Trabajo Fin de Máster

Clasificación de nubes de puntos tridimensionales de parcelas forestales mediante técnicas de aprendizaje profundo

Diego Laíño Rebollido

Máster en Técnicas Estadísticas

Curso 2021-2022

Propuesta de Trabajo Fin de Máster

Título en galego: Clasificación das nubes de puntos tridimensionais de parcelas forestais mediante técnicas de aprendizaxe profunda
Título en español: Clasificación de nubes de puntos tridimensionales de parcelas forestales mediante técnicas de aprendizaje profundo
English title: Classification of 3D point clouds from forest plots through deep learning techniques
Modalidad: Modalidad B
Autor/a: Diego Laíño Rebollido, Universidade da Coruña
Directores: Javier Roca Pardiñas, Universidad de Vigo, Carlos Cabo Gómez, Universidad de Oviedo
Breve resumen del trabajo: **1.** Introducción a la toma de datos de parcelas forestales: para qué se toman y cómo se toman. Las nubes de puntos y software para su tratamiento **2.** El problema de clasificación aplicado a las nubes de puntos tridimensionales de parcelas forestales. Aprendizaje profundo en este contexto. **3.** Exposición de los datos empleados. Tratamiento y modelización de los datos. **4.** Presentación y visualización de los resultados obtenidos. **5.** Discusión y consideraciones.
Recomendaciones: Conocimientos básicos de análisis multivariante y de aprendizaje estadístico, especialmente de aprendizaje profundo. Manejo de los lenguajes de programación R y Python.

Don Javier Roca Pardiñas, Doctor de la Universidad de Vigo, y don Carlos Cabo Gómez, Doctor de la Universidad de Oviedo, informan que el Trabajo Fin de Máster titulado

Clasificación de nubes de puntos tridimensionales de parcelas forestales mediante técnicas de aprendizaje profundo

fue realizado bajo su dirección por don/doña Diego Laíño Rebollido para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En A Coruña, a 30 de Agosto de 2022.

Los directores:

Don Javier Roca Pardiñas

ROCA PARDIÑAS JAVIER - 77592181W
Firmado digitalmente por ROCA PARDIÑAS JAVIER - 77592181W
Fecha: 2022.09.01 18:45:58 +02'00'

Don Carlos Cabo Gómez

CABO GOMEZ CARLOS - 11433734C
Digitally signed by CABO GOMEZ CARLOS - 11433734C
Date: 2022.09.01 18:32:16 +02'00'

El autor:

Don Diego Laíño Rebollido

LAIÑO REBOLLIDO DIEGO - 78591559F
Firmado digitalmente por LAIÑO REBOLLIDO DIEGO - 78591559F
Fecha: 2022.09.01 17:53:09 +01'00'

Agradecimientos

Me gustaría aprovechar este espacio del Trabajo de Fin de Máster reservado a agradecimientos para poner el foco en las personas que, si bien no se pueden denominar como autores de la obra, sin su apoyo este trabajo no habría visto la luz. En este documento hay incontables horas de investigación, etiquetado manual de datos, escritura de código en varios lenguajes, ensayo y error, comprobación de resultados, diseño de gráficos, formateo del documento... un sin fin de tareas llevadas a cabo en un periodo de tiempo de un año. Y tanto nunca se hace solo.

No se hace solo porque son los momentos con los seres queridos de desconexión lo que sirve para recargar las pilas y poder seguir escribiendo. Es solo a través de esos momentos cuando uno ve sentido en lo que hace, en el por qué y para qué terminar lo que se empezó y en querer hacerlo lo mejor posible. Y es que, en realidad, el trabajo académico no se puede desvincular de la vida privada, pues esta en el contexto en el que se desarrolla el primero. Quiero listar a continuación, a todas las personas que han hecho posible que esa vida privada haya sido la mejor posible, y que han sido los encargados de que yo haya sido capaz ahora de escribir estas líneas.

A mi familia, por su apoyo incondicional y constante durante este año, pero también durante toda mi vida. Son ellos quienes, sin duda, me han conducido hasta este mismo instante siendo mi principal fuente de inspiración, de buenos consejos y, cómo no, paño de lágrimas.

A mis amigos, siempre dispuestos a sacarme de casa cuando lo he necesitado y hacerme sentir que cada hora de trabajo tiene su recompensa en momentos de diversión. Cada vez que he sentido la necesidad de escapar momentáneamente de las obligaciones académicas, han estado ahí para asegurarse de que lo conseguía.

Y a mi pareja, que es quién ha visto más de cerca la evolución de este proyecto y, seguramente, la persona que ha tenido que soportar más veces mis idas y venidas. Ha sido capaz de mantenerse permanentemente paciente y le estoy enormemente agradecido por ello.

Pero no solo es de mis seres cercanos mi agradecimiento: también es de mis compañeros de clase, quienes estaban en las mismas que yo y te hacen ver que no estás solo en esta carrera de fondo. De mi tutor Carlos, quien ha sido seguido de cerca el desarrollo del trabajo y ha sido una fuente incesante de consejos, sabiduría y entendimiento. De mi director Javier, que ha conseguido la conexión entre estadística y gestión de recursos que ha hecho posible este trabajo y, de manera más general, a todos los profesores de la Universidad de A Coruña, la Universidad de Santiago y la Universidad de Vigo por brindarme los conocimientos y las herramienta necesarias para completar el máster.

¡Mil gracias a todos!

Con afecto,

Diego

Índice general

Resumen

Resumen en español

En este trabajo se contextualiza el uso de las técnicas de aprendizaje profundo en la clasificación de nubes de puntos tridimensionales de parcelas forestales como herramienta para mejorar los inventariados forestales y se desarrolla un modelo que clasifica las estructuras presentes en ellas. Con este fin, se introduce la voxelización de las nubes de puntos tridimensionales como una manera de reducir el coste computacional del posterior tratamiento de los datos, y el cómputo de descriptores geométricos a partir del análisis de componentes principales de un entorno de puntos como variables predictoras de las estructuras presentes en tal entorno. Tales descriptores geométricos serán después empleados como inputs de una red neuronal de perceptrón multicapa que separa los puntos presentes en las nubes de puntos tridimensionales asociadas a las parcelas forestales empleadas como material en las categorías 'arbusto', 'hierba', 'rama' y 'tronco'.

El objetivo final de este trabajo es desarrollar una técnica que permita acelerar la tarea de realizar inventariados forestales y facilitar la estimación de parámetros que permitan asesorarse del estado de conservación de los bosques, su potencial económico, su respuesta frente a agentes externos o, simplemente, sobre su estructura en cierto instante de tiempo.

English abstract

This document contextualizes the use of deep learning techniques in the classification of three-dimensional point clouds of forest plots as a tool to improve forest inventories and a model that classifies the structures present in these objects is developed for that matter. Thus, the voxelization of three-dimensional point clouds is introduced as a means to reduce the computational cost of subsequent data processing, and the computation of geometric descriptors from the principal component analysis of a point surroundings as predictors on the structures present in such space. Such geometric descriptors will then be used as inputs to a multilayer perceptron neural network that divides the points present in the three-dimensional point clouds associated with the forest plots used as material in the categories 'shrub', 'grass', 'branch' and ' trunk'.

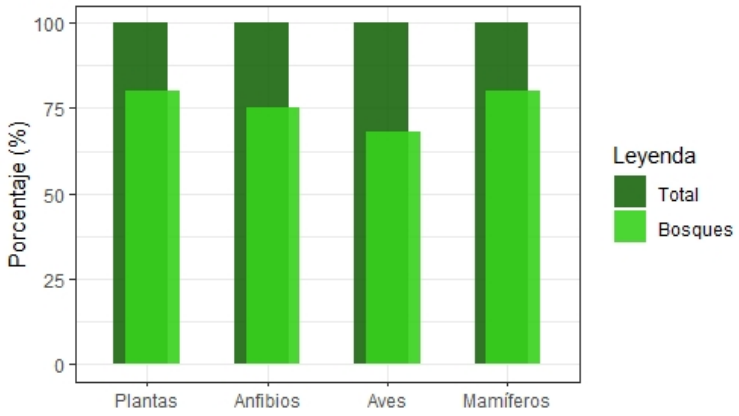
The main objective of this work is to develop a technique that speeds up the task of carrying out forest inventories and facilitates the estimation of parameters that would allow the assessment of the state of conservation of forests, their economic potential, their response to external agents or, simply, to monitor its structure at given times of interest.

Prefacio

Todo el mundo es consciente, en menor o mayor grado, de que los bosques son importantes, bien sea directamente para la humanidad o simplemente para otras especies que indirectamente nos beneficien. Aun así, es probable que, si se le preguntase a alguien por qué esto es así, no le sea fácil dar una respuesta concisa y explicativa que justifique su importancia. Puede que las posibles respuestas que se obtengan de esta pregunta estén relacionadas con la capacidad de los bosques de producir parte del oxígeno que respiramos, por su condición de estar formados por árboles, que sea por la cantidad de recursos, como la madera, que se extrae de ellos, por la variedad de formas de vida que los habitan o simplemente porque son un espacio de divertimento e incluso hogar para una parte considerable de la población humana. Me gustaría aprovechar el espacio reservado de este trabajo al prólogo para introducir, a través de un resumen informal de sus características más notables, qué es lo que hace de los bosques un ecosistema tan vital para nuestra supervivencia como especie y sociedad y, además, enmarcar cuál es el estado de los bosques con respecto de los principales agentes que amenazan su integridad actualmente.

Probablemente el dato con el que es más sencillo comprender el papel literalmente vital de los bosques es hablar de la cantidad y variedad de vida que albergan. Existen en el planeta alrededor de 390000 especies vegetales conocidas, de las que en torno a 60000 son árboles (Beech E et al. 2017). El 60 % de todas esas especies (no solo de los árboles, si no de todas las especies vegetales), aproximadamente, habitan en bosques tropicales, y un 20 % en otros tipos de bosque (Willis KJ 2017). Pero no solo eso: de cerca de las 70000 especies de vertebrados conocidas y descritas (IUCN 2019), los bosques proporcionan hogar a casi 5000 especies de anfibios (el 80 %), a 7500 especies de aves (75 %) y a más de 3700 especies de mamíferos (68 %) (FAO y UNEP 2020). Esto se visualiza en la Figura 1:

Figura 1: Porcentaje de especies en el bosque.

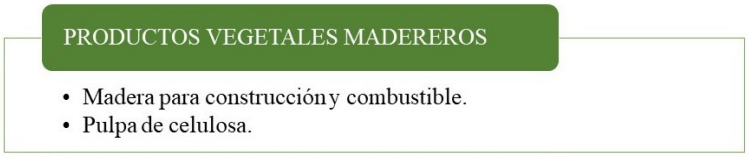


En el aspecto de animales invertebrados, las cifras son mucho mayores: De las más de 1300000 de especies descritas, la gran mayoría vive en bosques, aunque es difícil atestiguar cuál es el porcentaje real de las que habitan en el bosque puesto que la cifra no para de aumentar día a día. De estos invertebrados, el 77 %, aproximadamente, son insectos y, entorno al 8 %, arácnidos. A esta lista hay que sumar otros seres vivos, más difíciles de cuantificar, pero también muy numerosos: por ejemplo, las más de 15000 bacterias del suelo y los 97000 hongos descritos en los ecosistemas forestales (Orgiazzi et al. 2016).

Los bosques forman, por tanto, ecosistemas de elevadísima biodiversidad, y esto por sí solo ya es un hecho que coloca a los bosques en el pedestal de las cosas de las que debemos preocuparnos por cuidar y preservar. Tal biodiversidad es necesaria para que los ecosistemas sean funcionales y suministren oxígeno (en el caso de los bosques, proporcionan en torno al 10 % del oxígeno atmosférico global, como puede verse en Malhi et al. 2011), aire y agua limpios, se dé la polinización de las plantas, o se auto-controlen las plagas gracias a las interacciones depredador-presa de la cadena trófica. Y en realidad, en la mayoría de los casos, tales ecosistemas forestales ya se encuentran en una situación saludable y funcional debido tanto a actividades de gestión forestal pasadas como a las medidas de protección medioambientales vigentes. Así, proporcionan recursos, oportunidades y experiencias realmente diversas y únicas a la población.

Destacadamente, los bosques proporcionan la materia prima para la elaboración de un amplio abanico de productos, tanto de madera como de otros tipos, que son cruciales en muchos aspectos de la sociedad (FAO 2020). Entre los primeros se encuentra la madera usada en construcción, la pulpa de celulosa para la fabricación de papel, herramientas y muebles y la madera utilizada directamente como combustible. Esto se recoge en la Figura 2. Por ejemplo; se estima que, en 2021, unas 2.6 mil millones de personas (34 % de la población mundial en ese momento) dependían de la madera como combustible para cocinar, y que, durante el periodo del 2010 al 2020, al menos 1.3 mil millones de personas vivían en casas donde las paredes, techos o suelos estaban construidos principalmente de productos forestales (FAO 2022).

Figura 2: Resumen de los principales recursos madereros que se extraen de los bosques para su uso humano.



Por otra parte, y tal y como se recoge en la Figura 3, del bosque también se obtienen productos no madereros como el bambú, de uso en construcción, objetos del hogar, herramientas y tejidos; el ratán, empleado en la fabricación de muebles, ropa y objetos de decoración; el corcho, usado para embotellar y también en la construcción; cortezas, látex, gomas, y resinas, así como productos de origen animal: el cuero, la piel y la cera de abejas, entre muchos otros. No hay cifras recientes para el valor total que estos productos aportan a la economía mundial, pero estimaciones del mercado europeo de estos productos ponen una cifra de 1.6 mil millones de euros en venta de productos vegetales de origen forestal: un 47 % proveniente del mercado de plantas ornamentales, un 29 % de alimentos como hongos, bayas y nueces y un 24 % de otros productos vegetales. El mercado europeo de productos de origen animal provenientes del bosque, por su parte, se estima en 620 millones de euros, de los que el 51 %, aproximadamente, son resultado de la venta de carne de caza, el 46 % de miel y cera salvaje y 3 % restante de pieles, cuero, trofeos de caza y otros productos de origen animal (Forest Europe 2015).

Figura 3: Resumen de los principales recursos no madereros que se extraen de los bosques para su uso humano.

Sin embargo, en otros casos, la situación de algunos ecosistemas forestales no es sostenible sin un riesgo, inaceptable, de degradación por incendios descontrolados, insectos o enfermedades.

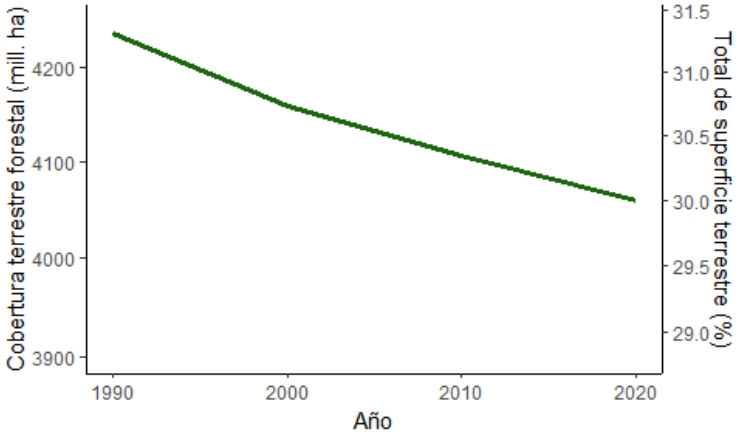
Un análisis global del área forestal afectada por incendios entre 2003 y 2012 estima que se quemaron, en media, 67 millones de hectáreas al año (FAO 2020 II), y un análisis de los incendios ocurridos en 2015 estima que en ese año ardieron de 98 millones de hectáreas (van Lierop et al. 2015). Estos fuegos forestales causan grandes pérdidas de vidas humanas, enormes daños en propiedades e infraestructuras, así como un inmenso daño medioambiental y económico, tanto en términos de recursos destruidos como en costes para tratar de suprimir los fuegos.

Pero los incendios forestales no son los únicos agentes que degradan directamente los bosques. En 2015, bajo estimaciones del mismo estudio, alrededor de 40 millones de hectáreas fueron afectadas por otros tipos de perturbaciones: especialmente plagas y enfermedades. Varias pestes forestales causadas por microorganismos son, en efecto, una amenaza para los bosques de todo el planeta (IPCC Secretariat 2021). El hongo *Cryphonectria parasitica* ha llevado al castaño americano a un estado crítico de conservación en América del Norte; la bacteria *Xylella fastidiosa* provoca la muerte de millones de olivos en Europa; el pseudohongo *Phytophthora kernoviae* ataca el endémico kauri de Nueva Zelanda y otra especie del mismo género, *P. pinifolia*, tiene en jaque a las poblaciones de pinos de Chile. Pero no solo hay pestes que se localicen en un único continente: el nemátodo *Bursaphelenchus xylophilus* es una de las pestes forestales más devastadoras y que afecta a los pinares, con un impacto significativo en ecosistemas naturales de Europa, África, América del Norte y Asia (CABI 2021).

Así con todo, la cobertura terrestre forestal es de aproximadamente 4 mil millones de hectáreas, aproximadamente el 30 % de la superficie terrestre total. Para completar este dato, es indicado apuntar que la pérdida neta anual de cobertura forestal entre 2010 y 2020 se estima que fue de 4.7 millones ha/año, comparada con las 5.2 millones ha/año de la década entre 2000 y 2010 y las 7.8 millones ha/año de la década anterior (FAO 2021). En la Figura 4 se muestra un gráfico de líneas que representa esta tendencia decreciente.

Es de notar cómo esta cifra es contraria con la dada anteriormente sobre la superficie quemada, que hablaba de muchas más hectáreas perdidas. Esta diferencia se debe a los esfuerzos colectivos de reforestación y la expansión natural de los bosques por el terreno favorable para su desarrollo. Además, una reducción en la deforestación total no implica que la deforestación en términos relativos haya disminuido. Aun así, el balance neto sigue siendo negativo: Se está perdiendo más bosque del que se recupera y, el que se recupera, lo hace de manera muy lenta. El ecosistema forestal que se pierde tan fácilmente en unas pocas jornadas de tala masiva o incendios tarda no menos de varias decenas

Figura 4: Evolución de la superficie terrestre total cubierta por vegetación.



de años en recuperarse, si llega a hacerlo. Una labor de suma importancia para frenar el retroceso de los bosques es la de desarrollar técnicas que permitan monitorizarlos y modelarlos. El recopilado de información sobre la estructura interna de los bosques y su comportamiento frente a los agentes con los que interactúan es crucial importancia para numerosas tareas de gestión forestal, tanto como para optimizar la extracción de los recursos económicos que ofrecen, como elaborar planes de mantenimiento de la biodiversidad y ecosistemas que estos albergan. Este trabajo es mi pequeño aporte a este amplio campo de investigación que tiene como tarea, en esencia, ayudar a entender los bosques.

Capítulo 1

La toma de datos de parcelas forestales

Estimaciones como la dada en el prefacio acerca de la cobertura forestal habitualmente se realizan a partir de fotografía aérea e imágenes satelitales y, si bien son una forma rápida y adecuada de medir el porcentaje de cobertura forestal, no dan apenas información sobre la estructura interna del bosque ni sobre el estado en el que se encuentra.

1.1. La biomasa forestal

Habitualmente, cuando es necesario asesorarse de la cantidad de recursos presentes en un bosque, su estructura o su estado antes y después de algún evento con capacidad de transformarlo, se realizan mediciones sobre su biomasa aérea. Esta se define como la cantidad total de materia orgánica viva por encima del nivel del suelo expresada en toneladas secas. Por ejemplo, la cantidad de madera que se puede talar tradicionalmente se evalúa haciendo estimaciones de la biomasa del bosque en cuestión. Así mismo, evaluar la densidad de biomasa aérea total de los bosques, expresada por unidad de área, es una forma útil de cuantificar la cantidad de recursos disponibles para todos los usos tradicionales del bosque.

En sí, la cantidad de biomasa en un bosque es el resultado de la diferencia entre la producción por fotosíntesis y el consumo por respiración celular, pero también se ve modulada por los procesos de transformación del medio que ocurran en el bosque. Por lo tanto, es una medida útil para evaluar los cambios en la estructura forestal que son inducidos por el hombre a través de actividades como la silvicultura, la cosecha y la degradación, pero también los producidos por la sucesión natural del ecosistema, el impacto sobre el medio de los incendios forestales y, de manera más general, el cambio climático. Así, la densidad de biomasa también es una variable útil para comparar los atributos estructurales y funcionales de los ecosistemas forestales en una amplia gama de condiciones ambientales. Pero no solo eso; la biomasa de los bosques también proporciona estimaciones de las reservas de carbono en la vegetación forestal porque alrededor del 50 % es carbono. En consecuencia, la biomasa se puede emplear como indicador de la cantidad potencial de carbono que se podría agregar a la atmósfera en forma de CO_2 cuando se tala y/o quema el bosque. De manera similar, las estimaciones de densidad de biomasa también son útiles para calcular la cantidad de CO_2 que se puede eliminar de la atmósfera mediante la regeneración de bosques o mediante la plantación, dado que establecen las tasas de producción de biomasa y los límites superiores para el secuestro de carbono.

Clásicamente se han empleado dos enfoques para estimar la densidad de biomasa de los bosques (Brown et al. 1989), y cada uno lleva asociada una serie de deficiencias metodológicas que los hacen susceptibles de ser mejorados o sustituidos. El primer enfoque se basa en el uso de estimaciones de

$BV = VOB \times WD \times BEF$

Cuadro 1.1: Ecuación general de estimación de densidad de biomasa en base al volumen.

volumen medidas existentes convertidas en densidad de biomasa (t/ha). A continuación, en el Cuadro 1.1, se muestra la ecuación general que se emplea bajo esta metodología:

Donde BV = densidad de biomasa forestal ($t \cdot ha^{-1}$); VOB = *volume over bark* ($m^3 \cdot ha^{-1}$), es decir, el volumen del tronco incluyendo la corteza; WD = *volume-weighted average wood density* ($g \cdot cm^{-3}$), es decir, la densidad de la madera media medida en el volumen de los troncos con los que se está estimando la biomasa total y BEF = *biomass expansion factor* o factor de expansión de biomasa, que es un coeficiente que expresa el ratio entre la biomasa seca medida en árboles completos y la biomasa seca del volumen inventariado (es decir, solo de los troncos). Esta metodología tiene el incoveniente obvio de que depende de qué tan bien se hayan realizado las estimaciones de VOB y WD, y que el coeficiente BEF es extremadamente dependiente de la muestra empleada: la cantidad de ramas de un árbol a otro, aún dentro de la misma especie, es enormemente variable. Además, por esta misma razón, el BEF calculado en una parcela forestal solo puede ser empleado para esa misma parcela y otras similares, por lo que habrá que recalcularlo cada vez que se quiera medir la biomasa de un bosque (hay que recordar que para calcularlo se requiere obtener la biomasa seca de varios árboles y este es un proceso destructivo y costoso). El valor de WD también sufre de limitaciones parecidas aunque en menor escala, ya que la densidad de la madera no varía tanto y, en la mayoría de los casos, se puede utilizar los valores obtenidos en estudios anteriores para especies similares de árboles y existen tablas de densidades de madera que se pueden consultar en fuentes como Reyes et al. 1992.

El segundo enfoque, por su parte, pasa por usar funciones de regresión que emplean como variables explicativas una o varias dimensiones del árbol, y que están basadas en mediciones de la biomasa seca al horno de una muestra vegetal. Se lista a continuación, en el Cuadro 1.2, algunos ejemplos de funciones de regresión, obtenidas también de Brown et al. 1989, que se emplean con este fin:

1	$Y = e^{-1,996+2,32 \cdot ln(D)}$
2	$Y = 10^{-0,535+log_{10}(BA)}$
3	$Y = 34,4703 - 11,7883D + 1,1926D^2$
4	$Y = 42,69 - 12,8D + 1,242D^2$
5	$Y = e^{-3,1141+0,9719 \cdot ln(D^2H)}$
6	$Y = e^{-2,134+2,53 \cdot ln(D)}$
7	$Y = 21,297 - 6,953D + 0,742D^2$
8	$Y = e^{-3,3012+0,9439 \cdot ln(D^2H)}$

Cuadro 1.2: Ejemplos de funciones de regresión de biomasa.

En este caso, la variable respuesta Y expresa la biomasa en kg por árbol y las variables explicativas D, BA y H son el diámetro del tronco a la altura del pecho, en cm, el área basal del tronco en cm^2 y la altura total del árbol en m, respectivamente. Tales ecuaciones se han obtenido tras quemar árboles previamente inventariados, medir su biomasa seca y obtener la función que mejor se ajustase a los datos, bien frente a D o frente a BA. Este método presenta una serie de claras limitaciones: el más aparente es que tales ecuaciones solo se pueden utilizar para estimar la biomasa de árboles cuyas dimensiones estén dentro del rango empleado en la muestra con la que se estimaron los coeficientes de la función, ya que no son extrapolables. Además, el empleo de una ecuación u otra depende de la región y del régimen de lluvias presente en la zona, lo que lleva a la necesidad de tener numerosas ecuaciones de regresión de biomasa. Por ejemplo, de las ecuaciones listadas en el Cuadro 1.2, las tres primeras ecuaciones son para bosques de zonas secas según el sistema de clasificación de zonas de vida de Holdridge (Holdridge 1967); las ecuaciones 4, 5 y 6 son para bosques de zonas húmedas y las dos últimas son para bosques de zonas muy húmedas. Además, el criterio de usar una ecuación u otra según el régimen de lluvias que lleva a un tipo de bosque u otro puede ser ambiguo y, en general, solo deben ser usado en altitudes próximas al nivel del mar: un régimen de lluvias de 1200 mm en tierras bajas se corresponde con una zona seca, pero esa misma pluviosidad se corresponde con una zona húmeda a partir de los 2500 m de altitud. Otra deficiencia de este método, y que comparte con el anterior, surge de la intención de modelizar la estructura compleja de un bosque con ecuaciones generales. Es habitual que las funciones de regresión de biomasa empleen un diámetro medio para representar a un grupo de árboles heterogéneo. Esto acarrea errores obvios de infraestimación o sobreestimación de la biomasa total, amplificados sobre todo por el hecho de que el volumen de un tronco aumenta aproximadamente con el cuadrado de su radio. Así, si en el bosque hay numerosos árboles con troncos gruesos, la estimación siempre será inferior al valor real.

En cualquier caso, sea el enfoque que se emplee para estimar biomasa (el uso de uno u otro dependerá de particularidades que no se discutirán aquí), ambos requieren de inventarios forestales con los que obtener una muestra para estimar los parámetros con los que, al fin y al cabo, se predecirá la biomasa total. Tales inventarios forestales se recolectan de grandes áreas de muestra utilizando métodos de muestreo diseñados para extraer datos representativos de la población de interés. Sin embargo, este inventariado tampoco está exento de problemas. Además de los problemas ecónomicos asociados a cualquier tipo de estudio estadístico (en este caso, se reflejan en el hecho de que los inventarios tienden a realizarse en bosques que se consideran con valor comercial), realizar inventarios de bosques a partir del muestreo llevado a cabo por observadores que se encargan de tomar mediciones de los árboles conlleva una serie de obstáculos metodológicos que, en la práctica, hacen que las estimaciones de la biomasa sean muy variables. Por ejemplo, se debe considerar el diamétro mínimo de los árboles en el inventario. Se suele tomar como umbral el que el diámetro de los árboles sea de 10 cm a la altura del pecho; esto excluye a una gran cantidad de árboles jóvenes o estrechos que, en realidad, también aportan biomasa al bosque. No solo eso, si no que también es frecuente que, a partir de cierto diámetro máximo (por ejemplo, 80 cm), todos los árboles se consideren de igual dimensión. Este procedimiento se justifica con la dificultad de medir el diámetro en árboles muy gruesos, pero inevitablemente lleva a infraestimar la biomasa real del bosque. Tampoco suelen incluir a los troncos muertos o que simplemente están caídos.

1.2. Teledetección

Debido a todos los motivos anteriormentes descritos, en las últimas décadas, se ha tratado de encontrar mejores maneras de estimar la biomasa; sobre todo, haciendo uso de la tecnología. Una manera efectiva de mejorar la estimación es haciendo más eficiente el inventariado forestal. Como ya se ha dicho, el proceso clásico de muestrear a mano los árboles que se consideren adecuados deja por el camino gran cantidad de información que, por practicidad, no es viable recoger y, que en diversas situaciones solo se realiza en parcelas con interés económico, tiene márgenes claros de mejora. Es en este

contexto donde entra en juego la teledetección y, más concretamente, las técnicas LiDAR (siglas del inglés *light detection and ranging*). Estas presentan una oportunidad sin precedentes de obtener datos espaciales de alta resolución, y ha venido siendo aplicada durante los dos últimos decenios (Wulder 1998) de forma masiva para obtener información sobre la estructura de los bosques.

LiDAR es una tecnología relativamente reciente de teledetección que emplea pulsos de luz láser para detectar y medir los objetos diana. Esta tecnología emergente tiene la capacidad de medir con precisión estructuras tridimensionales (por ejemplo, la estructura de los bosques) y está contribuyendo en el desarrollo de gran parte de las innovaciones realizadas en todos aquellos campos donde obtener información de las características geométricas del cuerpo en estudio sea deseable. Su funcionamiento se basa en medir la altura o distancia de los objetos en base al tiempo que tarda el pulso láser en salir y volver del sensor, empleando para ello luz del espectro electromagnético comprendido entre los 900 y los 1064 nm de longitud de onda (Lefsky 2002). Este escaneo produce una nube de puntos tridimensional, que no es más que un conjunto de puntos con coordenadas (x, y, z) asociadas y que pueden ser visualizadas fácilmente en un ordenador con el fin de analizar la estructura modelada. Estas serán tratadas en la siguiente sección, dedicada a las nubes de puntos tridimensionales y el *software* habitual para tratarlas.

Además, es habitual diferenciar clases de LiDAR atendiendo a la plataforma en la que se instalan los dispositivos de teledetección y desde la que se capturan los datos. Lo más habitual es hablar de *Escaneado Láser Satelital*, (*SLS* por sus siglas en inglés) cuando se toman los datos desde un satélite, *Escaneado Láser Aerotransportado* (*ALS*), para referirse a los dispositivos LiDAR implementados en drones aéreos, *Escaneado Láser Móvil* (*MLS*), cuando se instalan en vehículos móviles terrestres y *Escaneado Láser Terrestre* (*TLS*), cuando simplemente se capturan los datos desde un punto estático terrestre, que, habitualmente, suele un trípode. En el Cuadro 1.3, adaptado de **Cheng et al. 2018**, se resumen las principales características de las cuatro variantes.

Plataforma	**Abreviatura**	**Perspectiva de Escaneado**	**Densidad de la nube de puntos**	**Aplicaciones habituales**
Aérea	ALS	Vista cenital	Relativamente dispersa	Mapeado del terreno, monitoreo forestal, áreas urbanas
Vehículo terrestre	MLS	Vista lateral	Densa	Mapeado de vías, áreas urbanas
Trípode	TLS	Vista lateral	Densa	Monitorización de deformaciones, ingeniería inversa
Satélite	SLS	Vista cenital	Baja densidad	Monitoreo forestal, mediciones atmosféricas, monitoreo de nieve

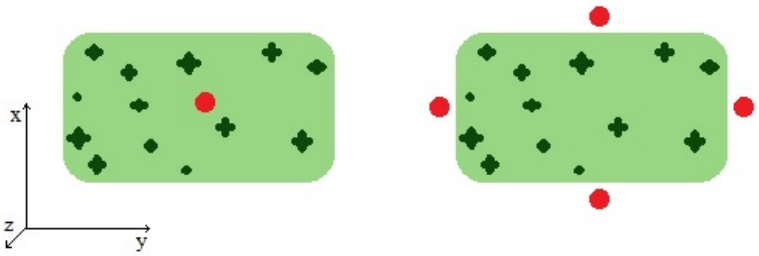
Cuadro 1.3: Tabla resumen de las principales variantes de LiDAR.

Cada una de estas variantes será más o menos útil dependiendo de la estructura que se pretenda escanear y, consecuentemente, unas son usadas con más frecuencia que otras en campos diferentes del conocimiento. Por ejemplo, el ALS proporciona nubes de puntos de tridimensionales desde una vista cenital del objeto, que, como suelen tomarse desde una distancia mayor, estarán conformadas por puntos relativamente dispersos. Esta técnica es ideal para escanear grandes superficies que no requieran demasiado detalle, como pueda ser el mapeado general del terreno, el monitoreo de parámetros

forestales como el avance o retroceso de los bosques o elaborar mapas urbanos. Por su parte, El MLS y el TLS toman datos desde una perspectiva frontal/lateral, desde una distancia habitualmente no muy lejana, y que son ideales para capturar estructuras de las que se requiera mayor detalle, como puedan ser las vías (con sus señales de tráfico) o la monitorización de procesos que impliquen un cambio estructural del que se desee tomar nota. Por último, el Escaneado Láser Satelital, de manera similar con el Aerotransportado, toma los datos desde una perspectiva cenital, aunque con todavía menos densidad de puntos, y es especialmente útil para registrar cambios en la superficie terrestre, como puede ser la acumulación de nieve o cambios en la cobertura forestal.

Para entender el funcionamiento de las tecnologías LiDAR en la toma de datos forestales se explica, a continuación, el funcionamiento del Escaneado Láser Terrestre. Este, como ya se dijo, se basa en el uso de instrumentos de escaneado láser fijos. Estos permiten una recolección rápida de mediciones de parámetros de inventariado forestal en forma de nubes de puntos tridimensionales precisas que representan la superficie de los árboles escaneados. Los aparatos de TLS se montan sobre un trípode y escanean un volumen esférico, que se obtiene de completar una rotación completa del aparato en el eje horizontal. Además, cuentan con un espejo rotatorio que permite escanear en el plano vertical y, a veces, incorporan cámaras réflex de lente única que permiten capturar imágenes coloreadas, lo que se traduce en que la nube de puntos incluya un campo RGB además de las coordenadas (x, y, z). (Dassot et al. 2011). En cuanto a su puesta en práctica, y para la toma de datos de parcelas forestales, en el Escaneo Láser Terrestre es habitual seguir uno de los dos siguientes métodos de toma de datos: escaneo simple o escaneo múltiple. En el primero, el escáner se sitúa en un único lugar, tomando datos de solo uno de los lados de los árboles u otros objetos; por su parte, en el escaneo múltiple, se varía la posición del aparato de medida para escanear desde todas las direcciones, para luego procesar todas las tomas conjuntamente y conformar el modelo tridimensional de las estructuras escaneadas.

Figura 1.1: Esquema de la toma de datos con TLS en una parcela forestal. El rectángulo redondeado representa una parcela con árboles (en verde oscuro) y los círculos rojos, los dispositivos LiDAR. A la izquierda se representa el escaneo simple, con un solo punto de recogida de datos. A la derecha se representa el escaneo múltiple, del que se obtiene un modelo completo de los árboles escaneados.

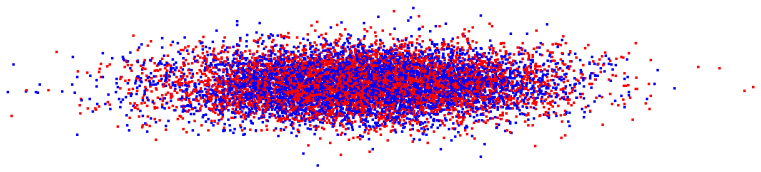


1.3. Nubes de puntos tridimensionales

En la sección anterior se habló de cómo el *output* habitual del proceso de escaneado de superficies tridimensionales con tecnologías LiDAR es una nube de puntos tridimensional. Esta no es más que un conjunto de puntos en el espacio, definidos por sus coordenadas cartesianas, y que pueden poseer o no más información en lo que se denominan 'campos'. En el sentido práctico, son el resultado del escaneo de un objeto o estructura de la que queremos obtener información para su posterior análisis y modelización. El desarrollo de nuevas metodologías para obtener lecturas geoespaciales del entorno rápidamente y de forma eficiente es un campo de investigación que aún está proliferando. En la Figura 1.2 se puede ver un ejemplo de nube de puntos tridimensional, que ha sido visualizada en CloudCompare (CloudCompare v2.12 2021). Este es un ejemplo de *software*, quizá el más conocido por su condición

de ser libre, de tratamiento de nubes de puntos tridimensionales y que permite un fácil manejo de estas. De esta manera, el escaneado de las estructuras forestales por medio de tecnología LiDAR es una forma conveniente de inventariar bosques puesto que existen tanto herramientas con las que es sencillo capturar la estructura del bosque de manera precisa, como programas que permiten una sencilla interacción con los *outputs* de estas últimas, todo ello a disposición del usuario. Las nubes

Figura 1.2: Nube de puntos tridimensional de ejemplo. Está formada por 10000 puntos, donde las coordenadas (x) siguen una distribución normal $(0, 5)$, las coordenadas (y) una normal $(0, 0{,}5)$ y las coordenadas z una normal $(0, 1)$.



de puntos tridimensionales obtenidas de parcelas forestales, por ende, albergan una gran cantidad de información sobre su estructura, si bien, estas deben ser primero tratadas adecuadamente. Atendiendo a la estructura de una parcela forestal antes y después de un incendio, por ejemplo, podrá evaluarse el impacto que haya tenido el fuego. Si se dispone de mediciones a lo largo del tiempo, en general, podrá cuantificarse el cambio producido por con capacidad de transformación del entorno, como tales incendios, la actividad humana, la llegada de especies foráneas o el cambio climático, entre otros. A fin de cuentas, estas son las tareas para las que precisamente se realizan los inventarios forestales, y la tecnología LiDAR ofrece, por todo lo explicado hasta ahora, una oportunidad de enorme facilitación de esta tarea. Por ejemplo, en relación con las estimaciones de biomasa descritas en la Sección 1.1, el enfoque de emplear una ecuación general para estimar la densidad de biomasa requiere de mediciones del volumen de madera; estas se pueden realizar de manera sencilla a través de la nube de puntos que represente a los árboles. Por su parte, las ecuaciones de regresión de biomasa dependían de varios parámetros físicos de los árboles, como el diámetro del tronco a la altura del pecho, su área basal o su altura total. De nuevo, estas son mediciones sencillas de computar desde la comodidad de un despacho en el que se disponga de un ordenador de sobremesa con el que analizar el resultado del escaneado de tales árboles.

1.4. Objetivo del trabajo

Estas tareas también son susceptibles de automatización. Si bien esta tarea ya no es tan trivial, comparada con la simpleza de que un operador mida, por ejemplo, la altura de los puntos asociados a la representación de un árbol (simplemente comprobando cuál es el valor de la coordenada (z) del punto más elevado del árbol y comparándolo con el sistema de referencia que se emplee). Una metodología en la que se ha habido un gran aporte de artículos es en la de tratar de diferenciar árboles de otras estructuras, sobre todo en el entorno urbano, para realizar estimaciones de la biomasa en ciudades o en lugares en los que hay pocos árboles y son fácilmente diferenciables. Otro enfoque es el de tratar de diferenciar especies vegetales en base a la estructura de las nubes de puntos tridimensionales asociadas a cada una, y que tiene gran utilidad, pues existen ya tablas con la densidad de la madera de cada especie y esto también permitiría acelerar el cómputo de la biomasa que aporta el conjunto de árboles. Ambas metodologías, de cualquier manera, requieren de lo que se conoce como técnicas de clasificación, que se encargan en establecer reglas con las que diferenciar objetos, estructuras, especies o, de manera más general, individuos, entre las posibles clases a las que puede pertenecer. Sin embargo, estas técnicas raramente han sido empleadas en una tarea que, a priori, también parece fundamental para realizar estimaciones automáticas de biomasa a partir de parcelas forestales: la de separar los troncos de las ramas, arbustos y la hierba que pueda haber en la parcela. Todas las maneras de estimar biomasa

que se han comentado requieren, específicamente, de mediciones efectuadas sobre los troncos. Además, como medir la cantidad de ramas, su grosor y su longitud es una tarea tediosa, habitualmente se hacen generalizaciones vagas sobre su contribución a la biomasa total, lo cual es poco preciso y propenso a grandes errores de estimación.

En este trabajo se propone una herramienta para clasificar los arbustos, hierba, ramas y troncos de las nubes de puntos tridimensionales obtenidas del escaneado por tecnologías LiDAR de parcelas forestales, con la que se pretende conseguir una mejora del inventariado forestal. Tal herramienta está basada en técnicas de clasificación por aprendizaje profundo: más concretamente, en una red neuronal de perceptrón multicapa. Con este fin, se introducen los cuatro próximos capítulos del trabajo: un capítulo dedicado a definir qué son y en qué se basan las técnicas de clasificación; un capítulo dedicado a los materiales y métodos empleados; un capítulo en el que se detallan los resultados obtenidos y un último capítulo, breve, en el que se discuten e interpretan tales resultados. En el Capítulo **Clasificación de las nubes de puntos tridimensionales**, se introducirá la teoría estadística en las que se basan las técnicas de clasificación, así como el paradigma del aprendizaje y las técnicas de aprendizaje profundo. Se ahondará también en el uso que se ha dado a estas a la hora de clasificar nubes de puntos tridimensionales de parcelas forestales en el pasado. En el capítulo **Material y métodos** se describirán los datos empleados en la elaboración del trabajo, que constan de 4 nubes de puntos procedentes del escaneado de 4 parcelas forestales de regiones diferentes, y el tratamiento que se ha dado a esos datos. Este incluye un muestreo semialeatorio de microparcelas que servirán como submuestra, una voxelización que tiene como fin reducir la complejidad de los datos originales y un proceso de extracción de descriptores geométricos del que se derivan los predictores que harán de inputs de la red. Así mismo, se detallarán las características del modelo empleado y los análisis de sensibilidad realizados para tunear sus hiperparámetros y valorar su capacidad predictiva. En el capítulo **Resultados** se mostrarán los resultados de tales análisis, el tiempo de computación asociado a los procesos de tratamiento y modelado de los datos y se representarán visualmente las parcelas originales clasificadas con el modelo. Por último, en el capítulo **Discusión**, como se mencionó anteriormente, se realizará una interpretación de los resultados, pero también se abrirá el debate sobre las consideraciones que se deben tomar a la hora de emplear el modelo en otros datos y de ajustes que se puedan realizar sobre este para futuras mejoras.

Capítulo 2

Clasificación de las nubes de puntos tridimensionales

En el Capítulo 1 se presentó la biomasa como un parámetro ideal para estudiar diversas características de interés de un bosque y cómo su estimación dependía, en gran medida, de la calidad con la que se realizan los inventarios forestales. Es a través de estos inventarios forestales de donde se obtienen mediciones clave sobre los árboles que conforman los bosques, que serán después empleadas como parámetros con los que estimar la biomasa total del bosque. Este inventariado, si se realiza por medios tradicionales de muestreo a mano es lento, costoso y deja gran cantidad de información detrás; es por esto que las tecnologías LiDAR han entrado en acción para abaratar y acelerar la tarea de medir los árboles y sus características (altura, diámetro, área basal, etc.). Tales tecnologías escanean con luz láser las superficies vegetales y las registran en forma de nubes de puntos tridimensionales, que no son más que entradas de datos con valores de coordenadas cartesianas (x, y, z) asociadas. Estas nubes son directamente medibles, pero realizado a mano, también resulta en un proceso tedioso, por lo que es de gran interés automatizarlo. Y es en este contexto donde se trata de aplicar técnicas de clasificación a este tipo de datos, que son de utilidad para separar al conjunto de registros en diferentes clases de interés.

2.1. El problema de clasificación

En esta sección se introduce formalmente qué es el problema de clasificación matemática y qué herramientas se pueden emplear para resolverlo. Para entenderlo, se hará primero una introducción de qué es el aprendizaje estadístico como disciplina que trata de entender los conjuntos de datos y realizar predicciones con ellos sobre alguna variable de interés -aprender de los datos- y dónde se ubican las técnicas de clasificación dentro de este. Los contenidos de esta sección están basados en los siguientes libros, cuyas referencias se pueden encontrar en la bibliografía del final: 'Mathematics for Machine Learning' (Deisenroth MP et al. 2020), 'Deep Learning with Python' (Chollet F 2017) y 'The Elements of Statistical Learning' (Hastie T et al. 2001).

2.1.1. El aprendizaje estadístico

Sea $X = (X_1, X_2, \ldots, X_p) \in \mathbb{R}^p$ un vector p-dimensional input de números reales y $Y \in \mathbb{R}$ una variable real output, ambos con distribución conjunta $Pr(X, Y)$. El aprendizaje estadístico trata de buscar una función $f(X)$ para predecir el valor de Y dados unos valores del input X. De esta manera, se dice que X es un vector formado por *variables explicativas o predictoras*, y que Y, por su parte, es una *variable objetivo o respuesta*. Esta teoría requiere, además, de una función $L(Y, f(X))$ que penalice los errores cometidos en tal predicción, y que se denota como *función de pérdida*. La más común y

conveniente es el error cuadrado: $L(Y, f(X)) = (Y - f(X))^2$. Esta nos conduce a un criterio para elegir la función f de la forma:

$$\begin{aligned} EPE(f) &= E(Y - f(X))^2 \\ &= \int [y - f(x)]^2 Pr(dx, dy), \end{aligned} \tag{2.1}$$

que se conoce como el *error de predicción esperado (cuadrado)* EPE y donde $E(f(x))$ representa la esperanza matemática. Al condicionar en X e integrando, el EPE se puede escribir como sigue:

$$\begin{aligned} EPE(f) &= E_X E_{Y|X}([Y - f(X)]^2|X) \\ &= \int [y - f(x)]^2 p(x, y) dx dy \\ &= \int_x \int_y [y - f(x)]^2 p(x, y) dx dy \\ &= \int_x \int_y [y - f(x)]^2 p(y|x) p(x) dx dy \\ &= \int_x \left(\int_y [y - f(x)]^2 p(y|x) dy \right) p(x) dx \\ &= \int_x E_{Y|X}([Y - f(X)]^2|X) p(x) dx \\ &= E_X E_{Y|X}([Y - f(X)]^2|X) \end{aligned} \tag{2.2}$$

En este desarrollo se ha tenido en cuenta que la densidad conjunta $Pr(X, Y) = Pr(Y|X)Pr(X)$, y $Pr(Y|X) = Pr(Y, X)/Pr(X)$. Derivando e igualando a 0, se tiene que:

$$f(x) = \arg\min_c E_{Y|X}([Y - c]^2|X)$$
$$\frac{\partial EPE(c)}{\partial f} = \frac{\partial}{\partial f}(E([Y - c]^2|X = x)) = 0$$

desarrollando el cuadrado y haciendo uso de las propiedades de la esperanza, se llega a:

$$\frac{\partial}{\partial f}(E([Y^2 - 2Yc + c^2]|X = x)) = 0$$
$$\frac{\partial}{\partial f}(E(Y^2|X = x) - 2c \cdot E(Y|X = x) + c^2) = 0$$

por último, sacando factor común c y despejando, se obtiene que la función que minimiza puntualmente el EPE es la que sigue:

$$0 - 2E(Y|X = x) + 2c = 0$$
$$c^* = E(Y|X = x)$$

La cual se conoce como la *esperanza condicional* o *función de regresión*. Así, la mejor predicción de Y en cualquier punto $X = x$ cuando se usa el error cuadrado como función de pérdida es la media condicional.

2.1.2. La clasificación matemática

Lo descrito hasta ahora es solo válido para la regresión; es decir, la predicción del valor de Y en base a X cuando la primera representa una función real continua. En el caso de que la función

de la que se quiera obtener predicciones solo tome valores enteros, es cuando podemos hablar del problema de clasificación. En este contexto, la variable respuesta suele denotarse por G, que se define como una variable categórica que toma valores en el conjunto $\mathbb{G} = \{1, 2, \ldots, K\}$, funcionando estos como etiquetas que identifican a las K posibles clases presentes en el conjunto de datos. En realidad, el caso de clasificación funciona bajo el mismo paradigma que el de regresión, como se mostrará en las siguientes líneas, con la salvedad de que es necesario cambiar la función de pérdida que penaliza los errores de predicción. Esta ahora puede representarse por medio de una matriz $\mathbf{L}$ $K \times K$, donde $K = card(\mathcal{G})$. Tal matriz $\mathbf{L}$ tomará valor 0 en la diagonal y algún valor no negativo $L(k, l)$ en cualquier otra posición. Tal valor es la penalización por clasificar una observación perteneciente a la clase $\mathcal{G}_k$ en la clase $\mathcal{G}_l$. Es habitual emplear la función de pérdida *cero-uno*, donde todas las clasificaciones incorrectas se penalizan con un valor de 1. Esta lleva a una expresión del error de predicción esperado de la forma:

$$EPE = E[L(G, \hat{G}(X))],$$

donde $\hat{G}(X)$ denota un estimador de G y donde si condicionamos en X obtenemos, de manera similar al caso de regresión:

$$EPE = E_X \sum_{k=1}^{K} L[\mathcal{G}_k, \hat{G}(X)] Pr(\mathcal{G}_k|X)$$

Notar que ahora aparece un sumatorio en el lugar donde antes aparecía una integral, debido a la condición discreta del problema. Para que un estimador minimice puntualmente tal expresión del EPE, solo es necesario que cumpla:

$$\hat{G}(x) = \arg\min_{g \in \mathcal{G}} \sum_{k=1}^{K} L(\mathcal{G}_k, g) Pr(\mathcal{G}_k|X = x)$$

que, usando la función de pérdida cero-uno, se simplifica a:

$$\hat{G}(x) = \arg\min_{g \in \mathcal{G}} [1 - Pr(g|X = x].$$

En el caso en el que la variable categórica G solo puede tomar dos valores; es decir, que solo existen dos clases entre las que predecir, y que la codificamos con una variable auxiliar binaria Y (conocida como variable *dummy*) que toma valor 0 o 1 en un caso o en el otro, se tiene que:

$$\begin{aligned} \hat{f}(X) &= E(Y|X) \\ &= Pr(G = \mathcal{G}_1|X) \end{aligned} \tag{2.3}$$

Si $\mathcal{G}_1$ se correspondió con $Y = 1$. En general, y así mismo, para el problema con K-clases:

$$\begin{aligned} \hat{f}(X) &= E(Y_k|X) \\ &= Pr(G = \mathcal{G}_k|X) \end{aligned} \tag{2.4}$$

Es a partir de este punto donde es conveniente introducir a los modelos de aprendizaje estadístico como herramienta para obtener las predicciones de las que se habló en los párrafos anteriores. Tales modelos tratan de, a partir de una muestra de datos $\{(x_{1p}, \ldots, x_{ip}, y_i) : i = 1, \ldots, n\}$, obtener predicciones del tipo $Y(\mathbf{x})$ para $\mathbf{X} = \mathbf{x} = (x_1, \ldots, x_p)$. Así, el modelado consiste en aproximar la función de regresión $f(x)$ o la densidad condicional $Pr(G|X)$ en el contexto de clasificación a través de funciones cuyos parámetros se puedan calcular con más o menos exactitud en base a tal muestra. Un ejemplo sencillo es el modelo lineal, que asume que la función de regresión $f(x)$ es aproximadamente lineal, y que esta se puede expresar de la siguiente manera:

$$f(x) \approx x^T \beta$$

Introduciendo este modelo lineal en la expresión del error de predicción esperado y diferenciando, se tiene que el valor teórico para β que minimiza tal error es: $\beta = [E(XX^T)]^{-1}E(XY)$, que se puede aproximar a partir de los datos muestrales y que, por tanto, es una herramienta cuanto menos accesible para obtener predicciones de Y.

Por otra parte, debido a su naturaleza discreta, es típico que los modelos que se emplean para resolver el problema de clasificación para un conjunto de datos devuelvan estimaciones de la probabilidad $\hat{p}$ de que una observación dada pertenezca a la clase de interés en lugar de predecir directamente si se da la clase en sí. Una regla de clasificación habitual en este contexto es la de predecir la clase más probable en la conocida como estrategia 'uno contra todos', siendo la predicción resultante:

$$G(\mathbf{X}) = \arg\max_x \{\hat{p}_k(\mathbf{x}) : k = 1, 2, \ldots, K\}$$

Otra regla, menos práctica cuando existen varias categorías posibles para la variable respuesta, es la de 'uno contra uno', en la que el problema original que se desea resolver se subdivide en $K(K-1)/2$ subproblemas, contemplando cada uno de ellos un solo par de clases de todas las combinaciones posibles. Esto requiere ajustar un modelo para cada subproblema y la predicción final que se dé sobre la clase a la que pertenece una entrada concreta será la que fue seleccionada por un mayor número de modelos.

2.1.3. Aprendizaje supervisado

Un término que aparece con frecuencia dentro del aprendizaje estadístico referido a la clasificación es el de *aprendizaje supervisado*. Volvamos al caso general: el objetivo es predecir valores de G en base a una muestra $\{(x_{1p}, \ldots, x_{ip}, y_i) : i = 1, \ldots, n\}$ que representa a X, empleando para ello un modelo que incorpora una función que aproxima se aproxima a la función objetivo y que es elegida en base a hipótesis razonables sobre la estructura de los datos subyacentes; por ejemplo, es habitual suponer una relación determinística entre las variables explicativas y la variable respuesta, que si existe una relación no determinística esta se puede recoger en un expresión ϵ de error con distribución normal de media 0, que tales errores están idénticamente distribuidos a lo largo del rango de predicción, etc.

Este modelado presenta algunos incovenientes obvios en el proceso de encontrar una función que se aproxime a la función objetivo real. El método más común para *ajustar* la función del modelo a la función objetivo, aunque hay otros, es el *método de los mínimos cuadrados*, que consiste en minimizar la *suma residual de cuadrados*, o *RSS* por sus siglas en inglés. Para una función arbitraria f, tiene la siguiente expresión:

$$RSS(f) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

Cualquier función $\hat{f}$ propuesta para modelar la relación entre X e Y es una solución al problema de minimización del RSS. En realidad el aprendizaje estadístico de lo que trata es, en base a la información presente en una muestra de datos a la que se tenga acceso, encontrar una función $\hat{f}$ que sea una buena solución a este problema de minimización; es decir, una función $\hat{f}$ cuya suma residual de cuadrados, que mide la distancia de esta función a las mediciones de y presenten en la muestra, sea pequeña. Si se dispone de una muestra lo suficientemente grande, es razonable pensar que si $\hat{f}$ se aproxima razonablemente bien a y, también lo hace a la función f de la que viene Y.

Es solo aquí donde entra el juego el *aprendizaje*: para realizar este ajuste de $\hat{f}$ a y y en consecuencia a f, las técnicas de aprendizaje estadístico supervisado tratan de aprender la forma de f mediante ejemplos dados por la muestra. Así, el modelo de aprendizaje supervisado construye una *muestra de entrenamiento* $\mathcal{T} = (x_i, y_i), i = 1, \ldots, N$. De esa muestra de entrenamiento, los valores observados x_i se pasan a un sistema artificial, conocido como *algoritmo de aprendizaje*, que produce outputs $\hat{f}(x_i)$

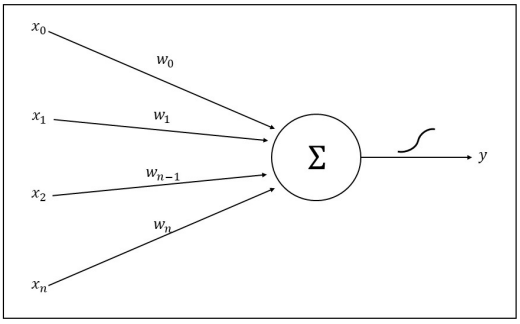
en base a esos inputs. Este algoritmo de aprendizaje tiene la capacidad de modificar su función $\hat{f}$ (que relaciona input/output) en respuesta a las diferencias entre los valores y_i presentes en la muestra de entrenamiento y los valores $\hat{f}(i)$ producidos. Este proceso se llama *aprendizaje mediante ejemplo* y, cuando se finaliza, se pretende que la respuesta medida real (los y_i) y la respuesta generada artificialmente (los $\hat{f}(x_i)$ se parezcan lo suficiente, con la esperanza de que cuando se obtengan nuevos inputs reales el algoritmo de aprendizaje pueda generar automáticamente respuestas artificiales que se asemejarían a la respuesta real.

Este paradigma de aprendizaje ha sido investigado intensivamente a lo largo de los últimos años, nutriéndose de los avances informáticos en la capacidad de computación y de los conjuntos de datos masivos que sirven como datos de entrenamiento, dando como resultado numerosos algoritmos de aprendizaje. Aquí no se entrará en detalle en los tipos de algoritmos que existen ni en qué se diferencian, puesto que hay infinidad de ellos (tanto para resolver el problema de regresión como el de clasificación) y no es la finalidad de esta obra. Sin embargo, en la siguiente sección se explicará con detalle las características de la técnica empleada en este trabajo para clasificar nubes de puntos tridimensionales de parcelas forestales y cómo esta ha sido empleada con anterioridad con este fin, aunque de maneras diferentes y con objetivos complementarios.

2.2. Las redes neuronales

En la sección anterior se introdujo el paradigma del aprendizaje para ajustar modelos de clasificación a una muestra de datos con el fin de realizar predicciones sobre una variable de interés. En este trabajo se emplea una técnica, conocida como *red neuronal artificial* (ANN), que es tan prevalente y aplicada en la práctica que ha dado lugar a toda una rama nueva dentro del aprendizaje estadístico y de la inteligencia artificial: el aprendizaje profundo. Sus orígenes se remontan a los años cuarenta, cuando McCulloch y Pitts (McCulloch WS et al. 1943) proponen su *neurona*: un modelo sencillo que pretende mimetizar el comportamiento de una célula neuronal animal. Tal neurona artificial funciona de la

Figura 2.1: Esquema de la neurona de McCulloch y Pitts.Los inputs $x_0, \ldots, x_n$ son ponderados con pesos w_0, w_n, sumados en la neurona y transformados por la función de activación representada por la curva sigmoidal, para producir el output y.



siguiente manera: los inputs $x_1, \ldots, x_n$ son recibidos por la neurona con ciertos pesos $w_1, \ldots, w_n$, que modulan la 'señal' asociada a cada input. Una vez ponderados por tales pesos, los inputs son sumados en la neurona y el resultado de tal sumatorio, que será un solo valor, es transformado por una *función de activación*. Esta es la que transforma los inputs en un output único asociado a la neurona por la que pasaron. En la Figura 2.1 se presenta un esquema de esta neurona. Aún con todo, la investigación

en este campo queda estancada hasta llegados los años 70-80. En este periodo, Werbos (Werbos 1974) propone un algoritmo, conocido como *propagación hacia atrás*, que permitió resolver ciertos problemas de inestabilidad a la hora de ajustar las redes neuronales artificiales compuestas por varis capas de neuronas de McCulloch y Pitts (se volverá sobre este algoritmo más adelante). Sin embargo, no es si no en la segunda década del siglo actual cuando el uso de las ANN se intensifica y ramifica, debido sobre todo a los avances realizados en la capacidad de computación de los ordenadores, la disponibilidad de conjuntos de datos masivos y la mejora de los algoritmos empleados durante las diferentes etapas de optimización de las ANN. Con estos ingredientes, las redes neuronales artificiales rápidamente sustituyeron a otros modelos de aprendizaje debido a su mejor desempeño y facilidad de despliegue.

En sí, una red neuronal artificial es un modelo de clasificación (o regresión, aunque se pondrá el foco en el primer caso) multi-etapa, que puede ser esquematizado con un diagrama de red en el que los nodos son neuronas, que se disponen en capas, y los arcos son 'sinapsis' que conectan neuronas de una capa con neuronas de otra: en realidad, cada neurona acepta un input numérico y manda un output a la siguiente. La idea central de su funcionamiento es extraer combinaciones lineales de los inputs como predictores derivados (derivado no en el sentido matemático de derivación) para después modelar la función objetivo como una función no lineal de tales predictores. Para el caso general de un problema de clasificación con K clases, y para una red de una sola capa oculta, llamada habitualmente 'perceptrón de una capa' -esto es, que solo tiene una capa de neuronas entre la capa de entrada y la capa de salida-, habrá una capa de entrada el número M de neuronas que se desee, pero que aceptan un vector P dimensional, siendo P el número de variables predictoras. Aquí, tal y como se detalló en la explicación del funcionamiento de una neurona artificial, los inputs son ponderados, sumados y transformados por una función de activación; en realidad, la función de activación σ transforma una combinación lineal de los inputs de la forma $\alpha_{0m} + \alpha_m^T X, m = 1, \ldots, M$, por lo que los predictores derivados Z_m que se obtienen del paso de los inputs por esta primera capa eson como siguen:

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \ldots, M$$

Donde $Z = (Z_1, Z_2, \ldots, Z_M)$ y donde $\alpha_{0m} + \alpha_m^T$ son los coeficientes que determinan el peso con el que entran los inputs a la neurona. Este mismo procedimiento se repite en la capa oculta, que recibe como inputs tales Z_m, solo que aquí las neuronas reciben un escalar y no un vector P dimensional; es de destacar que lo habitual es que se vuelva a emplear la misma función de activación σ en esta capa, llamada oculta por el hecho de que los valores Z_m no son directamente observables, aunque que se repita la activación no tiene por qué ocurrir. Finalmente, tras esto, las neuronas de la capa de salida, que serán K, reciben los outputs emitidos por las neuronas de la capa oculta, vuelven a realizar una combinación lineal $T_k, k = 1, \ldots, K$ de estos mismos realizan una última transformación $g_k(T)$ de los datos y generan el output final $f_k(X)$. Este tiene la forma:

$$T_k = \beta_{0k} + \beta_k^T Z \tag{2.5}$$

$$f_k(X) = g_k(T) \tag{2.6}$$

Donde $T = (T_1, T_2, \ldots, T_K), \quad k = 1, \ldots, K$. La función clásicamente empleada como activación ha sido la función sigmoide $\sigma(v) = 1/(1 + e^{-v})$, que fue después sustituida por la tangente hiperbólica: $\tanh v = (e^v - e^{-v})/(e^v + e^{-v})$, y que, desde la segunda década de este siglo, ha sido sustituida casi en la totalidad de los casos por la función rectificadora ReLU (*Rectified Linear Unit*), definida como la parte positiva de su argumento: $f(v) = v^+ = \text{máx}(0, v)$. Esta última presenta una serie de ventajas sobre sus antecesoras:

- Es computacionalmente eficiente.
- Es invariante ante cambios de escala: $\text{máx}(0, av) = a\,\text{máx}(0, v) \forall a \geq 0$.
- Conlleva menos problemas de desvanecimiento del gradiente; más sobre esto en el siguiente párrafo.

- Lleva a una menor activación de neuronas, puesto que solo aquellas que reciben un input positivo producirán un output no cero.

2.2.1. Propagación hacia atrás

Tales ventajas son notables, en particular, cuando se aumenta el número de capas ocultas, lo que conlleva una computación más intensiva y que ocurra el denominado desvanecimiento del gradiente. Este debe su nombre a la técnica empleada para ajustar la red neuronal a los datos muestrales con el fin de encontrar la combinación de pesos W que minimice el error de predicción producido por el modelo y ha sido el principal problema por el que el uso de las redes neuronales artificiales como técnica predictiva no se haya expandido si no hasta recientemente.

El conjunto de pesos, denotado por Θ, consiste en:

$$\{\alpha_{0m}, \alpha_m, \quad m = 1, \dots, M\} \quad M(p+1) \tag{2.7}$$
$$\{\beta_{0k}, \beta_k, \quad k = 1, \dots, K\} \quad K(M+1) \tag{2.8}$$

Por lo que hay $M(p+1) + K(M+1)$ pesos que optimizar; esto para una sola capa oculta. Si hubiese más, el segundo sumando sería $KH(M+1)$, siendo H el número de capas ocultas.

En sí, la función que se optimiza es la función de pérdida, que en el caso de clasificación puede ser la suma de errores al cuadrado:

$$R(\Theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2,$$

que es la misma que se usa en el contexto de regresión, si bien lo habitual suele ser emplear lo que se denomina función de entropía categórica cruzada:

$$R(\Theta) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log f_k(x_i),$$

con el criterio de clasificación $G(x) = \arg\max_k f_k(x)$ Esta es una función con una enorme cantidad de parámetros y que, además, si se tratase de encontrar el óptimo global, este seguramente se corresponda con una solución sobreajustada. Es por ello que se toma una solución alternativa de optimización, generalista: el descenso de gradiente (Haskell B 1944) (propagación hacia atrás en el contexto de redes neuronales artificiales). Se denomina gradiente a la generalización del concepto de derivada a funciones de inputs de dimensión mayor a 1 y, para una configuración dada de pesos en la red, es fácil calcular, a través de la regla del producto de la diferenciación, la derivada (gradiente) de la función de pérdida con respecto de los pesos actuales. En terminología matemática, el output y está conformado por una composición de funciones sobre los inputs x tal que: $y = (f_K \circ f_{K-1} \circ \dots f_1)(x) = f_K(f_{K-1}(\dots(f_1(x))\dots))$ donde cada función $f_i, \quad i = 1, \dots, K$ posee sus propios parámetros. No se entrará en detalle sobre las ecuaciones de la propagación hacia atrás, pero estas gozan de una propiedad que las hace realmente útiles para el propósito de encontrar la combinación de pesos que minimiza el error de predicción: sus coeficientes indican la dirección y la magnitud del cambio en el valor de la función de pérdida cuando se modifican los pesos. Así, se puede reducir el valor de la función de pérdida (y por tanto, mejorar el ajuste) moviendo el valor de los pesos en la dirección opuesta del gradiente. Este procedimiento se sigue algorítmicamente de la siguiente manera:

1. Toma una muestra de entrenamiento de predictores x y sus correspondientes respuestas y.

2. Calcula la predicción dada por la red para la combinación actual de pesos.

3. Calcula el valor de la función de pérdida para tal predicción.
4. Computa el gradiente de la función de pérdida con respecto de los pesos actuales (el pase hacia atrás, que da nombre al proceso).
5. Mueve el valor de los pesos en sentido opuesto al gradiente.

Cada pase del algoritmo se denomina *época de entrenamiento*, y el número de épocas que se realizan es un hiperparámetro del modelo a optimizar: demasiados pases conducirán a un sobreajuste a los datos de entrenamiento, pero pocos pasos provocarán un infraajuste; en ambos casos, la capacidad predictiva en datos nuevos de la red se verá mermada. Además, por construcción, durante el cálculo del gradiente, es posible que se efectuen numerosas multiplicaciones de números positivos pero próximos a 0; en especial, si se emplea como función de activación la función sigmoide o la tangente hiperbólica. Esto hace que, en una red de L capas, el gradiente decrezca exponencialmente a 0 en un factor de L e imposibilite el ajuste de los pesos, lo que se denomina *desvanecimiento del gradiente*. Es por este motivo que el uso de la función ReLU experimentó un gran auge como sustitución de las anteriores funciones de activación, pues esta no tiene por qué tomar valores próximos a 0 como sí lo hacían sus predecesoras.

Otro detalle a considerar es el tamaño del paso que se da cuando se mueven los pesos en sentido opuesto al gradiente: como se trata de buscar mínimos de la función de pérdida a optimizar, pasos muy pequeños pueden resultar en que el algoritmo se quede atrapado en un mínimo local no muy bueno, y pasos muy grandes en que evite mínimos locales adecuados. El tamaño del paso dado es lo que se conoce en este contexto como *tasa de aprendizaje*, o α, y el algoritmo que realiza el pase hacia atrás en la actualización de los pesos se denomina *optimizador*. En la actualidad, el optimizador usado con más frecuencia es *Adam*, acrónimo de 'Adaptative Moment Estimation' (Kingma D et al. 2014) y que tiene la propiedad de que computa tasas de aprendizaje adaptativas para cada peso, reteniendo información sobre cómo fue el cambio en el gradiente en los pasos anteriores. Para finalizar, el último gran avance en la estabilización del proceso de propagación hacia atrás que se destacará en esta sección es la *normalización por lotes*, *batch normalization* en inglés. Es frecuente que el optimizador haga los pases hacia atrás por lotes; es decir, por submuestras del conjunto de datos de entrenamiento. Esta metodología permite que el pase hacia atrás sea paralelizable, aunque conlleva a una curva de convergencia del algoritmo más inestable. Esto, sin embargo, presenta un incoveniente, y es que el proceso de entrenamiento de la red se complica debido a que, en redes profundas, la transformación de los inputs debido a los cambios producidos en los parámetros de la red en capas anteriores (anteriores durante el pase hacia atrás; es decir, en las capas más próximas a la de salida) se ve amplificada. La normalización de los lotes (Ioffe et al. 2015) reduce la dependencia de los gradientes de la escala de los parámetros, aliviando este efecto y permitiendo, además, tasas de aprendizaje más altas, lo que conlleva un ajuste todavía más rápido del modelo. Estos avances en el algoritmo de optimización de las redes neuronales aritificiales en la última década, sumados a los avances en desarrollo de GPU más potentes y la capacidad de implementar tales modelos en ellas, que aceleran el ajuste del modelo en órdenes de magnitud (Oh K et al. 2004) es lo que hace que un modelo como una red neuronal artificial profunda, a priori tan complejo por su elevadísimo número de parámetros, sea prácticamente la primera opción en cuanto a resolver problemas de clasificación.

2.3. Redes neuronales en la clasificación de nubes de puntos tridimensionales

Ya se vio en secciones anteriores como las técnicas de clasificación tienen su cabida en la clasificación de nubes de puntos tridimensionales de parcelas forestales, especialmente como herramienta para mejorar el inventariado forestal, y como las redes neuronales artificiales se presentan como un candidato ideal para resolver problemas de clasificación. En sí, las técnicas de aprendizaje profundo han venido

siendo aplicadas de manera progresiva en la última década para clasificar nubes de puntos tridimensionales, aunque sobre todo en áreas diferentes a las parcelas forestales. El artículo 'Deep Learning for Remote Sensing' (Zhu et al. 2017) se lista una gran cantidad de avances en redes neuronales para la clasificación de escenas y objetos en nubes de puntos tridimensionales. De él puede extraerse que los supuestos más habituales en los que se han desarrollado modelos basados en aprendizaje profundo son en la clasificación de mobiliario urbano en ambientes de ciudad, detección de vehículos en imágenes aéreas de la superficie terrestre, detección de embarcaciones en superficies marítimas y detección de aeronaves en el aire. Entre las diferentes variantes de redes neuronales aplicadas en los artículos recogidos en este trabajo bibliográfico, destaca sobre todo lo demás el uso intensivo de *redes neuronales convolucionales*, normalmente denominadas CNN por sus siglas en inglés o *convnets*. Estas fueron desarrolladas por primera vez en 1980 (Fukushima 1980) como una emulación del funcionamiento del cortex visual animal. En síntesis, son una red neuronal en la que cada neurona está especializada en aprender patrones locales sobre una submuestra de datos, en vez de aprender patrones globales sobre el espacio completo de variables explicativas. Las capas de la CNN dedicadas a este aprendizaje de patrones locales se denominan capas de convolución. Tienen propiedades teóricas que las hacen interesantes como técnica de clasificación, especialmente de imágenes: los patrones que aprenden son invariantes ante translaciones de los datos y, además, pueden aprender jerarquías espaciales de patrones, donde una capa aprende patrones sencillos como bordes y una segunda capa aprende patrones compuestos por los anteriores. Algunos ejemplos de trabajos en la aplicación de redes neuronales en datos LiDAR son Sothe et al. 2020, en el que se aplica una *convnet* para clasificar especies de árboles tropicales con una precisión global del 84 %, pero que solo usa la vista cenital de las parcelas forestales clasificadas para poder usar una CNN, y Li et al. 2022, proponen un modelo que denominan *red neuronal convolucional multivista*, o *MVCNN*, que usa nubes completas pero que implementa un algoritmo que, según detallan, tiene una eficiencia de convergencia baja, que emplean GPU de última tecnología para acelerarlo y que no puntualizan la precisión del modelo.

Figura 2.2: Evolución en el tiempo del número de publicaciones relacionadas con el aprendizaje profundo aplicado a datos de LiDAR. Tomado de Zhu et al. 2017.

Además, el artículo incorpora un gráfico de barras en el que se observa el incremento del número de publicaciones y avances realizados en el campo del aprendizaje profundo aplicado a la clasificación de datos obtenidos por tecnologías LiDAR en los años anteriores al estudio. Este gráfico se adjunta en la Figura 2.2, donde se aprecia la tendencia creciente en la investigación desarrollada en este campo. Aún así, antes de 2018 es difícil encontrar artículos que se especialicen en aplicar tales técnicas a las parcelas forestales y mucho menos a la clasificación de las nubes de puntos en tronco, rama, arbusto y hierba: búsquedas en este aspecto en las bases de datos científicas habituales como Scopus solo muestran artículos dedicados a la clasificación de especies forestales o a la identificación de árboles en ambientes urbanos, y la enorme mayoría de ellos son de 2019 en adelante. Tanto es así, que posiblemente este documento sea el primer trabajo en el que se aplican técnicas de aprendizaje profundo con el fin de realizar dicha clasificación en datos tomados por tecnologías LiDAR en parcelas forestales. Probablemente este hecho se deba a que las redes neuronales convolucionales son especialmente útiles cuando los datos que se trata de clasificar tienen forma de imagen rectangular formada por píxeles (que luego puede tener profundidad o no e incluir más dimensiones, cosa que sucede habitualmente) o similar, fácilmente expresable por medio de matrices no dispersas en las que buscar convoluciones de provecho. Este no es el caso de las nubes de puntos tridimensionales de parcelas forestales: aquí la estructura tridimensional es demasiado relevante, y prescindir de ella transformando los datos a alguna forma similar a una imagen dejaría atrás gran cantidad de información. Teniendo en cuenta esto, tratar de ajustar una CNN a la nube bruta también presenta un gran incoveniente, y es que los datos asociados a tal nube forman, en cualquier caso, una matriz dispersa, por lo que habría que transformarla de alguna otra manera para tratar de usar capas convolucionales.

En este trabajo se presenta una solución alternativa, basada en aprendizaje profundo pero alternativa a las redes neuronales convolucionales, para clasificar las nubes de puntos tridimensionales de parcelas forestales en rama, tronco, arbusto y hierba. Este trabajo es una continuación de las investigaciones previas del Departamento de Explotación y Prospección de Minas, área de Ingeniería Cartográfica, Geodésica y Fotogrametría de la Universidad de Oviedo. Tal investigación se ha centrado en la aplicación de técnicas de clasificación basadas en aprendizaje supervisado en parcelas forestales y urbanas, plasmadas en trabajos como 'Automatic Detection and Classification of Pole-like Objects for Urban Cartography using Mobile Laser Scanning Data' (Ordóñez C et al .2017), 'Multiscale Supervised Classification of Point Clouds with Urban and Forest Applications' (Cabo C et al. 2019) y 'A Distance Correlation Approach for Optimum Multiscale Selection in 3D Point Cloud Classification' (Oviedo-de la Fuente M et al. 2021). En el Apéndice A puede encontrarse una guía de uso de CloudCompare para el etiquetado manual realizado y en los Apéndices B y C el código Python y R, respectivamente, desarrollado en este trabajo.

Capítulo 3

Material y métodos

Una vez introducido qué son las nubes de puntos tridimensionales de parcelas forestales y qué técnicas se emplean habitualmente para su clasificación, en este apartado se describe un procedimiento para clasificar parcelas forestales en arbusto, hierba, rama, tronco. Este consta por un muestreo pseudo-aleatorio sencillo de microparcelas de tamaño reducido a partir de una parcela completa, para su posterior etiquetado manual. Tales microparcelas etiquetadas serán después reducidas en complejidad a través de un proceso de voxelización, y luego empleadas para entrenar un modelo basado en una red neuronal de perceptrón multicapa, que usa como predictores ciertas características geométricas asociadas a dichos vóxeles que representan los puntos iniciales.

Así, se construye un modelo de clasificación basado en datos tabulares, de relativamente bajo coste computacional y que ignora las dificultades asociadas a tratar de ajustar una red neuronal convolucional a datos en forma de matrices dispersas. En concreto, se mostrarán dos modelos alternativos: uno que divide los datos en las 4 categorías descritas antes ('arbusto', 'hierba', 'rama' y 'tronco') y otro que agrupa los datos etiquetados como 'arbusto' y como 'rama' en una sola categoría, que se denominará 'mata'. Esto es debido al hecho observado durante la clasificación manual de las microparcelas de robledales de que las ramas y los arbustos presentan estructuras tridimensionales extremadamente similares. De esta manera, se tendrán 3 categorías: una que se enfoca en los puntos que conforman el suelo, otra que recoge los troncos, que es la estructura de más interés y otra que engloba todo lo demás. Se pretende así que el modelo de 3 categorías se 'esfuerce' en separar los troncos del resto.

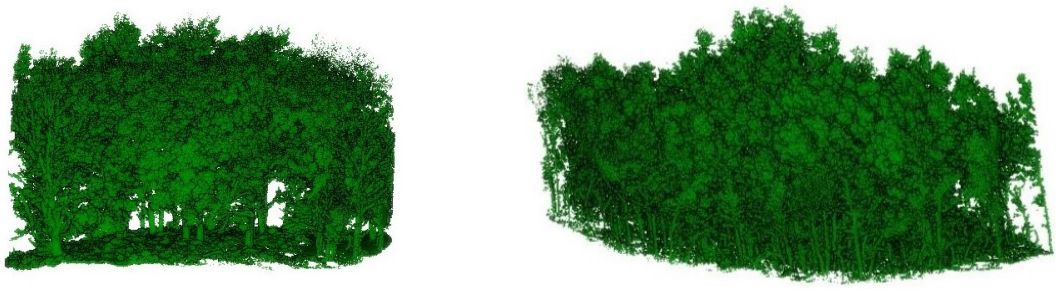
La parte experimental descrita a continuación de este punto se ha realizado en un ordenador portátil Acer®Swift 3 con las siguientes especificaciones: procesador AMD®Ryzen 7 4700U basado en x64 con Radeon Graphics de 16GB de memoria RAM y 8 x 2 GHz unidades de procesamiento, sistema operativo Windows®11 Home versión 21H2 de 64 bits. La herramientas de software empleadas para la visualización de las parcelas forestales y su etiquetado manual ha sido CloudCompare, la voxelización de las nubes de puntos tridimensionales y el cómputo de los descriptores geométricos asociados se ha realizado empleando el lenguaje de programación Python (Python 2022) y el ajuste del modelo a los datos y la creación de los gráficos originales presentes en el documento se han realizado en el software de computo estadístico R (R Core Team 2022).

3.1. Datos empleados

Los datos empleados en la elaboración de este trabajo provienen de 4 parcelas forestales extraídas de bosques de diversas partes del mundo y han sido cedidos por el Departamento de Explotación y Prospección de Minasde la Universidad de Oviedo. Se tienen dos parcelas de robledales: una primera extraída de *Loch Lomond*, Escocia, de *Quercus robur* y de una extensión aproximada de 2100 m^2 y

un segundo robledal de rebollos (*Quercus pyrenaica*) de superficie similar de Callecedo, en el municipo Polaciones, Cantabria; por otra parte, se dispone de dos parcelas de coníferas: una extraída de un pinar de pino de hoja larga (*Pinus palustris*) de *Pebble Hill*, *Georgia*, Estados Unidos y cuya dimensión es de, aproximadamente 2500 m^2 y una segunda parcela de 900 m^2 de la región de Wienerwald. Esta se encuentra ubicada en las tierras altas del pie noreste de los Alpes calizos septentrionales, entre Austria y Eslovaquia. Si bien el tipo de bosque más habitual de la región es el hayedo, en la parcela de la que se dispone la mayoría de individuos son de pícea común (*Picea abies*), aunque también hay ejemplares de haya europea (*Fagus sylvatica*), roble albar (*Quercus petraea*) y de carpe (*Carpinus betulus*).

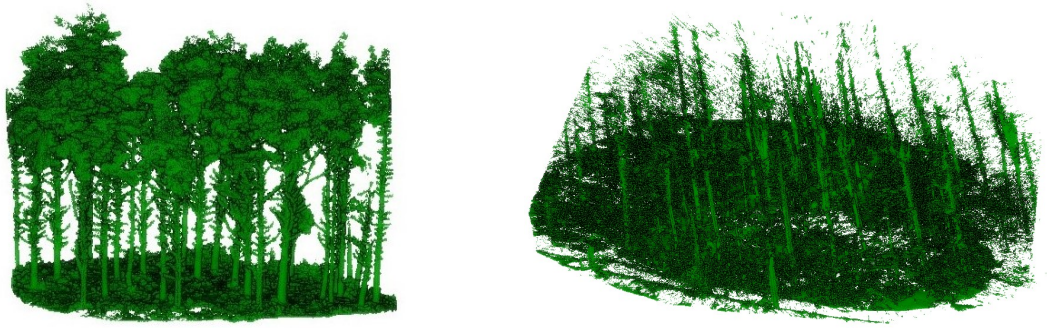
Así, se dispone de 2 robledales (Robledal 1 y Robledal 2 a partir de aquí) y 2 parcelas de coníferas que, por simplicidad y por falta de un término más descriptivo para los conjuntos de píceas y por u similitud con los pinos, se denotarán por Pinar 1 y Pinar 2. Todas ellas están conformadas por varias decenas de árboles y fueron registradas con tecnología TLS, excepto el Pinar 2, que fue registrado a través de un procedimiento denominado SfM (*Structure from Motion*), basado en imágenes fotogramétricas obtenidas con una cámara digital Nikon D800 digital de 28 *mm* de distancia focal. Esta técnica preserva el color de las superficies escaneadas y produce nubes de puntos más densas. En la Figura 3.1 se adjuntan visualizaciones de ambos robledales y en la Figura 3.2, por su parte, se adjuntan visualizaciones de ambos pinares.



(a) Robledal 1

(b) Robledal 2

Figura 3.1: a) Vista de la nube asociada al robledal 1, compuesta por 23,877,646 puntos. b) Vista de la nube asociada al robledal 2, compuesta por 19,480,027 puntos.



(a) Pinar 1

(b) Pinar 2

Figura 3.2: a) Vista de la nube asociada al Pinar 1, compuesta por 12,174,566 puntos. b) Vista de la nube asociada al Pinar 2, compuesta por 47,011,469 puntos.

De cada una de estas parcelas se han extraído 8 microparcelas de $2x2m$ de planta siguiendo el un proceso de muestreo pseudo-aleatorio. Para ello, se ha dividido el conjunto de datos inicial en microparcelas de 2x2 m, a las que se han asignado índices según iban siendo separadas de sus respectivas parcelas. Tras este paso, se ha establecido un criterio de validez de las microparcelas generadas para descartar aquellas que contenían demasiada poca información. Tal criterio ha sido filtrar las microparcelas que ocupasen más de 300 Kb de memoria. Después, se han generado 8 números aleatorios y se han seleccionado aquellas microparcelas cuyos índices se correspondiesen con tales números.

Tras esto, se ha visualizado cada una de las microparcelas seleccionadas en CloudCompare para cerciorar que todas eran representativas. En los casos en los que la microparcela no contenía troncos, estas se sustituyeron por la primera microparcela de índice posterior que contuviese puntos asociados a troncos. Además, este sencillo procedimiento pseudo-aleatorio de selección de microparcelas también da lugar, como es de esperar, a una variedad de nubes de puntos que representas estructuras arbóreas fragmentadas. Este hecho, que las microparcelas no representen a árboles completos, interferirá en la posterior construcción del modelo de clasificación. Por ello, las microparcelas que contuviesen estructuras muy dispares (fragmentos de troncos o ramas sueltas) también fueron sustituidos por la primera microparcela posterior que no tuviese estructuras antinaturales. Aún así, esta metodología no pretende realizar una búsqueda exhaustiva de los mejores individuos de la parcela y la extracción de su estructura completa; si no sentar una base sencilla, fácil de implementar y que aporte soluciones suficientemente buenas en un periodo de tiempo razonable.

(a) Microparcela con poca información

(b) Microparcela no representativa

Figura 3.3: a) Una microparcela con poca información. b) Una microparcela con suficiente información, pero no representativa.

Una vez que se dispuso de microparcelas que representan con un grado de fidelidad satisfactorio las estructuras tridimensionales que se pueden encontrar en un bosque, se procedió al etiquetado a mano de los puntos contenidos en ellas. Para ello, se tomó la decisión de disponer de 4 posibles etiquetas: hierba, arbusto, rama y tronco. Esta clasificación parte de criterios tanto botánicos como de gestión forestal: una clasificación botánica clásica de las plantas vasculares es aquella en las que estas se dividen en hierbas, arbustos y árboles. Por otra parte, es de interés separar las ramas del tronco, ya que este último es utilizado para medir diversas características del propio árbol o como recurso del que extraer madera. Una vez clasificados los puntos de todas las microparcelas, estas se han guardado en 4 archivos de texto, uno por cada categoría. A estas alturas, ya están listas para ser exportadas de CloudCompare y comenzar su tratamiento estadístico.

(a) Microparcela separada

(b) Microparcela etiquetada

Figura 3.4: a) Una microparcela representativa. b) Esa misma microparcela ya etiquetada manualmente.

3.2. Voxelización

Una vez las microparcelas han sido adecuadamente clasificadas a mano, están se han voxelizado, para generar una representación de menor complejidad de sí mismas y de más fácil manipulación; esta es una técnica que se usa con frecuencia para trabajar con nubes de puntos tridimensionales de parcelas forestales, y cuyo primeros usos se remontan a los primeros años del siglo presente (Riaño et al. 2003). Tal voxelización de los datos puede entenderse como una discretización del espacio en el que estos se encuentran en una malla de cubos de dimensión regular y a la agregación de los puntos que se encuentran en dichos cubos en un solo dato. Esta suele realizarse a partir de un paralelepípedo auxiliar que acota el dominio real de la nube de puntos, y que se conoce como *Axis-Aligned Bounding Box*, o *ABB*. Para una nube de puntos:

$$P = \{p_i(x_i, y_i, z_i), \quad i = 1, \ldots, n\}$$

donde n es el número de puntos, p_i es el punto i-ésimo y (x_i, y_i, z_i) las coordenadas de tal punto, el paralelepípedo auxiliar se define como sigue:

$$ABB = \{(x, y, z) | x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}, z_{min} \leq z \leq z_{max}\}$$

donde $(x_{min}, y_{min}, z_{min})$ y $(x_{max}, y_{max}, z_{max})$ son las coordinadas mínimas y máximas de la nube de puntos, definidos estos como $x_{min}(y_{min}, z_{min}) = \text{mín}\{(x_i, y_i, z_i), \quad i = 1, \ldots, n\}$ y $x_{max}(y_{max}, z_{max}) = \text{máx}\{(x_i, y_i, z_i), \quad i = 1, \ldots, n\}$. La voxelización se define, de esta manera, como una subdivisión de este paralelepípedo en filas, columnas y capas, resultando en el conjunto de vóxeles:

$$V = \{v_j(r_j, c_j, l_j), \quad j = 1, \ldots, m\}$$

donde ahora m es el número de vóxeles, v_j es el vóxel j-ésimo y (r_j, c_j, l_j) son las coordenadas o localización de tal vóxel.

Este proceso lleva irremediablemente a la pérdida de información de la estructura de datos original, y a tomar decisiones basadas en criterios subjetivos sobre cómo etiquetar tales vóxeles: consecuentemente con haber etiquetado los puntos originales, puesto que esto proporciona una variable categórica

respuesta de interés, es necesario asignar una etiqueta a los vóxeles que contienen a dichos puntos, así como unas coordenadas. El criterio habitual es construir los vóxeles centrados alrededor de puntos de la nube original y asignar al vóxel las coordenadas y la etiqueta de tal punto central. Esta es la metodología que se empleará en este trabajo.

Queda, aún así, otra decisión que tomar: la de la escala o resolución de vóxel, que se puede denotar como:

$$\Delta x, \quad \Delta y, \quad \Delta z$$

que son las resoluciones en cada eje x, y, z respectivamente y que no tienen por qué ser iguales. La elección de esta escala del vóxel es problemática; escalas pequeñas preservan más información del entorno que seleccionan, a costa de incluir variaciones locales espúreas que pueden llevar, en este caso, a establecer diferencias entre vóxeles que si se tomasen con diferente escala serían clasificados igual. Por otra parte, escalas demasiado grandes se traducen en una pérdida de información importante, que difumina la información subyacente y que proporcionará clasificaciones sesgadas que no tienen en cuenta la estructura real de los datos. En la metodología seguida en este trabajo se ha optado, teniendo estas consideraciones en cuenta, en tomar una escala de vóxel pequeña, que retenga mucha información de la nube original, pero lo suficientemente grande para reducir el tiempo de cómputo de las operaciones posteriores lo suficiente para poder desarrollar el modelo en un periodo de tiempo tratable y cómodo. Tras ensayo y error, se ha establecido que la menor escala que satisface este requisito es $\Delta x = \Delta y = \Delta z = 5\ cm$. Por último, durante la voxelización se tomó registro de qué puntos se recogieron bajo qué voxel, para poder transformar la nube de puntos tridimensional voxelizada ya clasificada a la escala original.

3.3. Extracción de descriptores geométricos

Tras la voxelización de los datos se dispone de una representación de los datos iniciales de menor coste computacional y que recoge gran parte de la información sobre la estructura presente en las microparcelas originales. Estos vóxeles, aún así, solo contienen información de las coordenadas (x, y, z) del punto sobre el que están centrados, y difícilmente se pueden considerar como predictores por sí mismos de la clase que les corresponde. Por ello, es en este punto donde se busca un procedimiento que permita extraer información sobre cómo se comporta un vóxel con respecto a los que están en su entorno. En lugar de tratar de encontrar convoluciones, como ya se explicó al final del capítulo anterior, en este trabajo se opta por seguir un enfoque que se ha desarrollado en los trabajos de clasificación de nubes de puntos tridimensionales anteriores del Departamento de Explotación y Prospección de Minas de la Universidad de Oviedo: extraer descriptores en base a la geometría de vecindarios de vóxeles.

El objetivo de usar estos predictores es no depender de las coordenadas de cada punto / vóxel a la hora de clasificar. A priori, si se pretende realizar una clasificación basada en características geométricas, las coordenadas parecen un gran indicador de a qué categoría puede pertenecer un punto cualquiera. Por ejemplo, no es de esperar que haya hierba en cotas altas de la nube de puntos, ni ramas a ras de suelo. Sin embargo, el sistema de referencia con el que se asignan coordenadas a los puntos de la nube es, casi siempre, variable entre diferentes nubes. No solo eso, sino que también es frecuente que, dentro de una misma nube, el suelo esté a diferentes alturas según cambie la posición (x, y), cuando el terreno presenta irregularidades o está en pendiente. Esto lleva a que haya puntos de hierba cuya coordenada (z) sea 0 y también puntos de hierba con una coordenada (z) equivalente a 3 m de altura en una misma nube. En cuanto a las coordenadas (x, y), no es de esperar que todas las parcelas estén escaneadas con la misma orientación (cuando esto tiene siquiera sentido, porque lo habitual es que la orientación sea variable durante el escaneado de una misma parcela) y, además, reduce la capacidad de extrapolación del modelo, ya que lo más seguro es que haya pares (x, y) que no estén presentes en la muestra de entrenamiento pero que sí lo estén en la nube completa. Con esta metodología se consigue evitar el

problema de tratar los datos como una matriz dispersa y, además, el cómputo de tales descriptores es computacionalmente eficiente, como se explicará a continuación. Después, los datos tabulares que se obtienen son empleados como inputs de una red neuronal relativamente sencilla, que clasificará los vóxeles exclusivamente en función de sus características geométricas.

Para definir los descriptores geométricos que se emplearán como herramienta predictora es necesario, primero, introducir algunos conceptos. Se define el vecindario $\mathcal{V}_P^r$ de un punto P a una escala r como el conjunto de puntos P_k que verifican:

$$P_k \in \mathcal{V}_P^r \iff ||P - P_k|| \leq r$$

Definido de esta manera, el vecindario es un volumen esférico de radio r. Las características geométricas se extraerán de los puntos contenidos en vecindarios de la forma anterior, a través del enfoque empleado en los trabajos anteriores. Este pasa por realizar un análisis de componentes principales de los puntos, que resulta en tres vectores ortogonales centrados en el centroide del vecindario y que modelan las principales direcciones y magnitudes de variación de la distribución de puntos alrededor del centro de gravedad. Tales magnitudes se pueden combinar para definir descriptores geométricos de los puntos en el entorno considerado.

3.3.1. Análisis de componentes principales

Sea $\mathbf{x}_i = (x_i, y_i, z_i)^T$, y $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ el centro de gravedad de los n puntos registrados en $\mathcal{V}_P^r$. Dada la matriz $\mathbf{M} = (\mathbf{x}_1 - \overline{\mathbf{x}} \dots \mathbf{x}_n - \overline{\mathbf{x}})^T$, el tensor $\mathbf{C}$ de tal estructura tridimensional se define por $\mathbf{C} = \frac{1}{n}\mathbf{M}^T\mathbf{M}$. Por la manera en que está construido, este tensor es una matriz definida positiva simétrica, por lo que se puede descomponer en autovalores, y tal descomposición se puede expresar como $\mathbf{C} = \mathbf{R}\mathbf{\Lambda}\mathbf{R}^T$, donde $\mathbf{R}$ es una matriz de rotación formada por los autovectores de $\mathbf{C}$ y $\mathbf{\Lambda}$ es la matriz de autovalores, definida y positiva. Estos autovalores son positivos y ordenados, de manera que $\lambda_1 \geq \lambda_2 \geq \lambda_3 > 0, \forall j \in [1, 3]$. Además, a partir de un λ_j se define la desviación estándar a lo largo del autovector $\overrightarrow{v}_j$ como $\sigma_j = \sqrt{\lambda_j}$. De esta manera, el análisis de componentes principales permite describir la forma de $\mathcal{V}_P^r$ como un elipsoide orientado, con las σ_j como las longitudes de sus semi-ejes que revela la linealidad, planaridad y volumetría del vecindario de puntos atendiendo a cómo se distribuyen tales puntos en las 3 dimensiones.

3.3.2. Descriptores geométricos empleados

Los descriptores geométricos que se emplean en este trabajo son una combinación de los que se han empleado en los trabajos anteriores del Departamento con otros que también se han empleado con frecuencia en otras investigaciones. El Cuadro 3.1 recoje los descriptores empleados que pueden obtenerse directamente de los autovalores λ: De los descriptores del cuadro, los más comunes y empleados habitualmente para describir un volumen son los tres primeros. Se cumple que, si $\lambda_1 \gg \lambda_2, \lambda_3$, y $\lambda_2, \lambda_3 \simeq 0$, la linealidad será mayor que la planaridad y la volumetría. Por su parte, la planaridad, P_λ, prevalecerá sobre la linealidad cuando $\lambda_1, \lambda_2 \gg \lambda_3 \simeq 0$ el comportamiento plano será el que prevalezca. El empleo de la esfericidad S_λ se justifica de la siguiente manera: por construcción de los tres descriptores, todos toman valores en el intervalo $[0, 1]$ y cuando se cumple que $\lambda_1 \simeq \lambda_2 \simeq \lambda_3$, se cumple que $S_\lambda = 1$, $L_\lambda \simeq 0$ y $P_\lambda \simeq 0$, lo que permite identificar vecindarios en los que los puntos que están en su interior forman una estructura que no es ni lineal ni plana.

Si bien la linealidad, planaridad y volumetría del vecindario son fácilmente extraíbles del elipsoide conformado por las componentes principales del tensor $\mathbf{C}$, estas no son las únicas características geométricas que se pueden extraer de la información contenida en esta matriz. A lo largo de las últimas décadas se han definido varios descriptores geométricos obtenidos por combinaciones u operaciones sobre los λ_j y sus correspondientes desviaciones típicas.

Descriptor	Abreviación	Fórmula
Linealidad	L_λ	$(\lambda_1 - \lambda_2)/\lambda_1$
Planaridad	P_λ	$(\lambda_2 - \lambda_3)/\lambda_1$
Esfericidad	S_λ	λ_3/λ_1
Anisotropía	A_λ	$(\lambda_1 - \lambda_3)/\lambda_1$
Omnivarianza	O_λ	$\sqrt[3]{\lambda_1\lambda_2\lambda_3}$
Suma de autovalores	Σ_λ	$\lambda_1 + \lambda_2 + \lambda_3$
Variación superficial	C_λ	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$
Autoentropía	E_λ	$-\sum_{i=1}^{3} \lambda_i ln(\lambda_i)$
'Componente Principal 1'	$PCA1_\lambda$	$\lambda_1/(\lambda_1 + \lambda_2 + \lambda_3)$
'Componente Principal 2'	$PCA2_\lambda$	$\lambda_2/(\lambda_1 + \lambda_2 + \lambda_3)$

Cuadro 3.1: Descriptores geométricos basados en los autovalores del análisis de componentes principales de un vecindario de puntos.

La anisotropía A_λ del tensor $\mathbf{C}$ se define como la propiedad de ser direccionalmente dependiente, y se calcula de la siguiente manera:

$$A_\lambda = \frac{\lambda_1 - \lambda_3}{\lambda_1}$$

La omnivarianza O_λ es una medida de la inhomogeneidad de los puntos del vecindario:

$$O_\lambda = \sqrt[3]{\lambda_1\lambda_2\lambda_3}$$

La suma de autovalores Σ_λ también ha sido propuesta como una característica geométrica de interés en la descripción de vecindarios de puntos de nubes tridimensionales.

$$\Sigma_\lambda = \lambda_1 + \lambda_2 + \lambda_3$$

Por su parte, la variación superficial C_λ (Rusu RB 2009) también se ha explotado en trabajos anteriores como descriptor:

$$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$$

Y la autoentropía, que no es más que la entropía de los autovectores, es también empleada como medida de la cantidad de información que contiene un vecindario (Demantke J et al. 2012). Se calcula de la siguiente manera:

$$E_\lambda = -\sum_{i=1}^{3} \lambda_i ln(\lambda_i)$$

Existen varias librerías públicas que incorporan funciones para computar estos descriptores; por ejemplo, mismamente CloudCompare permite su cálculo. Sin embargo, en este trabajo se ha optado por emplear la librería *jakteristics* de Python, puesto que está bien optimizada y los computa rápidamente, y por practicidad. Esta librería incorpora, además, otros dos descriptores, similares a la variación superficial, pero calculados a lo largo de los autovectores $\overrightarrow{v}_1$ y $\overrightarrow{v}_2$ en vez del $\overrightarrow{v}_3$, y que denomina *componente principal 1* y *componente principal 2*.

$$PCA1_\lambda = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

$$PCA2_\lambda = \frac{\lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}$$

Además de los descriptores mencionados anteriormente, en este trabajo también se propone el uso de estos dos, de los vectores normales a cada eje, N_x, N_y, N_z, asociados al vecindario (también obtenibles directamente desde jakteristics) y de dos predictores más, computados a partir de código hecho en R con esta finalidad. Estos se basan en la cuenta de vóxeles en cada columna y se detallan en la siguiente sección, dedicada al tratamiento de los datos.

3.4. Tratamiento de los datos

Tras su etiquetado manual en CloudCompare, las nubes de puntos tridimensionales fueron tratadas con una combinación de código Python y código R para su posterior modelización. Es de destacar en este punto que todo el código empleado se ha desarrollado en el IDE RStudio (RStudio 2022), para lo que se ha realizado una integración de ambos lenguajes en un solo flujo de trabajo mediante la librería *Reticulate* (Allaire JJ et al. 2017).

Tras la voxelización a 5 *cm* de las nubes de puntos y el cómputo de los descriptores geométricos detallados en la sección anterior mediante código Python, que se puede encontrar en el Apéndice B, se computaron, además, dos predictores más: la cantidad de vóxeles total en cada columna y la cantidad de vóxeles de cada objeto. Como la voxelización implica la discretización del espacio en el que se encuentran los datos, es posible listar cada combinación (x, y) presente en los datos, por lo que se puede contar cuántos vóxeles (que se diferenciarán por su coordinada (z) cumplen la condición de estar en una coordenada (x, y) determinada. A esta cuenta se le ha denominado *altura_ total*. Así mismo, es de suponer que no estarán presentes todas las coordenadas (z) registradas en la nube en cada columna erigida sobre una combinación (x, y) cualquiera. Teniendo esto en cuenta, se pueden ordenar los vóxeles de cada columna, y establecer cuál es la longitud de todas las rachas ininterrumpidas de puntos sobre otros. Se ha asociado a cada punto la longitud de la racha ininterrumpida que lo contiene, y a este predictor se le ha denominado como *altura_ racha*. El código para computar estos dos predictores puede consultarse en el Apéndice C dedicado al código R.

3.4.1. Arquitectura de la red

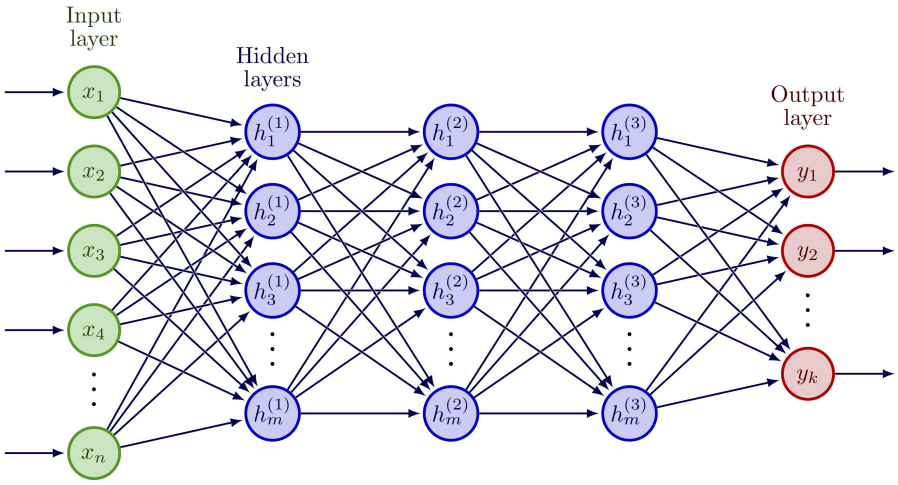
Después de computar todos los predictores, se descartaron las coordenadas (x, y, z) asociadas a cada vóxel, y los susodichos predictores fueron empleados como inputs de la red neuronal diseñada para clasificarlos: la clase predicha por la red se compara por la clase real, representada por las etiquetas asignadas a los datos, y se cuenta el número de aciertos como medida del error de predicción. En cuanto al modelo en sí, se trata de una red neuronal perceptrón multicapa, como ya se comentó previamente, cuya arquitectura consta de:

- Capa de entrada de 500 unidades, con función de activación ReLU
- 3 Capas ocultas con 500 unidades, cada una con función de activación ReLU
- Capa de salida con 4 unidades, con función de activación softmax
- Entropía cruzada categórica como función de pérdida
- Optimizador Adam

Y el algoritmo de propagación hacia atrás usa el optimizador Adam, con la entropía cruzada categórica como función de pérdida (la más habitual en redes neuronales dedicadas a clasificación), con normalización en lotes y 15 épocas de entrenamiento. La red ha sido programada utilizando la librería *Keras*

(Chollet et al. 2015), que usa una sintaxis sencilla para construir y entrenar la red y realizar predicciones con ella. En la Figura 3.4 puede encontrarse una representación esquemática de la arquitectura de la red empleada.

Figura 3.5: Esquema de la red neuronal empleada. En ella, $n = m^{(1)} = m^{(2)} = m^{(3)} = 500$, siendo n el número de unidades en la capa de entrada y $m^{(1)}$,$m^{(2)}$),$m^{(3)}$ el número de unidades en las capas ocultas 1, 2 y 3, y $k = 4$, siendo k el número de categorías y el número de unidades en la capa de salida.



3.5. Ajuste del modelo

La metodología seguida para entrenar el modelo consta de tres etapas. En primer lugar, se ha realizado un análisis de la sensibilidad del radio del vecindario sobre el que se computan los descriptores geométricos asociados a los puntos basado en un procedimiento de validación cruzada, en el que se entrena la red con 7 microparcelas y se deja una para validar el modelo. De esta manera, se obtiene un *error de validación cruzada* de la forma:

$$E_{CV} = \sum_{i=1}^{K} E_i, \quad K = 8$$

, donde E_i es el error de predicción medido en una microparcela i. El proceso puede esquematizarse de la siguiente manera:

1. Se deja fuera la microparcela 1, de validación y se entrena el modelo sobre las otras 7 restantes, de entrenamiento.

2. Con el modelo recién entrenado, se predice a qué categoría pertenece cada punto de la microparcela de validación.

3. Se mide el error de la predicción realizada con respecto de las categorías verdaderas registradas en la microparcela de validación

4. Se deja fuera la microparcela 2 y se repiten los pasos anteriores hasta completar las 8.

5. Se promedia el error de predicción, obteniendo así el error de validación cruzada.

Así, para una parcela, se toman las 8 microparcelas que se escogieron como submuestra a través del muestreo semialeatorio y que están voxelizadas y etiquetadas, se fija un radio (que se denotará como escala) y se realiza una validación cruzada dejando 1 de ellas fuera en cada vuelta. Tras esto, se ha promediado el error de validación cruzada en cada escala y se ha comparado entre parcelas. Las escalas empleadas han sido 5 cm, 15 cm, 25 cm, 35 cm, 45 cm, 55 cm, 65 cm, 75 cm, 85 cm y 95 cm.

En segundo lugar, una vez seleccionada la escala que minimiza el error de validación, se ha comprobado si un modelo basado en menos predictores producía mejores tasas de error de clasificación y, para ello, se ha realizado un análisis de la sensibilidad de eliminar predictores del modelo. Se han considerado 4 combinaciones de predictores diferentes, que se han denominado A, B, C y D:

- A: Todos los predictores
- B: Todos los predictores menos *altura_total* y *altura_racha*
- C: Todos los predictores menos Σ_λ, O_λ, E_λ y L_λ
- D: Todos los predictores menos N_x, N_y y N_z

La justificación de la combinación B se basa en que el cómputo de los predictores *voxeles_cuenta* y *stack_size* se realiza en una escala diferente al cómputo de los descriptores geométricos: estos últimos toman un radio de varios vóxeles, mientras que los descartados en esta combinación solo contabilizan los vóxeles de una columna y podrían introducir confusión. En cuanto a la combinación C, el estudio de Weinmann de 2013 (Weinmann M et al. 2013) realiza un análisis minucioso sobre la importancia de varios de los descriptores geométricos en la predicción realizada por diferentes modelos de clasificación bajo diferentes supuestos, y determina que Σ_λ, O_λ, E_λ y L_λ son los predictores que menos veces resultaron importantes. Por su parte, en la combinación D se descartan las normales, puesto que no se basan en la los autovalores del análisis de componentes principales y, como en el caso B, podrían introducir confusión. De nuevo, la medida de error que se ha evaluado es la precisión de validación, puesto que se ha vuelto a replicar el procedimiento seguido en la evaluación de la mejor escala: para cada parcela, se ha realizado una validación cruzada con las 8 microparcelas dejando 1 fuera en cada vuelta para cada combinación de predictores. Los resultados de ambas etapas de entrenamiento pueden consultarse en Capítulo 4, dedicado a analizarlos.

Una vez se dio con la mejor escala y la mejor combinación de variables de las analizadas, se empleó la red neuronal entrenada con los descriptores geométricos seleccionados computados a esa escala para clasificar las parcelas enteras. Para ello, primero se voxelizaron las parcelas enteras y se computaron los descriptores en ellas. Después se realizaron predicciones sobre ellas de 4 maneras alternativas:

- Empleando el modelo entrenado en cada parcela exclusivamente para tal parcela y utilizando las categorías 'arbusto', 'hierba', 'rama' y 'tronco' (que han sido codificadas como 1, 2, 3 y 4 respectivamente y luego representadas por variables auxiliares binarias)
- Empleando el modelo entrenado en cada parcela exclusivamente para tal parcela y utilizando las categorías 'hierba', 'mata' y 'tronco' (que han sido codificadas como 1, 2, 3 respectivamente y luego representadas por variables auxiliares binarias)
- Empleando un modelo que combina todas las microparcelas y utilizando las categorías 'arbusto', 'hierba', 'rama' y 'tronco' (que han sido codificadas como 1, 2, 3 y 4 respectivamente y luego representadas por variables auxiliares binarias)
- Empleando un modelo que combina todas las microparcelas y utilizando las categorías 'hierba', 'mata' y 'tronco' (que han sido codificadas como 1, 2, 3 respectivamente y luego representadas por variables auxiliares binarias)

Las alternativas basadas en un modelo general entrenado con todas las microparcelas se justifican por el hecho de que, idealmente, sería deseable tener un modelo que sirva para clasificar parcelas procedentes de cualquier tipo de bosque, y este procedimiento es un acercamiento razonable a conseguir un modelo que tenga ese objetivo. Las alternativas que juntan a las ramas y los arbustos en una misma clase, por su parte, se basan en que estas dos presentan estructuras tridimensionales considerablemente parecidas: muchas veces una masa de puntos sin forma determinada. El hecho de juntar ambas en una clase facilita que el modelo se concentre en discernir entre lo que es tronco y lo que no, y esta separación es, en realidad, la más importante. Todos los resultados obtenidos se detallan en el siguiente capítulo, en el que, además, se incluyen visualizaciones de las parcelas clasificadas con los diferentes modelos.

Capítulo 4

Resultados

En el capítulo anterior se detalló la metodología empleada para obtener el modelo que clasificará, finalmente, las nubes de puntos tridimensionales asociadas a las parcelas forestales. En este capítulo se exponen los resultados de los diferentes ajustes de modelos realizados: para ello, se mostrará la precisión media obtenida en cada supuesto, se destacará cuál es la mejor combinación de hiperparámetros para en base a tal precisión y se visualizará la clasificación obtenida de cada parcela a través del modelo final.

4.1. Análisis de sensibilidad de la escala

Primero se presentan los resultados del análisis de sensibilidad de la escala más apropiada para computar los descriptores geométricos asociados a los vóxeles. Este análisis pasa por ajustar un modelo con la arquitectura especificada en la Sección 3.4. para cada combinación de parcela y escala, y medir su precisión mediante el procedimiento de validación cruzada. En él, se deja 1 microparcela fuera como muestra de validación y se ajusta un modelo sobre las otras 7, que sirven de muestra de entrenamiento, y se repite el procedimiento con cada microparcela como validación. En el Cuadro 4.1 se muestra la precisión media aproximada de cada combinación para el modelo que contempla las 4 categorías ('arbusto', 'hierba', 'rama' y 'tronco'). A la izquierda, se muestra la precisión de entrenamiento; es decir, la precisión que obtuvo la red neuronal al finalizar su última época de entrenamiento. A la derecha se presentan las mismas mediciones pero para la precisión de validación cruzada.

De este cuadro se extrae que, para las 15 épocas de entrenamiento, la red encontró una clasificación de los datos de entrenamiento más adecuada para que las predicciones se asemejasen a las categorías reales usando los predictores computados a escala 0,45 m, donde se obtuvo una precisión de entrenamiento de, aproximadamente, 87 %. Por su parte, bajo las condiciones del experimento, la escala que minimizó el error de validación cruzada (la que maximizó la precisión) fue 0,45 m, obteniéndose una precisión de validación de 81 %.

Escala (m)	Robledal 1	Robledal 2	Pinar 1	Pinar 2	Total
0.05	71-63	70-58	71-62	72-64	**71-62**
0.15	75-65	81-80	82-82	82-79	**80-77**
0.25	77-55	82-77	84-78	84-83	**82-73**
0.35	77-68	86-80	86-80	87-81	**84-77**
0.45	82-74	87-81	87-82	92-88	**88-81**
0.55	82-74	88-79	89-80	91-85	**87-80**
0.65	77-76	76-76	80-80	85-85	**80-79**
0.75	80-70	89-77	86-82	88-83	**86-78**
0.85	82-69	80-76	86-80	89-83	**84-77**
0.95	71-65	73-68	77-72	79-73	**75-70**

Cuadro 4.1: Precisión media de entrenamiento y validación en cada escala para el modelo de 4 categorías.

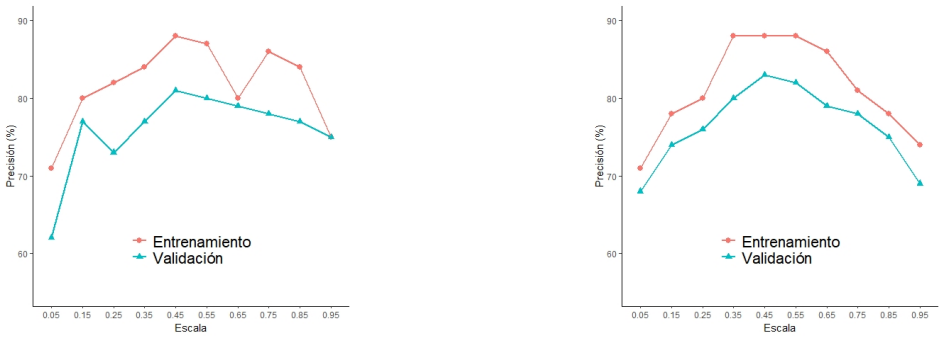
A continuación, se presentan los resultados del mismo análisis pero realizado con una red neuronal con 3 nodos de salida, en los que los datos empleados solo contienen 3 categorías: 'hierba', 'mata' y 'tronco'. Para ello, se ha agrupado en la misma categoría los datos que anteriormente estaban recogidos en las categorías 'arbusto' y 'rama'. En el Cuadro 4.2 se recoge la precisión media de este modelo:

Escala	Robledal 1	Robledal 2	Pinar 1	Pinar 2	Total
0.05	72-67	70-66	71-72	73-66	**71-68**
0.15	75-70	74-69	78-79	82-77	**78-74**
0.25	76-72	77-71	84-79	83-81	**80-76**
0.35	87-75	83-72	88-83	93-87	**88-80**
0.45	84-79	84-79	90-85	93-88	**88-83**
0.55	83-78	84-77	90-85	91-86	**88-82**
0.65	81-74	82-76	87-82	88-83	**86-79**
0.75	80-75	77-73	85-82	84-80	**81-78**
0.85	73-69	77-73	81-78	81-79	**78-75**
0.95	71-66	70-65	77-72	79-74	**74-69**

Cuadro 4.2: Precisión media de entrenamiento y validación en cada escala para el modelo de 3 categorías.

De donde se extrae que, para el modelo de 3 categorías, las escalas que producen un mejor ajuste a los datos por parte del modelo en las 15 épocas de entrenamiento son 0,35, 0,45 y 0,55 m; en todas ellas

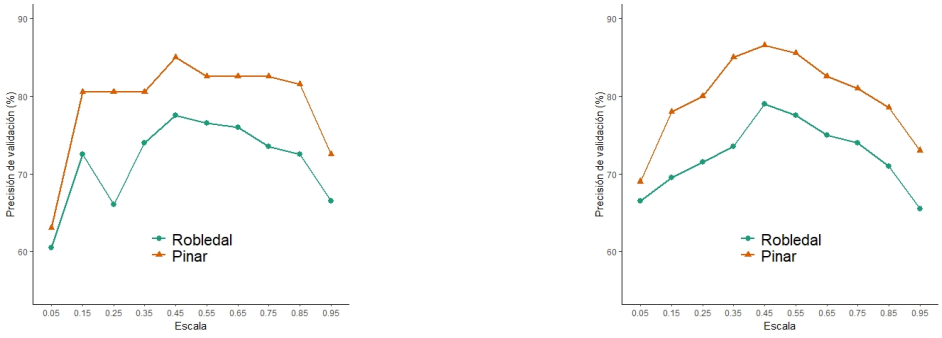
se obtuvo una precisión del 88 %. Tal y como en el caso anterior, la máxima precisión de validación cruzadase obtiene con la escala 0,45 m, donde se obtuvo una precisión de clasificación de los datos 'no vistos' por el modelo del 83 %. La información contenida en ambos cuadros puede observarse de manera gráfica en la Figura 4.1, en la que se observa cómo, en ambos modelos, la tendencia general es que la precisión media del modelo en las diferentes parcelas crece a medida que se aumenta la escala hasta llegar a un máximo en 0,45 m, punto en el que la tendencia comienta a ser negativa.



(a) 4 categorías (b) 3 categorías

Figura 4.1: Gráfico de líneas de la precisión de entrenamiento y validación media obtenida con las diferentes escalas. a) Los resultados para el modelo de 4 categorías. b) Los resultados del modelo de 3 categorías.

Así, 0,45 m será la escala seleccionada para continuar con el ajuste de los modelos, y la que se empleará en la etapa de selección de predictores. En esta etapa también se considerarán las dos variantes del modelo, de número diferentes de categoría. Además, en los Cuadros 4.1 y 4.2 puede entreverse un comportamiento previsible del modelo: la precisión media obtenida en los pinares es superior a la obtenida en los robledales. Esto se ilustra en la Figura 4.2:



(a) 4 categorías (b) 3 categorías

Figura 4.2: Gráfico de líneas comparando la precisión de validación media obtenida con las diferentes escalas para los robledales y los pinares. a) Los resultados para el modelo de 4 categorías. b) Los resultados del modelo de 3 categorías.

En esta figura, la línea anarajanda representa la precisión de validación media obtenida para la clasificación de los robledales y la línea verdosa representa la precisión equivalente promediando la

obtenida en los pinares. Puede comprobarse claramente cómo con los pinares, en la totalidad de las escalas, el modelo tiene más capacidad predictiva. Es más, en los robledales, bajo ningún modelo, no se llega a una precisión del 80 %, mientras que clasificando datos nuevos de pinares se llega a una precisión promedio del 86 %, que alcanza un máximo de 88 % en el Pinar 2 con la escala de 0,45 m. Este hecho probablemente se deba a las diferencias en la estructura del tronco y ramas de robles y pinos: los primeros tienden a ser árboles simpodiales, en los que el tronco deja de ser el eje principal para dividirse en ramas que pueden ser tan grandes o más que el tronco en sí. Por su parte, los pinos crecen con una estructura monopodial, en la que hay un principal eje de crecimiento del que parten ramas secundarias de mucho menor calibre y, habitualmente, perpendiculares a este. Es razonable que esta diferencia en la estructura del tronco dificulte la clasificación de las ramas y troncos de robles, puesto que, en realidad, no hay una diferencia clara entre los dos.

Una vez concluidas las interpretaciones de los resultados obtenidos en el análisis de sensibilidad de la escala elegida con la que computar los predictores geométricos, se procede a mostrar los resultados pertinentes a las diferentes combinaciones de predictores propuestas en la siguiente sección.

4.2. Análisis de sensibilidad de los predictores empleados

En esta sección se muestran los resultados del análisis de sensibilidad de los predictores empleados por la red. En este, tal y como en el análisis previo sobre la escala adecuada en la que computar los descriptores geométricos, se realizó una validación cruzada con las 8 microparcelas de cada parcela, para estimar la precisión con la que la red las clasifica. Para ello, se seleccionaron 4 combinaciones de predictores: una en la que se emplea el conjunto de todos los predictores computados y 3 en las que se prueban diferentes subconjuntos. Si bien se tiene ya, con los resultados obtenidos en el análisis anterior empleando todos los predictores, un modelo que tiene una precisión del 81-83 %, es posible que algunos de los predictores empleados proporcionen confusión a la hora de clasificar los datos y eliminarlos aumente la capacidad predictiva de la red o que, simplemente, se pueda obtener una precisión similar con un modelo más sencillo. Bajo estas premisas, se computó la precisión media de las cuatro combinaciones en las diferentes parcelas, y en los Cuadros 4.3 y 4.4 se muestran los resultados obtenidos.

Combinación	**Robledal 1**	**Robledal 2**	**Pinar 1**	**Pinar 2**	**Total**
A	82-74	87-81	87-82	92-88	**88-81**
B	84-78	88-86	87-81	94-85	**88-83**
C	85-77	89-68	85-73	89-74	**87-73**
D	78-63	77-64	85-64	80-60	**80-63**

Cuadro 4.3: Precisión media de entrenamiento y validación en cada combinación para el modelo de 4 categorías.

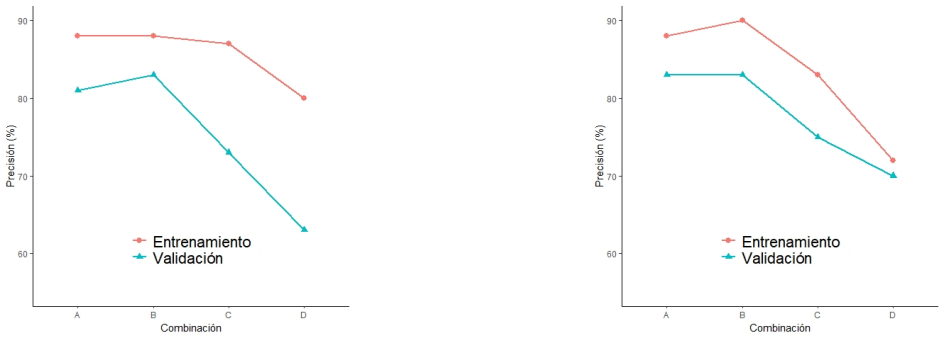
En ellos, la información referente a la combinación A no es más que la contenida en la fila correspondiente a la escala de 0,45 m de los Cuadros 4.1 y 4.2. Puede observarse, sin embargo, que la combinación B -que prescinde de los predictores *altura_total* y *altura_racha*- produce una precisión mayor o igual que la combinación que contempla el uso de todos los predictores simultáneamente: en el caso del modelo de 4 categorías, aumenta la precisión de validación cruzada en 2 puntos, mientras que en el modelo de 3 categorías, de manera menos relevante, es la precisión de entrenamiento la que aumenta 2 puntos. Es notable también cómo los predictores que resultaron menos relevantes según el

Combinación	Robledal 1	Robledal 2	Pinar 1	Pinar 2	Total
A	84-79	84-79	90-85	93-88	**88-83**
B	88-84	85-80	92-82	94-85	**90-83**
C	83-71	83-75	82-77	84-78	**83-75**
D	69-67	69-67	78-70	72-74	**72-70**

Cuadro 4.4: Precisión media de entrenamiento y validación cada combinación para el modelo de 4 categorías.

estudio de Weinmann de 2013 en este caso sí son relevantes, ya que eliminarlos reduce la capacidad predictiva de ambos modelos en un 8 % y 10 % respectivamente, y que si se eliminan las normales N_x, N_y y N_z la capacidad predictiva se reduce todavía más; hasta en un 20 %. Esto las convierte en predictores especialmente importantes en este modelo.

Para finalizar con los resultados relativos a los análisis de la precisión del modelo, se muestra en la Figura 4.3 el gráfico de líneas de la precisión obtenida con cada combinación para ambos modelos, de manera similar a lo que se mostraba en la Figura 4.2.



(a) 4 categorías (b) 3 categorías

Figura 4.3: Gráfico de líneas de la precisión de entrenamiento y validación media obtenida con las diferentes combinaciones de predictores. a) Los resultados para el modelo de 4 categorías. b) Los resultados del modelo de 3 categorías.

En esta Figura 4.3 se visualiza el máximo de la precisión correspondiente a la Combinación B seguido de cerca por la precisión obtenida con la combinación A, pero lo más destacable es la gran disminución de la capacidad predictiva en cuanto se eliminan las variables contempladas en la Combinación C y en la Combinación D. En la siguiente sección, por otra parte, se muestra visualmente la clasificación hecha por el modelo de las 4 parcelas tratadas en este trabajo, en las que se hará hincapié en la capacidad de separar los puntos correspondientes a troncos del resto de puntos.

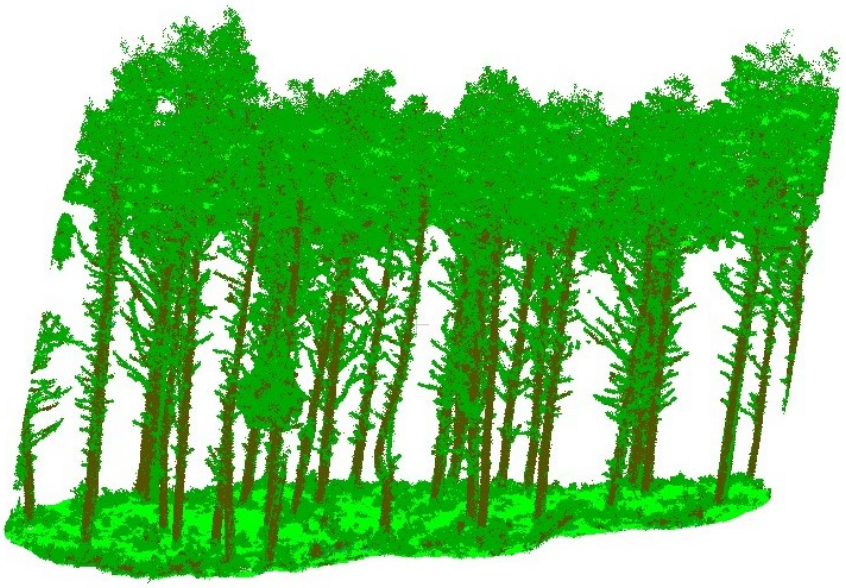
4.3. Clasificación de las parcelas

En esta sección se trata de validar los resultados obtenidos de clasificar las parcelas completas con la red creada a partir de los descriptores geométricos de la Combinación B y computados a escala

0,45 m. Esta validación es, en realidad, notablemente difícil: de la única herramienta que se dispone para valorar qué tan buena ha resultado la clasificación es visualizar las parcelas tras haber sido coloreadas de acuerdo a las etiquetas asignadas por el modelo. Es conveniente recordar aquí que las nubes de puntos completas contienen decenas de millones de puntos, y acceder a las zonas interiores de la nube requeriría de realizar numerosos cortes, lo que es tedioso y, de aún así, inpreciso. Además, cualquier forma de calcular la precisión de la clasificación de manera cuantitativa llevaría comparar las etiquetas artificiales predichas por el modelo con alguna suerte de etiqueta real, lo que implicaría haber clasificado antes la parcela entera a mano y ahorrarse esteo es, precisamente, el problema que se trata de solucionar en este trabajo. Es por ello que en esta sección se presentan los resultados visuales de la clasificación a modo de herramienta de validación del procedimiento, haciendo hincapié en la separación de los troncos del resto de puntos. Esto es fácil de observar y razonablemente preciso y, en realidad, es el punto de mayor interés para calcular parámetros como la biomasa del bosque.

Debido a lo razonado anteriormente y que, por consecuencia de los resultados obtenidos con las validaciones cruzadas se tiene que tanto el modelo de 4 categorías como el de 3 categorías funcionan razonablemente bien, se han clasificado todas las parcelas con ambas versiones de la red y se valorado visualmente cuál funciona mejor en qué caso. Lo que se observó va en concordancia con los resultados discutidos hasta ahora: a la hora de generalizar sobre datos no vistos anteriormente por la red, el modelo de 4 categorías funciona mejor en los pinares, mientras que el de 3 categorías funciona mejor en los robledales (aunque ningún modelo funciona realmente bien con estos últimos).Primeramente, se ilustrará la clasificación realizada para el Pinar 1 en la Figura 4.4:

Figura 4.4: Pinar 1 clasificado con el modelo de 4 categorías.



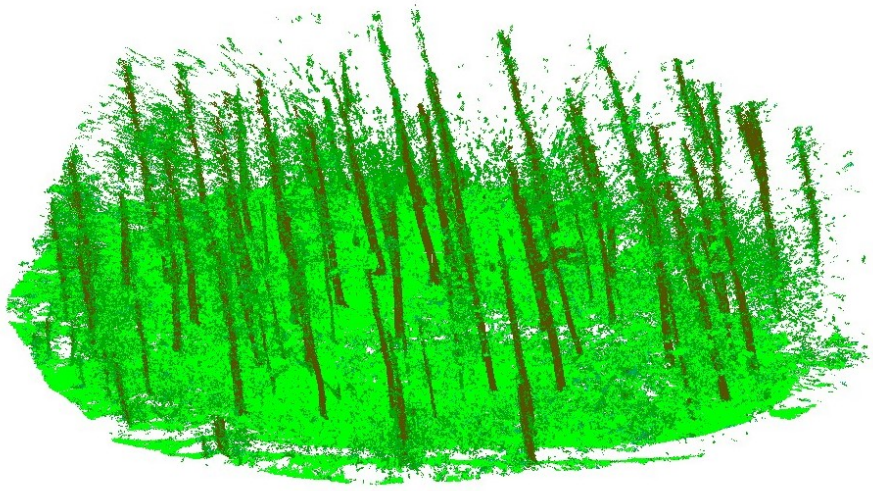
De esta visualización puede apreciarse cómo hay bastante confusión entre las ramas y los arbustos; estos últimos han sido etiquetados casi en su totalidad como ramas. Esto es razonable, teniendo en cuenta que presentan estructuras tridimensionales realmente parecidas, como se discutió en el Capítulo 3. La hierba, por su parte, ha sido clasificada casi en su totalidad correctamente, si bien hay algunos parches en las ramas clasificados como hierba en zonas planas. Esto de todas formas no es un inconveniente, ya que son fácilmente etiquetables como rama con un simple filtro de altura. Los troncos,

sin embargo, han sido clasificados con bastante precisión: en la Figura 4.5 se muestran los puntos etiquetados como 'tronco' filtrados del resto.

Figura 4.5: Puntos del Pinar 1 asignados como pertenecientes a la categoría 'tronco'.

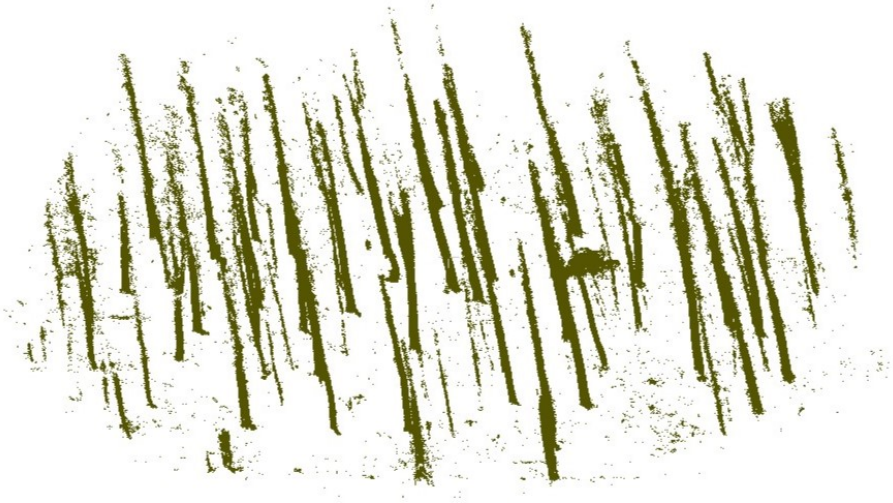
Puede verse como el modelo ha conseguido aislar los troncos de una manera satisfactoria y que ha sido incluso capaz de reconocer un tronco tirado en el suelo, aunque haya clasificado algunos puntos del suelo y de ramas como 'tronco'. En cuanto al pinar 2, los resultados de la clasificación se muestran en la Figura 4.6, a continuación.

Figura 4.6: Pinar 2 clasificado con el modelo de 4 categorías.



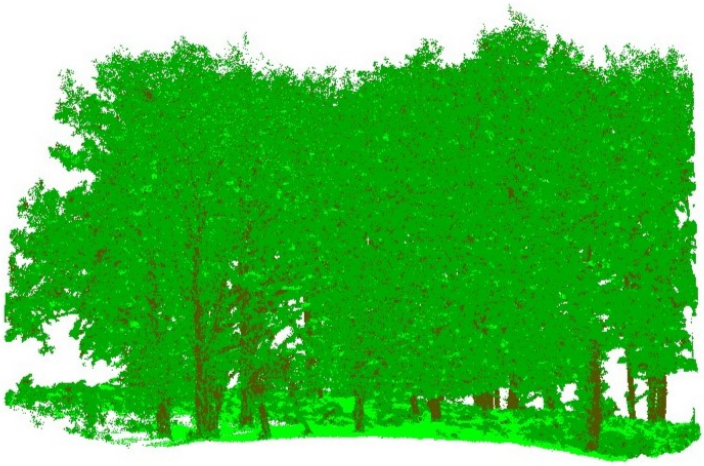
Esta parcela, tal y como se esperaba de los valores de precisión obtenidos en los análisis, ha sido etiquetada de manera más precisa. De nuevo, la práctica totalidad de la hierba ha sido clasificada correctamente y, en este caso, las ramas se han separado aún mejor de los troncos. Esto se corrobora en la Figura 4.7:

Figura 4.7: Puntos del Pinar 2 asignados como pertenecientes a la categoría 'tronco'.



Se observa como claramente los troncos han sido aislados con éxito, siendo raros los puntos que se han clasificado bajo la etiqueta 'tronco' y que realmente no pertenezcan a uno. En la imagen, destaca un tocón del suelo que también ha sido clasificado en esta categoría. A continuación, en la Figura 4.8, se ilustra la clasificación realizada de la nube de puntos tridimensional asociada a la parcela del Robledal 1

Figura 4.8: Robledal 1 clasificado con el modelo de 4 categorías.

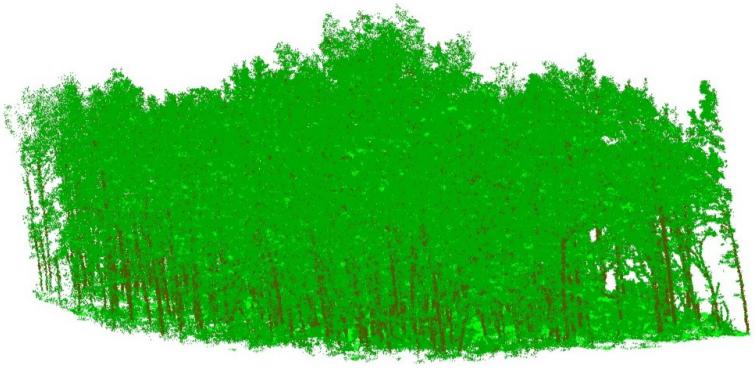


En esta ocasión la clasificación ha sido realizada con el modelo de 3 categorías y no ha sido buena. Observando la nube con todos los puntos ya puede observarse como los troncos y las ramas se entremezclan: esto probablemente sea debido a que es más difícil establecer una regla clara de qué es qué, debido a la edificación simpodial del tronco: en estos árboles, la yema terminal que formaría el eje principal de crecimiento detiene su desarrollo y las yemas auxiliares originan grandes ramas oblicuas. En la Figura 4.9 se adjunta una ilustración de los puntos asignados como 'tronco' aislados de los demás:

Figura 4.9: Puntos del Robledal 1 asignados como pertenecientes a la categoría 'tronco'.

Efectivamente, las ramas y los troncos aparecen entremezclados en gran medida. Para esta parcela, ni el modelo de 4 categorías ni el de 3 ha realizado una distinción satisfactoria de los troncos. Para finalizar con las visualizaciones de las parcelas completas, se muestra ahora, en la Figura 4.10, la equivalente ilustración del Robledal 2.

Figura 4.10: Robledal 2 clasificado con el modelo de 4 categorías.



En esta parcela, aunque se corresponde con un robledal, los troncos forman con más frecuencia ejes principales de crecimiento y la clasificación parece más acertada. La hierba ha sido clasificada con precisión, formando un manto continuo correspondiente al suelo del bosque, y apenas aparece en cotas superiores; de cualquier manera, como ya se dijo, este es un problema sencillo de resolver sin más que filtrar esos puntos por altura. Por ejemplo, cualquier punto de hierba localizado en una altura superior a 0,4 m se puede reetiquetar como 'mata'. Como en casos anteriores, se adjunta una imagen con los troncos aislados:

Figura 4.11: Puntos del Robledal 2 asignados como pertenecientes a la categoría 'tronco'.

En la Figura 4.11, así, se puede observar como en el caso de este robledal los troncos se han aislado algo mejor, aunque sigue habiendo cierta confusión con las ramas. Tras esta última visualización se procede a mostrar los tiempos de computación totales asociados a, finalmente, generar tales imágenes.

4.4. Tiempos de computación

En esta sección se muestran los tiempos de computación asociados al proceso completo de clasificación de las parcelas. Para ello, se detallará el tiempo empleado en los procesos más computacionalmente costosos, a saber: la voxelización de las parcelas enteras, el cómputo de los descriptores geométricos asociados a las parcelas voxelizadas, el entrenamiento de los modelos finales y la clasificación de las parcelas enteras por parte del modelo. En el Cuadro 4.5 se muestra un resumen de los tiempos aproximados de ejecución de estos procesos en todas las parcelas tratadas.

Puede verse como el proceso de voxelización de las parcelas no es demasiado costoso: apenas lleva unos 10 s en el peor de los casos. El cómputo de los descriptores geométricos en la escala seleccionada para el modelo, de 0,45 m, lleva algo más de tiempo: entre 1 y 2 minutos aproximadamente. El entrenamiento del modelo con la Combinación B de parámetros conlleva un tiempo parecido al combinado en los dos procesos anteriores, demorando entre 1 minuto y 2 y medio. Por último, el tiempo de clasificación -de etiquetado de las parcelas por parte del modelo- es lo más costoso computacionalmente, conllevando un tiempo comprendido entre los 7 minutos en el caso del Pinar 1 hasta los 11 minutos en

Proceso	Robledal 1	Robledal 2	Pinar 1	Pinar 2	Promedio
Voxelización	10 s	10 s	5 s	10 s	**10 s**
Descriptores geométricos	85 s	110 s	65 s	50 s	**80 s**
Entrenamiento	90 s	120 s	150 s	60	**105 s**
Clasificación	540 s	660 s	420 s	600 s	**555 s**
Total	**12 min**	**15 min**	**11 min**	**11 min**	**13 min**

Cuadro 4.5: Tiempo total aproximado de clasificación de las parcelas forestales mediante el modelo diseñado.

el caso del Robledal 1. Así con todo, el tiempo medio del proceso completo de clasificación, pasando por todos los cálculos mencionados es de tan solo 13 minutos, y nunca más de 15 minutos.

Con esto, finaliza la sección de Resultados, en la que se ha visto cómo se ha evaluado el tamaño adecuado de la escala para computar los descriptores geométricos y la mejor combinación de predictores para clasificar las parcelas forestales en base a sendos procedimientos de validación cruzada basados en el uso de microparcelas previamente extraídas de las parcelas completas. La bondad de clasificación sobre las susodichas parcelas se ha valorado mediante la visualización de las nubes de puntos tridimensionales asociadas coloreadas y el aislamiento de los puntos etiquetados como pertenecientes a la categoría 'tronco'. Finalmente, se han expuesto los tiempos de ejecución de la clasificación de las parcelas completas, divididos en los principales procesos llevados a cabo para la consecución del etiquetado final. En el siguiente y último capítulo se discutirán los resultados obtenidos y las conclusiones que se pueden extraer de ellos.

Capítulo 5

Discusión

En los capítulos anteriores se introdujo el tema principal del trabajo, que es la clasificación de nubes de puntos tridimensionales de parcelas forestales mediante técnicas de aprendizaje automático, contextualizado bajo la necesidad de realizar mejores inventarios forestales con el fin de calcular parámetros biológicos que sean de ayuda para entender mejor la estructura y estado de los bosques. Este conocimiento es de gran utilidad para un sin fin de objetivos sociales, económicos y científicos y en los últimos años, con el surgimiento de las tecnologías LiDAR, ha aumentado el interés en aplicar técnicas de aprendizaje automático, especialmente aprendizaje profundo, sobre nubes de puntos tridimensionales de parcelas forestales escaneadas mediante esta tecnología con la finalidad de mejorar y automatizar el inventariado forestal.

A continuación, en este trabajo se han repasado los usos principales de las técnicas de aprendizaje profundo en este campo de investigación, concluyendo que, si bien existe abundante bibliografía sobre la identificación de árboles en ambientes urbanos y la clasificación de distintas especies, apenas hay trabajos previos en la clasificación de la estructura forestal en sus diferentes componentes. Bajo esta premisa, se ha desarrollado un modelo de clasificación basado en una red neuronal de perceptrón multicapa que clasifica los puntos de las nubes tridimensionales asociadas a parcelas forestales en arbusto, hierba, rama y tronco. Para ello se han empleado datos de 4 parcelas forestales cedidas por el Departamento de Explotación y Prospección de Minas, área de Ingeniería Cartográfica, Geodésica y Fotogrametría de la Universidad de Oviedo. Con ellas se ha llegado a una combinación de predictores basados en descripciones geométricas de los puntos y sus entornos, previa voxelización de los datos con el fin de reducir el coste computacional asociado a trabajar con archivos tan grandes y que contienen información excesiva, que permite realizar la clasificación descrita.

Tras la visualización de las parcelas clasificadas con el modelo obtenido, se ha visto cómo este funciona notablemente mejor en parcelas que contienen árboles de ramificación monopodial, en la que el tronco forma un eje principal de crecimiento del que parten ramas oblicuas. En estas, la separación de los troncos del resto de puntos, que es el paso que despierta mayor interés, se realiza acertadamente. En las parcelas en las ramas no se disponen de esta manera, si no que los troncos presentan un crecimiento simpodial y poseen ramas de calibre similar al tronco, esta separación no se realiza correctamente y no se puede recomendar el uso de este modelo con esta finalidad. También se vio cómo la confusión que existía entre las categorías 'hierba', 'arbusto' y 'rama' en el caso del modelo de 4 categorías y entre 'hierba' y 'mata' era de una importancia mejor, ya que es fácilmente solucionable con unos simples filtros de altura para atajar tal confusión. Como ejemplo se muestra, a continuación, las parcelas de Pinar 1 (Figura 5.1) y Pinar 2 (Figura 5.2) tras una breve edición en CloudCompare.

La edición realizada ha consistido, simplemente, en etiquetar los puntos clasificados en las categorías 'hierba' y 'arbusto' que no estuviesen a ras de suelo como 'rama', y los puntos clasificados bajo la

Figura 5.1: Pinar 1 etiquetado tras una breve edición en CloudCompare. Se han filtrado los puntos de hierba y arbusto que estaban en el aire y se han etiquetado como 'rama', y los puntos de 'rama' que estaban a ras de suelo y se han clasificado como 'arbusto'.

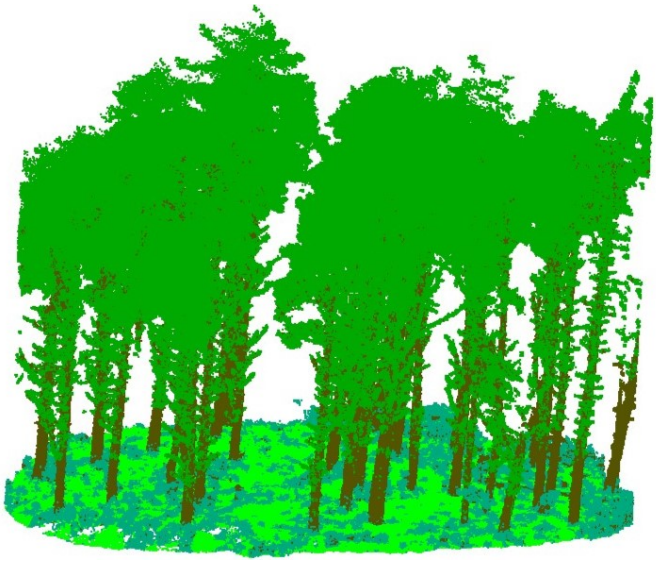
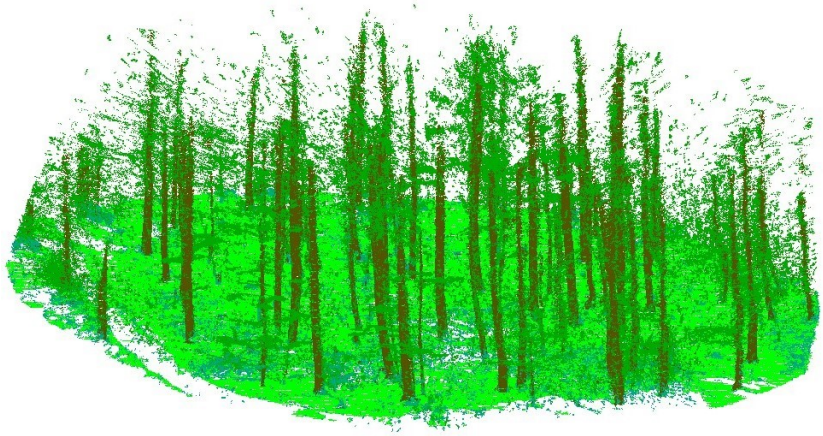


Figura 5.2: Pinar 2 etiquetado tras una breve edición en CloudCompare. Se han filtrado los puntos de hierba y arbusto que estaban en el aire y se han etiquetado como 'rama', y los puntos de 'rama' que estaban a ras de suelo y se han clasificado como 'arbusto'.



categoría 'rama' que estuviesen a ras de suelo como 'arbusto'. Este cambio solo conlleva unos pocos minutos de manipulación y, sumado al etiquetado preexistente realizado por el modelo, proporciona una clasificación realmente satisfactoria. Esta edición puede sustituirse fácilmente por un sencillo algoritmo de filtrado que recategorice los puntos que se encuentren a partir de cierta cota (z) tras la asignación inicial dada por el modelo. Esta es una mejora fácil de implementar que corrige el comportamiento deficiente del modelo de asignar una etiqueta equivocada a puntos que obviamente no pueden pertenecer a tal categoría. Así, en el estado actual del modelo, puede ser concebido como una herramienta para separar rápidamente troncos del resto de puntos en parcelas con árboles de ramificación monopodial

como los pinos o abetos. El proceso de etiquetado manual de una parcela forestal entera es una tarea laboriosa que requiere de decenas de horas de trabajo: este modelo permite acelerar a tan solo unos pocos minutos el separar los troncos, y a unas decenas de minutos más separar el resto de estructuras.

Aún así, existe una serie de pequeños ajustes que se pueden realizar en el proceso de tratamiento de los datos que seguramente mejoren la calidad de las predicciones obtenidas. Una de ellas pasa por muestrear las microparcelas de manera que se incluya un borde exterior de radio igual a la escala en la que se computen los descriptores geométricos, con el fin de evitar el conocido como efecto frontera, que ocurre en contextos como este. Ocurre que cuando se computan los descriptores de los puntos se hace en un vecindario con forma esférica: tal esfera, cuando se centre sobre puntos que están en los bordes o cerca de ellos, se verá truncada, y los descriptores no se computarán como se esperaría. A mayor radio de vecindario, más problemático será este efecto frontera. Si se muestreasen las microparcelas tal y como se acaba de indicar, se computasen los descriptores geométricos y luego se descartasen los bordes añadidos como holgura, la descripción geométrica del entorno de los datos se realizaría de forma más precisa. Si no se ha hecho así en este trabajo ha sido por mantener la filosofía de ofrecer un procedimiento sencillo con el que obtener unas primeras clasificaciones veloces de las nubes de puntos tridimensionales. Además, en general, el proceso de muestreo semialeatorio propuesto es fácilmente mejorable, aunque, seguramente, cualquier mejora conllevará un aumento del tiempo necesario para obtener el modelo final. Tal consideración se deja a elección del posible usuario.

Una última posible mejora es la de computar los descriptores propuestos en este trabajo, los denominados por el autor como *altura_total* y *altura_racha* en la misma escala que el resto de descriptores. Si bien finalmente se ha descartado su uso tras el análisis de sensibilidad de los predictores, es razonable pensar que tales variables puedan ser un buen indicador de si un punto pertenece a un tronco o no: es de esperar que los puntos que describan estructuras asociadas a troncos posean valores altos de ambos predictores, mientras que los demás, no. Finalmente, se quiere dejar sobre la mesa una posible extensión de la metodología propuesta que puede resultar de notorio interés: el modelo puede ser generalizado, de manera que se entrene con microparcelas procedentes de varias parcelas a la vez, y potencialmente conseguir un modelo que fuese capaz de clasificar las estructuras presentes en cualquier tipo de bosque. Sería necesario adaptar el proceso de validación cruzada presentado en este trabajo o buscar alguna alternativa y, seguramente, el modelo resultante sería computacionalmente más costoso de emplear porque se debería aumentar en gran medida el número de datos de entrenamiento. En realidad, esto último no es un problema inatajable; en este trabajo ni siquiera se ha empleado una GPU para acelerar el proceso de entrenamiento ni de clasificación, pero esto es perfectamente plausible y conseguiría reducir todavía más el tiempo de etiquetado. No es descartable, así, que un modelo entrenado con un número considerablemente mayor de microparcelas provenientes de diversss tipos de bosques fuese capaz de producir clasificaciones satisfactorias de diversidad de parcelas o, cuanto menos, de parcelas de árboles monopodiales.

Apéndice A

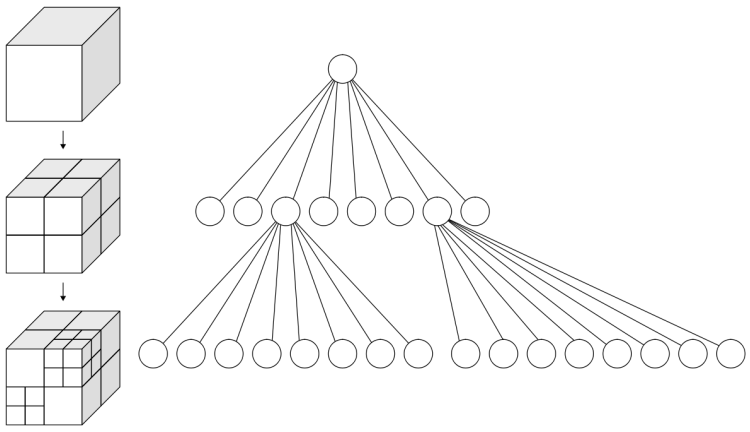
Clasificación manual de los datos en CloudCompare

En este apéndice se adjunta una guía sencilla de la clasificación manual de los datos de las microparcelas realizado en CloudCompare y se describen aquí las principales características y funcionalidades del software empleadas con tal finalidad.

A.1. Características del software

CloudCompare es un software libre de edición y procesado de nubes de puntos tridimensionales. Originariamente diseñado para realizar comparaciones directas entre nubes de puntos, el programa ha ido incorporando numerosas herramientas y algoritmos para realizar distintas operaciones. En la actualidad dispone de funcionalidad para remuestrear datos, añadir campos escalares y vectoriales a los ficheros que contienen a las nubes de puntos tridimensionales, realizar operaciones estadísticas, segmentar los datos, etc., así como de herramientas para mejorar la visualización de los datos con los que se esté trabajando. Su funcionamiento se basa en la partición recursiva del espacio muestral en octantes para mejorar el rendimiento del procesado y del cómputo de operaciones sobre los datos.

Figura A.1: Esquema de la división en del espacio en octantes y su equivalente octárbol, que permiten tratar archivos de datos de gran tamaño eficientemente.

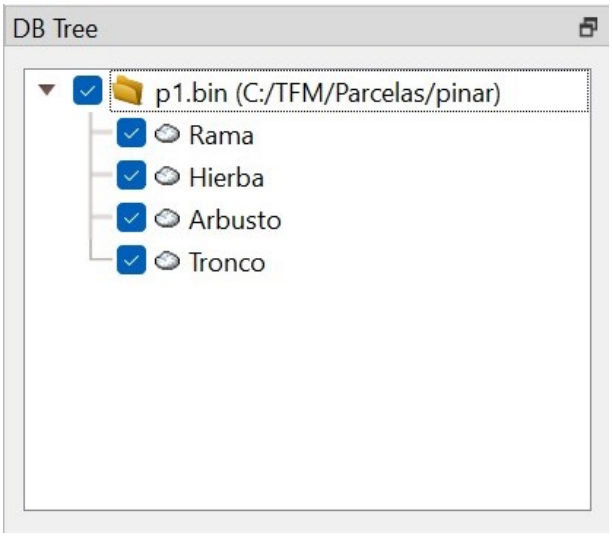


Dado un volumen cúbico de espacio, los octantes se forman dividiendo tal cubo en 8 subcubos, también denominados celdas. Después, cada octante se convierte en un nodo de un octárbol, que es una estructura de datos en árbol (con 8 hijos a partir de cada nodo). Tales nodos equivalen a un punto tridimensional, lo que los asocia con la nube de puntos tridimensional original. En la figura A.1, tomada de *Wikimedia Commons*, se presenta un esquema de tal subdivisión.

A.2. Funcionalidades empleadas

Menú DB Tree Las entidades cargadas en el programa (en este caso, las microparcelas) se almacenan en el menú DB tree, al que se puede acceder a través de una ventana gráfica que se encuentra en la parte izquiera por defecto. En la Figura A.2 adjunta se muestra su aspecto.

Figura A.2: Menú DB Tree. Se sitúa en la parte superior izquierda de la ventana gráfica de CloudCompare, y es el que muestra las entidades cargadas. En este caso, hay cargado un archivo padre, que es una microparcela del Pinar 1 y 4 archivos hijo, que contienen a los puntos ya clasificados manualmente en las 4 categorías 'arbusto', 'hierba', 'rama' y 'tronco'.



Este menú tiene forma de árbol jerárquico, en el que se puede agrupar las instancias en carpetas que incluyen unas a otras en niveles de jerarquía. Esto puede realizarse fácilmente sin más que clicando en una (tanto en su vista 3D como en su entrada en el menú DB tree) y arrastrándola hasta el nivel deseado de la estructura padre-hijo. Además, se pueden seleccionar varias instancias a la vez manteniendo la tecla CTRL o la tecla SHIFT mientras se clica encima de todas las que se desee involucrar en la selección.

Herramienta Segmentation. Puede accederse a esta herramienta a través del icono de la barra de herramientas o a través del menú de edición. Permite al usuario trazar una línea poligonal en el objeto para separar los puntos que quedan a un lado y al otro y generar un recorte con cualquiera de los dos subconjuntos generados. El procedimiento para separar puntos con esta herramienta es el siguiente:

1. Seleccionar una o varias entidades e inicializar la herramienta. Una nueva barra de herramientas aparecerá en la esquina superior derecha de la vista 3D. Por defecto, la herramienta empezará con el modo de edición de "línea poligonal", que es el que se ha empleado en este trabajo.

2. Con esta, puedes comenzar a trazar un polígono con el que encerrar los puntos deseados. Simplemente haciendo clic izquierdo, creas el primer vértice del polígono, y con sucesivos clics se definen las posiciones de los posteriores vértices, que son automáticamente encadenados entre sí con líneas rectas.

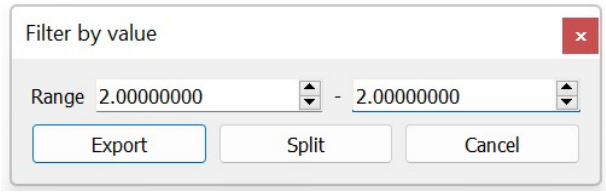
3. Una vez que se ha confinado a todos los puntos que se desea que estén dentro, basta con hacer clic derecho para parar la edición del polígono.

4. Concluida la selección, se puede descartar la selección o se puede guardar. Para hacer esto último, se debe decidir si se desea guardar los puntos que han quedado dentro o fuera del polígono, clicando en sendos botones de la barra de herramientas asociada al proceso de recorte. Una vez elegida una, los otros puntos desaparecen. Tras esto, automáticamente se crea una entidad nueva con los puntos recortados.

Herramienta Merge. Esta herramienta es accesible desde el menú de edición, o con el icono desde la barra principal de herramientas. Su función es fusionar dos o más entidades. Al fusionarlas, las nubes originales serán eliminadas; a veces será necesario guardar una copia en otro directorio o clonarlas si se quisiesen preservar. De cualquier manera, una vez se fusionan dos o más entidades, CloudCompare preguntará al usuario si se quiere generar un campo escalar con una etiqueta que asigna a los puntos su nube de procedencia.

Filtrado por valor de campo escalar A esta herramienta se accede desde el menú superior a través de Edit >Scalar Fields >Filter by Value. Tras esto, se abrirá una ventana, que se ilustra en la Figura A.3.

Figura A.3: Ventana para filtrar por un valor de campo escalar. En este caso, se muestra el ajuste necesario para filtrar por el valor 2, equivalente a la clase 'hierba'.



Una vez que se ha realizado la clasificación de la nube de puntos de la parcela completa, este filtrado permite separar los puntos en base a las etiquetas asignadas por el modelo, sin más que filtrar por el valor del campo escalar pertinente

Herramienta Save Esta herramienta es accesible desde el menú de edición, o con el icono desde la barra principal de herramientas. Con ella se guardan las entidades seleccionadas en alguno de los formatos compatibles con el software. De entre ellos, destaca el formato BIN (.bin), que es un archivo binario que contiene la nube de puntos serializada y que solo es deserializable a través de una librería interna de CloudCompare.

Apéndice B

Código Python

En este apéndice se adjunta el código R empleado en la elaboración de los resultados obtenidos en este trabajo. Las librerías empleadas en su elaboración se describen aquí:

- **numpy** Librería extremadamente popular de Python para el manejo de datos numéricos a través de *arrays*. Su nombre es una abreviatura de *numerical Python*.
- **pandas** Otra de las librerías más populares de Python. Esta es empleada de manera principal, por su parte, para trabajar con datos tabulares. Su nombre es una apreviatura de *panel data*.
- *jakteristics* Librería específicamente diseñada para computar descriptores geométricos a partir de nubes de puntos tridimensionales.
- *timeit* Librería empleada para medir el tiempo de ejecución del código relacionado con el cómputo de los descriptores geométricos.

A partir de este punto, se adjunta el código Python empleado.

Voxelización Función empleada para voxelizar las nubes de puntos a una cierta resolución. La función permite elegir una resolución diferente para cada eje, aunque se ha empleado 5 *cm* para todos. Incluye también el código para generar un indicador de qué puntos se corresponden con qué vóxel y *vice versa*.

```
def voxelizar(nube_, ladoXY = 0.05, ladoZ = 0.05,
        n_digitos_cada_coord = 5,
        Campo_X = 0, Campo_Y = 1, Campo_Z = 2,
        nube_voxel_XYZnPtos = True):

    t = timeit.default_timer()

    min_nube = np.min(nube_[:, [Campo_X, Campo_Y, Campo_Z]], axis = 0)

    nube_[:,Campo_X]=nube_[:,Campo_X]-min_nube[Campo_X];
    nube_[:,Campo_Y]=nube_[:,Campo_Y]-min_nube[Campo_Y];
    nube_[:,Campo_Z]=nube_[:,Campo_Z]-min_nube[Campo_Z];

    elapsed = timeit.default_timer() - t

  # print("%.2f" % elapsed,'s: escalado y trasladado')
```

```
    codigo = np.floor(nube_[:,2]/ladoZ)*10**
        (n_digitos_cada_coord*2) +     np.floor(nube_[:,1]/ladoXY)*10**
        n_digitos_cada_coord +
        np.floor(nube_[:,0]/ladoXY)

    elapsed = timeit.default_timer() - t

  # print("%.2f" % elapsed,'s: codificado')

    ind_orden_voxel = np.argsort(codigo)

    elapsed = timeit.default_timer() - t

  # print("%.2f" % elapsed,'s: 1a ordenacion')

    ind_orden_voxel_inverso = np.argsort(ind_orden_voxel)

    codigo=codigo[ind_orden_voxel]

    elapsed = timeit.default_timer() - t

  # print("%.2f" % elapsed,'s: 2a ordenacion')

    codigos_unicos, idPrimerPtoVoxel,
        inverseId, ptos_en_cada_voxel =
        np.unique(codigo, return_index=True,
            return_inverse=True, return_counts=True)

    elapsed = timeit.default_timer() - t

  # print("%.2f" % elapsed,'s: valores unicos extraidos')

    corresp_voxel_nube = inverseId[ind_orden_voxel_inverso]

    corresp_nube_voxel = ind_orden_voxel[idPrimerPtoVoxel]

    nubevoxel = np.zeros((np.size(codigos_unicos,0),3))

    nubevoxel[:,2]=np.floor(codigos_unicos/10**(n_digitos_cada_coord*2))

    nubevoxel[:,1] = np.floor(
        (codigos_unicos-np.floor(
        codigos_unicos/10**(n_digitos_cada_coord*2))*10**
        (n_digitos_cada_coord*2))/10**n_digitos_cada_coord)

    nubevoxel[:,0] = codigos_unicos-np.floor(
        codigos_unicos / 10 **
        (n_digitos_cada_coord*2)) * 10 **
        (n_digitos_cada_coord*2) - np.floor(
        (codigos_unicos-np.floor(codigos_unicos / 10 **
        (n_digitos_cada_coord*2))*10**
        (n_digitos_cada_coord*2))/10**
```

```
        n_digitos_cada_coord)*10**
        n_digitos_cada_coord

    elapsed = timeit.default_timer() - t

  # print("%.2f" % elapsed,'s: codigo descompuesto')

    nubevoxel[:,2]=nubevoxel[:,2]*ladoZ+min_nube[2]+ladoZ/2

    nubevoxel[:,1]=nubevoxel[:,1]*ladoXY+min_nube[1]+ladoXY/2

    nubevoxel[:,0]=nubevoxel[:,0]*ladoXY+min_nube[0]+ladoXY/2


    if nube_voxel_XYZnPtos==True:
        nubevoxel =
            np.append(nubevoxel,ptos_en_cada_voxel[:,np.newaxis],axis=1)

    elapsed = timeit.default_timer() - t

  # print("%.2f" % elapsed,'s:
        reescalado y trasladado al origen')

    nube_[:,Campo_X]=nube_[:,Campo_X]+min_nube[Campo_X];

    nube_[:,Campo_Y]=nube_[:,Campo_Y]+min_nube[Campo_Y];

    nube_[:,Campo_Z]=nube_[:,Campo_Z]+min_nube[Campo_Z];

    return nubevoxel,corresp_voxel_nube,corresp_nube_voxel
```

Cómputo de los descriptores geométricos Función para computar los descriptores geométricos. Usa internamente la función *compute_features* de la librería jakteristics.

```
def calculate_geometric_features(xyz_array, scales):

    geom_features = np.empty((0, xyz_array.shape[0]))

    for i in scales:

        geom_features_at_scale = compute_features(
            xyz_array, search_radius=i).T

        geom_features = np.append
            (geom_features, geom_features_at_scale, axis=0)

    geom_features = geom_features.T

    return geom_features
```

Apéndice C

Código R

En este apéndice se adjunta el código R empleado en la elaboración de los resultados obtenidos en este trabajo. Los paquetes empleados en su elaboración son los que se describen a continuación:

- **tidyverse.** Metapaquete que contiene una variedad de paquetes empleados en la lectura, escritura, manipulación, limpieza y visualización de datos, entre otras muchas funcionalidades. En este trabajo se han empleado las siguientes funciones de tidyverse:
  - *%>%*: Operador que emplea como primer argumento de la función que se escriba a continuación el objeto precedido por dicho operador.
  - *mutate_at*: Función para crear nuevas variables en un data.frame, incluso a partir de otras ya existentes en él.
  - *arrange*: Función empleada para ordenar las entradas de un data.frame de acuerdo a los valores de una columna.
  - *distinct*: Función que permite seleccionar las filas únicas de un data.frame. Las filas únicas son aquellas que solo aparecen una vez.
  - *left_join*: Función que implementa un *left join*.
  - *inner_join*: Función que implementa un *inner join*.
  - *slice_sample*: Función que permite seleccionar una muestra aleatoria de filas de un data.frame.
  - *drop_na*: Función que elimina las filas de un data.frame que contengan algún NA.
  - *write_csv2*: Función para escribir archivos de extensión .csv de una forma mucho más rápida que con las funciones del paquete base de R equivalentes.
  - *read_csv2*: Similar a la función anterior, pero para leer en vez de escribir archivos.
- **stringi** Paquete con diversas funciones para manipular cadenas de caracteres. Solo se ha empleado la función *stri_split_fixed* para partir cadenas de caracteres de acuerdo a algún patrón.
- **data.table** Paquete para tratar con conjuntos de datos grandes de manera rápida y eficiente. En este trabajo se han empleado las siguientes funciones de data.table:
  - *rbindlist*: Función que une por filas, cuando sea compatible, los elementos de una lista.
  - *fread*: Función para efectuar lecturas de archivos de manera rápida.
- **scales** Paquete con funcionalidades para reescalar datos. En este trabajo solo se ha empleado, de este paquete, la función *rescale*. Esta permite reescalar datos numéricos a un rango deseado.

- **reticulate** Paquete que permite conectar Python con R. Proporciona a la sesión interoperatividad para usar funciones de Python con código R y vice versa, crear celdas de código Python en RMarkdown, importar librerías y scripts de Python, así como acceder a la consola de Python desde RStudio.
- **tensorflow** Paquete que permite acceder a la librería de Tensorflow desde R para implementar modelos de aprendizaje profundo.
- **keras** Paquete que permite acceder al marco de trabajo de Keras-Tensorflow desde R. Keras tiene una sintaxis clara y sencilla que permite crear, entrenar y validar redes neuronales con pocas líneas de código.
- **ramify** Paquete con varias funciones para trabajar con matrices. Solo se ha empleado la función de conveniencia *argmax*, que permite extraer el máximo de cada fila de una matriz. Es especialmente útil cuando se quiere convertir las predicciones dadas por un modelo en forma de probabilidades en unos y ceros.

A continuación, se muestra el código escrito en R que se ha utilizado a lo largo del trabajo.

Lectura de los datos Funciones para cargar los datos de las microparcelas clasificadas a mano.

```
lectura_datos <- function() {

  nombres <- c("Campo_X", "Campo_Y", "Campo_Z")

  carpetas <- list.dirs()[-1]

  n_carpetas <- length(carpetas)

  parcelas <- vector(mode = "list", length = n_carpetas)

  for (i in 1:n_carpetas) {

    setwd(carpetas[i])

    archivos <- list.files()

    n_archivos <- length(archivos)

    tablas <- vector(mode = "list", length = n_archivos)

    etiquetas_pre <- etiquetas <- character(n_archivos)

    for (k in 1:n_archivos) {

      etiquetas_pre[k] <-
        stri_split_fixed(archivos[k], pattern = "_")[[1]][2]

      etiquetas[k] <-
        stri_split_fixed(etiquetas_pre[k], pattern = ".")[[1]][1]

      tablas[[k]] <-
        fread(archivos[k], select = 1:3, col.names = nombres)
```

```
      tablas[[k]][,
        ':='(etiqueta = rep(etiquetas[k], nrow(tablas[[k]])))]
    }

    parcelas[[i]] <- rbindlist(tablas)

    setwd("..")

  }

  names(parcelas) <- paste0("parcela_", as.character(1:n_carpetas))

  return(parcelas)

}
```

Extracción de etiquetas Función que extrae las etiquetas dadas manulmente de las microparcelas, para poder operar después con las coordenadas.

```
saca_etiquetas <- function(x) {

  etiquetas <- vector(mode = "list", length = length(x))

  for (i in 1:length(x)) {

    etiquetas[[i]] <- x[[i]][, 4]

  }

  return(etiquetas)

}
```

Reescalado de datos Función para reescalar las coordenadas de las parcelas y microparcelas de manera que el mínimo sea 0. Esto evita problemas numéricos en la voxelización.

```
reescalar <- function(x) {

  n <- length(x)

  parcelas_reescaladas <- vector(mode = "list", length = n)

  for (i in 1:n) {

    parcelas_reescaladas[[i]] <- x[[i]][, 1:3] %>%

      mutate_at(colnames(x[[i]][, 1:3]), ~
        (rescale(., to = c(0, max(.) - min(.))) %>% as.vector))

  }

return(parcelas_reescaladas)
```

```
}
```

Voxelización y cómputo de predictores Bucle que selecciona las microparcelas de cada parcela y las voxeliza, computa sus descriptores geométricos y calcula los predictores *altura_ total* y *altura_ racha*. Para realizar las mismas operaciones sobre las parcelas, ignorar el código referente a las etiquetas y a los marcadores.

```
carpetas <- c("pinar_1", "pinar_2", "robledal_1", "robledal_2")

n_carpetas <- length(carpetas)

escalas <- c(seq(0.05, 0.95, by = 0.10))

escalas_ch <- c("005", "015", "025", "035", "045",
    "055", "065", "075", "085", "095")

n_escalas <- length(escalas)

carac_nombres <- c('eigenvalue_sum', 'omnivariance', 'eigenentropy',
    'anisotropy', 'planarity', 'linearity', 'PCA1', 'PCA2',
    'surface_variation', 'sphericity', 'nx', 'ny', 'nz')

voxel_nombres <- c("X", "Y", "Z")

for (i in 1:n_carpetas) {

    carpeta <- carpetas[i]

    setwd(paste0("C:/ruta/", carpeta))

    lista_datos <- lectura_datos()

    n <- length(lista_datos)

    etiquetas <- saca_etiquetas(lista_datos)

    lista_datos <- reescalar(lista_datos)

    # Voxelizo las parcelas haciendo uso de codigo Python

    repl_python(input = "py_arrays = [np.asarray(datos) for
        datos in r.lista_datos]")

    repl_python(input = "datos_voxel = [voxelizar(py_array) for
        py_array in py_arrays]")

    repl_python(input = "voxeles, indices_original, indices_vox =
    [], [], []")

    repl_python(input = "voxeles =
        [voxeles + [datos_voxel[d][0][:, 0:3]] for d in range(len(datos_voxel))]")
```

```
repl_python(input = "indices_original" =
    [indices_original + [datos_voxel[d][2]] for d in range(len(datos_voxel))]")

repl_python(input = "indices_vox" =
    [indices_vox + [datos_voxel[d][1]] for d in range(len(datos_voxel))]")

for (j in 1:n_escalas) {

  escala <- escalas[j]

  escala_ch <- escalas_ch[j]

  etiquetas_voxel <- lon_etiquetas_voxel <-
    lon_caracteristicas <- xy_unicos <- tabla <- nombres_X <-
    nombres_y <- cuenta_sucia <- cuenta <- xy_unicos_cuenta <-
    altura_total <- xy_etiqueta <- xy_join <- z_etiqueta <-
    z_unicos <- z_join <- datos_xy <- datos_xyz <- out <-
    stack_size <- datos_completo <- marcador <-
    vector(mode = "list", length = n)

  # Computo los predictores geometricos haciendo uso de codigo Python

  repl_python(input = "voxeles_array =
    [np.asarray(v) for v in voxeles]")

  repl_python(input = "voxeles_array =
    [np.squeeze(v) for v in voxeles]")

  repl_python(input = "caracteristicas =
    [calculate_geometric_features(v, scales = [r.escala]) for
    v in voxeles_array]")

  # Tranformo los datos de Python en datos de R

  caracteristicas <- lapply(py$caracteristicas, as.data.frame)

  voxeles <- lapply(py$voxeles, as.data.frame)

  for (k in 1:n) {

    colnames(caracteristicas[[k]]) <- carac_nombres

    colnames(voxeles[[k]]) <- voxel_nombres

  }

  for (k in 1:n) {

    etiquetas_voxel[[k]] <-
        unlist(etiquetas[[k]])[unlist(py$indices[[k]])]
```

```
  etiquetas_voxel[[k]] <- factor(etiquetas_voxel[[k]])

  etiquetas_voxel[[k]] <- data.frame(etiqueta =
      as.numeric(etiquetas_voxel[[k]]))

}

for (k in 1:n) {

  lon_etiquetas_voxel[[k]] <- nrow(etiquetas_voxel[[k]])

  lon_caracteristicas[[k]] <- nrow(caracteristicas[[k]])

}

lon_etiquetas_voxel <- unlist(lon_etiquetas_voxel)

lon_caracteristicas <- unlist(lon_caracteristicas)


for (k in 1:n) {

  if (lon_etiquetas_voxel[k] != lon_caracteristicas[k]) {

    etiquetas_voxel[[k]] <- data.frame(etiqueta =
      rbind(etiquetas_voxel[[k]][1, ], etiquetas_voxel[[k]]))

  }

  else {

    etiquetas_voxel[[k]] <- etiquetas_voxel[[k]]

  }

}

# Creo una matriz en la que solo hay las combinaciones X, Y unicas,
# para despues calcular cuantos voxeles hay en esa columna.

for (k in 1:n) {

  xy_unicos[[k]] <- voxeles[[k]] %>% distinct(X, Y) %>%
    arrange(X, Y, .by_group = T)

  tabla[[k]] <- table(voxeles[[k]][, 1:2])

  nombres_X <- paste0(rep("X_", nrow(tabla[[k]])),
      rownames(tabla[[k]]))

  nombres_Y <- paste0(rep("Y_", ncol(tabla[[k]])),
      colnames(tabla[[k]]))
```

```
}

# Aqui saco la cuenta de cuantos voxeles hay en cada columna y calculo
# el predictor altura_total

for (k in 1:n) {

  cuenta_sucia[[k]] <- as.vector(matrix(t(tabla[[k]]),
      ncol = 1, byrow = T))

  cuenta[[k]] <- cuenta_sucia[[k]][cuenta_sucia[[k]] > 0]

  xy_unicos_cuenta[[k]] <- cbind(xy_unicos[[k]], cuenta[[k]])

  altura_total[[k]] <- voxeles[[k]] %>%
    inner_join(xy_unicos_cuenta[[k]], by = c("X" = "X", "Y" = "Y"))

  colnames(altura_total[[k]]) <- c("X", "Y", "Z", "cuenta")

}

# Aqui calculo el predictor altura_racha

for (k in 1:n) {

  xy_etiqueta[[k]] <- 1:nrow(xy_unicos[[k]])

  xy_join[[k]] <- cbind(xy_unicos[[k]], xy_etiqueta[[k]])

  z_unicos[[k]] <- voxeles[[k]] %>% distinct(Z)

  z_etiqueta[[k]] <- 1:nrow(z_unicos[[k]])

  z_join[[k]] <- cbind(z_unicos[[k]], z_etiqueta[[k]])

}

for (k in 1:n) {

  datos_xy[[k]] <- voxeles[[k]] %>%
      left_join(xy_join[[k]], by = c("X" = "X", "Y" = "Y")) %>%
          arrange(X, Y, .by_group = T)

  datos_xyz[[k]] <- datos_xy[[k]] %>%
      left_join(z_join[[k]], by = c("Z" = "Z"))

  out[[k]] <- split(datos_xyz[[k]]$z_etiqueta,
      cumsum(c(TRUE, abs(diff(datos_xyz[[k]]$z_etiqueta)) != 1)))

  altura_racha[[k]] <- data.frame(altura_racha =
      rep(lengths(out[[k]]), lengths(out[[k]])))
```

```
}

# Asigno un marcador a los datos para saber de que microparcela son.
# Esto facilita la posterior validacion cruzada.

for (k in 1:n) {

  marcador[[k]] <-
      as.data.frame(rep(k, nrow(caracteristicas[[k]])))

  colnames(marcador[[k]]) <- "marcador"

}

# Union de los todos los datos: el marcador, las etiquetas,
# los predictores creados y los descriptores geometricos.

for (k in 1:n) {

  datos_completo[[k]] <- cbind(marcador[[k]],
                               etiquetas_voxel[[k]],
                               altura_total[[k]],
                               altura_racha[[k]],
                               caracteristicas[[k]])

}

# Empato las listas en un unico data.frame.

datos_parcela <- rbindlist(datos_completo)

# Arreglo nombres

colnames(datos_parcela[, 1:2]) <- c("marcador", "etiqueta")

# Guardo los datos

setwd(paste0("C:/Ruta/", carpeta, "/", escala_ch))

fwrite(datos_parcela,
  paste0("datos_", carpeta, "_", escala_ch, ".csv"))

# Entre parcela y parcela borro los datos de la anterior para
# liberar espacio en memoria.

rm("etiquetas_voxel", "lon_etiquetas_voxel", "lon_caracteristicas",
  "xy_unicos", "tabla", "nombres_X", "nombres_y", "cuenta_sucia",
  "cuenta", "xy_unicos_cuenta", "altural_total", "xy_etiqueta",
  "xy_join", "z_etiqueta", "z_unicos", "z_join", "datos_xy",
  "datos_xyz", "out", "altura_racha", "datos_completo",
  "marcador", "datos_parcela")
```

```
    }

}
```

Red neuronal Función para generar la red neuronal con la arquitectura especificada en el Capítulo 3. Usa la sintaxis de Keras.

```
red_neuronal <- function(d) {

  modelo <- keras_model_sequential()

  # la version de 3 categorias es igual pero con 4 unidades en la capa de
  # salida creada con la ultima llamada a layer_dense().
  modelo %>%
    layer_dense(units = 500, activation = "relu",
        input_shape = c(d), kernel_initializer='normal') %>%
    layer_batch_normalization() %>%
    layer_dense(units = 500, activation = 'relu',
        kernel_initializer='normal') %>%
    layer_batch_normalization() %>%
    layer_dense(units = 500, activation = 'relu',
        kernel_initializer='normal') %>%
    layer_batch_normalization() %>%
    layer_dense(units = 500, activation = 'relu',
        kernel_initializer='normal') %>%
    layer_batch_normalization() %>%
    layer_dense(4, activation = 'softmax',
        kernel_initializer = 'normal')

  modelo %>% compile(
    loss = "categorical_crossentropy",
    optimizer = 'adam',
    metrics= c('accuracy')
  )

  return(modelo)

}
```

Análisis de sensibilidad de la escala Código con el que se realiza la validación cruzada para cada escala y para cada parcela.

```
escalas <- c(seq(0.05, 0.95, by = 0.10))

escalas_ch <- c("005", "015", "025", "035", "045", "055",
    "065", "075", "085", "095")

n_escalas <- length(escalas)

b_size <- 1500
```

```
epocas <- 15

# Primer bucle: Parcela

for (i in 1:n) {

  parcela <- parcelas[i]

  # Segundo bucle: Escala

  for (k in 1:length(escalas_ch)) {

    escala <- escalas_ch[k]

    setwd(paste0("C:/TFM/Datos/", parcela, "/", escala))

    datos <- fread(paste0("datos_", parcela, "_", escala, ".csv"))

    d <- ncol(datos) - 5

    m <- length(unique(datos$marcador))

    datos <- drop_na(datos)

    score <- vector(mode = "list", length = m)

    marcador <- paste0(rep("marcador_", m), 1:m)

    # Tercer bucle: Validacion cruzada

    for (j in 1:m) {

      test  <- datos[datos$marcador == j, -1]

      train <- datos[datos$marcador != j, -1]

      X_train <- normalize(as.matrix(train[, -(1:4)]))

      X_test  <- normalize(as.matrix(test[, -(1:4)]))

      Y_train <- train[,  1]

      Y_train <- to_categorical(Y_train)[,-1]

      Y_test <- test[, 1]

      Y_test <- to_categorical(Y_test)[, -1]

      setwd(paste0("C:/ruta/", parcela, "/", escala))

      modelo <- red_neuronal(d)
```

```
      historia <- modelo %>% fit(
        X_train, Y_train,
        epochs = epocas,
        batch_size = b_size,
        validation_split = 0.2
        )

      score[[j]] <- modelo %>% evaluate(X_test, Y_test,
        batch_size = b_size)

      score[[j]] <- c(score[[j]], historia[[2]][[2]][epocas])

      predicciones <- modelo %>% predict(X_test)

      predicciones_int <- as.factor(argmax(predicciones))

      parcela_etiquetada <-
        cbind(test[, 2:4], test[, 1], predicciones_int)

      colnames(parcela_etiquetada) <-
        c("X", "Y", "Z", "Observado", "Prediccion")

      write_csv2(as.data.frame(parcela_etiquetada),
        paste0(parcela, "_", escala, "_", marcador[j], ".csv"))

      rm(X_test, Y_test, X_train, Y_train, modelo, parcela_etiquetada)
    }

    score <- as.matrix(bind_rows(score))

    setwd(paste0("C:/ruta/", parcela))

    write.table(cbind(1:m, score),
        paste0("precision_", parcela, "_", escala, ".txt"))

  }

}
```

Análisis de sensibilidad II Código con el que se realiza la validación cruzada para cada combinación de predictores y cada parcela, ya fijada la escala obtenida con la validación anterior.

```
combinaciones <- list(A = c(1:2, 6:21),
                      B = c(1:2, 8:21),
                      C = c(1:2, 6:7, 11:12, 14:21),
                      D = c(1:2, 6:18))

n_combinaciones <- length(combinaciones)

combinaciones_letra <- LETTERS[1:4]

for (i in 1:n_parcelas) {
```

```
parcela <- parcelas[i]

setwd(ruta_datos)

datos_completo <- read_csv(paste0("datos_", parcela, "_045.csv"))

datos_completo <- drop_na(datos_completo)

for (j in 1:n_combinaciones) {

  combinacion <- combinaciones[[j]]

  letra <- combinaciones_letra[j]

  datos <- datos_completo[, combinacion]

  d <- ncol(datos) - 2

  m <- length(unique(datos$marcador))

  score <- vector(mode = "list", length = m)

  marcador <- paste0(rep("marcador_", m), 1:m)

  for (k in 1:m) {

    setwd(paste0("C:/ruta/", parcela, "/", letra))

    test  <- datos[datos$marcador == j, -1]

    train <- datos[datos$marcador != j, -1]

    X_train <- normalize(as.matrix(train[, -1]))

    X_test  <- normalize(as.matrix(test[, -1]))

    Y_train <- train[,  1]

    Y_train <- to_categorical(Y_train)[,-1]

    Y_test <- test[, 1]

    Y_test <- to_categorical(Y_test)[, -1]

    modelo <- red_neuronal(d)

    historia <- modelo %>% fit(
      X_train, Y_train,
      epochs = epocas,
      batch_size = b_size,
      validation_split = 0.2
```

```
        )

      score[[j]] <- modelo %>%
        evaluate(X_test, Y_test, batch_size = b_size)

      score[[j]] <- c(score[[j]], historia[[2]][[2]][epocas])

      predicciones <- modelo %>% predict(X_test)

      predicciones_int <- as.factor(argmax(predicciones))

      parcela_etiquetada <-
        cbind(test[, 2:4], test[, 1], predicciones_int)

      colnames(parcela_etiquetada) <-
        c("X", "Y", "Z", "Observado", "Prediccion")

      write_csv2(as.data.frame(parcela_etiquetada),
        paste0(parcela, "_", letra, "_", marcador[k], ".csv"))

      rm(X_test, Y_test, X_train, Y_train, modelo,
        parcela_etiquetada, historia)

    }

    score <- as.matrix(bind_rows(score))

    setwd(paste0("C:/ruta/", parcela))

    write.table(cbind(1:m, score),
        paste0("precision_", parcela, "_", letra, ".txt"))
  }

}
```

Clasificación de la parcela completa Código para entrenar un modelo con la mejor combinación de predictores en la mejor escala y etiquetar la parcela completa correspondiente.

```
B = c(2, 8:21)

for (i in 1:n_parcelas) {

  parcela <- parcelas[i]

  setwd(paste0(ruta_datos, parcela))

  datos <- fread(paste0("datos_", parcela, "_025.csv"))

  datos <- drop_na(datos)

  datos <- datos[, ..B]
```

```
# para el modelo de 3 categorias
# datos[datos$etiqueta == 3, 1] <- 1
# datos[datos$etiqueta == 4, 1] <- 3

X <- as.matrix(datos[, -1])

X <- normalize(X)

Y <- datos[,  1]

Y <- to_categorical(Y)[,-1]

d <- ncol(X)

modelo <- red_neuronal(d)

historia <- modelo %>% fit(
  X, Y,
  epochs = epocas,
  batch_size = b_size
  )

# Cargado del bosque ya voxelizado y con los predictores calculados #

parcela_completa <-
  as.data.frame(t(fread(paste0(parcela, "_voxelizado_025.csv"))))

parcela_completa <- drop_na(parcela_completa)

caracteristicas <- parcela_completa[, 6:19]

caracteristicas <- normalize(as.matrix(caracteristicas))

# Prediccion de las categorias por el modelo #

t <- Sys.time()

prediccion <- modelo %>% predict(caracteristicas)

prediccion_int <- argmax(prediccion)

t_etiquetado[i] <- Sys.time() - t

print(paste0("tiempo etiquetado parcela ",
  parcela, ": ", t_etiquetado[i], " s"))

# Guardado de tales predicciones #

setwd(ruta_resultados)

parcela_etiquetada <- cbind(parcela_completa[, 1:3], prediccion_int)
```

```
  fwrite(as.data.frame(parcela_etiquetada),
    paste0(parcela, "_vox_etiquetado_045_4cat.csv"))

  # Guardado del modelo #

  save_model_hdf5(modelo, paste0("modelo_", parcela, "_045_4cat.h5"))

  # Tiempos de etiquetado #

  write.table(t_etiquetado[i],
    paste0("t_etiquetado_", parcela, "_045_4cat.txt"))

  rm(parcela, datos, X, Y, modelo, parcela_etiquetada, historia,
    parcela_completa, caracteristicas, prediccion, prediccion_int, t)

}
```

Vuelta a la nube original El código siguiente asigna a los puntos de la nube original las predicciones obtenidas para los vóxeles.

```
nube_original_ind <- cbind(parcela_completa, unlist(indices_original))

nube_vox_pred_ind <- cbind(parcela_etiquetada, unlist(indices_vox))

colnames(nube_original_ind) <- c("X", "Y", "Z", "ind")

colnames(nube_vox_pred_ind) <- c("X", "Y", "Z", "pred", "ind")

parcela_con_pred <- nube_original_ind %>%
  inner_join(nube_vox_ind, by = c("ind" = "ind")) %>%
  select("X", "Y", "Z", "pred")
```

Bibliografía

[1] Allaire JJ, Ushey K,Tang Y, Eddelbuettel D (2017) reticulate: R Interface to Python.

[2] Beech E, Rivers M, Oldfield S, Smith PP (2017) Global Tree Search: the first complete global database of tree species and country distributions. Journal of Sustainable Forestry, 36(5):454–489.

[3] Brown S, Gillespie AJR, Lugo AE (1989) Biomass estimation methods for tropical forests with applications to forest inventory data. Forest Science 35:881-902.

[4] CABI (2021) Bursaphelenchus xylophilus (pine wilt nematode) datasheet. In Invasive Species Compendium. CABI, Wallingford.

[5] Cabo C, Ordóñez C, Sáchez-Lasheras F, Roca-Pardiñas J, Cos-Juez AJ (2019) Multiscale Supervised Classification of Point Clouds with Urban and Forest Applications. Sensors 19(20):4523. Basilea.

[6] Cheng L, Chen S, Liu X, Xu H, Wu Y, Li M, Chen Y (2018) Registration of Laser Scanning Point Clouds: A Review. Sensors 18(5):1641-1666.

[7] Chollet F (2015) Keras. GitHub.

[8] Chollet F (2017) Deep learning with python. Manning Publications. New York.

[9] CloudCompare (version 2.12, GPL Software). 2021.

[10] Curry HB (1944) The method of steepest descent for non-linear minimization problems. Quarterly of Applied Mathematics, 2(3):258–261.

[11] Dassot M, Constant T, Fournier M (2011) The use of terrestrial lidar technology in forest science: application fields, benefits and challenges. Annals of Forest Science 68(5):959-974.

[12] Deisenroth MP, Faisal AA, Ong CS (2020) Mathematics for Machine Learning. Cambridge University Press. Cambridge.

[13] Demantké J, Mallet C, David N, Vallet B (2011). Dimensionality based scale selection in 3D lidar point clouds. Proceedings of the ISPRS Workshop Laser Scanning. 38:97-102.

[14] FAO (2020) Biodiversity for food and agriculture and ecosystem services. Thematic Study for The State of the World's Biodiversity for Food and Agriculture. Rome.

[15] FAO (2020) Global Forest Resources Assessment 2020: Main report. Rome.

[16] FAO (2021) The State of the World's Land and Water Resources for Food and Agriculture: Systems at breaking point. Synthesis report 2021. Rome.

[17] FAO (2022) State of the World's Forests. Forest pathways for green recovery and building inclusive, resilient and sustainable economies. Rome.

[18] FAO, UNEP (2020) The State of the World's Forests 2020. Forests, biodiversity and people. Rome.

[19] Forest Europe (2015) State of Europe's Forests 2015. Madrid.

[20] Fukushima K. (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybernetics 36:193-202.

[21] Hastie T, Tibshirani R, Friedman JH (2001) The elements of statistical learning: Data mining, inference, and prediction. Springer. New York.

[22] Holdridge L, (1967) Life zone ecology. Tropical Science Center, San Jose.

[23] IPPC Secretariat (2021) Scientific review of the impact of climate change on plant pests: A global challenge to prevent and mitigate plant pest risks in agriculture, forestry and ecosystems. Rome.

[24] Ioffe S and Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on International Conference on Machine Learning 37: 448-456.

[25] IUCN (2022) The IUCN red list of threatened species. Version 2022-1.

[26] Kingma D, Ba J (2014) Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

[27] Lefsky M, Cohen WB, Parker GG, Harding DJ (2002) Lidar Remote Sensing for Ecosystem Studies: Lidar, an emerging remote sensing technology that directly measures the three-dimensional distribution of plant canopies, can accurately estimate vegetation structural attributes and should be of particular interest to forest, landscape, and global ecologists. BioScience 52(1):19-30.

[28] Li C, Xia Y, Yang M, Wu X (2022) Study on TLS Point Cloud Registration Algorithm for Large-Scale Outdoor Weak Geometric Features. Sensors 22(14):5072. Basilea.

[29] Malhi Y, Doughty C, Galbraith D (2011) The allocation of ecosystem net primary productivity in tropical forests. Philosophical transactions of the Royal Society of London. Series B, Biological sciences. 366:3225-3245.

[30] McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5:115–133.

[31] Oh K, Keechul J (2004) GPU implementation of neural networks. Pattern Recognit. 37: 1311-1314.

[32] Ordóñez C, Cabo C, Sanz-Ablanedo E (2017) Automatic Detection and Classification of Pole-Like Objects for Urban Cartography Using Mobile Laser Scanning Data. Sensors 17(7):1465. Basilea.

[33] Orgiazzi A, Bardgett R, Barrios E, Behan Pelletier V, Briones MJI, Chotte JL, De Deyn G (2016) Global Soil Biodiversity Atlas. European Commission, Publications Office of the European Union, Luxembourg.

[34] Oviedo-de la Fuente M, Cabo C, Ordóñez C, Roca J (2021) A Distance Correlation Approach for Optimum Multiscale Selection in 3D Point Cloud Classification. Mathematics 9:1328.

[35] 40. R Core Team (2022) R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna.

[36] Reyes G, Brown S,Chapman J, Lugo AE (1992) Wood densities of tropical tree species. En: USDA Forest Service, General Technical Report SO-88. Southern Forest Experiment Station, New Orleans.

[37] Riaño D, Meier E, Allgöwer B, Chuvieco E, Ustin S (2003) Modeling airborne laser scanning data for the spatial generation of critical forest parameters in fire behavior modeling. Remote Sensing of Environment 86:177-186.

[38] RStudio Team (2022) RStudio: Integrated Development Environment for R. RStudio, PBC, Boston.

[39] Rusu RB (2009) Semantic 3D object maps for everyday manipulation in human living environments. PhD thesis, Computer Science department, Technische Universität München. Múnich.

[40] Sothe C, De Almeida CM, Schimalski MB, La Rosa LEC, Castro JDB, Feitosa RQ, Dalponte M, Lima CL, Liesenberg V, Miyoshi GT, Tommaselli AMG (2020) Comparative performance of convolutional neural network, weighted and conventional support vector machine and random forest for classifying tree species using hyperspectral and photogrammetric data. GIScience and Remote Sensing 57(3):369-394.

[41] van Lierop P, Lindquist E, Sathyapala S, Franceschini G (2015) Global forest area disturbance from fire, insect pests, disease and severe weather events. Forest Ecology and Management 352: 78–88.

[42] Werbos P, John P (1974) Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. Thesis, Harvard University. Cambridge.

[43] Willis KJ (2017) State of the World's Plants 2017. Richmond, Surrey, Kew Publishing.

[44] Wulder M (1998). Optical remote-sensing techniques for the assessment of forest inventory and biophysical parameters. Progress in Physical Geography: Earth and Environment 22(4):449-476.

[45] Zhu XX Tuia D, Mou L, Xia GS, Zhang L, Xu F, Fraundorfer F (2017) Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. IEEE Geoscience and Remote Sensing Magazine 5(4):8-36.