, debe de enfrentarse a una cuestión que sea **capaz de resolver haciendo uso de los medios de los que dispone**, lo que supone que el problema ha de tener solución, aunque esta no tenga que ser única.

Supóngase, por ejemplo, que cierto sistema de resolución de problemas sirve para la conducción de vehículos, se le podría pedir que guie un automóvil, por ejemplo, a Madrid. Si no se especifica nada más, con que fuese capaz de conducir hasta cualquier punto de esta ciudad se podría considerar que ha cumplido su objetivo, pero se debe tener en cuenta que, para poder llegar hasta dicha población, el sistema ha de estar dotado de los recursos que posibiliten la resolución del problema. Es decir, debe contar con capacidades de visión por computador que le permita saber por dónde va conduciendo y si hay obstáculos en su camino, ha de tener un sistema de posicionamiento, una base de datos de mapas, elementos de cálculo, etc. Así, los recursos de los que disponga el sistema serán los que definan qué acciones se pueden acometer y cuáles no. El caso del vehículo autónomo resulta de gran complejidad por el elevado número de tecnologías que comprende y, por tanto, se trabajará con ejemplos más sencillos.

A la hora de formular un problema para garantizar que un sistema de resolución de problemas sea capaz de alcanzar la solución, es necesario que **el medio donde este se formula cumpla una serie de características:**

- En primer lugar, que **sea observable** para el sistema de resolución de problemas. Es decir, que de alguna forma este sistema pueda reconocer y explorar el medio en su totalidad

- En segundo lugar, con el fin de garantizar que se pueda obtener una solución, resulta conveniente que el medio sea **finito y determinista**, entendiendo como tal que ejercer la misma acción sobre el medio, conduzca siempre al mismo resultado.

Así, si el medio es observable, finito y determinista, la resolución de cualquier problema consistirá en la ejecución de un número finito de pasos.

El proceso mediante el que, haciendo uso de una secuencia de acciones, el sistema de resolución de problemas alcanza el objetivo, se denomina búsqueda. Si el medio es determinista, el agente sabrá exactamente dónde se encuentra después de cada una de las acciones que lleve a cabo.

Dado un sistema de resolución de problemas al que se le propone la consecución de un objetivo en un medio observable, finito y determinista, el sistema comienza un proceso

de búsqueda para la consecución del objetivo. Este proceso es lo que se conoce como **proceso de ejecución**.

Para que un sistema de resolución de problemas sea capaz de resolver cierto problema, es necesario disponer de un conjunto de componentes, que son los siguientes:

- **Estado inicial**: se trata del estado desde el que se comenzará el proceso de búsqueda.

- **Acciones que se pueden ejecutar:** en función del problema con el que se esté trabajando, el sistema dispondrá de una serie de posibles acciones que le permitirán pasar de un estado a otro.

- **Modelo de transición:** explica lo que hará cada una de esas acciones.

- **Espacio de estados del problema:** se trata del conjunto de todas las posibles situaciones a las que se puede llegar a partir del punto de partida.

- **Verificación** de que se ha alcanzado el objetivo, que determina si la posición lograda es una de las posiciones objetivo.

- **Coste de ejecución**: en muchos casos, puede resultar de interés el saber cuánto cuesta (en tiempo, dinero, consumo de energía, distancia, número de pasos, etc.) ejecutar cada una de las acciones que son necesarias para llegar al objetivo. Nótese que, en caso de existir más de una solución, estas se pueden priorizar en función del coste de ejecución necesario para alcanzarlas.

Ejemplo 2

Sistema de resolución de problemas.

La Figura 2.2 muestra un laberinto en el que el círculo azul representa la posición inicial. Para salir del laberinto, se debe llegar a alguna de las dos puertas de este, a las que se accede a través de las casillas con el aspa roja.

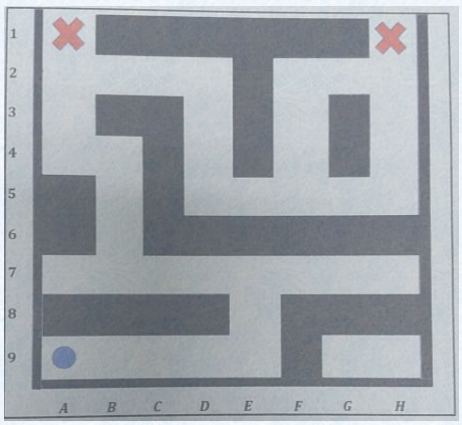

Figura 2.2 Laberinto

En este caso, el entorno donde se resuelve el problema es el propio laberinto, que es observable, finito y determinista, entendiendo como tal que se puede examinar por completo, existe un número finito de posiciones y que, si se elige avanzar desde una posición en cierta dirección hacia otra posición, siempre se llegará al mismo lugar (es decir, desde la posición A9 avanzando dos posiciones hacia la derecha siempre se llegaría a C9).

El estado inicial es la casilla A9, desde la que comienzan los movimientos. Las acciones que se pueden ejecutar se limitan a moverse de cada vez una casilla hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha, sin que esté permitido realizar desplazamientos en diagonal ni tampoco atravesar las paredes del laberinto.

Como se puede observar, estas acciones serán suficientes para poder alcanzar ambas salidas del laberinto. Si se comienza en la posición A9 y se aplica la acción mover una unidad hacia la derecha, que en ese caso es la única dirección posible, el resultado que se obtendrá será siempre la posición B9.

Partiendo de la posición A9, es posible pasar por todas las posiciones que no son de pared, salvo G9 y H9 dado que no son accesibles. Por tanto, el resto de las casillas que no son de pared constituyen **el espacio de estados de este problema.** Estas posiciones son las que aparecen en la Figura 2.3.

A9 B9 C9 D9 E9 E8 A7 B7 C7 D7 B6 B5 E7 F7 G7 H7 B6 B5
B5 D5 E5 F5 G5 H5 A4 B4 D4 F4 H4 A3 D3 F3 H3 A2
B2 C2 D2 G2 H2 A1 H1

Figura 2.3 Relación completa de casillas que constituyen el espacio de estados del problema del laberinto.

Dado el laberinto que se presenta en la Figura 2.2, resulta obvio que es posible alcanzar alguna solución, es decir, llegar, partiendo de A9 hasta alguna de las dos salidas. Así, por ejemplo, con la secuencia B9, C9, D9, E9, E8, E7, D7, C7, B7, B6, B5, B4, A4, A3, A2 y A1, se alcanzaría una de las dos salidas, mientras que con B9, C9, D9, E9, E8, E7, D7, C7, B7, B6, B5, B4, A4, A3, A2, B2, C2, D2, D3, D4, D5,E5, F5, G5, H5, H4,H3,H2 y H1 se llegaría a la otra.

Si se asigna un coste de ejecución de una unidad a cada movimiento, el coste de ejecución de la primera solución sería de 16 unidades y el de la segunda, de 29 unidades. Por tanto, si lo que se pretende es minimizar el coste de ejecución, la primera solución sería mejor que la segunda.

2. 3 Modelos de sistemas de inteligencia artificial

En general, se puede definir **un modelo de sistema de inteligencia artificial** como un **software que ha sido programado o entrenado para reconocer cierto tipo de patrones y/o datos y actuar en consecuencia.** Este aprendizaje puede realizarse bien de manera determinista o bien estadística. En caso de tratarse de un modelo basado en aprendizaje estadístico, se requiere disponer de un conjunto de datos que se empleará para el entrenamiento del modelo.

2.3.1. Los hiperparámetros de un modelo de inteligencia artificial.

En el desarrollo de un modelo de inteligencia artificial intervienen tanto los denominados parámetros como los hiperparámetros.

Vocabulario.
Parámetros de un modelo: **aquellas variables que se estiman durante el proceso de entrenamiento del mismo: es decir, durante su proceso de construcción.**

La forma más habitual de estimar los parámetros de la mayoría de los modelos es mediante el uso de metodologías de optimización en el proceso de entrenamiento. Aquellas variables que se pueden modificar a lo largo del proceso de optimización que se realiza durante el entrenamiento del modelo son las que se denominan **hiperparámetros**. Los valores más adecuados que se deben asignar a cada hiperparámetro dependen del problema objeto de estudio.

Ejemplo 3

TensorFlow Playground (https://playground.tensorflow.org) permite la creación de una red neuronal en la que resulta posible la modificación de gran cantidad de sus hiperparámetros. TensorFlow es una librería open source para machine learning e inteligencia artificial desarrollada por Google. En este ejemplo únicamente se pretende mostrar cómo se podría diseñar, con la ayuda de TensorFlow Playground, una red neuronal, sin entrar en detalles técnicos.

En la Figura 2.4 se muestra una captura de pantalla correspondiente de TensorFlow Playground. Debajo de DATA, se encuentran los distintos conjuntos de datos disponibles y que serán los que se utilicen para entrenar el modelo. Como variables de entrada se emplearán las que se seleccionen debajo de FEATURES. Aunque en todos los conjuntos de datos disponibles existen únicamente dos variables distintas, resulta posible realizar operaciones sobre dichas variables para crear otras nuevas. Así, por ejemplo, se pueden elevar las variables al cuadrado, multiplicar entre sí, o calcular el seno de cada variable.

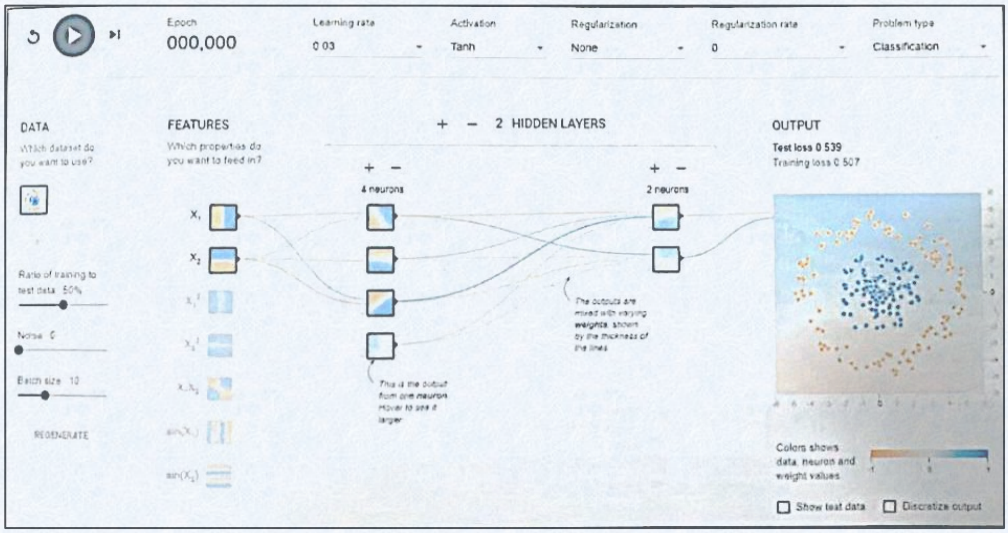

Figura 2.4 Captura de pantalla de TensorFlow Playground.

Una vez seleccionadas las variables de entrada, es posible determinar el número de capas ocultas y neuronas en cada una de ellas. Cuanto mayor sea el número de capas y neuronas por capa, más compleja será la red y más costoso su entrenamiento. Pero, obsérvese también que, en general, los modelos que obtienen los mejores resultados requieren de una arquitectura de cierta complejidad.

Aunque antes del entrenamiento de la red neuronal se puede considerar que todas las neuronas de cada capa de la red están igualmente relacionadas con los de la capa anterior y con las de la capa siguiente, el proceso de entrenamiento hará que algunas de estas relaciones salgan fortalecidas y otras se debiliten.

En la misma pantalla de TensorFlow Playground resulta muy sencillo modificar una serie de hiperparámetros, que son los siguientes:

- **Tasa de aprendizaje (learning rate):** se trata de un hiperparámetro que influirá en la velocidad con la que se actualizan los parámetros de la red.
- **Función de activación (activation function):** se puede escoger entre distintas funciones de activación. Aunque no se entrará en esta unidad en los significados de dichas funciones, cabe señalar que la función de activación es aquella que se aplica a la información de salida de una neurona.
- **Regularización (regularization):** se trata de un parámetro sobre el que es posible actuar con el fin de evitar el sobreajuste. En TensorFlow Playground también resulta posible elegir una tasa de regularización (regulation rate).
- **Tipo de problema (problem type):** se puede elegir entre modelos de clasificación, en los que la salida de la red será una variable categórica que indica a qué grupo pertenece cada conjunto de datos, y modelos de regresión, en los que el valor de la variable de salida es numérico.

Se sugiere al lector que elija el conjunto de datos denominado «circle» para el entrenamiento de un modelo de red neuronal. Una vez seleccionado dicho modelo, se hará uso, como variables de entrada, de X, y X, y se definirán de dos capas ocultas, la primera de cuatro neuronas y la segunda de dos, con una tasa de aprendizaje de 0.03. La función de activación elegida será la de tangente hiperbólica (Tanh) y se seleccionará como tipo de problema el de clasificación (classification). Seguidamente se ejecutará la red durante al menos 1.000 epochs. Con el fin de ver la bondad de los resultados

obtenidos, se recomienda activar los datos de test y dejar el resto de hiperparámetros no mencionados en sus valores por defecto.
Tal y como se puede apreciar en la Figura 2.5, el modelo entrenado clasifica de forma correcta el conjunto de datos empleados para la verificación del modelo. El grosor de las líneas que unen las distintas neuronas muestra la fortaleza de las relaciones que se establecieron entre ellas durante el entrenamiento.

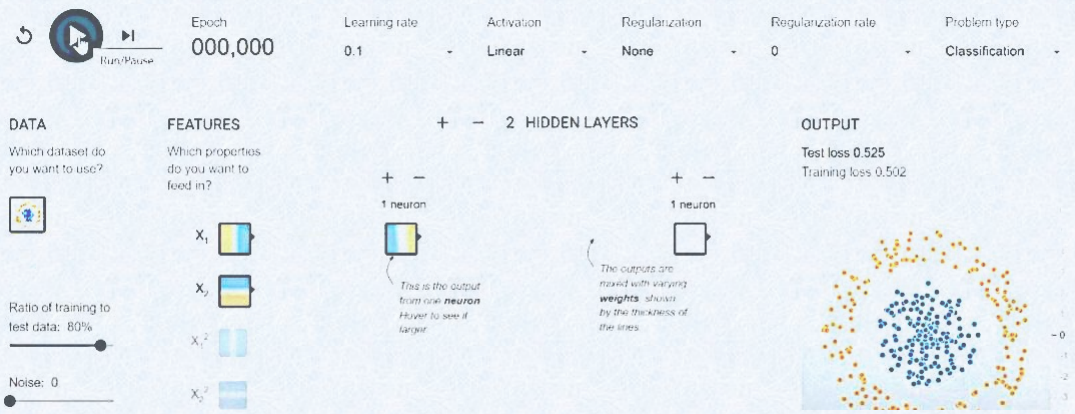

Figura 2.5 Captura de pantalla de los resultados obtenidos con Tensorflow Playground.

Para recordar...
Una hoja de cálculo permite introducir cierta información y obtener una serie de resultados de manera automática, pero no hace uso de ninguna técnica de inteligencia artificial, sino de operaciones matemáticas.

2.3.2 Automatización de tareas

La **automatización de tareas repetitivas** permite liberar capacidades que se pueden usar para realizar otras tareas de mayor complejidad. Efectivamente, una de las mayores utilidades de la informática es la de automatizar tareas repetitivas; sin embargo, no todos los trabajos que se pueden automatizar requieren de inteligencia artificial.

Los avances que se han producido en el campo de la inteligencia artificial, así como la aparición del *deep learning*, **han hecho que comience una nueva era en la automatización de tareas de complejidad elevada y que las máquinas sean capaces de realizarlas más rápido que los humanos**. Dichas tareas incluyen tanto aquellos trabajos que requieren de la fuerza física como otros de carácter intelectual. Actualmente las máquinas pueden llevar a cabo todo este tipo de actividades de forma más rápida y barata que los humanos.

Un ejemplo de automatización de tareas podría ser las líneas de producción en las que se instalan cámaras con capacidades de visión artificial que son capaces de retirar las unidades defectuosas. Esta labor, aunque también la podría realizar una persona, se trata de un trabajo monótono y repetitivo que supone un esfuerzo importante para quien la realiza y que, en muchas aplicaciones, puede ser llevada a cabo de manera satisfactoria por un sistema inteligente.

Otro ejemplo de automatización de tareas podría ser un hospital que recomiende a los familiares de los enfermos allí ingresados, en función de su perfil y nivel de renta, lugares a los que puedan ir a comer o en los que hospedarse. Aunque pueda parecer una tarea poco relacionada con el núcleo de competencias requeridas para la buena gestión de un hospital, precisamente por eso, el no ocupar al personal del centro en estas labores permite disponer de una serie de recursos adicionales que se pueden emplear en tareas de mayor utilidad desde el punto de vista sanitario.
Entre los proyectos típicos de automatización de tareas basadas en inteligencia artificial, **se encuentran aquellos relacionados con la gestión administrativa** de la compañía en la que no se necesita establecer una relación con el cliente.

2.4 Redes neuronales artificiales

El desarrollo de las **redes neuronales artificiales** comenzó en la década de 1950. En un principio, se pretendía simular con ellas el funcionamiento del cerebro huma-no, emulando el comportamiento que tiene una neurona, sus interconexiones y su mecanismo de aprendizaje. Se perseguía crear así un sistema capaz de aprender de forma similar a como lo hace un ser humano.

Con el paso del tiempo, han ido surgiendo nuevas ideas que mejoran el funcionamiento de las redes neuronales, como el **algoritmo de retropropagación**, los filtros convolucionales o su implementación en tarjetas de procesamiento gráfico para acelerar su funcionamiento.

El éxito de las redes neuronales en su aplicación práctica viene ligado a su capacidad de generalización. Así, si se dispone de una **red neuronal artificial entrenada** con datos procedentes de cierta distribución de probabilidad, se ha comprobado que cuando a dicha red se le presentan nuevos datos procedentes de la misma distribución, las estimaciones que realiza son correctas.

Existen **dos claves en el funcionamiento** de las redes neuronales. Una es su algoritmo de aprendizaje, que utiliza entradas y salidas conocidas para calcular los pesos que poseen las distintas neuronas que integran una red. Mediante este proceso, denominado entrenamiento, es posible crear un modelo capaz de encontrar características no solo ante entradas conocidas, sino que además puede generalizar dichos descubrimientos a otros casos que no se han utilizado en el proceso de entrenamiento. La segunda clave es la utilización de los filtros convolucionales. Dichos filtros ayudan a extraer las características que poseen los datos de entrada con un bajo coste computacional.

En los últimos años, la utilización de redes neuronales se ha extendido enormemente en diferentes campos, Algunas de sus aplicaciones más importantes se encuentran en el procesamiento de imágenes, el reconocimiento de voz, o el análisis y traducción de textos. Todos estos avances, desembocaron en lo que se ha llamado deep learning o aprendizaje profundo, Para abordar la teoría del deep learning, es necesario conocer una serie de conceptos relacionados con disciplinas tales como el cálculo, la teoría de la aproximación, la optimización, el álgebra lineal o la computación científica.

Para recordar…
Los filtros convolucionales ayudan a extraer las características de los datos de entrada con un bajo coste computacional.

El término *deep learing* hace referencia a modelos que son capaces de aprender haciendo uso de composición de funciones, sin que tenga relación directa con término neurológico alguno. El motivo por el que en la actualidad no se hace uso de los modelos biológicos para la configuración de las redes de *deep learning* es, fundamentalmente, la falta de información relativa al cerebro humano. A pesar de ello, dado que la investigación biológica ha demostrado que, por ejemplo, los hurones son capaces de aprender a ver con la parte del cerebro destinada a la audición, eso sugiere que este animal hace uso de un único algoritmo para resolver distintas tareas. Este descubrimiento motivó que en el campo del machine learning se especule con la posibilidad de entrenar un mismo algoritmo para realizar también distintas tareas.

Para recordar…
El desarrollo que tuvo lugar entre 2012 y 2015 en las redes neuronales convolucionales dedicadas a la clasificación de imágenes fue muy importante.

2.4.1 Redes neuronales preentrenadas

Seguidamente se introducen algunas redes neuronales preentrenadas destacando sus características fundamentales.

- AlexNet es una red neuronal convolucional diseñada por Alex Krizhevsky, Ilya Suts-kever y Geoffrey Hinton. Si bien no se trata de la primera red convolucional de la historia, dado que ya se habían probado algunas arquitecturas de este tipo más de 20 años antes de la creación de este modelo, sí se puede decir que se trata de la red neuronal convolucional que mayor influencia ha tenido sobre desarrollos posteriores. La popularidad de AlexNet comenzó tras su victoria en el concurso ImageNet Large-Scale Visual Recognition Challen-ge de 2012, siendo el modelo computacional con un mejor rendimiento ante un problema de clasificación de imágenes. AlexNet está formada por ocho capas, de las que las cinco primeras son de tipo convolucional y las tres últimas son totalmente conectadas.

- GoogLeNet es otra red neuronal convolucional que se presentó al concurso ImageNet Large-Scale Visual Recognition Challenge en su edición de 2014, ganando dicha edición con mejores resultados que los obtenidos por AlexNet en 2012. Esta red está formada por 27 capas. Las imágenes de entrada que recibe la red para su clasificación son de 224 x 224 píxeles y, a medida que la imagen se va transmitiendo de unas capas a otras, el tamaño de la misma se reduce, pero tratando de conservar la máxima información posible.

En el año 2014, también participó en este concurso una red denominada **VGG**, obteniendo resultados también de gran interés. Como evolución de dicha red, surgió la denominada **VGG-19,** se trata de una red convolucional, formada por 19 capas. Esta red es capaz de clasificar imágenes en 1.000 categorías y hace uso también como entrada de imágenes de 224 x 224 píxeles.

ResNet-50 (Residual Networks 50) se trata de una red neuronal convolucional que está formada por 50 capas. La popularidad de este modelo de red neuronal comenzó tras ser el ganador del concurso ImageNet Large-Scale Visual Recognition Challenge de 2015. La explicación en profundidad de la arquitectura de **ResNet50** va más allá del

alcance de este texto. Sin embargo, se recomienda el uso de la API de deep lerning Keras con el fin de practicar el uso de alguno de los modelos pre-entrenados de ResNet50.

Existen versiones de esta red que han sido entrenadas con millones de imágenes procedentes de la base de datos ImageNet (www.image-net.org). Esta red neuronal preentrenada es capaz de clasificar cualquier imagen que se le presente como perteneciente a alguna de las 1.000 categorías preestablecidas.

Para recordar…
En el año 2012 en el concurso ImageNet Large-Scale Visual Recognition Challenge AlexNet alcanzó una tasa de errores de clasificación del 15,3%; en el de 2014 GoogLeNet lo redujo a 6,67 %, y en el de 2015 ResNet-50 al 3,57 %.

2.5 Sistemas de razonamiento impreciso.

Los sistemas de razonamiento pueden describirse por la exactitud que necesitan al realizar cualquier paso de su proceso de razonamiento. Los sistemas de razonamiento más preciso son aquellos que solo se ocupan de las relaciones lógicamente válidas, conocidas con certeza.

Ejemplos de este tipo de sistemas de razonamiento son **los sistemas aristotélicos clásicos, las matemáticas y otras lógicas clásicas**. En estos casos, las conclusiones son verdaderas, si las premisas son verdaderas.

Ejemplo 4

En la lógica aristotélica, el razonamiento consiste en un encadenamiento de juicios en el que se parte de una proposición conocida y se llega a otra u otras no conocidas. Considérese la siguiente frase:

Si apruebas, irás de vacaciones.

Supongamos que se ha producido el hecho:

Has aprobado.

Haciendo uso de la lógica aristotélica resulta que:

Luego irás de vacaciones.

La noción central del sistema lógico de Aristóteles es el silogismo. Un silogismo es un discurso en el que establecido cierto hecho, resulta necesariamente una consecuencia.

Un ejemplo clásico de silogismo es el siguiente:

Todos los hombres son mortales.
Todos los griegos son hombres.
Por tanto, todos los griegos son mortales.

Existen metodologías que permiten el razonamiento bajo condiciones de incertidumbre. El uso de estos sistemas cobra gran importancia cuando se trata de construir agentes que van a operar en situaciones reales en las que, por tanto, se manejará la incertidumbre.

Entre las aproximaciones existentes para el manejo de la incertidumbre se incluyen el uso de métodos probabilísticos como la inferencia bayesiana y la lógica difusa.

2.6 Inferencia bayesiana.

La inferencia bayesiana tiene sus fundamentos en los trabajos efectuados en el siglo XVIII por Thomas Bayes quien estudió el problema de la determinación de la probabilidad de las causas mediante los efectos observados.

Es decir, la inferencia bayesiana permite obtener conclusiones acerca de los datos disponibles teniendo también en cuenta lo que se conoce acerca del resultado.

De manera intuitiva, se puede afirmar que la probabilidad de ocurrencia de un evento viene dada por el cociente entre el número de posibles ocurrencias de dicho evento dividido por el número total de eventos posibles.

Así, la probabilidad de ocurrencia de un evento A se expresa como:

$$p(A) = \frac{\text{número ocurrencias de } A}{\text{número total de eventos}}$$

Asimismo, la probabilidad condicionada de ocurrencia de un evento A se define como la probabilidad de que ocurra A condicionada a la ocurrencia de otro evento, por ejemplo B, y se expresa como:

$$p\,(A \,/\, B) = \frac{p\,(A \cap B)}{p\,(B)}$$

Es decir, se trata del cociente de la probabilidad de que ocurran A y B simultáneamente y que representamos por p (A n B), dividido por la probabilidad de que ocurra B a la que se denomina p (B).

Ejercicio resuelto 1

En una empresa de 200 empleados, 40 de ellos tienen conocimientos de programación en lenguaje Python y 120 hacen uso habitual del sistema operativo Linux. Además, de entre los 40 con conocimientos de Python, hay 20 que usan de forma habitual Linux.

En caso de tomar un empleado al azar, ¿cuál es la probabilidad de que dicho empleado sea un usuario habitual del sistema operativo Linux si se sabe que posee conocimientos de programación en lenguaje Python?

Solución

Sea A el hecho de que un empleado tenga conocimientos de lenguaje Python y B el hecho de que el empleado sea usuario habitual del sistema operativo Linux, se calculará primeramente cuál es la probabilidad de que ambos eventos se produzcan de manera simultánea:

$$p\,(A \cap B) = \frac{20}{200} = 0{,}1$$

Además, la probabilidad de que un empleado sea usuario habitual del sistema operativo Linux viene dada por:

$$p\,(B) = \frac{120}{200} = 0{,}6$$

Por tanto, la probabilidad de que un empleado tomado al azar sea un usuario habitual del sistema operativo Linux si se sabe que tiene conocimientos de programación en lenguaje Python viene dada por:

$$p(A/B) = \frac{p(A \cap B)}{p(B)} = \frac{0,1}{0,6} = 0,167$$

A la hora de estudiar la probabilidad de algunos eventos, conviene tener en cuenta que, si dados dos eventos A y B estos se consideran independientes, es decir, si el uno no depende del otro, entonces se verifica que p (A / B) = p (A).

Conviene, por tanto, recordar la siguiente fórmula con su correspondiente simplificación:

p (An B) = p (A / B) • p (B) = p (A) • p (B)

Para estos dos mismos eventos, la probabilidad de ocurrencia de la unión de A y B viene dada por:

p (AU B) = p (A) + p (B) - p (An B)

Si lo que se tiene es un conjunto completo de eventos, la probabilidad condicionada de uno de estos elementos A, está definida por el teorema de Bayes.

Teorema de Bayes.

Sean {A_1, A_2,... , A_n} un conjunto completo de eventos, es decir que cumplen A_1 U A_2 U ..., U A_n = I siendo I el evento seguro (100% de probabilidad de ocurrencia), todos ellos con probabilidad mayor que cero e incompatibles dos a dos, y sea B otro evento del que se conocen las probabilidades condicionadas p (A_i / B).
Entonces, se verifica que:

$$p(A_i/B) = \frac{p(B/A_i)\cdot p(A_i)}{p(B)} = \frac{p(B/A_i)\cdot p(A_i)}{\sum_{k=1}^{n} p(B/A_k)\cdot p(A_k)}$$

Ejercicio resuelto 2

Una empresa dedicada al desarrollo de software comercializa tres productos distintos. El 60% de sus ventas corresponden al software K, el 30% al software L y el 10% restante al software M. Según datos históricos de los que dispone la propia empresa, la probabilidad de que un cliente que ha comprado el software K llame al servicio de asistencia técnica es del 2%, mientras que el que lo haga uno que ha comprado el software L es del 4% y que lo haga alguien que ha comprado el software M es del 1%. En estos momentos está entrando una nueva llamada en el servicio de asistencia técnica, ¿cuál es la probabilidad de que la llamada sea para pedir soporte del software K?

Solución:

En primer lugar, se plantean todas las probabilidades conocidas a la vista del enunciado expresadas en tanto por uno.

p (K) = 0,6	p (L) = 0,3	p (M) = 0,1
p (llamada / K) = 0,02	p (llamada / L) = 0,04	p (llamada / M) = 0,01

Se aplica seguidamente la fórmula del teorema de Bayes particularizándola para este problema:

$$p\,(llamada\,/\,K) = \frac{p\,(llamada\,/\,K)\cdot p\,(K)}{p\,(llamada\,/\,K)} =$$

$$= \frac{p\,(llamada\,/\,K)\cdot p\,(K)}{p\,(llamada\,/\,K)\cdot p\,(K) + p\,(llamada\,/\,L)\cdot p\,(L) + p\,(llamada\,/\,M)\cdot p\,(M)}$$

Sustituyendo los valores obtenidos se llega al siguiente resultado:

$$p\,(llamada\,/\,K) = \frac{0{,}02\cdot 0{,}6}{0{,}02\cdot 0{,}6 + 0{,}04\cdot 0{,}3 + 0{,}01\cdot 0{,}1} = 0{,}48$$

Es decir, la probabilidad de que la llamada que está entrando sea para pedir soporte del software K es del 48%.

2.7. La lógica difusa

La lógica difusa se puede considerar como una extensión de la lógica booleana, basada únicamente en dos estados, en el sentido de que trata de cuantificar situaciones intermedias. Por ejemplo, si alguien dice «hace calor», se trata de una percepción subjetiva, dado que para una persona cierta temperatura puede ser percibida como frío y para otra como calor. Expresiones del tipo «hace un poco de calor» encajan dentro del campo de la lógica difusa.

La lógica difusa es capaz de procesar distintos grados de verdad, pero estos no deben ser interpretados en el sentido de probabilidades, pues lo que representan dentro del **ámbito de la lógica difusa es la probabilidad de pertenencia a cierto conjunto**, en vez de la probabilidad de que ocurra un evento.

Si se dispone de una botella numerada como 1 con agua que tiene un grado de pertenencia difuso del 80% al conjunto de agua potable. Por otro lado, la botella numerada como 2 tiene una probabilidad del 80% de ser potable. ¿De qué botella es menos arriesgado beber? Se ha de interpretar que la primera botella tiene un contenido de agua que es bastante similar al de otras botellas que son potables, alrededor de un 80%, mientras que en el caso de la segunda botella lo que se quiere decir es que, si se bebe el agua de una botella de ese tipo, el 80% de las veces que se han analizado contenían agua potable. Pero, cuidado, que en un 20% de las ocasiones su contenido completo era de agua no potable.

Por tanto, la lógica difusa permite asignar un valor de pertenencia a cualquier conjunto de entre 0 y 1. Es decir, por medio de la lógica difusa es posible asignar la pertenencia parcial a un conjunto.

Ejemplo 5

Sea x la edad de un individuo cualquiera, esta siempre será igual o mayor que 0. Así, por ejemplo, un adolescente puede pensar que él es absolutamente joven y que la gente es completamente joven hasta los 25 años, siendo ya «nada joven», a partir de los 40 años. Desde el punto de vista matemático, esto se pues modelizar por medio de la siguiente ecuación:

$$Y(x) = \begin{cases} 1 \; si \; x \leq 25 \\ \frac{40 - x}{15} \; si \; 25 < x \leq 40 \\ 0 \; si \; x > 40 \end{cases}$$

La Figura 2.6 muestra la representación gráfica de la ecuación. Tal y como se puede apreciar en ella, hasta los 25 años, la pertenencia al conjunto joven es de 1, disminuyendo de forma lineal dicha pertenencia hasta los 40 años en los que se ha reducido a 0.

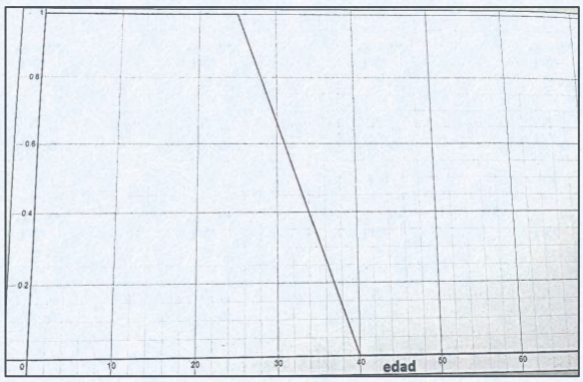

Figura 2.6 Pertenencia al conjunto joven en función de la edad.

Este ejemplo muestra cómo se podría modelar el concepto de «ser joven».

2.8. Sistemas basados en reglas

El diseño de los sistemas basados en reglas comenzó en la década de 1960. Un sistema basado en reglas suele tener la forma de un conjunto de reglas del tipo SI... ENTONCES, conocidas en inglés como IF...THEN.

Los sistemas basados en reglas, más conocidos como **sistemas expertos,** hacen uso de estas con el fin de realizar deducciones o de elegir entre posibles alternativas.

Estos sistemas evolucionan desde un estado inicial haciendo uso exclusivamente del conjunto de reglas existente.

Todo sistema basado en reglas consta, al menos, de cuatro componentes principales:

- Una **lista de reglas**, que constituye la base de datos de conocimiento que empleará.
- Un **motor** para realizar inferencias que combina la información disponible en cada momento con la lista de reglas para producir la evolución del sistema.
- Una **memoria** para almacenar la información requerida en cada momento.
- Una **interface** de usuario que permita algún tipo de conexión con el mundo exterior.

Los sistemas basados en reglas son, en general, programas de ordenador capaces de simular algunas de las características del conocimiento humano con el fin de realizar tareas que normalmente llevan a cabo únicamente personas expertas.

Los dos componentes más importantes para un sistema basado en reglas son el propio sistema de reglas y el motor para realizar inferencias:

- La primera parte, el sistema de reglas, representa hechos acerca del mundo.
- La segunda parte, el motor de inferencia, permite que se infiera nuevo conocimiento haciendo uso de las reglas previamente definidas.

La construcción de un sistema basado en reglas se apoya tanto en el empleo de conocimiento experto como en el uso de información acumulativa disponible en bases de datos.

La **ventaja fundamental** de los sistemas basados en reglas es que se trata de sistemas cuyo mecanismo de funcionamiento es fácil de entender, se pueden construir haciendo uso de conocimiento experto y se aplican en cualquier campo. La relación entre la causa y el efecto es clara y conocida. Ofrece una metodología adecuada para la modelización de muchos de los mecanismos mentales básicos y sirve para mecanizar el proceso de razonamiento.

Entre las **desventajas** de los sistemas basados en reglas se encuentra que, para hacer un sistema de este tipo generalmente se requiere un dominio del campo de conocimiento en el que se va a aplicar. Además, debe tenerse en cuenta que, en problemas difíciles, la generación de reglas se puede volver un proceso de alta complejidad

Ejemplo 6

Sistema basado en reglas

La Figura 2.7 muestra de forma gráfica un ejemplo de un sistema basado en reglas. En este ejemplo, se pretende clasificar cuatro tipos de animales haciendo uso de tres preguntas. En primer lugar, se pregunta si el animal tiene pelo. En caso de ser así, si el animal tiene las garras retráctiles se supone que es un gato y si no, un perro. En el caso de que el animal no tenga pelo, se pregunta si éste sabe nadar, y un animal que sabe nadar y no tiene pelo sino plumas es el pato, mientras que un ejemplo de animal que tampoco tiene pelo, pero no sabe nadar podría ser el gorrión.

Figura 2.7 Gráfico de un sistema basado en reglas.

Ejemplo 7

Sistema basado en reglas

Seguidamente se presenta un código fuente escrito en el lenguaje Python que implementa este ejemplo. Aunque se ha escrito en Python que es un lenguaje de propósito general, existen otros lenguajes de programación como LISP y PROLOG enfocados a la programación de sistemas basados en reglas, fundamentalmente sistemas expertos.

```
print(" ¿Tiene pelo? (sí/no)")
pelo = input()
y=5
if pelo = = "sí":
     print("¿Tiene garras retráctiles? (sí/no)")
     garras=input()
     if garras = = "sí":
          print("Es un gato")
     else:
          print("Es un perro")
else:
     print(" ¿Sabe nadar? (si/no)")
```

```
     nadar=input ()
     if nadar = ="si":
          print("Es un pato")
else:
          print("Es un gorrión")
```

Figura 2.8 Código Fuente

Si la construcción es de un sistema basado en reglas algo más complejas, se emplea el diagrama de cajas, que condensa la información relativa a todos los casos disponibles mediante un gráfico.

En estos diagramas, la línea horizontal inferior de cada una de las cajas representa el valor del percentil 25 (o primer cuartil) de los datos objeto de estudio, mientras que la línea central es la mediana y la línea de la parte superior de la caja representa el percentil 75 (o tercer cuartil) de ese conjunto de datos. La distancia comprendida entre el primer y el tercer cuartil se denomina rango intercuartílico. Los bigotes que sobresalen hacia arriba y hacia abajo de la caja lo hacen, por la parte inferior hasta una distancia que es igual al valor del primer cuartil menos el rango intercuartílico multiplicado por 1,5 y por arriba hasta una distancia que es la suma del tercer cuartil más el rango intercuartílico multiplicado también por 1,5. Cualquier valor que se obtenga más allá de esa distancia, tanto por arriba como por debajo, se considerará un espurio y normalmente se representará por medio de un círculo o un aspa (Figura 2.9).

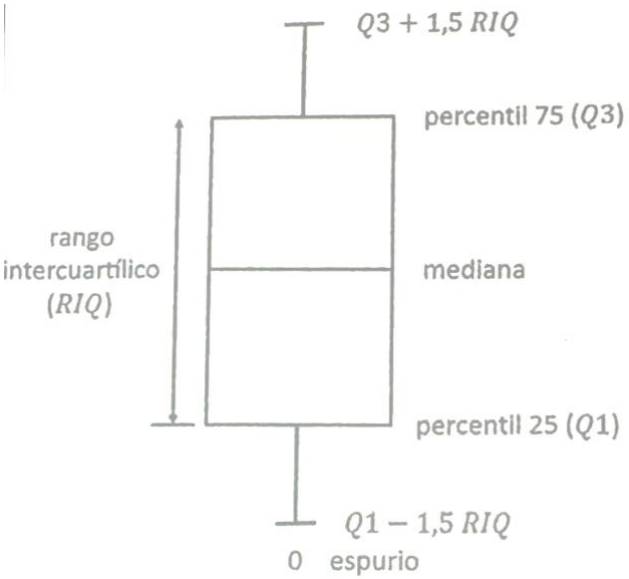

Figura 2.9 Ejemplo de un diagrama de cajas o boxplot.

Para recordar...

El diagrama de cajas también es conocido por su nombre en inglés boxplot.

Ejemplo 8

Construcción de un sistema basado en reglas algo más complejo.

Se empleará una base de datos de características de flores de iris formada por 150 muestras. Se trata de una base de datos muy conocida que introdujo Ronald Fisher (1890-1962) y que se emplea para ilustrar todo tipo de problemas relacionados con el machine leaming y la estadística. Esta base de datos contiene información relativa a la longitud y ancho de los pétalos y sépalos de tres especles diferentes de esta planta, denominadas setosa, versicolor y virginica.

Se quiere que este sistema basado en reglas sea capaz de diferenciar entre las tres variedades de iris en función de las características de sus flores. Dado que no se poseen conocimientos expertos en botánica que permitan realizar esta distinción, se analizará la información disponible con el fin de encontrar posibles diferencias entre las especies. Para ello, en primer lugar, se hará uso de los diagramas de cajas.

Así, en este caso, en lo referente a la longitud del sépalo, la especie setosa es la que presenta un valor inferior en su mediana, seguida de la especie versicolor y de la especie virginica. Para esta variable, únicamente la especie virginica presenta un espurio (Figura 2.10).

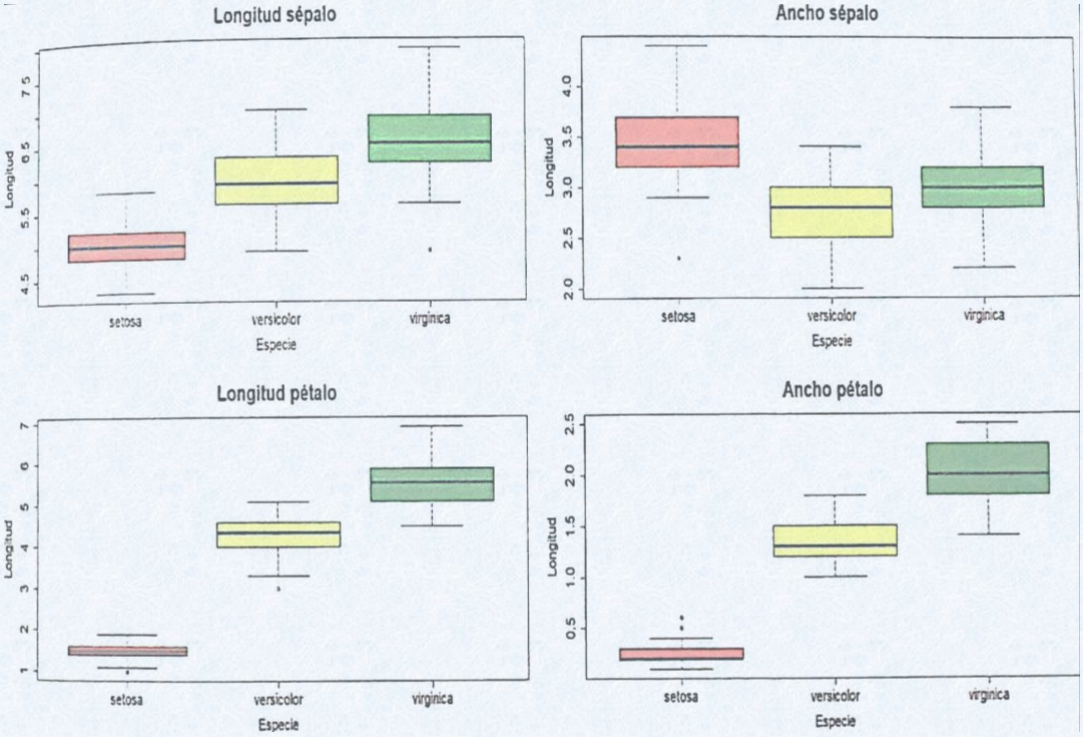

Figura 2.10 Diagramas de cajas de las variables longitud sépalo, ancho sépalo, longitud pétalo y ancho pétalo de la base de datos de las flores de iris.

En el caso de la variable ancho del sépalo, los valores más altos corresponden a la especie setosa, cuyo valor de percentil 25 supera ampliamente al valor del percentil 75 de la especie versicolor y tiene un valor muy próximo al percentil 75 de la especie virginica. En esta variable, únicamente la especie setosa presenta algún espurio.

En el caso de las variables longitud y ancho de pétalo, se observa cómo existe una clara diferencia en los valores que toman estos valores en las distintas especies, aunque la existencia de cierto solape entre ellas hace que no sea posible hacer uso de una de estas variables como clasificador exclusivo de la especie a la que pertenece cada una de las

plantas analizadas. Es decir, aunque la longitud del pétalo o el ancho del mismo permitan discriminar a las setosas de las otras dos especies, la discriminación que se consigue entre las especles versicolor y virginica haciendo uso únicamente de estas dos variables no es perfecta.

Para realizar la clasificación, se calculará un modelo cuyo código fuente se recoge en la Figura 2.11. Dicho modelo hace uso de la librería party del software estadístico R. Se trata de una librería en la que se implementa la partición recursiva. La partición recursiva es una metodología multivariante cuyo objetivo es el de construir árboles de decisión en los que se recoja la influencia de las variables independientes de un modelo sobre la variable dependiente.

```
# carga de la base de datos

data(iris)

# traducción de los nombres de las variables al español

colnames(iris)<-c("longitud  sépalo","ancho  sépalo","longitud  pétalo","ancho
pétalo","especie")

# gráfico de cajas de las cuatro variables de la base de datos

par(mfrow=c(2,2))

boxplot(iris$'longitud  sépalo'~iris$'especie',xlab="Especie",ylab="Longitud",
col=c('red','yellow',          'green'),cex.lab=1.5,cex.axis=1.5,main="Longitud
sépalo",cex.main=2)

boxplot(iris$'ancho     sépalo'~iris$'especie',xlab="Especie",ylab="Longitud",
col=c('red','yellow','green'),cex.lab=1.5,cex.axis=1.5,main="Ancho     sépalo",
cex.main=2)

boxplot(iris$'longitud  pétalo'~iris$'especie',xlab="Especie",ylab="Longitud",
col=c('red','yellow',          'green'),cex.lab=1.5,cex.axis=1.5,main="Longitud
pétalo",cex.main=2)

boxplot(iris$'ancho     pétalo'~iris$'especie',xlab="Especie",ylab="Longitud",
col=c('red','yellow','green'),cex.lab=1.5,cex.axis=1.5,main="Ancho     pétalo",
cex.main=2)

# en caso de que no estén instaladas, resulta necesario proceder a la instalacion
de las

# librerías antes de cargarlas

install.packages("party")

# se procede a la carga de las librerías

library(party)

# se crea modelo de clasificación

modelo  <-  ctree(iris$'especie'  ~  iris$'longitud  sépalo'+iris$'ancho
sépalo'+iris$'longitud pétalo'+iris$'ancho pétalo', data = iris)

# representación gráfica del modelo de clasificación generado

plot(modelo)
```

Figura 2.11 Código fuente aplicado para la generación de un modelo de clasificación.

Más adelante se profundiza en la teoría necesaria para la construcción de este tipo de modelos, centrándonos en este momento en interpretar los resultados que se obtienen de la aplicación de este código (Figura 2.12).

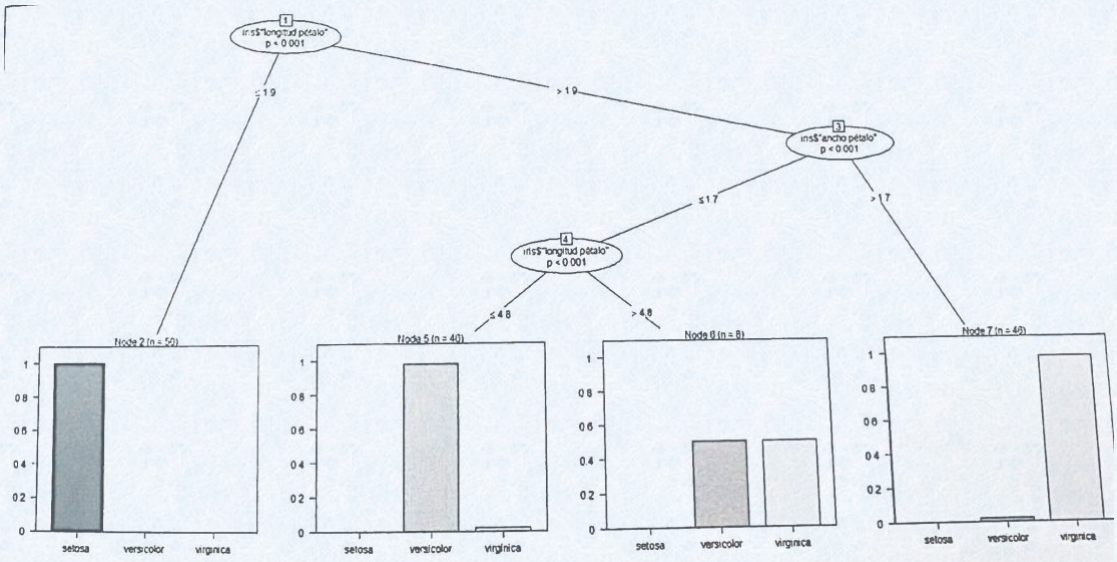

Figura 2.12 Resultados del modelo de partición recursiva aplicado a la base de datos de las flores de iris.

Tal y como ya se ha indicado anteriormente, la Figura 2.12 recoge los resultados del modelo de partición recursiva aplicado a la base de datos de las flores de iris. Dado un elemento de la base de datos, si su longitud de pétalo es menor o igual que 1,9, se considera que se trata, con una probabilidad del 100% de un miembro de la especie setosa. En caso de que la longitud sea mayor que 1,9, si el ancho del pétalo es mayor que 1,7, será con una probabilidad muy elevada de la especie virginica, pero dicha probabilidad no es del 100%, dado que también sería posible que perteneciera a la especie versicolor. Finalmente, en el caso de flores con una longitud de pétalo mayor de 1,9 y ancho de pétalo menor o igual que 1,7, en el caso de aquellas longitudes de pétalo menores o iguales que 4,8, con muy alta probabilidad serían de la especie versicolor, existiendo también una cierta probabilidad de pertenecer a la especie virginica. Finalmente, en caso de longitudes de pétalo por encima de 4,8, la probabilidad de ser de la especie versicolor o de la especie virginica es prácticamente la misma. De manera simplificada y sin tener en cuenta aquellas opciones con bajas probabilidades, el árbol de decisión de la Figura 2.12 se presenta en forma de pseudocódigo en la Figura 2.13.

```
SI LONGITUD PÉTALO > 1.9

ENTONCES
        SI ANCHO PÉTALO > 1.7
            ENTONCES la planta es de la especie virgínica
            SI NO
                    SI LONGITUD PÉTALO > 4.8
                        ENTONCES la planta es versicolor (50%) o virgínica (50%)
                        SI NO la planta es versicolor
SI NO la planta es de la especie setosa
```

Figura 2.13 Pseudocódigo que recoge de forma simplificada los resultados del modelo de partición recursiva obtenido.