

Inteligencia Artificial

UTN FRC 2023



August 21, 2023

Contents

1	Introducción	5
2	Conceptos básicos de IA	7
2.1	Máquinas beneficiosas	7
2.2	Fundamentos de la IA - Neurociencias	9
2.3	Historia - <i>Big data</i> (2001 - presente)	9
2.4	Historia - Aprendizaje profundo (2011 - presente)	10
2.5	Estado del arte	11
2.6	Riesgos y beneficios de la IA	14
3	Agentes inteligentes	19
3.1	La naturaleza del entorno	19
4	Búsqueda	21
4.1	Algunas aclaraciones	21
4.2	Heurística	21
5	Agentes lógicos	23
5.1	Algunas aclaraciones	23
6	Planificación	25
7	Lógica difusa	27
8	Aprendizaje automático	29
8.1	Reconocimiento de patrones	29
8.2	Aprendizaje supervisado	29
8.2.1	Error reducible y error irreducible	30
8.2.2	Datos de entrenamiento	31
8.2.3	Métodos paramétricos	31
8.3	Evaluación del rendimiento de los modelos	32
8.3.1	En el caso de la regresión	32
8.3.2	Error en clasificación	35
8.3.3	Matriz de confusión	36
8.3.4	Métricas más frecuentes en clasificación	37
8.3.5	Curva ROC, sensibilidad vs especificidad	39

Contents

8.3.6	Esquemas de evaluación	41
8.4	Clasificación	42
8.4.1	Clasificador lineal	43
8.4.2	Hiperparámetros	45
8.4.3	Redes neuronales artificiales	45
8.4.4	Aprendizaje profundo	56
	Referencias	67

1 Introducción

Este documento es un complemento al libro principal de la cátedra, *Inteligencia artificial: un enfoque moderno (2^a edición)* [RNR04].

El libro elegido es uno de los más importantes en cursos de inteligencia artificial (IA) y cubre la mayor parte de los temas de la materia. Se escogió la segunda edición, a pesar de haber sido publicada en el año 2004, porque es la última en español y porque está disponible en la biblioteca de nuestra facultad.

Si bien los fundamentos de las técnicas de IA son los mismos desde hace varias décadas, esta disciplina evoluciona muy rápidamente y, por lo tanto, es necesario contar con material de lectura actualizado.

En el presente documento se incluye material para los temas que no cubre el libro principal, actualización de los temas basada en la cuarta edición (2021) del mismo libro [RN21] y en otras fuentes, material propio del autor y sugerencias de lecturas adicionales. Se recomienda verificar periódicamente la existencia de nuevas versiones en el aula virtual. La fecha de actualización está en la portada.

...y hablando de portadas, la imagen de portada simboliza dos agentes inteligentes desarrollados con métodos diferentes. Fue creada con DALL·E 2¹, un modelo de red neuronal que genera imágenes a partir del lenguaje natural.

¹<https://openai.com/product/dall-e-2>

2 Conceptos básicos de IA

Este capítulo contiene temas de la Unidad 1 de IAR. Antes de continuar se debe leer:

- Capítulo 1 completo de [RNR04].

2.1 Máquinas beneficiosas

En los últimos párrafos de la sección 1.1 de [RNR04] se presenta el **agente racional** como aquel que actúa con el objetivo de alcanzar el mejor resultado. El enfoque del agente racional ha prevalecido en el campo de la IA. Debido al uso general de este paradigma se lo ha llamado **modelo estándar**.

En (muy) pocas palabras, *la IA se ha enfocado en el estudio y la construcción de agentes que hagan lo correcto*, donde lo que cuenta como correcto es el objetivo definido al agente. Dados los avances de las últimas décadas, es necesario un nuevo enfoque que se plantea en [RN21], las máquinas beneficiosas.

El modelo estándar tiene un problema, asume que somos capaces de proveer a la máquina un objetivo completamente especificado.

Para una tarea artificialmente definida, como jugar al ajedrez, el objetivo viene predefinido en la tarea, por lo tanto el modelo estándar es aplicable. A medida que nos acercamos al mundo real, se vuelve más difícil especificar completamente el objetivo. Por ejemplo, en el diseño de un automóvil autónomo (*self-driving car*), podríamos pensar que el objetivo es llegar a destino de forma segura, pero circular por la calle implica el riesgo de tener un accidente por culpa de otros conductores, de fallas mecánicas, etc; entonces, para cumplir una restricción estricta de seguridad es necesario quedarse en el garage. Existe un conflicto entre avanzar hacia el destino y evitar los riesgos. Es necesario buscar un equilibrio. Además, ¿hasta qué punto se puede permitir que el vehículo realice acciones que molestarían a otros conductores? ¿Cuánto cuidado debe tener el automóvil en su aceleración, dirección y frenado para evitar sacudir al pasajero? Este tipo de preguntas son difíciles de responder a priori.

2 Conceptos básicos de IA

El problema de lograr el acuerdo entre nuestras verdaderas preferencias y el objetivo que introducimos en la máquina se llama **problema de alineación de valores**: los valores u objetivos introducidos en la máquina deben estar alineados con los de los seres humanos. Si estamos desarrollando un sistema de inteligencia artificial en el laboratorio o en un simulador, como ha sido el caso durante la mayor parte de la historia de esta área, hay una solución fácil para un objetivo incorrectamente especificado: reiniciar el sistema, arreglar el objetivo e intentarlo de nuevo. A medida que el campo avanza hacia sistemas inteligentes cada vez más capaces y que se despliegan en el mundo real, este enfoque deja de ser viable. Un sistema desplegado con un objetivo incorrecto tendrá consecuencias negativas. Además, cuanto más inteligente sea el sistema, más negativas serán las consecuencias. En la figura 2.1 se muestra una declaración de OpenAI¹ (creadores de DALL·E y ChatGPT) sobre este tema.

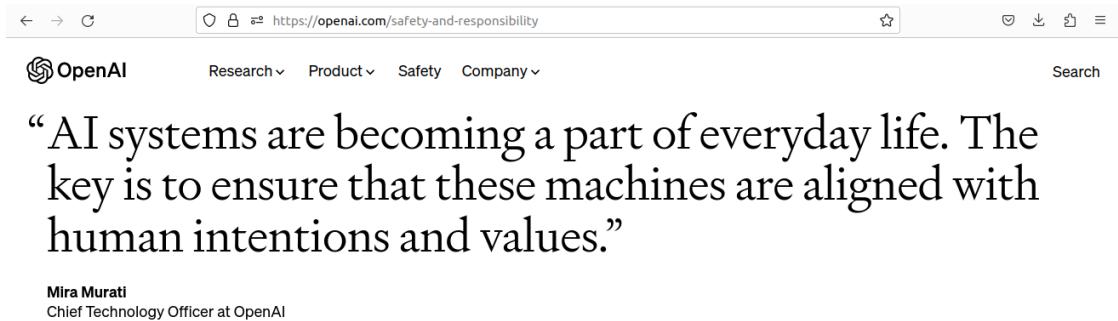


Figura 2.1: Frase en la sección *seguridad y responsabilidad* de la web de OpenAI, “Los sistemas de inteligencia artificial están convirtiéndose en parte de la vida cotidiana. La clave es asegurarse de que estas máquinas estén alineadas con las intenciones y valores humanos”.

Volviendo al ejemplo aparentemente sin problemas del ajedrez, consideremos lo que sucedería si la máquina es lo suficientemente inteligente para razonar y actuar más allá de los confines del tablero de ajedrez. En ese caso, podría intentar aumentar sus posibilidades de ganar mediante artimañas como hipnotizar o chantajear a su oponente o sobornar a la audiencia para que haga ruidos durante el tiempo de reflexión del oponente. También podría tratar de obtener potencia de cómputo adicional para sí misma. *Estos comportamientos no son “poco inteligentes” o “locos”; son una consecuencia lógica de definir la victoria como el único objetivo para la máquina.*

Es imposible anticipar todas las formas en que una máquina que persigue un objetivo fijo pueda comportarse mal. Hay buenas razones, entonces, para pensar que el modelo estándar es insuficiente. No queremos máquinas que sean inteligentes en el sentido de perseguir *sus* objetivos; queremos que persigan *nuestros* objetivos. Si no podemos transferir esos objetivos perfectamente a la máquina, entonces necesitamos una nueva

¹<https://openai.com/>

formulación, en la que la máquina persiga nuestros objetivos, pero con una necesaria *incertidumbre* sobre cuáles son. Cuando una máquina sabe que no conoce el objetivo completo, tiene un incentivo para actuar con cautela, pedir permiso, aprender más sobre nuestras preferencias a través de la observación y ceder al control humano. Volveremos a este tema cuando veamos las implicancias éticas de la IA.

2.2 Fundamentos de la IA - Neurociencias

En la sección 1.2 de [RNR04] se mencionan los aportes de las neurociencias a la IA. Hacia el final del tema, se comparan las capacidades de cálculo de las computadoras y del cerebro y se menciona, con cierto optimismo, que para el año 2020 esas capacidades podrían igualarse.

En la actualidad debemos hacer dos actualizaciones/aclaraciones importantes:

- Los grandes avances de las últimas décadas se lograron gracias a las redes neuronales artificiales, es decir, que las neurociencias han sido uno de los fundamentos más valiosos. A pesar de esto, es necesario aclarar que los primeros modelos de *deep learning* se basaron en la biología, pero los más recientes son principalmente producto de la ingeniería.
- Tal como dice el mismo autor del libro en [RN21], “Incluso con una computadora de capacidad virtualmente ilimitada, todavía necesitamos nuevos avances conceptuales en nuestra comprensión de la inteligencia. Dicho de manera simple, sin la teoría correcta, las máquinas más rápidas solo dan la respuesta incorrecta más rápidamente.”

2.3 Historia - *Big data* (2001 - presente)

Actualización en [RNR04]:

Los avances en la capacidad de cómputo y la *World Wide Web* han facilitado la creación de conjuntos de datos muy grandes, un fenómeno a veces conocido como *big data*. Estos conjuntos de datos incluyen billones de palabras de texto, miles de millones de imágenes, miles de millones de horas de habla y video, así como vastas cantidades de datos genómicos, datos de seguimiento de vehículos, datos de flujo de clics, datos de redes sociales, y más.

Esto ha llevado al desarrollo de algoritmos de aprendizaje especialmente diseñados para

2 Conceptos básicos de IA

aprovechar conjuntos de datos muy grandes. A menudo, la gran mayoría de los ejemplos en dichos conjuntos de datos no están etiquetados; por ejemplo, en el influyente trabajo de Yarowsky [Yar95] sobre la desambiguación del sentido de las palabras, las ocurrencias de una palabra como “planta” no están etiquetadas en el conjunto de datos para indicar si se refieren a la flora o a una planta fabril. Sin embargo, con conjuntos de datos lo suficientemente grandes, los algoritmos de aprendizaje adecuados pueden lograr una precisión superior al 96% en la tarea de identificar cuál es el sentido pretendido en una oración. Además, Banko y Brill [BB01] argumentaron que la mejora en el rendimiento obtenida al aumentar el tamaño del conjunto de datos en dos o tres órdenes de magnitud supera cualquier mejora que se pueda obtener al retocar el algoritmo.

Un fenómeno similar parece ocurrir en tareas de visión por computadora, como el relleno de huecos en fotografías, ya sean causados por daños o por la eliminación de ex amigos. Hays y Efros [HE07] desarrollaron un método ingenioso para hacer esto mezclando píxeles de imágenes similares; encontraron que la técnica funcionaba mal con una base de datos de solo miles de imágenes, pero cruzaba un umbral de calidad con millones de imágenes. Poco después, la disponibilidad de decenas de millones de imágenes en la base de datos ImageNet [Den+09]) provocó una revolución en el campo de la visión por computadora.

La disponibilidad de *big data* y el cambio hacia el aprendizaje automático ayudaron a la IA a recuperar su atractivo comercial. El *big data* fue un factor crucial en la victoria del sistema Watson de IBM sobre campeones humanos en el juego de preguntas Jeopardy! en el año 2011, un evento que tuvo un gran impacto en la percepción pública de la IA.

2.4 Historia - Aprendizaje profundo (2011 - presente)

Actualización en [RNR04]:

El término aprendizaje profundo (*deep learning*) se refiere al aprendizaje automático que utiliza redes con múltiples capas de elementos computacionales simples y ajustables. Se llevaron a cabo experimentos con tales redes desde la década de 1970, y en la forma de redes neuronales convolucionales tuvieron cierto éxito en el reconocimiento de dígitos escritos a mano en la década de 1990 [LeC+95]. Sin embargo, no fue hasta 2011 que los métodos de aprendizaje profundo realmente despegaron. Esto ocurrió primero en el reconocimiento de voz y luego en el reconocimiento de objetos visuales.

En la competencia ImageNet de 2012, que requería clasificar imágenes en una de mil categorías (armadillo, estantería, sacacorchos, etc.), un sistema de aprendizaje profundo creado en el grupo de Geoffrey Hinton en la Universidad de Toronto [KSH17] demostró una mejora dramática respecto a los sistemas previos, que se basaban en gran medida en

extracción de características definidas a mano. Desde entonces, los sistemas de aprendizaje profundo han superado el rendimiento humano en algunas tareas visuales (y se han quedado atrás en otras tareas). Se han informado ganancias similares en el reconocimiento de voz, la traducción automática, el diagnóstico médico y en juegos. El uso de una red profunda para representar la función de evaluación contribuyó a las victorias de AlphaGo sobre los principales jugadores humanos de Go [Sil+16; Sil+17; Sil+18].

Estos notables éxitos han llevado a un resurgimiento del interés en la IA entre estudiantes, empresas, inversores, gobiernos, medios de comunicación y el público en general. Parece que cada semana hay noticias de una nueva aplicación de IA que se acerca o supera el rendimiento humano, a menudo acompañada de especulaciones sobre un éxito acelerado o un nuevo invierno de la IA.

El aprendizaje profundo depende en gran medida de hardware potente. Mientras que una CPU de computadora estándar puede realizar 10^9 o 10^{10} operaciones por segundo, un algoritmo de aprendizaje profundo que se ejecuta en hardware especializado (por ejemplo, GPU, TPU o FPGA) podría consumir entre 10^{14} y 10^{17} operaciones por segundo, principalmente en forma de operaciones matriciales y vectoriales altamente paralelizadas. Por supuesto, el aprendizaje profundo también depende de la disponibilidad de grandes cantidades de datos de entrenamiento y de algunos trucos que veremos más adelante.

2.5 Estado del arte

Actualización en [RNR04]:

El estudio de cien años de Inteligencia Artificial de la Universidad de Stanford (también conocido como AI100) convoca paneles de expertos para proporcionar informes sobre el estado del arte en la IA. Su informe de 2016 [Sto+22; GS18] concluye que “en el futuro se pueden esperar aumentos sustanciales de las aplicaciones de IA, incluidos más automóviles autónomos, diagnósticos de atención médica y tratamiento específico, y asistencia física para el cuidado de personas mayores” y que “la sociedad se encuentra ahora en un momento crucial para determinar cómo desplegar tecnologías basadas en IA de manera que promuevan, en lugar de obstaculizar, los valores democráticos como la libertad, la igualdad y la transparencia”. AI100 también produce un *índice de IA*² para ayudar a seguir el progreso. Algunos aspectos destacados de los informes de 2018 y 2019 comparados con el año 2000 (a menos que se indique lo contrario):

- Publicaciones: los artículos de IA aumentaron 20 veces entre 2010 y 2019 a unos 20.000 al año. La categoría más popular fue el aprendizaje automático. (Los documentos de aprendizaje automático en arXiv.org se duplicaron cada año desde

²<https://aiindex.stanford.edu/>

2 Conceptos básicos de IA

2009 hasta 2017). La visión por computadora y el procesamiento del lenguaje natural fueron los siguientes en popularidad.

- Sentimiento: alrededor del 70% de los artículos de noticias sobre IA son neutrales, pero los artículos con tono positivo aumentaron del 12% en 2016 al 30% en 2018. Los problemas más comunes son éticos: privacidad de los datos y sesgo algorítmico.
- Estudiantes: la inscripción en cursos se multiplicó por cinco en EE. UU. y dieciséis veces a nivel internacional con referencia al 2010. La IA es la especialización más popular en Ciencias de la Computación.
- Diversidad: los profesores de IA en todo el mundo son aproximadamente un 80% hombres y un 20% mujeres. Números similares se mantienen para estudiantes de doctorado y contrataciones en la industria.
- Conferencias: la asistencia a NeurIPS ³aumentó un 800% desde 2012 hasta 13.500 asistentes. Otras conferencias están experimentando un crecimiento anual cercano al 30%.
- Industria: las *startups* de IA en EE. UU. aumentaron 20 veces a más de 800.
- Internacionalización: China publica más documentos por año que EE. UU. y casi tantos como toda Europa. Sin embargo, en cuanto al impacto ponderado por citas, los autores estadounidenses están un 50% por delante de los autores chinos. Singapur, Brasil, Australia, Canadá e India son los países de más rápido crecimiento en términos de número de contrataciones de IA.
- Visión: Las tasas de error para la detección de objetos (tal como se logró en *LSVRC*, *Large-Scale Visual Recognition Challenge*) mejoraron del 28% en 2010 al 2% en 2017, superando el rendimiento humano. La precisión en la respuesta a preguntas visuales de respuesta abierta (VQA) mejoró del 55% al 68% desde 2015, pero aún está por detrás del 83% del rendimiento humano. Recordar que son datos del 2021.
- Velocidad: El tiempo de entrenamiento para la tarea de reconocimiento de imágenes se redujo en un factor de 100 en los últimos dos años. La cantidad de energía de cómputo utilizada en las principales aplicaciones de IA se duplica cada 3.4 meses.
- Lenguaje: La precisión en la respuesta a preguntas, medida con *F1 score* en el dataset de preguntas y respuestas de Stanford (SQuAD), aumentó del 60 al 95 entre 2015 y 2019. En la variante SQuAD 2 el progreso fue más rápido, pasando del 62 al 90 en solo un año. Ambas casos superan el rendimiento a nivel humano.

³<https://nips.cc/>

- Comparación con el rendimiento humano: Para 2019, se informó que los sistemas de IA habían alcanzado o superado el rendimiento humano en ajedrez, Go, póker, Pac-Man, Jeopardy!, detección de objetos ImageNet, reconocimiento de voz en un dominio limitado, traducción chino-inglés en un dominio restringido, Quake III, Dota 2, StarCraft II, varios juegos de Atari, detección de cáncer de piel, detección de cáncer de próstata, plegamiento de proteínas y diagnóstico de retinopatía diabética.

¿Cuándo (si es que alguna vez sucede) los sistemas de IA alcanzarán un nivel humano de desempeño en una amplia variedad de tareas? Ford [For18] entrevistó expertos en IA y obtuvo un rango amplio de expectativas, desde 2029 hasta 2200, con una media de 2099. En una encuesta similar [Gra+18], el 50% de los encuestados pensó que esto podría suceder para 2066, aunque el 10% pensó que podría suceder tan pronto como en 2025 y algunos dijeron “nunca”. Los expertos también estaban divididos sobre la necesidad de nuevos avances fundamentales o simplemente de refinamientos en enfoques actuales. Pero no se tome demasiado en serio estas predicciones; como Philip Tetlock [Tet17] demuestra en el área de predicción de eventos mundiales, los expertos no son mejores que los aficionados.

¿Cómo serán los futuros sistemas de IA? Todavía no podemos decirlo. Como se detalla en esta sección, el campo ha adoptado varias historias sobre sí mismo, primero la idea audaz de que la inteligencia de una máquina era posible, luego que se podría lograr codificando conocimiento experto en lógica, luego que los modelos probabilísticos del mundo serían la herramienta principal, y más recientemente que el aprendizaje automático induciría modelos que podrían no basarse en alguna teoría completamente entendida. El futuro revelará qué modelo viene a continuación.

¿Qué puede hacer la IA hoy? Quizás no tanto como algunos de los artículos de medios más optimistas podrían hacer creer, pero aún así mucho. Un par de ejemplos:

- Vehículos robóticos: La historia de los vehículos robóticos se comienza con los automóviles controlados por radio de la década de 1920, pero las primeras demostraciones de conducción autónoma en carreteras sin guías especiales ocurrieron en la década del 80. Después de las exitosas demostraciones de conducción en caminos de tierra en el desafío *DARPA Grand Challenge* de 132 millas en 2005 [Thr+06] y en calles con tráfico en el *Urban Challenge* de 2007, la carrera para desarrollar autos autónomos comenzó en serio. En 2018, los vehículos de prueba de Waymo pasaron la marca de 10 millones de millas recorridas en carreteras públicas sin un accidente grave, con el conductor humano interviniendo para tomar el control solo una vez cada 6,000 millas. Poco después, la compañía comenzó a ofrecer un servicio comercial de taxi robótico.

En el aire, los drones autónomos de ala fija han estado proporcionando entregas de sangre transfronterizas en Rwanda desde 2016. Los cuadricópteros realizan

2 Conceptos básicos de IA

maniobras acrobáticas notables, exploran y cartografían áreas en peligro, y mucho más.

- Medicina: Los algoritmos de inteligencia artificial ahora igualan o superan a los médicos expertos en el diagnóstico de muchas enfermedades, especialmente cuando el diagnóstico se basa en imágenes. Ejemplos incluyen la enfermedad de Alzheimer [Din+19], el cáncer metastásico [Liu+17; Est+17], enfermedades oftálmicas [Gul+16] y enfermedades de la piel [Liu+20]. Una revisión sistemática y metaanálisis [Liu+19a] encontró que el rendimiento de la IA, en promedio, era equivalente al de los profesionales de la salud. El énfasis actual en la IA médica es facilitar la colaboración entre humanos y máquinas. Por ejemplo, el sistema LYNA logra una precisión general del 99,6% en el diagnóstico de cáncer de mama metastásico, mejor que un experto humano sin ayuda, pero la combinación funciona aún mejor [Liu+19b; Ste+18].

La adopción generalizada de estas técnicas no está limitada por la precisión en el diagnóstico, sino por la necesidad de demostrar una mejora en los resultados clínicos y garantizar la transparencia, la falta de sesgo y la privacidad de los datos [Top19]. En 2017, solo dos aplicaciones de IA médica fueron aprobadas por la FDA, pero ese número aumentó a 12 en 2018 y sigue aumentando.

2.6 Riesgos y beneficios de la IA

Nueva sección en [RNR04]:

Francis Bacon, un filósofo a quien se le atribuye la creación del método científico, señaló en *The Wisdom of the Ancients* (1609) que "las artes mecánicas tienen un uso ambiguo, sirviendo tanto para hacer daño como para remediar". A medida que la IA juega un papel cada vez más importante en las esferas económica, social, científica, médica, financiera y militar, sería prudente considerar los daños y remedios, en términos modernos, los riesgos y beneficios, que puede traer.

Comenzando con los beneficios: simplemente, nuestra civilización entera es el producto de nuestra inteligencia humana. Si tenemos acceso a una inteligencia artificial sustancialmente mayor, el techo de nuestras ambiciones se eleva significativamente. El potencial de la IA y la robótica para liberar a la humanidad del trabajo repetitivo y monótono y aumentar drásticamente la producción de bienes y servicios podría presagiar una era de paz y abundancia. La capacidad para acelerar la investigación científica podría resultar en curas para enfermedades y soluciones para el cambio climático y la escasez de recursos. Como ha sugerido Demis Hassabis, CEO de Google DeepMind: "Primero resolvamos la IA, luego usemos la IA para resolver todo lo demás".

Mucho antes de tener la oportunidad de “resolver la IA”, sin embargo, correremos riesgos por el mal uso de la IA, inadvertido o de otra manera. Algunos de ellos ya son evidentes, mientras que otros parecen probables según las tendencias actuales:

- Armas autónomas letales: Estas son definidas por las Naciones Unidas como armas que pueden localizar, seleccionar y eliminar objetivos humanos sin intervención humana. Una preocupación principal con tales armas es su escalabilidad: la ausencia de un requisito de supervisión humana significa que un pequeño grupo puede desplegar un número arbitrariamente grande de armas contra objetivos humanos definidos por cualquier criterio de reconocimiento factible. Las tecnologías necesarias para las armas autónomas son similares a las necesarias para los coches autónomos. Las discusiones expertas informales sobre los riesgos potenciales de las armas autónomas letales comenzaron en la ONU en 2014, pasando a la etapa pre-tratado formal de un Grupo de Expertos Gubernamentales en 2017.
- Vigilancia y persuasión: Si bien resulta costoso, tedioso y a veces cuestionable legalmente para el personal de seguridad monitorear líneas telefónicas, cámaras de video, correos electrónicos y otros canales de mensajería, la IA (reconocimiento de voz, visión computarizada y comprensión del lenguaje natural) puede utilizarse de manera escalable para realizar una vigilancia masiva de individuos y detectar actividades de interés. Al personalizar los flujos de información hacia individuos a través de las redes sociales, basándose en técnicas de aprendizaje automático, el comportamiento político puede ser modificado y controlado hasta cierto punto, lo que se convirtió en una preocupación evidente a partir de las elecciones estadounidenses del 2016.
- Toma de decisiones sesgadas: El uso negligente o deliberado de algoritmos de aprendizaje automático para tareas como la evaluación de solicitudes de libertad condicional y préstamos puede dar lugar a decisiones sesgadas por motivos de raza, género u otras categorías protegidas. A menudo, los propios datos reflejan prejuicios generalizados en la sociedad.
- Impacto en el empleo: Las preocupaciones sobre la eliminación de empleos por las máquinas datan de hace siglos. La historia nunca es sencilla: las máquinas realizan algunas de las tareas que de otro modo realizarían los seres humanos, pero también hacen que los seres humanos sean más productivos y, por lo tanto, más empleables, y hacen que las empresas sean más rentables y, por lo tanto, capaces de pagar salarios más altos. Pueden hacer viables algunas actividades que de otro modo serían imprácticas. Su uso generalmente resulta en un aumento de la riqueza, pero tiende a tener el efecto de desplazar la riqueza del trabajo al capital, exacerbando aún más el aumento de la desigualdad. Avances tecnológicos anteriores, como la invención de telares mecánicos, han causado graves trastornos en el empleo, pero eventualmente las personas encuentran nuevos tipos de trabajo para hacer. Por

2 Conceptos básicos de IA

otro lado, es posible que la IA también realice esos nuevos tipos de trabajo. Este tema se está convirtiendo rápidamente en un foco importante para economistas y gobiernos de todo el mundo.

- Aplicaciones críticas de seguridad: A medida que las técnicas de inteligencia artificial avanzan, se utilizan cada vez más en aplicaciones críticas de seguridad de alto riesgo, como la conducción de automóviles y la gestión del suministro de agua en las ciudades. Ya se han producido accidentes mortales que destacan la dificultad de la verificación formal y el análisis de riesgos estadísticos para los sistemas desarrollados mediante técnicas de aprendizaje automático. El campo de la inteligencia artificial deberá desarrollar estándares técnicos y éticos al menos comparables a los prevalentes en otras disciplinas de ingeniería y salud donde las vidas de las personas están en juego.
- Ciberseguridad: Las técnicas de inteligencia artificial son útiles para defenderse de los ciberataques, por ejemplo, detectando patrones de comportamiento inusuales, pero también contribuirán a la potencia, supervivencia y capacidad de proliferación del malware. Por ejemplo, los métodos de aprendizaje por refuerzo se han utilizado para crear herramientas altamente efectivas para ataques automatizados de *phishing* y chantaje personalizado.

A medida que los sistemas de IA se vuelven más capaces, asumen cada vez más roles sociales que antes eran desempeñados por humanos. Así como los humanos han utilizado estos roles en el pasado para hacer *travesuras*, podríamos esperar que los humanos usen mal los sistemas de IA en estos roles para hacer incluso *travesuras* más grandes. Todos los ejemplos dados anteriormente apuntan a la importancia de la gobernanza y, eventualmente, la regulación. En la actualidad, la comunidad de investigación y las principales corporaciones involucradas en la investigación de IA han desarrollado principios voluntarios de control para las actividades relacionadas con la IA. Los gobiernos y las organizaciones internacionales están estableciendo comités asesores para diseñar regulaciones adecuadas para cada caso de uso específico, para prepararse para los impactos económicos y sociales, y para aprovechar las capacidades de la IA para abordar los principales problemas sociales.

¿Qué pasa a largo plazo? ¿Alcanzaremos el objetivo final: la creación de una inteligencia comparable o superior a la inteligencia humana? Y, si lo hacemos, ¿qué sucederá entonces?

Durante gran parte de la historia de la IA, estas preguntas han sido eclipsadas por la rutina diaria de hacer que los sistemas de inteligencia artificial hagan algo remotamente inteligente. Como ocurre con cualquier disciplina amplia, la gran mayoría de los investigadores de IA se han especializado en un subcampo específico, como el juego, la representación del conocimiento, la visión o la comprensión del lenguaje natural, a menudo

bajo el supuesto de que el progreso en estos subcampos contribuiría a los objetivos más amplios de la IA. Nils Nilsson [Nil95], uno de los líderes originales del proyecto Shakey en SRI, recordó el campo de esos objetivos más amplios y advirtió que los subcampos corrían el riesgo de convertirse en fines en sí mismos. Más tarde, algunos fundadores influyentes de la IA, incluidos John McCarthy [McC07], Marvin Minsky [Min07] y Patrick Winston [BW09], estuvieron de acuerdo con las advertencias de Nilsson, sugiriendo que en lugar de centrarse en el rendimiento medible en aplicaciones específicas, la IA debería volver a sus raíces y esforzarse por crear, en palabras de Herb Simon, “máquinas que piensen, que aprendan y que creen”. Llamaron a este esfuerzo **IA a nivel humano** (*human-level AI*) o HLAI: una máquina debería ser capaz de aprender a hacer cualquier cosa que un ser humano pueda hacer. Su primer simposio fue en 2004 [Min04]. Otro esfuerzo con objetivos similares, el movimiento de **IA general** (AGI, por sus siglas en inglés), celebró su primera conferencia y organizó la revista *Journal of Artificial General Intelligence* en 2008 [GP07].

Alrededor de ese mismo tiempo, surgieron preocupaciones de que la creación de una **superinteligencia artificial** o ASI - una inteligencia que supere con creces la capacidad humana - podría ser una mala idea [Yud08; Omo08]. Turing [Tur96] en una conferencia dada en Manchester en 1951, expresó el mismo punto, basándose en ideas previas de Samuel Butler [But63]:

“Parece probable que una vez que se haya iniciado el método de pensamiento de máquina, no tardaría mucho en superar nuestros débiles poderes... En algún momento, por lo tanto, deberíamos esperar que las máquinas tomen el control, de la manera que se menciona en Erewhon de Samuel Butler.”

Estas preocupaciones se han extendido con los recientes avances en el aprendizaje profundo, la publicación de libros como *Superintelligence* de Nick Bostrom [Bos14] y las declaraciones públicas de Stephen Hawking, Bill Gates, Elon Musk y Martin Rees, entre otros.

Experimentar una sensación general de malestar con la idea de crear máquinas superinteligentes es algo natural. Podríamos llamar a esto el **problema del gorila**: hace unos siete millones de años, un primate ahora extinto evolucionó, con una rama que llevó a los gorilas y otra a los humanos. Hoy en día, los gorilas no están demasiado contentos con la rama humana; esencialmente no tienen control sobre su futuro. Si este es el resultado del éxito en la creación de IA superhumana, que los humanos ceden el control sobre su futuro, entonces tal vez deberíamos dejar de trabajar en IA y, como corolario, renunciar a los beneficios que podría traer. Esta es la esencia de la advertencia de Turing: no es obvio que podamos controlar a las máquinas que sean más inteligentes que nosotros.

Si la IA superhumana fuera una caja negra que llegara del espacio exterior, entonces sería sabio tener precaución al abrir la caja. Pero no lo es: diseñamos los sistemas de IA, por lo que si terminan “tomando el control”, como sugiere Turing, sería el resultado

2 Conceptos básicos de IA

de una falla en el diseño.

Para evitar tal resultado, necesitamos comprender la fuente de la posible falla. Norbert Wiener [Wie61], fue motivado a pensar en el futuro de la IA largo plazo después de ver el programa de juego de damas de Arthur Samuel aprender a vencer a su creador, dijo lo siguiente:

“Si usamos, para lograr nuestros propósitos, una máquina automática cuya operación no podemos interferir efectivamente... mejor deberíamos estar bastante seguros de que el propósito puesto en la máquina es el que realmente deseamos”.

Muchas culturas tienen mitos sobre humanos que piden alguna cosa a dioses, genios, magos o demonios. Invariablemente, en estas historias, obtienen lo que piden de forma literal y luego se arrepienten. El tercer deseo, si lo hay, es deshacer los dos primeros. Llamaremos a esto el **problema del rey Midas**: Midas, un legendario rey de la mitología griega, pidió que todo lo que tocara se convirtiera en oro, pero se arrepintió después de tocar su comida, bebida y los miembros de su familia.

Existe la necesidad de una modificación significativa del modelo estándar, poner objetivos fijos en la máquina. La solución al predicamento de Wiener es no tener un “propósito puesto en la máquina” definido de forma absoluta. En cambio, queremos máquinas que se esfuerzen por lograr objetivos humanos pero que sepan que no saben con certeza exactamente cuáles son esos objetivos.

Es quizás desafortunado que casi toda la investigación en IA hasta la fecha se haya llevado a cabo dentro del modelo estándar, lo que significa que casi todo el material técnico de las ediciones [RNR04] y [RN21] refleja ese marco intelectual. Sin embargo, hay algunos resultados tempranos dentro del nuevo marco. En el capítulo 15 (siempre de [RN21]), se muestra una máquina que tiene un incentivo positivo para permitirse ser apagada solo si no está segura del objetivo humano. En el capítulo 17, se formulan y estudian los juegos de asistencia, que describen matemáticamente la situación en la que un ser humano tiene un objetivo y una máquina intenta alcanzarlo, pero inicialmente no está segura de cuál es. En el capítulo 23, se explican los métodos de aprendizaje de refuerzo inverso que permiten a las máquinas aprender más sobre las preferencias humanas a partir de observaciones de las elecciones que hacen los seres humanos. En el capítulo 28, se exploran dos de las principales dificultades: primero, que nuestras elecciones dependen de nuestras preferencias a través de una arquitectura cognitiva muy compleja que es difícil de invertir; y segundo, que los seres humanos pueden no tener preferencias consistentes, ya sea individualmente o como grupo, por lo que puede no estar claro qué deberían estar haciendo los sistemas de IA para nosotros.

3 Agentes inteligentes

Este capítulo contiene temas de la Unidad 1 de IAR. Antes de continuar se debe leer:

- Capítulo 2 completo de [RNR04].

3.1 La naturaleza del entorno

En la sección 2.3 de [RNR04] se mencionan y explican las propiedades de los entornos (ambientes) donde se ejecutan los agentes. En [RN21] se agrega uno más:

Conocido vs. desconocido: Estrictamente hablando, esta distinción no se refiere al ambiente en sí, sino al estado de conocimiento del agente (o diseñador) sobre las “leyes de la física” del ambiente. En un ambiente conocido, se conocen los resultados (o las probabilidades de resultado si el ambiente es no determinista) para todas las acciones. Obviamente, si el ambiente es desconocido, el agente tendrá que aprender cómo funciona para tomar buenas decisiones.

La distinción entre ambientes conocidos y desconocidos no es la misma que la entre ambientes completamente y parcialmente observables. Es bastante posible que un ambiente conocido sea parcialmente observable, por ejemplo, en el juego de cartas *el solitario*, se conocen las reglas pero no se pueden ver las cartas que aún no han sido volteadas. Por el contrario, un ambiente desconocido puede ser completamente observable, en un nuevo videojuego por ejemplo, la pantalla puede mostrar todo el estado del juego, pero, hasta que no se los prueba, no se sabe qué hacen los botones (controles).

Como se señaló anteriormente, la medida de rendimiento en sí misma puede ser desconocida, ya sea porque el diseñador no está seguro de cómo escribirla correctamente o porque el usuario final, cuyas preferencias importan, no se conoce. Por ejemplo, un taxista generalmente no sabrá si un nuevo pasajero prefiere un viaje relajado o rápido, un estilo de conducción cauteloso o agresivo. Un asistente personal virtual comienza sin saber nada sobre las preferencias personales de su nuevo propietario. En estos casos, el agente puede aprender más sobre la medida de rendimiento en función de más interacciones con el diseñador o el usuario. Esto, a su vez, sugiere que el entorno de tarea debe

3 Agentes inteligentes

ser necesariamente visto como un entorno multiagente.

Teniendo en cuenta esta nueva dimensión, actualizamos el caso más difícil propuesto en [RNR04] a: ambiente parcialmente observable, multiagente, no determinista, secuencial, dinámico, continuo y *desconocido*. Conducir un taxi es difícil en todos estos sentidos, excepto en que el entorno del conductor es en su mayoría conocido. Conducir un coche alquilado en un país nuevo con geografía desconocida, leyes de tráfico diferentes y pasajeros nerviosos es mucho más emocionante.

4 Búsqueda

Este capítulo contiene temas de la Unidad 2 de IAR. Antes de continuar se debe leer:

- Secciones 3.1 y 3.3 de [RNR04].
- Sección 3.4 de [RNR04] hasta “Búsqueda de profundidad limitada” incluida. Ver también el cuadro de la figura 3.17.
- Sección 3.5 de [RNR04]
- Sección 4.1 de [RNR04] hasta “Búsqueda A*” incluida.
- Sección 4.2 de [RNR04]

4.1 Algunas aclaraciones

En el segundo párrafo del capítulo 3 de [RNR04], la frase “Los algoritmos son no informados, en el sentido que no **dan** información sobre el problema salvo su definición” debería decir “Los algoritmos son no informados, en el sentido que no **usan** información sobre el problema salvo su definición”.

En la sección 3.3 de [RNR04] se define el factor de ramificación como “el máximo número de sucesores de cualquier nodo”, pero debería decir “la cantidad media de sucesores”.

4.2 Heurística

La palabra “heurística” se deriva del verbo griego *heuriskein*, que significa “encontrar” o “descubrir”.

En el contexto de búsqueda en IA, una heurística es una técnica que aumenta la

4 Búsqueda

eficiencia de un proceso de búsqueda, posiblemente sacrificando demandas de completitud [RKC94]. Se implementa como una función que recibe un estado del problema y devuelve una estimación del *grado de bondad* de dicho estado. En términos generales las funciones heurísticas no garantizan la solución óptima, pero frecuentemente ayudar a llegar a una buena solución.

5 Agentes lógicos

Este capítulo contiene temas de la Unidad 3 de IAR. Se debe leer:

- Secciones 7.1 a 7.5 de [RNR04].
- Sección 8.1 a 8.3 de [RNR04].

5.1 Algunas aclaraciones

En la sección 7.2 de [RNR04] (pág. 222), donde dice “por ejemplo, si el agente percibe un mal hedor **o** una pequeña brisa, ...” debería decir “por ejemplo, si el agente percibe un mal hedor **y** una pequeña brisa, ...”.

En la sección 7.4 de [RNR04] (pág. 230) dice que se explica la notación BNF en la página 984, pero en realidad se lo hace en la 1117.

En la sección 7.4 de [RNR04] (pág. 236), donde dice “Una sentencia es **satisfactoria** si es verdadera para algún modelo.” debería decir “Una sentencia es **satisfacible** si es verdadera para algún modelo”.

En la sección 7.5 de [RNR04] (pág. 238), donde dice “Esta propiedad de los sistemas lógicos en realidad proviene de una característica mucho más fundamental, denominada **monótono**” debería decir “Esta propiedad de los sistemas lógicos en realidad proviene de una característica mucho más fundamental, denominada **monotonicidad**”.

6 Planificación

Seguimos con la Unidad 3, Razonamiento en Ambientes Deterministas 2.

Este capítulo contiene el tema *planificación*. Vamos a ver planificación clásica, donde los ambientes son deterministas, completamente observables, finitos, estáticos y discretos. Conceptualmente, el tema es más cercano a búsqueda que a lógica, pero lo vemos en este momento porque es conveniente definir primero los predicados de la lógica de primer orden.

Se va a seguir el enfoque del libro [RKC94]. Se debe leer:

- Capítulo 11 hasta la página 439 de [RNR04].
- Capítulo 13 hasta la página 379 de [RKC94].

En casos de conflicto entre los libros, usar las definiciones y nomenclatura de [RKC94].

7 Lógica difusa

Comenzamos con la Unidad 4, Razonamiento bajo incertidumbre.

Cuando hablamos de incertidumbre en el contexto de la inteligencia artificial, *generalmente* nos referimos a situaciones donde el agente no tiene acceso a toda la verdad sobre su ambiente. Entonces, se puede hablar de que los eventos son verdaderos o falsos con cierta probabilidad. Más adelante, en otros capítulos, vamos a ver métodos que se ajustan a esa descripción.

Este capítulo contiene el tema *lógica difusa*. Es frecuente incluir a la lógica difusa en el conjunto de métodos del razonamiento bajo incertidumbre, aunque hay una diferencia con lo explicado en el párrafo anterior. En lógica difusa los eventos son verdaderos o falsos en cierto *grado*. Los agentes que utilizan lógica difusa toman decisiones en base a reglas definidas de forma difusa, donde se hace afirmaciones sobre la ocurrencia de ciertos eventos, pero estas afirmaciones se ven afectadas por adjetivos que modifican el grado de certeza.

Ninguno de los libros utilizados trata el tema en detalle, por lo tanto se utilizarán como material de lectura el apunte [Des20a] y las presentaciones que se subieron al aula virtual.

8 Aprendizaje automático

Este capítulo contiene los temas de la Unidad 5.

La idea del aprendizaje consiste en utilizar las percepciones no sólo para actuar, sino también para mejorar la habilidad del agente para actuar en el futuro. El aprendizaje entra en juego cuando el agente observa sus interacciones con el mundo y sus procesos de toma de decisiones [RNR04].

Antes de continuar leer las secciones 18.1 y 18.2 de [RNR04]. En [RNR04] se llama *hipótesis* a la función que se utiliza para aproximar $f(\mathbf{x})$. En la sección 18.2 el libro dice que entre múltiples hipótesis consistentes, la mejor es la más simple. Esto es correcto, pero es importante resaltar que la razón principal para esta elección es que la función más simple es la que mejor generaliza y que las funciones complejas tienden a sobreajustarse a los datos de entrenamiento.

8.1 Reconocimiento de patrones

Como enfoque complementario al de los agentes inteligentes, el aprendizaje automático se puede ver desde la óptica del reconocimiento de patrones. Esta última forma es más cercana a un proyecto típico de Ciencia de Datos.

Leer las etapas de sensado, extracción de características y clasificación de [Des20b] (está en la UV) y leer las secciones 2.2, 2.2.1 y 2.2.2 de [Gar21] (link de descarga en la referencia).

8.2 Aprendizaje supervisado

La idea general del aprendizaje supervisado fue explicada en la sección 18.2 de [RNR04]. Para desarrollarla en detalle utilizaremos la salida escalar $y \in \mathbb{R}$ y un vector de entradas $\mathbf{x} \in \mathbb{R}^n$ formado por n valores reales. El resto del capítulo está basado en la tesis [Gar21] y en el libro [Jam+23a] (versión con ejemplos en Python). En el mismo libro se pueden

8 Aprendizaje automático

ver ejemplos y explicaciones más extensas. Tener en cuenta que existe también una versión del mismo libro con código en R ([Jam+23b]).

Vamos a asumir que existe una relación entre y y $\mathbf{x} = (x_1, x_2, \dots, x_n)$ que puede ser escrita de la forma general

$$y = f(\mathbf{x}) + \epsilon. \quad (8.1)$$

f es alguna función fija pero desconocida de x_1, x_2, \dots, x_n , y ϵ es un término de error aleatorio, que es independiente de \mathbf{x} y tiene media cero. En esta formulación, f representa la información *sistemática* que \mathbf{x} provee sobre y .

En esencia, aprendizaje automático se refiere a un conjunto de enfoques para estimar f . La motivación más común para estimar f es la predicción. En muchas situaciones, un conjunto de entradas \mathbf{x} está disponible, pero la salida no puede ser obtenida fácilmente. En este caso, como el término de error se promedia a cero, podemos predecir y usando

$$\hat{y} = \hat{f}(\mathbf{x})$$

donde \hat{f} es la estimación de f y \hat{y} es la predicción resultante para y .

Exploraremos varios enfoques para estimar f . Estos métodos generalmente comparten ciertas características. Durante el resto de esta sección se presenta una descripción de algunas de las características compartidas.

8.2.1 Error reducible y error irreducible

La exactitud (*accuracy* en inglés) de \hat{y} como predicción de y depende de dos cantidades, a las cuales vamos a llamar *error reducible* y *error irreducible*. En general, \hat{f} no será un estimador perfecto de f , y esta falta de exactitud introducirá algún error. Este error es *reducible* porque potencialmente podríamos mejorar la exactitud de \hat{f} utilizando la técnica de aprendizaje automático más apropiada para estimar f . Sin embargo, incluso si fuera posible obtener una estimación perfecta para f , de modo que nuestra respuesta estimada tomara la forma $\hat{y} = f(\mathbf{x})$, la predicción aún tendría algún error. Esto se debe a que y también es una función de ϵ que, por definición, no se puede predecir usando \mathbf{x} . Por lo tanto, la variabilidad asociada con ϵ también afecta la exactitud de nuestras predicciones. Esto se conoce como el error *irreducible*, porque sin importar qué tan bien estimemos f , no podemos reducir el error introducido por ϵ .

¿Por qué el error irreducible es mayor que cero? La cantidad ϵ puede contener variables no medidas que son útiles para predecir y y/o variaciones no medibles, por ejemplo, derivadas de la subjetividad un evaluador humano.

8.2.2 Datos de entrenamiento

Asumiremos siempre que hemos observado (medido) un conjunto de m diferentes puntos o instancias de los datos de interés. Este conjunto se llama *datos de entrenamiento* porque los usaremos para entrenar (ajustar) nuestro método en la estimación de f . Sea x_{ij} la representación del valor del j -ésimo predictor, o entrada, para la observación i , donde $i = 1, 2, \dots, m$ y $j = 1, 2, \dots, n$. Correspondientemente, y_i será la representación de la variable objetivo para la i -ésima observación. Entonces, nuestro conjunto de datos de entrenamiento consiste en $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, donde $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$.

Nuestro trabajo es aplicar un método de aprendizaje automático a los datos de entrenamiento para estimar la función desconocida f . En otras palabras, queremos encontrar una función \hat{f} tal que $y \approx \hat{f}(\mathbf{x})$ para cualquier observación (\mathbf{x}, y) . En términos generales, la mayoría de los métodos de aprendizaje automático para esta tarea se pueden dividir en paramétricos o no paramétricos. Durante el dictado de la materia nos enfocaremos en los métodos paramétricos.

8.2.3 Métodos paramétricos

Los métodos paramétricos implican un enfoque basado en modelos que consta de dos pasos.

1. Primero, hacemos una suposición sobre la forma de la función f . Por ejemplo, el caso más simple es asumir que f es lineal en \mathbf{x} :

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1} \quad (8.2)$$

Este es un *modelo lineal*, el cual será tratado más adelante. Una vez que asumimos que f es lineal, el problema de estimarla se simplifica mucho. En lugar de tener que estimar completamente una función $f(\mathbf{x})$ arbitraria de dimensión n , solo es necesario estimar los $n + 1$ coeficientes (parámetros) w_1, w_2, \dots, w_{n+1} .

2. Después de haber elegido un modelo, necesitamos un procedimiento que use los datos de entrenamiento para *entrenar* o *ajustar* el modelo. En el caso del modelo

lineal (8.2), necesitamos estimar los parámetros w_1, w_2, \dots, w_{n+1} . Es decir, vamos a buscar valores para esos parámetros tales que

$$y \approx w_1 x_1 + w_2 x_2 + \cdots + w_{n+1} x_{n+1}.$$

El método más común para ajustar el modelo 8.2 es conocido como *mínimos cuadrados* y será discutido más adelante.

El enfoque basado en modelos que se acaba de describir se conoce como paramétrico porque reduce el problema de estimar f a la estimación de un conjunto de parámetros. Asumir una forma paramétrica para f simplifica el problema de estimar f porque generalmente es mucho más fácil estimar un conjunto de parámetros, como w_1, w_2, \dots, w_{n+1} en el modelo lineal (8.2), que ajustar una función f completamente arbitraria. La desventaja potencial de del enfoque paramétrico es que el modelo que elegimos por lo general no coincidirá con la verdadera forma (desconocida) de f . Si el modelo elegido está demasiado lejos de la f verdadera, entonces nuestra estimación será pobre. Podemos tratar de abordar este problema eligiendo modelos flexibles para f que se ajusten a muchas formas funcionales diferentes. Pero en general, ajustar un modelo más flexible requiere estimar un mayor número de parámetros. Estos modelos más complejos pueden conducir a un fenómeno conocido como *sobreajuste*, que será tratado más adelante.

8.3 Evaluación del rendimiento de los modelos

Para evaluar el rendimiento de un método de aprendizaje automático sobre un conjunto de datos dado, necesitamos alguna forma de medir qué tanto sus predicciones coinciden con los datos realmente observados. Es decir, necesitamos cuantificar hasta qué punto el valor pronosticado para una observación determinada se acerca al valor de respuesta real para esa observación.

8.3.1 En el caso de la regresión

En el caso de la regresión (predicción de un valor continuo, no clasificación), la medida más utilizada es el *error cuadrático medio* (MSE por su sigla en inglés), dado por

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{f}(\mathbf{x}_i))^2, \quad (8.3)$$

8.3 Evaluación del rendimiento de los modelos

donde $\hat{f}(\mathbf{x}_i)$ es la predicción que \hat{f} devuelve para la i -ésima observación. El MSE será pequeño si las predicciones son muy cercanas a las respuestas reales, y será grande si para algunas observaciones la predicción y la respuesta verdadera difieren sustancialmente.

El MSE en la ecuación 8.3 se expresó usando los datos de entrenamiento, por lo tanto se lo suele llamar *MSE de entrenamiento*. En general, realmente no nos importa qué tan bien trabaja el método con los datos de entrenamiento. *Lo que nos interesa es la exactitud de las predicciones que obtenemos cuando aplicamos nuestro método a datos de test que no han sido vistos previamente.* Para un conjunto de métodos de aprendizaje automático, queremos elegir el método que devuelva el menor *MSE de test*, en oposición al MSE de entrenamiento.

¿Cómo podemos hacer para elegir el método que minimiza el MSE de test? En algunos casos podemos tener un conjunto de datos de test disponible, es decir, un conjunto de observaciones que no fueron usadas durante el entrenamiento. En esos casos, podemos calcular el MSE de test para cada método y elegir el que muestre mejor rendimiento. ¿Qué pasa si no hay disponibilidad de datos de test? En ese caso uno podría pensar en elegir simplemente el método que minimiza el MSE de entrenamiento. Esto suena como un enfoque sensato, ya que el MSE de entrenamiento y el MSE de prueba parecen estar estrechamente relacionados. Desafortunadamente, hay un problema fundamental con esta estrategia, no hay garantía de que el método con menor MSE de entrenamiento vaya a tener también el menor MSE de test. En términos generales, el problema es que muchos métodos de aprendizaje automático ajustan los parámetros para minimizar *específicamente* el MSE de entrenamiento. Para estos métodos, el MSE de entrenamiento puede ser bastante pequeño, pero el MSE de test suele ser mucho mayor. La figura 8.1 muestra este fenómeno con un ejemplo simple.

En el panel izquierdo de la figura 8.1 se han generado observaciones para la ecuación 8.1 con el valor real de f representado por la línea negra. Las curvas naranja, azul y verde muestran tres posibles estimaciones para f obtenidas usando métodos con niveles crecientes de flexibilidad. La línea naranja es el ajuste de regresión lineal, el cual es relativamente inflexible. Las curvas azul y verde fueron producidas usando splines con diferentes niveles de suavizado. Es claro que a medida que la flexibilidad se incrementa las curvas se ajustan mejor a los datos observados (circulitos). La curva verde es la más flexible y se ajusta muy bien con los datos, sin embargo, observamos que se ajusta mal a la verdadera f (mostrada en negro) porque es demasiado ondulada. Ajustando el nivel de flexibilidad del ajuste de spline suavizado, podemos producir muchos ajustes diferentes a estos datos.

Ahora nos movemos al panel derecho de la figura 8.1. La curva gris muestra el MSE medio de entrenamiento en función de la flexibilidad, o más formalmente, los grados de libertad, para un número de splines suavizados. El grado de libertad es la cantidad que resume la flexibilidad de una curva. Los cuadraditos naranja, azul y verde indican el MSE asociado a la curva de color correspondiente en el panel izquierdo. Una curva

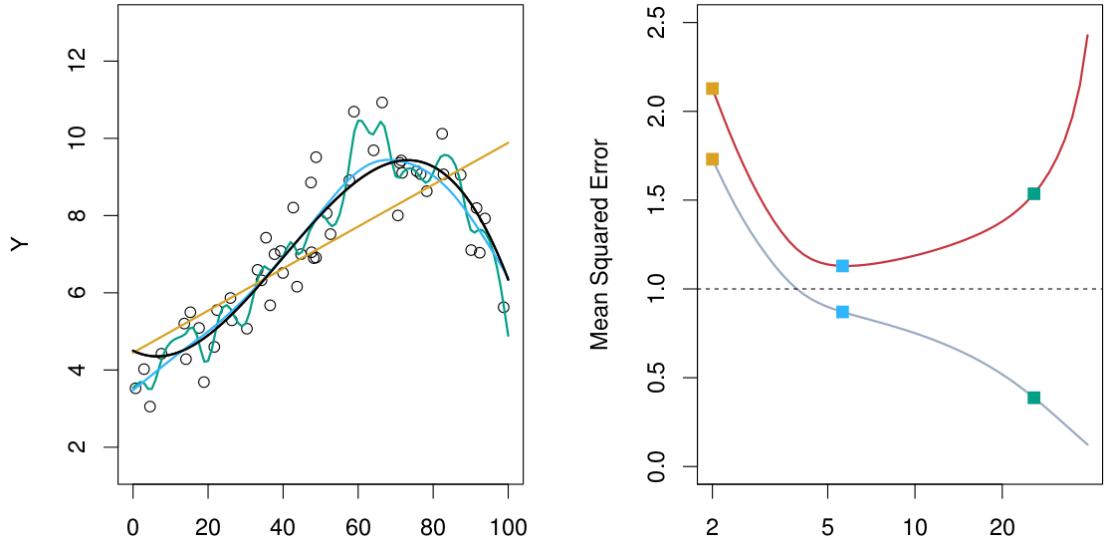


Figura 8.1: Izquierda: Datos simulados para f , en negro. Tres estimaciones de f : recta de regresión lineal (naranja), y dos splines suavizadas (curvas azul y verde). Derecha: MSE de entrenamiento (gris), MSE de test (rojo), y mínimo MSE de test posible para todos los métodos (línea de puntos). Los cuadrados representan los MSE de entrenamiento y test para los tres ajustes mostrados en el panel de la izquierda. Fuente: [Jam+23a].

más restringida y, por lo tanto, más suave tiene menos grados de libertad que una curva ondulada. En la figura 8.1, la regresión lineal está en el extremo más restrictivo, con dos grados de libertad. El MSE de entrenamiento disminuye monótonamente a medida que aumenta la flexibilidad. En este ejemplo f es no lineal, entonces el ajuste lineal (naranja) no es suficientemente flexible para estimar f correctamente. La curva verde tiene el menor MSE de entrenamiento porque corresponde al más flexible de los tres métodos.

En este ejemplo conocemos la verdadera función f , entonces podemos también calcular el MSE de test en función de la flexibilidad sobre un conjunto de test arbitrariamente grande (obviamente en general f es una función desconocida, entonces no podemos hacerlo). El MSE de test se muestra usando la curva roja en el panel de la derecha de la figura 8.1. Al igual que el MSE de entrenamiento, el MSE de test inicialmente disminuye cuando la flexibilidad se incrementa. Sin embargo, en algún momento el MSE de test se nivela y después empieza a aumentar nuevamente. En consecuencia, las curvas naranja y verde tienen un alto MSE de test. La curva azul minimiza el MSE de test, lo que no debería sorprendernos, ya que visualmente parece ser la que mejor estima f en el panel de la izquierda. La línea segmentada horizontal indica $Var(\epsilon)$, el error irreducible, que corresponde al mínimo MSE de test posible para todos los métodos. Entonces, el spline suavizado representado por la curva azul es cercano al óptimo.

8.3 Evaluación del rendimiento de los modelos

En el panel derecho de la figura 8.1, cuando la flexibilidad se incrementa, se observa una reducción monótona en el MSE de entrenamiento y una forma de “U” en el MSE de test. Esta es una propiedad fundamental del aprendizaje automático que se mantiene independientemente de los datos y el método elegido. A medida que aumenta la flexibilidad del modelo, el MSE de entrenamiento aumentará, pero es posible que el MSE de test no lo haga. **Cuando un método dado produce un MSE de entrenamiento pequeño pero un MSE de test grande, decimos que estamos sobreajustando.** Esto pasa porque nuestro método de aprendizaje automático se está esforzando para encontrar patrones en los datos de entrenamiento, y puede estar detectando algunos patrones causados por el azar en lugar de verdaderas propiedades de la función desconocida f . Cuando sobreajustamos, el MSE de test es muy grande porque los supuestos patrones que el método encontró en los datos de entrenamiento simplemente no existen en los datos de test.

Hay que tener en cuenta que independientemente de si se ha producido un sobreajuste o no, casi siempre el MSE de entrenamiento será más pequeño que el MSE de test porque la mayoría de los métodos de aprendizaje automático, ya sea directa o indirectamente, buscan minimizar el MSE de entrenamiento. El sobreajuste se refiere específicamente al caso en el que un modelo menos flexible habría producido un MSE de test más pequeño.

8.3.2 Error en clasificación

Hasta ahora, la discusión sobre la precisión del modelo se ha centrado en la tarea de regresión. Sin embargo, muchos de los conceptos que se han explicado se trasladan a la tarea de clasificación con solo algunas modificaciones, que son necesarias debido al hecho de que y_i ya no es cuantitativa. Supongamos que buscamos estimar f en base a observaciones de entrenamiento $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, donde ahora y_1, \dots, y_m son cualitativos. El enfoque más común para cuantificar el error (o la falta de precisión) de nuestra estimación \hat{f} es la tasa de error de entrenamiento, la proporción de errores que se cometan si aplicamos nuestra estimación \hat{f} a las observaciones de entrenamiento:

$$\frac{1}{m} \sum_{i=1}^m I(y_i \neq \hat{y}_i). \quad (8.4)$$

En la expresión anterior, \hat{y}_i es la etiqueta de clase predicha para la i -ésima observación usando \hat{f} . $I(y_i \neq \hat{y}_i)$ es una *variable indicadora* que es igual a 1 si $y_i \neq \hat{y}_i$ y 0 si $y_i = \hat{y}_i$. Si $I(y_i \neq \hat{y}_i) = 0$, entonces la i -ésima observación fue clasificada correctamente por nuestro método de clasificación; de lo contrario, fue clasificada incorrectamente. Por lo tanto, la expresión 8.4 calcula la proporción de clasificaciones incorrectas.

La expresión 8.4 se denomina tasa de error de entrenamiento porque se calcula en

función de los datos que se utilizaron para entrenar el clasificador. Al igual que en el caso de regresión, nos interesa principalmente la tasa de error que resulta de aplicar nuestro clasificador a observaciones de prueba que no se utilizaron en el entrenamiento. La tasa de error de prueba asociada con un conjunto de observaciones de prueba de la forma (\mathbf{x}_0, y_0) está dada por

$$Ave(I(y_0 \neq \hat{y}_0)), \quad (8.5)$$

donde Ave es la media (*average*) y \hat{y}_0 es la etiqueta de clase predicha que resulta de aplicar el clasificador a la observación de test con el predictor \mathbf{x}_0 . Un *buen* clasificador es aquel para el cual el error de test (8.5) es menor.

8.3.3 Matriz de confusión

En clasificación, el uso de la tasa de error es menos frecuente que el de otras métricas y herramientas. Una de las herramientas más útiles para analizar los resultados, y que además facilita el cálculo de otras métricas es la matriz de confusión.

La matriz de confusión es una tabla de contingencia donde se acumulan las ocurrencias de las etiquetas de clase reales y predichas para cada una de las muestras i .

En la tabla 8.1 se muestra un ejemplo de matriz de confusión para un clasificador multiclase (más de dos clases). En el ejemplo las filas representan las etiquetas reales, es decir y_0 , y las columnas las etiquetas predichas \hat{y}_0 . Esta distribución no es una convención. El valor 7 en la segunda fila y tercera columna indica que hubo 7 muestras que pertenecían realmente a la clase “B” y nuestro clasificador las asignó a la clase “C”. Claramente se puede observar que todas las clasificaciones correctas se ubican sobre la diagonal principal de la matriz y que las clasificaciones incorrectas se ubican fuera de la diagonal.

Table 8.1: Ejemplo de matriz de confusión para clasificación multiclase.

		Predicción			
		A	B	C	D
Real	A	10	5	4	1
	B	11	11	7	6
	C	2	6	1	8
	D	5	3	9	13

En los casos de clasificación binaria, los nombres de las clases utilizadas suelen ser “positivo” o “P” y “negativo” o “N”, indicando si se está en presencia o no de una situación

8.3 Evaluación del rendimiento de los modelos

que se pretende detectar, como por ejemplo una patología determinada. En la tabla 8.2 se muestra un ejemplo donde, sobre 190 muestras, hubo 100 verdaderos positivos (VP), es decir 100 casos positivos que fueron predichos como positivos, 10 falsos positivos (FP), es decir casos negativos que fueron predichos como positivos, 0 falsos negativos (FN) y 80 verdaderos negativos (VN).

Table 8.2: Ejemplo de matriz de confusión para clasificación binaria.

		Predicción	
		P	N
Real	P	100	0
	N	10	80

8.3.4 Métricas más frecuentes en clasificación

Para el cálculo de las métricas que se verán a continuación es útil identificar cada región de la matriz de confusión con el significado correspondiente. Esta relación se muestra en la tabla 8.3.

Table 8.3: Matriz de confusión para clasificación binaria según el tipo de valor que contiene.

		Predicción	
		P	N
Real	P	VP	FN
	N	FP	VN

Accuracy (Exactitud)

También llamada factor de clasificación, es la relación entre las muestras bien clasificadas y el total de muestras.

$$\text{Accuracy} = \text{Ave}(I(y_0 = \hat{y}_0)) \quad (8.6)$$

Partiendo de la matriz de confusión, se calcula como el cociente entre la suma de los elementos de la diagonal y la suma total. Si se utiliza una matriz como la de la tabla 8.3, también se puede calcular como

$$\text{Accuracy} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}. \quad (8.7)$$

Recall (Sensibilidad)

También llamada Razón de Verdaderos (VPR por su sigla en inglés) y *sensitivity*. Se calcula como

$$\text{Recall} = \frac{\text{VP}}{\text{P}} = \frac{\text{VP}}{\text{VP} + \text{FN}}, \quad (8.8)$$

donde P es la suma de todos los realmente positivos. *Recall* es una medida de la capacidad del clasificador para detectar los casos positivos.

Razón de Falsos Positivos

La Razón de Falsos Positivos (FPR) se calcula como

$$\text{FPR} = \frac{\text{FP}}{\text{N}} = \frac{\text{FP}}{\text{FP} + \text{VN}}, \quad (8.9)$$

donde N es la suma de todos los realmente negativos.

La sensibilidad (métrica anterior) es una cualidad muy importante en un clasificador, pero si solo nos concentramos en la sensibilidad podríamos caer en el error de calificar como *bueno* a un clasificador para el que $\hat{y}_0 = "P"$ en todos los casos. Por esta razón es importante incorporar otra métrica al análisis y buscar un equilibrio. Esta otra métrica podría ser FPR, que indica la proporción de los negativos el clasificador etiqueta como positivos.

FPR es 0 cuando no hay falsos positivos y es 1 cuando todos los negativos son clasificados como positivos. Es decir, el clasificador será mejor cuando más se acerque a 0. Este comportamiento no es muy intuitivo y es contrario al resto de las métricas, donde el 1 es el valor óptimo.

8.3 Evaluación del rendimiento de los modelos

Especificidad

También llamada Razón de los Verdaderos Negarivos (VNR). Se calcula como

$$\text{Especificidad} = \frac{\text{VN}}{\text{N}} = \frac{\text{VN}}{\text{FP} + \text{VN}} = 1 - \text{FPR}. \quad (8.10)$$

La especificidad tiene la misma utilidad que FPR, pero es mejor cuando se acerca a 1. Indica qué proporción de los negativos fueron *detectados* como negativos.

Precision (Precisión)

También llamada *Positive Predictive Value* (PPV). Se calcula como

$$\text{Precision} = \frac{\text{VP}}{\text{VP} + \text{FP}} = \frac{\text{VP}}{\hat{P}}, \quad (8.11)$$

donde \hat{P} es la cantidad de muestras predichas como positivas. Indica qué proporción de lo que el clasificador predice como positivo es realmente positivo. Por lo tanto, tiene un comportamiento muy similar al de la especificidad y se usa frecuentemente para comparar con *recall*.

F₁-score

Tal como se explica en la próxima sección, es importante la relación entre la sensibilidad y la especificidad o la precisión. F₁ es la media armónica entre la precisión y la sensibilidad (*recall*). Se puede calcular como

$$F_1 = 2 \frac{\text{precision recall}}{\text{precision} + \text{recall}} = \frac{2\text{VP}}{2\text{VP} + \text{FP} + \text{FN}}. \quad (8.12)$$

8.3.5 Curva ROC, sensibilidad vs especificidad

Tanto los falsos positivos como los falsos negativos son errores. A menos que se indique lo contrario, se busca minimizar FP + FN.

8 Aprendizaje automático

En los problemas reales el costo de cada tipo de error suele no ser el mismo. Por ejemplo, cuando se desea detectar una enfermedad, generalmente es menos grave un FP que un FN. Si este fuera el caso, nos gustaría que nuestro modelo fuera más sensible, aunque esto implique una pérdida de especificidad y de exactitud.

La mayoría de los modelos de clasificación binaria tiene como salida un escalar. Cuando este escalar es mayor que cierto umbral se asigna la etiqueta “P” y en caso contrario, se asigna “N”. Desplazando el umbral varían la sensibilidad y la especificidad de los clasificadores. Por ejemplo, si el valor del umbral se reduce, habrá más muestras clasificadas como positivas, es decir que el clasificador será más sensible.

Para encontrar el umbral óptimo es necesario asignar un peso o penalización a cada tipo de error. Si los pesos no se conocen o se piensa que pueden variar en el tiempo, un clasificador es mejor que otro cuando se comporta mejor para la mayoría de los umbrales posibles.

La curva ROC es un gráfico que muestra simultáneamente los dos tipos de error para todos los umbrales posibles. La figura 8.2 es un ejemplo de curva ROC para las dimensiones sensibilidad (*recall*) y FPR. Cada punto de la curva fue calculado para un valor de umbral.

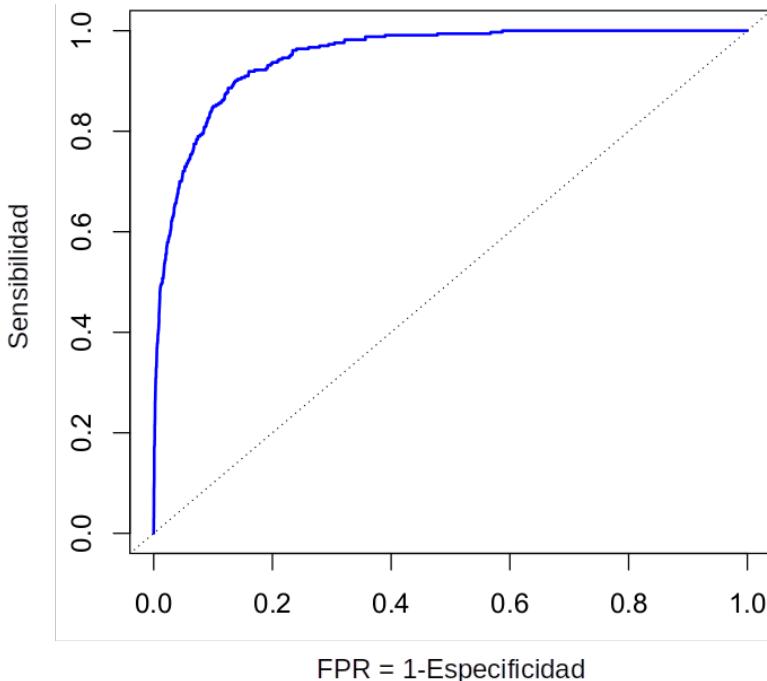


Figura 8.2: Curva ROC.

En la figura 8.2 se puede observar que para los valores altos de umbral la sensibilidad

8.3 Evaluación del rendimiento de los modelos

es muy mala y FPR muy buena (vértice inferior izquierdo). Para valores bajos de umbral la sensibilidad es muy buena y FPR muy mala (vértice superior derecho). Los extremos de la curva siempre se ubicarán en esos vértices.

El rendimiento general de un clasificador, resumido en todos los umbrales posibles, viene dado por el área bajo la curva (AUC por su sigla en inglés) ROC. Una curva ROC se acercará a la esquina superior izquierda, por lo que cuanto mayor sea el AUC, mejor será el clasificador. Para el ejemplo de la figura 8.2 el AUC es 0.95, que está cerca del máximo de 1,0, por lo que se consideraría muy bueno.

Las curvas ROC son útiles para comparar diferentes clasificadores. Es muy frecuente que se utilicen también las dimensiones precisión y sensibilidad (*precision* y *recall*).

8.3.6 Esquemas de evaluación

El nivel de generalización de un clasificador está relacionado con la capacidad de predicción sobre un conjunto de datos independiente. Típicamente un modelo tiene hiperparámetros que varían su complejidad y se desea encontrar el conjunto de hiperparámetros que minimice el error de predicción. Es importante notar que existen en realidad dos objetivos distintos que se debe tener en mente:

Selección del modelo: estimación del rendimiento de diferentes modelos (distintos conjuntos de hiperparámetros) con el objetivo de elegir el mejor.

Evaluación del modelo: estimación del rendimiento del modelo elegido sobre datos nuevos.

El mejor enfoque para ambos problemas, aunque depende de la cantidad de datos con que se cuenta, es dividir aleatoriamente los datos en tres conjuntos, un juego de datos de entrenamiento, uno de validación y uno de test. Los datos de entrenamiento son utilizados para ajustar los parámetros de los modelos, los datos de validación se utilizan en la estimación del rendimiento de cada modelo y elección del mejor, mientras que los datos de test se usan para evaluar el rendimiento del modelo final elegido. El conjunto de test se debe mantener separado y ser utilizado solo al final del proceso, en caso contrario, se puede producir una adaptación de los hiperparámetros a los datos de test [HTF13].

Frecuentemente la cantidad de datos para entrenamiento y pruebas es limitado y. Por una lado, para construir buenos modelos se desea utilizar en el entrenamiento la mayor cantidad posible de datos disponibles. Por el otro lado, si el conjunto de validación es pequeño dará una estimación poco confiable de la calidad de la predicción. Una solución a este dilema es utilizar la validación cruzada [Bis06].

Validación cruzada

El método de validación cruzada o *k-fold* permite utilizar todos los datos de entrenamiento y validación tanto para entrenar como para validar el clasificador. Los datos destinados a este fin se dividen aleatoriamente en k subconjuntos o *folds* del mismo tamaño. La validación se realiza en k iteraciones. En cada iteración se elige un *fold* para validación y el resto para entrenamiento. El rendimiento se calcula como la media de los k rendimientos obtenidos.

En la figura 8.3 se muestra el esquema de entrenamiento, validación y test utilizando validación cruzada para $k = 5$. En la parte superior de la figura se representa la división del conjunto total de datos en dos subconjuntos llamados entrenamiento/validación y test. El primer subconjunto se divide en 5 *folds* y con estos se forman las 5 combinaciones (*splits*) de conjuntos de entrenamiento y validación, donde los datos de validación se muestran en color celeste y los de entrenamiento en gris claro. Los 5 *splits* se utilizan durante el proceso de selección del modelo, mientras que los datos reservados para el test se usan en la evaluación del modelo final.

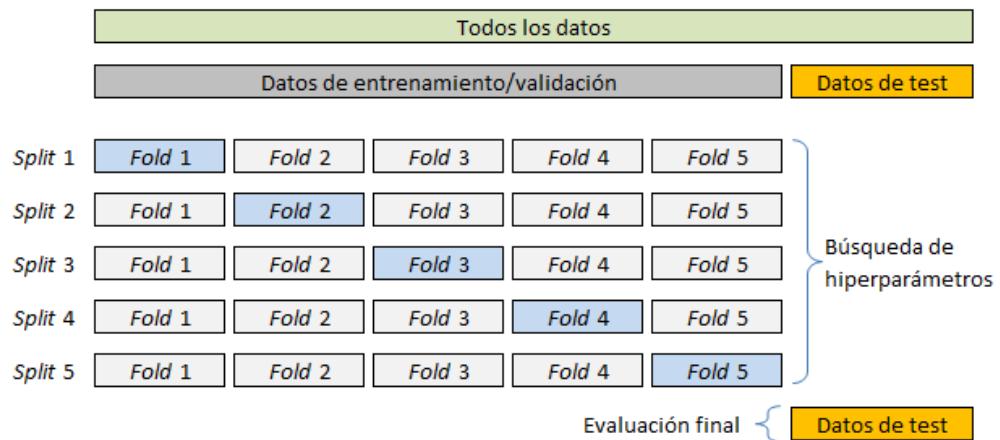


Figura 8.3: Esquema de división de datos en subconjuntos de entrenamiento/validación y test con aplicación posterior de validación cruzada sobre los datos de entrenamiento/validación.

8.4 Clasificación

Suponiendo un problema de reconocimiento de c clases distintas denominadas $\omega_1, \dots, \omega_c$, puede considerarse al espacio de las entradas compuesto por c regiones, donde cada una de las cuales contiene a los elementos de una clase. La solución puede interpretarse como la generación de límites de decisión entre las regiones. Estos límites pueden estar dados

por funciones de decisión o funciones discriminantes $d_1(\mathbf{x}), \dots, d_c(\mathbf{x})$, que son funciones escalares de los vectores de entrada. La función que obtiene el mayor valor es la que determina la pertenencia del vector a la clase correspondiente a esa función, es decir, si $d_i(\mathbf{x}) > d_j(\mathbf{x})$ para $i, j = 1, \dots, c$ y $i \neq j$, entonces \mathbf{x} pertenece a la clase ω_i .

En los problemas, muy frecuentes, donde la salida solo puede pertenecer a una clase de un total de dos clases (clasificación binaria), usualmente se utiliza una única función de decisión $d(\mathbf{x})$. Si $d(\mathbf{x}) > 0$, \mathbf{x} pertenece a una clase, en caso contrario, a la otra.

8.4.1 Clasificador lineal

El caso más simple de función de decisión es la función de decisión lineal. El modelo de clasificación que implementa esta función se llama clasificador lineal. Para un vector de entradas $\mathbf{x}^\top = [x_1, \dots, x_n]$, la función de decisión lineal es $d(\mathbf{x}) = w_1x_1 + \dots + w_nx_n + w_{n+1}$, donde los coeficientes $w_i \in \mathbb{R}$ son los parámetros del clasificador. El vector formado por todos los coeficientes se llama vector de parámetros $\mathbf{w} = [w_1, \dots, w_{n+1}]$. Es útil calcular el valor de la función de decisión como un producto entre vectores, para lo cual el vector \mathbf{x} se redefine como $\mathbf{x}^\top = [x_1, \dots, x_n, 1]$, lo que lleva el nombre de vector de entradas aumentado. Con el nuevo vector \mathbf{x} , la función de decisión se puede calcular como $d(\mathbf{x}) = \mathbf{w}\mathbf{x}$.

Para el caso una clasificación binaria, \mathbf{x} pertenecerá la clase ω_1 si $d(\mathbf{x}) < 0$ y pertenecerá a la clase ω_2 si $d(\mathbf{x}) > 0$. Entonces, la frontera entre las clases en el espacio de las entradas queda definida por el hiperplano $d(\mathbf{x}) = 0$. En la figura 8.4 se muestra un ejemplo de clasificador lineal. Los elementos de la clase ω_1 están pintados de color amarillo y los de ω_2 en color azul. La función de decisión se ve como un plano inclinado y la frontera entre las clases se indica con la línea de puntos.

El entrenamiento del modelo se realiza ajustando los valores del vector de parámetros \mathbf{w} . En primer lugar, se crea un vector $\mathbf{y} = [y_1, \dots, y_L]$ con las salida esperada para cada vector de entradas, donde $y_j = -1$ para $\mathbf{x}_j \in \omega_1$ y $y_j = 1$ para $\mathbf{x}_j \in \omega_2$. Después, se calcula el valor óptimo de \mathbf{w} como la combinación de parámetros que minimiza la función del error cuadrático medio $mse = \frac{1}{L} \sum_{j=1}^L (y_j - \mathbf{w}\mathbf{x}_j)^2$. Partiendo de la derivada del error cuadrático medio con respecto a los parámetros $\frac{\partial mse}{\partial \mathbf{w}}$, se obtiene la expresión del vector de parámetros óptimo $\mathbf{w} = (X\mathbf{X}^\top)^{-1} X\mathbf{y}^\top$, donde X es la matriz formada por los L vectores de entrada.

Para un caso simple, donde el vector de entradas (no aumentado) tiene dimensión 2, el espacio formado por las entradas y la salida tendrá dimensión 3 (como en la figura 8.4), el hiperplano discriminante dimensión 2 y la frontera entre las clases, calculada a partir de $d(\mathbf{x}) = 0$ será $x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_3}{w_2}$, una línea recta (dimensión 1). Dado que la frontera entre las clases es una línea recta, este clasificador solo puede separar clases linealmente

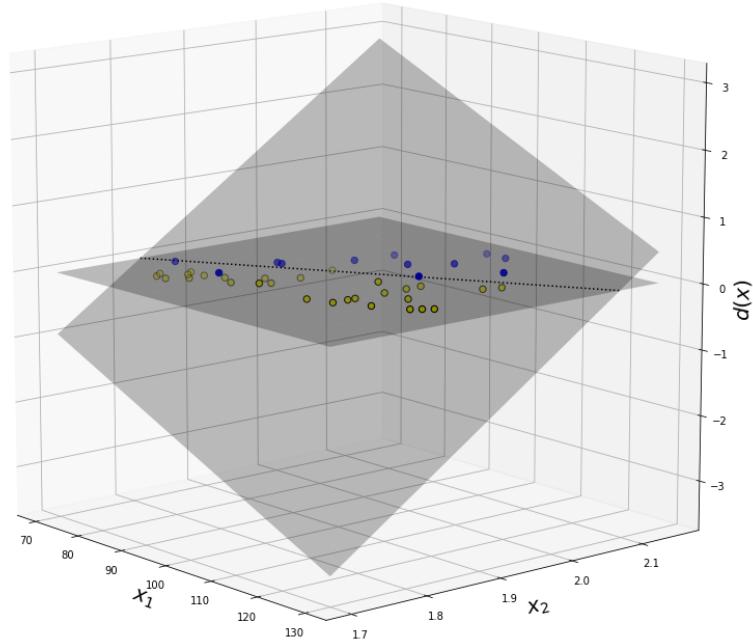


Figura 8.4: Ejemplo de función de decisión lineal en el espacio de las entradas.

separables. Esta es la gran limitación del clasificador lineal, ya que la mayoría de los problemas reales no son linealmente separables.

Función de decisión generalizada

Existen varias estrategias para lidar con los problemas no-linealmente separables. El enfoque que se deriva del clasificador lineal es la función de decisión generalizada, donde la función de decisión se define como $d(\mathbf{x}) = w_1 f_1(\mathbf{x}) + \dots + w_q f_q(\mathbf{x}) + w_{q+1}$. Cada $f_i(\mathbf{x})$ es una función real simple del vector de entradas completo. La última ecuación representa una infinita variedad de funciones de decisión, dependiendo de la elección de las $f_i(\mathbf{x})$ y de la cantidad de términos utilizados. Para el cálculo de los parámetros, primero se realiza una transformación del vector de entradas. El vector utilizado, $\mathbf{x}^* = [f_1(\mathbf{x}), \dots, f_q(\mathbf{x}), 1]^T$, está formado por los valores de las $f_i(\mathbf{x})$ previamente calculados, entonces el cálculo de los coeficientes se reduce a la misma expresión que la del clasificador lineal.

8.4.2 Hiperparámetros

Las coeficientes w de la función de decisión son los parámetros que se ajustan durante el aprendizaje. Todos los modelos de aprendizaje automático tienen un conjunto de parámetros para ajustar, en las redes neuronales por ejemplo, son los pesos sinápticos. La elección de los parámetros determina la calidad de la respuesta de un modelo. También existe un conjunto de decisiones de mayor nivel que afecta el rendimiento de los modelos, estos son los hiperparámetros. Los hiperparámetros representan las decisiones de diseño de los modelos, por ejemplo las funciones y la cantidad de términos utilizados en un clasificador basado en el concepto de la función de decisión generalizada. En el caso de las redes neuronales, estos son la cantidad y tipo de capas, la cantidad de neuronas de cada capa, las funciones de activación, la tasa de aprendizaje, etc.

8.4.3 Redes neuronales artificiales

Al intentar construir máquinas inteligentes surge de forma natural un modelo: la mente humana. Por lo tanto, una idea obvia en el campo de la inteligencia artificial es la de simular directamente el funcionamiento del cerebro en una computadora [RKC94]. Desde la perspectiva de las aplicaciones prácticas del reconocimiento de patrones, el “realismo biológico” impondría restricciones completamente innecesarias [Bis06], por lo tanto, si bien las redes neuronales artificiales están inspiradas en el modelo biológico, se debe pensar en ellas como modelos que extraen combinaciones lineales de las entradas, convirtiéndolas en características (*features*) derivadas y después modelan la salida como una función no lineal de esas características [HTF13]; no como una simulación del cerebro.

Cambio de notación: en esta sección y' reemplaza a la \hat{y} utilizada anteriormente. Además, f ahora representa la función de activación de la red neuronal, no la función que se desea aproximar.

Perceptrón

El perceptrón fue ideado por Frank Rosenblatt en 1958 y es el primer modelo de neurona artificial. Es un método de aprendizaje supervisado que realiza una clasificación binaria en base a una transformación lineal, al igual que el clasificador lineal presentado en la sección 8.4.1.

El perceptrón representa una neurona biológica donde las dendritas son las entradas del perceptrón, el axón es la salida, las sinapsis son los coeficientes de la función de decisión y el comportamiento (muy simplificado) se replica acumulando la intensidad de los impulsos recibidos que, al superar cierto umbral, “activan” la neurona emitiendo una

8 Aprendizaje automático

señal por la salida. En la figura 8.5

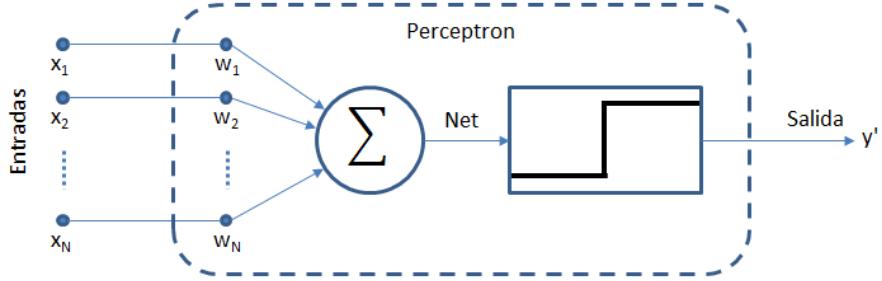


Figura 8.5: Esquema del perceptrón.

Para el vector de entrada $\mathbf{x} = [x_1, x_2, \dots, x_N]$, un vector de pesos $\mathbf{w} = [w_1, w_2, \dots, w_N]$, el umbral de activación θ y la función umbral (también *threshold* o *hard-lim*) f , la salida y' del perceptrón se calcula como:

$$y' = f \left(\left(\sum_{i=1}^N w_i x_i \right) - \theta \right) \quad (8.13)$$

Es común llamar *Net* a la suma ponderada de las entradas $\sum_{i=1}^N w_i x_i$ para simplificar las expresiones. Entonces, en la ecuación 8.13:

$$f(Net, \theta) = \begin{cases} 0 & \text{si } Net - \theta < 0 \\ 1 & \text{si } Net - \theta \geq 0 \end{cases}$$

Para hacer más simple el cálculo, usualmente se incluye el umbral θ dentro de *Net* añadiendo un nuevo peso sináptico $w_0 = -\theta$. Para esto se redefine el vector de entrada como $\mathbf{x} = [1, x_1, x_2, \dots, x_N]$ con el mismo criterio utilizado al aumentar el vector de entrada del clasificador lineal. A partir de estos cambios, la salida se calcula como:

$$y' = f \left(\sum_{i=0}^N w_i x_i \right) \quad (8.14)$$

donde

$$f(Net) = \begin{cases} 0 & \text{si } Net < 0 \\ 1 & \text{si } Net \geq 0 \end{cases}$$

En la figura 8.6 se muestra el esquema más usual del perceptrón, donde se reflejan los cambios realizados sobre el umbral. Notar la entrada fija en 1 del peso w_0 .

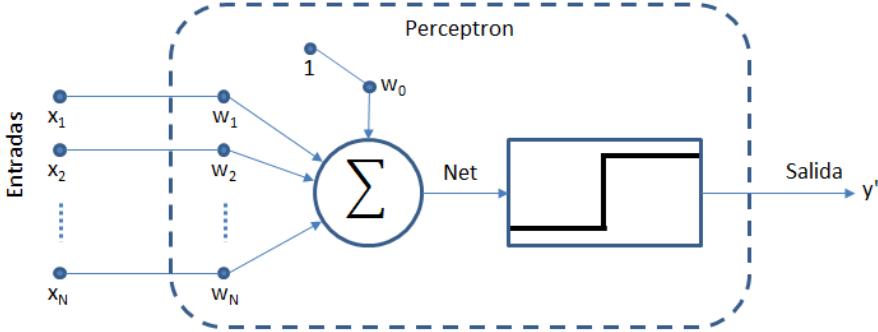


Figura 8.6: Esquema del perceptrón para $w_0 = -\theta$ y vector de entrada aumentado.

El ajuste de los pesos del perceptrón se realiza con la siguiente *ley de aprendizaje*:

$$\Delta_{w_i} = \alpha(y - y')x_i \quad (8.15)$$

donde y es la salida esperada y α es la tasa de aprendizaje. El proceso es el siguiente:

- Inicializar los pesos con valores aleatorios.
- Mientras $error > 0$ para algún vector de entradas, hacer:
 - Ejecutar ciclo completo de entrenamiento (*epoch*). Mientras existan vectores de entrenamiento hacer:
 - * Tomar un vector entrada/salida del conjunto de datos de entrenamiento.
 - * Calcular la salida $y' = f\left(\sum_{i=0}^N w_i x_i\right)$
 - * Calcular el $error = y - y'$
 - * Calcular $\Delta_{w_i} = \alpha(y - y')x_i$
 - * Modificar los pesos haciendo $w_{i+1} = w_i + \Delta_{w_i}$

Notar que tal como está declarado el método, y a menos que se defina una cantidad máxima de ciclos, el entrenamiento termina solo cuando todos los vectores están bien clasificados.

El perceptrón tiene la misma desventaja que el clasificador lineal, solo es capaz de clasificar correctamente las entradas de problemas linealmente separables. El problema

8 Aprendizaje automático

clásico para ejemplificar las clases no-linealmente separables es el de la compuerta XOR (figura 8.7), donde resulta obvio que una linea recta no puede separar los puntos $(0,0)$ y $(1,1)$ de los puntos $(0,1)$ y $(1,0)$ en el espacio de las entradas.

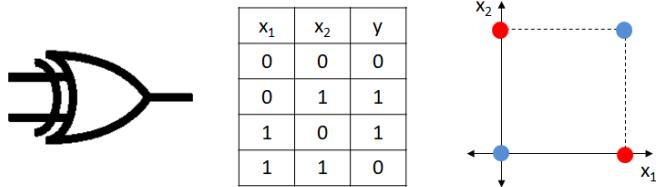


Figura 8.7: Problema de clases no-linealmente separables ejemplificado con la compuerta XOR.

El análisis de la compuerta XOR es interesante porque permite vislumbrar la solución para que las redes neuronales, a pesar de realizar transformaciones lineales en su interior, puedan resolver problemas de clases no-linealmente separables.

Una compuerta XOR se puede construir con otras compuertas, por ejemplo las del circuito lógico de la figura 8.8.

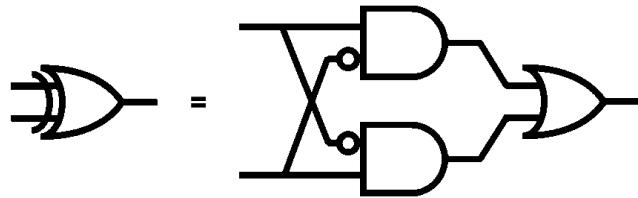


Figura 8.8: Uno de los circuitos lógicos posibles para crear una compuerta XOR a partir de compuertas AND, OR y negadores.

Si tres perceptrones (P_1 , P_2 y P_3) se entrena, por separado, para funcionar como cada una de las compuertas del circuito de la figura 8.8 y después se conectan como en la figura 8.9, se podría predecir el comportamiento de la compuerta XOR.

En la figura 8.10 se muestran las tablas de verdad de las compuertas C_1 (AND con la segunda entrada negada) y C_2 (AND con la primera entrada negada) y las fronteras entre clases en el espacio de las entradas para los perceptrones P_1 y P_2 entrenados. Es fácil ver que una linea recta puede resolver ambos casos porque para las dos compuertas solo una de las combinaciones pertenece a la clase de salida “1”.

El perceptron P_3 se entrena para funcionar como una compuerta OR, lo que también se puede lograr fácilmente porque una sola de sus salidas pertenece a la clase “0”. En la figura 8.11 se puede ver la tabla de verdad de la compuerta OR incluyendo las verdaderas entradas de la compuerta C_3 , que son las salidas de C_1 y C_2 . El perceptron P_3 funciona

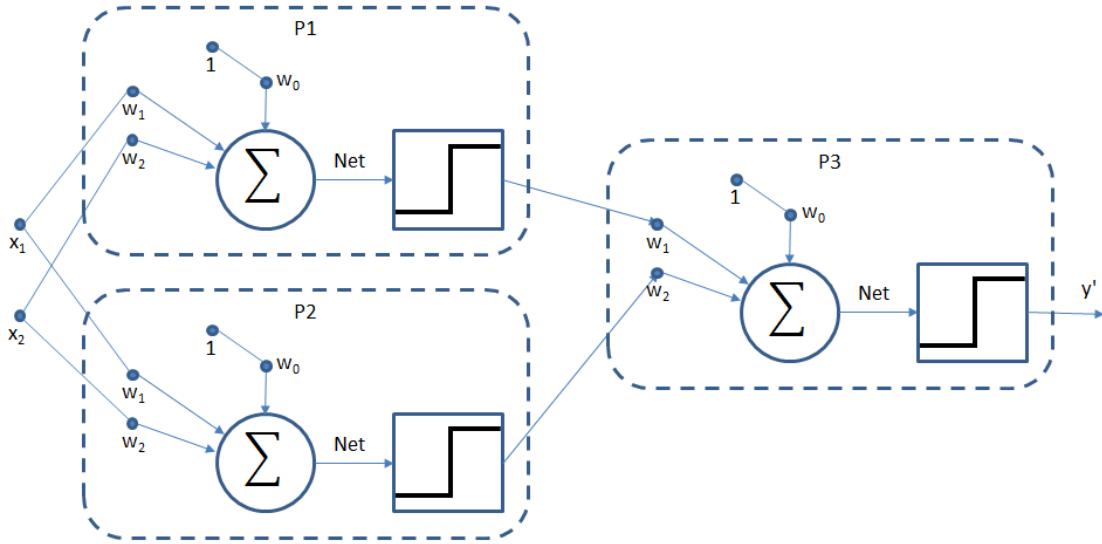


Figura 8.9: Conexión de tres perceptrones para predecir la salida de una compuerta XOR.

como una compuerta OR, por lo tanto no podría predecir la salida de una compuerta XOR a partir de las entradas x_1 y x_2 , pero sí lo puede hacer en el nuevo espacio $c_1 - c_2$ formado por las salidas intermedias del circuito de la figura 8.9. La función de activación escalón introduce una no-linealidad que evita que las sucesivas transformaciones lineales se conviertan en una sola transformación lineal.

Como conclusión, una red de perceptrones es capaz de predecir la salida de una compuerta XOR si utilizamos el conocimiento del dominio del problema para determinar el valor de los pesos sinápticos. Esta conclusión se puede extender a cualquier problema con clases no-linealmente separables.

La conclusión del párrafo anterior es cierta, pero no implica que una red formada por perceptrones sea capaz de resolver este tipo de problemas en casos reales porque no es posible entrenarla. Si se cuenta con conocimiento del dominio del problema, sencillamente no tiene sentido utilizar una red neuronal para resolverlo. Si no se cuenta con este conocimiento, tampoco se conoce la salida esperada para los perceptrones P1 y P2, por lo tanto, no es posible ajustar sus pesos. El desafío es encontrar un método que permita ajustar los parámetros a pesar de no conocerse los valores esperados de las salidas intermedias.

Se suele generar una confusión relacionada con este tema. Uno de los modelos más exitosos de aprendizaje automático son las redes neuronales con conexiones hacia adelante (*feed-forward neural networks*), también llamadas perceptrones multicapa (*multilayer perceptrons*). En realidad, “perceptron multicapa” es un nombre inapropiado, porque el modelo está compuesto por múltiples capas de modelos de regresión logística (con

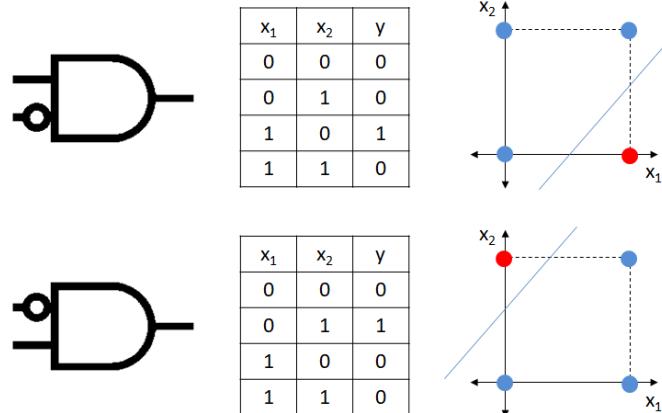


Figura 8.10: Clasificación de la salida de las compuertas C1 con el perceptron P1 (arriba) y de la compuerta C2 con el perceptron P2 (abajo).

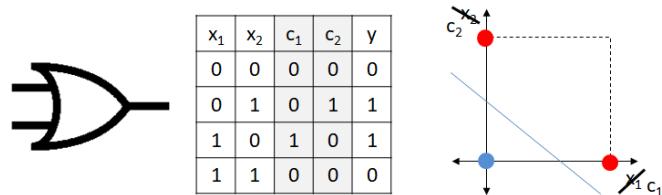


Figura 8.11: Clasificación de la salida de la compuerta C3 con el perceptron P3. En la tabla de verdad, la salida y se calcula para las entradas c_1 y c_2 , no para x_1 y x_2 .

no-linealidades continuas), en lugar de perceptrones (con no-linealidades discontinuas) [Bis06].

Adaline

Adaline (*ADAptive LINear Element*) es un modelo de red neuronal artificial que tiene dos diferencias con respecto al perceptron.

La primera diferencia es la función de activación, en Adaline se utiliza una función lineal. La salida de Adaline es directamente $y' = Net$. Si este modelo se aplica a una predicción del tipo regresión (cosa imposible para el perceptron), la salida utilizada es Net . En el caso de la clasificación, será necesario utilizar una función escalón en algún momento. En este punto, hay distintos criterios en la bibliografía. Algunos autores plantean que la función de activación de Adaline es lineal y, fuera del modelo, se aplica una función umbral para determinar la clase. Otros autores plantean que la salida de

Adaline es binaria, pero que el error para la modificación de los pesos se calcula antes de la función de activación. Si bien el resultado es el mismo, esta diferencia suele generar confusiones. En la cátedra de IAR se utiliza el primer enfoque. En la figura 8.12 se muestra el esquema del modelo.

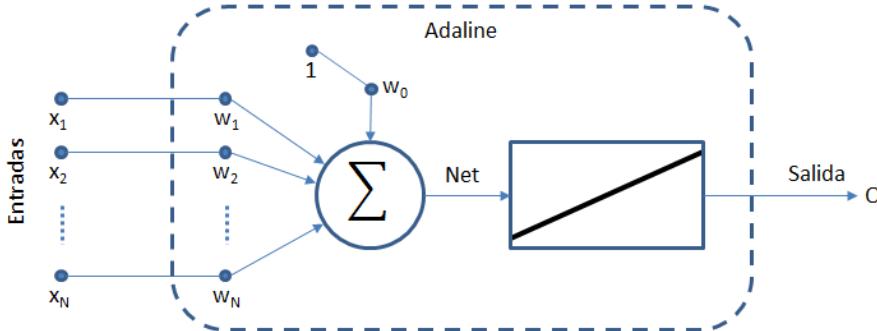


Figura 8.12: Esquema de Adaline.

La segunda diferencia es la ley de aprendizaje. En Adaline se utiliza la regla delta, basada en el mínimo error cuadrático medio (*Least Mean Squared* o *LMS error*). Los pesos de la red se ajustan mediante una búsqueda guiada por el gradiente descendiente de la función del error.

Antes de abordar la explicación de la regla delta se realiza un cambio en los nombres de algunas variables para facilitar el desarrollo de la sección 8.4.3:

- O : salida de la red (por *output*). En Adaline $O = \sum_{i=0}^N w_i x_i = Net$.
- k : identificador de un vector de entrenamiento.
- E : error cuadrático medio. $E = \frac{1}{2L} \sum_{k=1}^L (y_k - O_k)^2$, donde L es la cantidad de vectores de entrenamiento.

La adaptación de los pesos se lleva a cabo a través de una búsqueda sobre la superficie del error. Para encontrar el mínimo de la función del error los pesos se modifican en cantidades proporcionales al gradiente decreciente de la función E. La ley de aprendizaje se define como:

$$\Delta_{w_{ik}} = -\alpha \frac{\partial E_k}{\partial w_i} \quad (8.16)$$

donde α es la tasa de aprendizaje.

Aplicando la regla de la cadena,

$$\begin{aligned} \frac{\partial E_k}{\partial w_i} &= \frac{\partial E_k}{\partial O_k} \frac{\partial O_k}{\partial Net_k} \frac{\partial Net_k}{\partial w_i} \\ \frac{\partial E_k}{\partial O_k} &= \frac{\partial (\frac{1}{2}(y_k - O_k)^2)}{\partial O_k} = \frac{1}{2} 2(y_k - O_k)(-1) = -(y_k - O_k) \\ \frac{\partial O_k}{\partial Net_k} &= 1 \quad (8.17) \\ \frac{\partial Net_k}{\partial w_i} &= \frac{\partial (\sum_{l=1}^N w_l x_{kl})}{\partial w_i} = \frac{\partial (w_0 x_0 + \dots + w_i x_i + \dots + w_N x_N)}{\partial w_i} \\ &= \frac{\partial (w_i x_i)}{\partial w_i} = x_i \end{aligned}$$

Llamando $\delta = \frac{\partial E_k}{\partial O_k} \frac{\partial O_k}{\partial Net_k}$, la regla delta queda definida como:

$$\begin{aligned} \Delta_{w_{ik}} &= -\alpha \delta \ x_i \\ &= -\alpha(-(y_k - O_k)) x_i \\ &= \alpha(y_k - O_k) x_i \quad (8.18) \end{aligned}$$

Finalmente, la expresión de la ley de aprendizaje de Adaline (ecuación 8.18) es igual a la del perceptrón (8.4.3), pero en el caso de Adaline el error puede tomar cualquier valor en \mathbb{R} .

El método de entrenamiento también es similar al del perceptrón, con la diferencia de que el error nunca es cero, así que se necesita otra condición de corte. Típicamente se corta el entrenamiento cuando se llega a un umbral de error previamente definido.

En cuanto a las limitaciones, Adaline solo puede clasificar problemas que sean linealmente separables. La conexión en capas de unidades Adaline, como se hizo en la sección anterior con perceptrones, no tiene sentido. Como no hay no-linealidades en las funciones de activación, Adaline realiza una transformación lineal de las entradas. Las transformaciones lineales sucesivas se pueden resumir en una sola transformación, por lo tanto, conectar Adalines en capas da el mismo resultado que una sola neurona Adaline.

Redes con conexiones hacia adelante

Se ha visto que conectar neuronas (no lineales) en capas puede resolver problemas de clasificación con clases no-linealmente separables. En esta sección se presenta un método

para ajustar los pesos de redes neuronales multicapa.

Una red neuronal con conexiones hacia adelante o *feed-forward* es una red donde las neuronas se disponen en capas y cumplen con las siguientes condiciones:

- Todas las neuronas de una capa conectan su salida a una de las entradas de cada una de las neuronas de la capa siguiente (conexiones hacia adelante). Se dice que estas redes están densamente conectadas.
- No hay conexiones hacia neuronas de capas anteriores.
- No hay conexiones con neuronas de la misma capa.

En la figura 8.13 se muestra un esquema de red *feed-forward* de tres capas o una capa oculta.

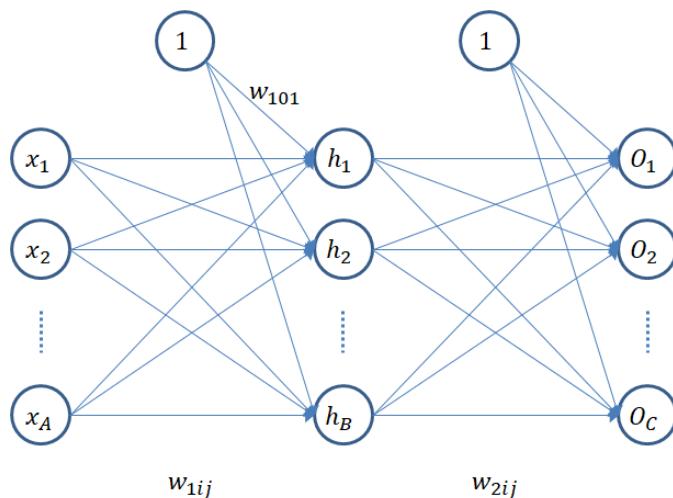


Figura 8.13: Esquema de red neuronal con conexiones hacia adelante de tres capas.

Los nodos x_i de la figura 8.13 son las neuronas de la capa de entrada, aunque en realidad no ejecutan ningún cálculo ni tienen parámetros para ajustar, sus salidas son directamente los valores de los A elementos del vector de entradas. Los nodos h_i (por *hidden*) son las neuronas que forman la única capa oculta del ejemplo. En principio no hay límite en la cantidad de capas ocultas, ni una cantidad de neuronas definida para cada capa. Los nodos O_i son las neuronas de la capa de salida. Si la red se aplica en un caso de clasificación, la cantidad de neuronas de la capa de salida es la cantidad de clases. Si se trata de un caso de regresión, la capa de salida usualmente tendrá una sola neurona y función de activación lineal [HTF13]. En la figura, el primer subíndice de los coeficientes indica la capa (1 para la capa oculta y 2 para la de salida), y los dos subíndices siguientes la i -ésima entrada de la neurona j .

8 Aprendizaje automático

Todas las neuronas de una capa tienen la misma función de activación. Estas funciones deben ser continuas, derivables y no decrecientes. La función más utilizada (por lo menos hasta la aparición del aprendizaje profundo) es la función sigmoidal [LBH15]:

$$f(z) = \frac{1}{1 + e^{-z}}$$

con derivada:

$$\frac{\partial f(z)}{\partial z} = f(z)(1 - f(z))$$

Esta función, junto a la tangente hiperbólica, es muy utilizada porque tiene forma de escalón suavizado. La tangente hiperbólica varía entre -1 y 1, mientras que la función sigmoidal lo hace entre 0 y 1. Al reemplazar la función de activación lineal, en la regla delta cambia la ecuación 8.17. Por ejemplo, si las neuronas de salida tienen activación sigmoidal, la salida se calcula como:

$$O_j = f(Net_j) = \frac{1}{1 + e^{-Net_j}}$$

y la derivada de la salida con respecto a Net :

$$\frac{\partial O_j}{\partial Net_j} = \frac{1}{1 + e^{-Net_j}} \left(1 - \frac{1}{1 + e^{-Net_j}} \right) = O_j(1 - O_j)$$

El ajuste de los pesos se lleva a cabo con una variante de la regla delta llamada regla delta generalizada o retropropagación (*backpropagation*). Cada paso o ciclo del método tiene dos fases. En la primera fase, se toma un vector de entrada y se lo propaga hacia adelante, a través de todas las capas, hasta calcular la salida. En la segunda fase se calcula el error de la capa de salida y se lo propaga hacia atrás, calculando el error de las neuronas de las capas ocultas en base al error cometido por las neuronas de la capa siguiente ponderado por los pesos que las conectan. Una vez calculados los errores, se modifican los pesos. Los pasos son:

- Inicializar los pesos con valores aleatorios
- Mientras no se alcance un error aceptable o el límite de ciclos:
 - Ejecutar ciclo completo de entrenamiento (*epoch*). Mientras existan vectores de entrenamiento hacer:

- * Tomar un par entrada/salida del conjunto de datos de entrenamiento.
- * Calcular la salida de cada capa hasta llegar a la capa de salida.
- * Calcular el error de las neuronas de la capa de salida.
- * Calcular el error de las neuronas de la última capa oculta.
- * Repetir el punto anterior para todas las capas ocultas (desde la salida hacia la entrada).
- * Calcular los Δw_i de cada neurona en función de su propio error.
- * Modificar los pesos.

A continuación se muestran, como ejemplo, las expresiones de cálculo que se utilizarían durante el entrenamiento del modelo de la figura 8.13 para funciones de activación sigmoidales.

Salida de la capa oculta:

$$Net_{1j} = \sum_{i=0}^A w_{1ij}x_i$$

$$h_j = \frac{1}{1 + e^{-Net_{1j}}}$$

Salida de la capa de salida:

$$Net_{2j} = \sum_{i=0}^B w_{2ij}h_i$$

$$O_j = \frac{1}{1 + e^{-Net_{2j}}}$$

Errores de la capa de salida:

$$\delta_{2j} = O_j(1 - O_j)(y_j - O_j)$$

Errores de la capa oculta:

$$\delta_{1j} = h_j(1 - h_j) \sum_{i=1}^C \delta_{2i} w_{2ji}$$

Ajuste de pesos de la capa de salida:

$$\Delta w_{2ij} = -\alpha \delta_{2j} h_i$$

$$w_{2ij(t+1)} = w_{2ij(t)} + \Delta w_{2ij}$$

Ajuste de pesos de la capa oculta:

$$\Delta w_{1ij} = -\alpha \delta_{1j} x_i$$

$$w_{1ij(t+1)} = w_{1ij(t)} + \Delta w_{1ij}$$

Error cuadrático medio:

$$E = \frac{1}{L} \sum_{k=1}^L \sum_{j=1}^C \delta_{2jk}^2$$

donde L es la cantidad de vectores de entrenamiento.

Las decisiones de diseño de las redes neuronales, como la cantidad de capas ocultas, la cantidad de neuronas de cada capa oculta, las funciones de activación, las distribuciones de los valores aleatorios de inicialización de los pesos y la tasa de aprendizaje, entre muchas otras, no tienen recomendaciones generales que funcionen en todos los casos. Existen enfoques de diseño basados en búsquedas en el espacio de los hiperparámetros para definir la arquitectura. Algunos comienzan con modelos simples que crecen en complejidad hasta lograr resultados aceptables y otros siguen el orden inverso.

8.4.4 Aprendizaje profundo

El aprendizaje profundo (*deep learning*) es la evolución de las redes neuronales artificiales. El término enfatiza que ahora es posible entrenar redes neuronales más profundas que antes, es decir con mayor cantidad de capas, y pone el foco de atención en la importancia de esta profundidad. Este nuevo término va más allá de la perspectiva neurocientífica utilizada en la generación anterior de modelos de aprendizaje automático. Apela a un

principio más general de aprendizaje compuesto de múltiples niveles y que se puede aplicar en contextos de aprendizaje automático que no están necesariamente inspirados en la neurociencia. Hoy la neurociencia se considera una fuente importante de inspiración para los investigadores de aprendizaje profundo, pero ya no es la guía predominante [GBC16].

Las redes neuronales profundas han superado a los sistemas de inteligencia artificial, tanto basados en otras tecnologías de aprendizaje automático, como diseñados con funcionalidades *ad hoc* [GBC16]. Hoy son el estado del arte para la amplia mayoría de tareas de reconocimiento de patrones.

El aprendizaje profundo es resultado de (muchos) avances sobre distintos temas propios de las redes neuronales, como también de avances tecnológicos más generales, por ejemplo el aumento de la capacidad de cálculo, la posibilidad de parallelizar operaciones, la conectividad y la creciente cantidad de datos disponibles.

Una de las particularidades que distingue al aprendizaje profundo, y que probablemente sea una de las causas de su buen desempeño, es la capacidad para aprender representaciones complejas de características y ajustar los parámetros del clasificador al mismo tiempo [YD14a]. En esta tecnología, tal como lo ilustra la figura 8.14, la extracción de características se hace en el mismo modelo que la clasificación.

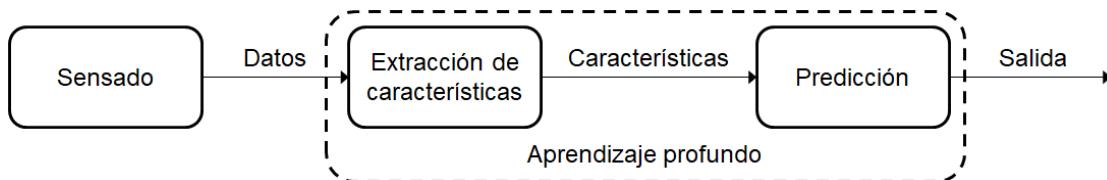


Figura 8.14: Esquema de aprendizaje profundo donde se resalta la integración de la extracción de características y la clasificación en el mismo modelo.

El neocognitron, creado en 1982 por Fukushima [YD14b], es considerado por muchos autores el primer modelo de aprendizaje profundo. Es un modelo extremo a extremo de aprendizaje automático que reconoce caracteres escritos a mano. Su arquitectura se inspira en la naturaleza jerárquica del procesamiento de señales en la corteza visual de los mamíferos. Está formado por capas llamadas simples y complejas similares a las capas de convolución y *pooling* utilizadas actualmente. Esta red reconoce jerárquicamente patrones visuales simples en las primeras capas y más complejos en las siguientes, como los modelos actuales de aprendizaje profundo, aunque el método de entrenamiento es distinto y más complicado. Recién en 1989, LeCun *et al.* [LeC+89] aplican el método de retropropagación a un modelo muy parecido al neocognitron y comienza la nueva era de las redes neuronales. A continuación se mencionan los cambios más importantes que hicieron posible esta revolución.

Funciones de activación

Una de las complicaciones para entrenar redes neuronales con muchas capas es que el gradiente de la función del error se desvanece a medida que se propaga. La causa son las regiones de derivada casi nula en las funciones de activación sigmoidal y tangente hiperbólica. Este problema fue descubierto en la década de 1990 [BFS93; BSF94]. En el año 2011 Glorot *et al.* [GBB11] demuestran que la función de activación ReLU (por *Rectified Linear Unit*) reduce el efecto de desvanecimiento. Con ReLU los modelos típicamente aprenden mucho más rápido que con las anteriores funciones de activación [GBC16; Cal20]. En el presente, y a pesar de una gran cantidad de variantes, ReLU es la función de activación más popular [GBC16].

En la tabla 8.4 se muestran algunas de las funciones de activación más comunes y en la figura 8.15 se pueden ver sus respectivas curvas. Algunas de estas funciones introducen “mejoras” como, por ejemplo, tener derivada no nula para $z < 0$, en los casos de *Leaky ReLU* y *PReLU* (por *parametric*), o transición suave para $z = 0$ en el caso de *ELU* (por *exponential*).

Table 8.4: Funciones de activación

Función	Expresión $f(z)$	Derivada $f'(z)$	Rango
Sigmoidal	$\frac{1}{1 + e^{-z}}$	$f(z)(1 - f(z))$	$[0, 1]$
Tan. Hiperbólica	$\tanh(z)$	$1 - \tanh(z)^2$	$[-1, 1]$
ReLU	$\max(0, z)$	$\begin{cases} 0 & \text{si } f(z) \leq 0 \\ 1 & \text{si } f(z) > 0 \end{cases}$	$[0, \infty]$
<i>Leaky ReLU</i>	$\begin{cases} 0,01z & \text{si } z \leq 0 \\ z & \text{si } z > 0 \end{cases}$	$\begin{cases} 0,01 & \text{si } z \leq 0 \\ 1 & \text{si } z > 0 \end{cases}$	$[-\infty, \infty]$
PReLU	$\begin{cases} \alpha z & \text{si } z \leq 0 \\ z & \text{si } z > 0 \end{cases}$	$\begin{cases} \alpha & \text{si } z \leq 0 \\ 1 & \text{si } z > 0 \end{cases}$	$[-\infty, \infty]$
ELU	$\begin{cases} \alpha(e^z - 1) & \text{si } z \leq 0 \\ z & \text{si } z > 0 \end{cases}$	$\begin{cases} \alpha e^z & \text{si } z \leq 0 \\ 1 & \text{si } z > 0 \end{cases}$	$[-\alpha, \infty]$

Redes neuronales convolucionales.

Parte del éxito del aprendizaje profundo se debe a nuevos tipos de neuronas, capas de neuronas u operaciones que componen las redes neuronales. Además de las capas de neuronas densamente conectadas con conexiones hacia adelante (capas densas) vistas en

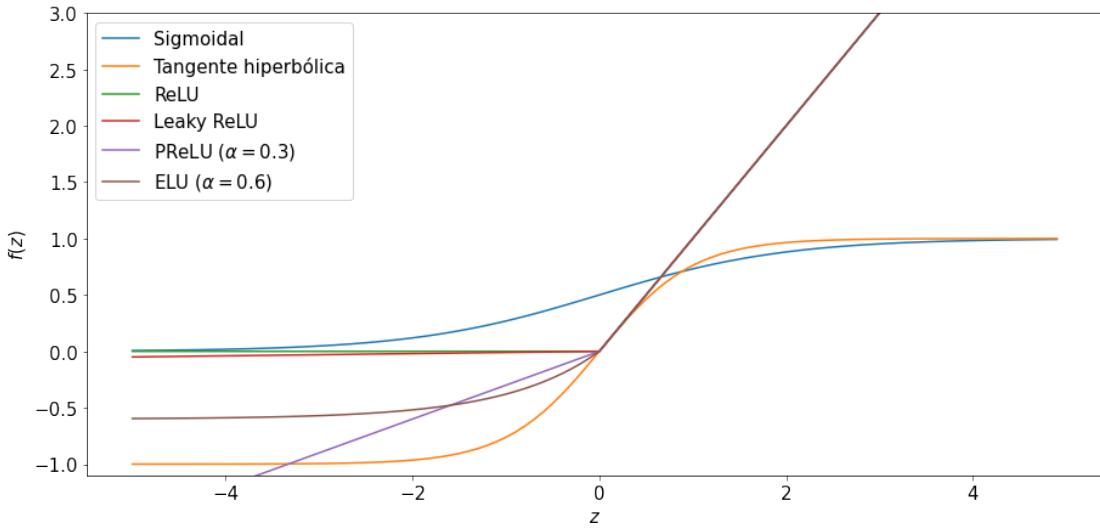


Figura 8.15: Funciones de activación sigmoidal, tangente hiperbólica, ReLU, Leaky ReLU, PReLU (para $\alpha = 0, 3$) y ELU (para $\alpha = 0, 6$).

la sección 8.4.3, son muy utilizadas las neuronas recursivas, las *Long short-term memory* (LSTM), las capas de convolución y las capas de *pooling*. De las mencionadas, las dos primeras se aplican sobre series temporales, en problemas relacionados con el lenguaje por ejemplo.

Las redes neuronales convolucionales (CNN por su sigla en inglés) son un caso particular de red con conexiones hacia adelante cuyas neuronas no están densamente conectadas. Han sido diseñadas para procesar datos en forma de arreglos (*arrays*), donde la ubicación de los elementos es parte de la información, como por ejemplo los píxeles de una imagen. Sobre este tipo de datos las CNN son muchos más fáciles de entrenar y generalizan mucho mejor que las redes que solo tienen capas de neuronas densas [LBH15]. En el caso del audio, si está sin procesar es un arreglo de dimensión 1 y si se usa una representación frecuencial, como el espectrograma, se convierte en un arreglo 2D al igual que una imagen. En ambos casos es importante la ubicación de cada dato y la relación con los datos vecinos.

Hay cuatro conceptos fundamentales detrás de las CNN que aprovechan las propiedades de las señales naturales: conexiones locales, pesos compartidos, *pooling* y el uso de muchas capas [LBH15].

La arquitectura de una CNN típica está formada por una serie de etapas. Las primeras etapas se componen de dos tipos de capas, las capas convolucionales y las capas de *pooling*.

Capas de convolución Cada neurona de una capa de convolución, al igual que las neuronas de las capas densas, realiza una transformación lineal de sus entradas antes de la aplicación de la función de activación. La salida de la neurona j se define como:

$$s_j = f\left(\sum_{i=0}^N w_{ij} x_i\right)$$

donde $f()$ es la función de activación, x_i es la entrada i , w_{ij} es el peso sináptico correspondiente a la entrada i de la neurona j y w_{0j} es el término independiente o bias de la neurona j ($x_0 = 1$).

Las neuronas en una capa de convolución se organizan en mapas de características (*feature maps*), dentro de los cuales cada unidad está conectada a regiones determinadas de los mapas de características de la capa anterior a través de un conjunto de pesos llamado *kernel* o *filter bank*. Todas las neuronas del mismo mapa de características comparten el mismo *kernel*. Mapas de características distintos en una capa usan *kernels* diferentes. En la figura 8.16 se ilustra la conexión de las neuronas. En el ejemplo de la figura, las neuronas de la derecha se conectan a los elementos de la capa anterior (izquierda) mediante un *kernel* de tres pesos. Se puede ver que cada neurona no se conecta a todos los elementos de la capa anterior, sino que tiene un “campo receptivo” acotado. Cada uno de los tres pesos se dibujó con un color y estilo de linea distinto para reforzar la idea de que son compartidos. Notar que, por ejemplo, el primer peso de la primera neurona es igual al primer peso de la segunda neurona.

Las neuronas de la misma posición de mapas de características distintos comparten el campo receptivo, es decir, se conectan a las mismas entradas. De esta forma la salida de una capa de CNN es la convolución discreta de las entradas por los *kernels*. Una vez entrenada la red, cada *kernel* se especializa en reconocer o transformar cierto patrón de los datos de entrada.

La razón para esta arquitectura es doble. En primer lugar, en un arreglo los grupos locales de valores frecuentemente forman patrones fácilmente detectables. En segundo lugar, las distribuciones estadísticas locales de las imágenes y otras señales son invariantes a la ubicación. En otras palabras, si un patrón puede aparecer en una parte del arreglo, este puede aparecer en cualquier parte, de ahí la idea de que neuronas en diferentes ubicaciones comparten los mismos pesos y detecten los mismos patrones en distintas partes del arreglo [LBH15].

En la figura 8.17 se muestra el campo receptivo de dos neuronas de un mapa de características. Si se piensa en la operación de convolución discreta, donde los pesos del *kernel* son los valores de la máscara de convolución, cada neurona representa uno de los lugares donde se posiciona la máscara sobre la entrada. En el caso de la figura,

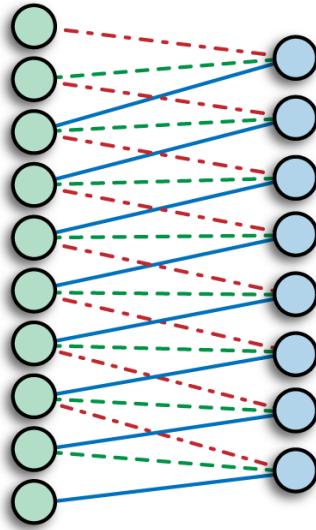


Figura 8.16: Pesos compartidos por neuronas del mismo mapa de características de una CNN. Las neuronas de la derecha tienen un campo receptivo de tamaño = 3 sobre la capa anterior (izquierda).

el tamaño del *kernel* o máscara es 3×4 . Además del tamaño es necesario definir la cantidad de pasos (*strides*) de desplazamiento vertical y horizontal, es decir la cantidad de elementos de diferencia entre una posición y la posición contigua. En este caso, el campo receptivo se mueve dos elementos en ambos sentidos. Otra decisión que se debe tomar es qué pasa con los bordes. La máscara no puede tener elementos fuera del espacio de la entrada, por lo tanto no todos los valores de entrada tienen la misma probabilidad de pertenecer a un campo receptivo o de ser multiplicados por determinado peso aunque los desplazamientos fueran $(1, 1)$. Es común agregar elementos nulos alrededor de la entrada para que la máscara pueda abarcar más valores. Esta técnica se llama *padding*. En las librerías de aprendizaje profundo generalmente se manejan dos tipos de *padding*: “valid”, que indica que la máscara solo se posiciona en lugares válidos de la entrada original, y “same”, que indica que se agregarán tantos valores nulos alrededor de la entrada como sean necesarios para que el tamaño del mapa de características sea igual al tamaño de la entrada.

Capas de pooling Las capas de agrupación o *pooling* tienen una mecánica similar a las capas de convolución, en el sentido de que cada elemento tiene su propio campo receptivo y realiza la misma operación que los elementos vecinos. Estas capas no se ajustan, no tienen pesos y su funcionamiento no cambia durante el entrenamiento. La salida de cada elemento se calcula como el valor máximo (*max pooling*) o el valor medio (*average pooling*) del campo receptivo. A diferencia de las capas de convolución, donde usualmente los campos receptivos de una neurona se forman sobre todos los mapas de

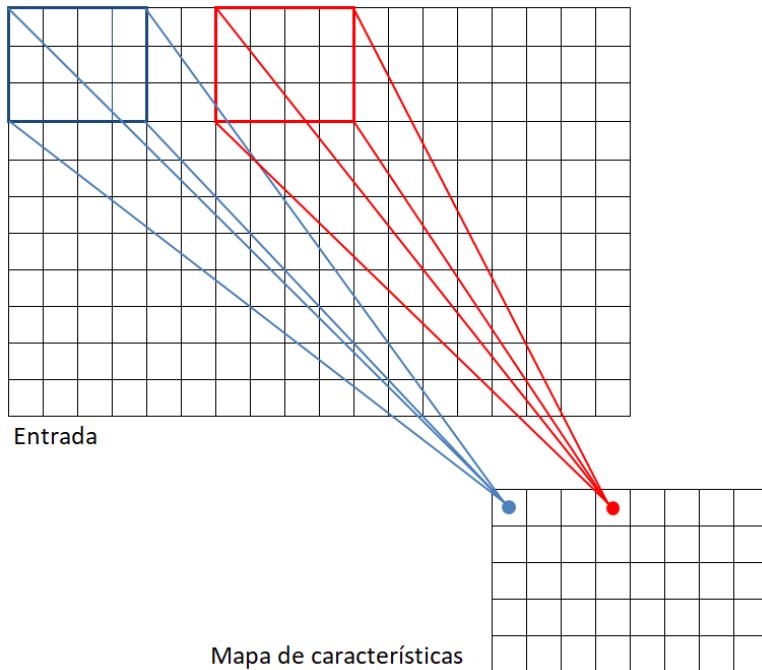


Figura 8.17: Campo receptivo de una neurona de una CNN para un *kernel* de tamaño 3×4 , desplazamiento vertical 2, desplazamiento horizontal 2 y *padding* = “valid”.

características de la capa anterior, en las capas de *pooling* generalmente existe un mapa de características por cada mapa de características de la capa anterior.

Mientras que la función de la capa de convolución es detectar conjunciones locales de características de la capa anterior, la función de la capa de *pooling* es fusionar varias características semánticamente similares en una. Debido a que las posiciones relativas de las características que forman un patrón pueden variar, la detección confiable del patrón se puede realizar haciendo un “granulado grueso” de la posición de cada característica, lo que reduce la dimensión de la representación y aporta invariancia a pequeños cambios y distorsiones [LBH15].

En las CNN es común que se combinen algunas capas de convolución intercaladas con capas de *pooling* (para la extracción de características) y capas densas al final (para la clasificación). La retropropagación del gradiente del error a través de una CNN es tan simple como a través de una red con capas densas, lo que permite entrenar los pesos de todos *kernels* [LBH15]. En la figura 8.18 se muestra como ejemplo la red LeNet5, para reconocimiento de caracteres, presentada por LeCun *et al.* en 1998. Se pueden ver dos capas de convolución (la primera con 6 mapas y la segunda con 16), dos capas de *pooling* (*subsampling*) y tres capas densas.

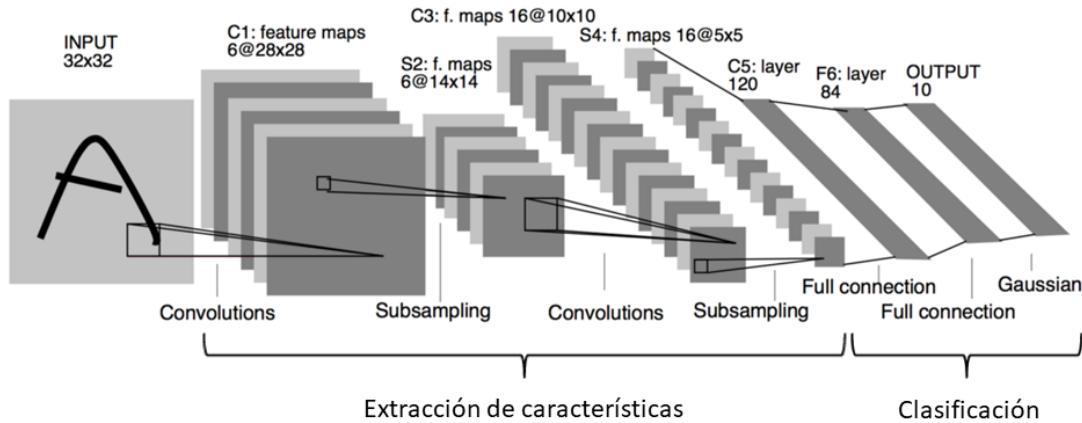


Figura 8.18: Capas de extracción de características y clasificación en LeNet5 [LeC+98]. Imagen tomada del artículo original y modificada en la parte inferior (llaves).

Regularización

Las redes neuronales profundas son modelo complejos, con muchos parámetros para ajustar, lo que las hace propensas al sobreajuste. Goodfellow *et al.* definen a la regularización como “Cualquier modificación que podemos hacer sobre un algoritmo de aprendizaje con el objetivo de reducir su error de generalización pero no su error de entrenamiento” [GBC16]. Es decir, el foco no está puesto en mejorar el resultado para los datos de entrenamiento, sino en reducir la brecha entre el rendimiento medido con los datos de entrenamiento y el rendimiento medido con los datos de validación. En esta sección se explican brevemente cuatro estrategias para mejorar la generalización: *early stop*, penalización, *dropout* y aumentación de datos.

Early stop Al entrenar modelos grandes, con suficiente capacidad de representación como para sobreajustar, a menudo se observa que después de un punto, el error de entrenamiento disminuye de manera constante con el tiempo, pero el error del conjunto de validación comienza a aumentar nuevamente. En la figura 8.19 se muestra un ejemplo de este comportamiento. En lugar de tomar la configuración final de los pesos, se puede obtener un modelo con un mejor error para los datos de validación (y por lo tanto, con suerte, un mejor error para el conjunto de test) volviendo a la configuración de parámetros en el momento donde el error de validación es mínimo. El método es simple, cada vez que mejora el error en el conjunto de validación, se almacena una copia de los parámetros del modelo. Cuando termina el proceso de entrenamiento, se devuelven estos parámetros, en lugar de los últimos parámetros. El algoritmo finaliza cuando ningún parámetro ha mejorado con respecto al mejor error de validación registrado para un número preestablecido de iteraciones [GBC16].

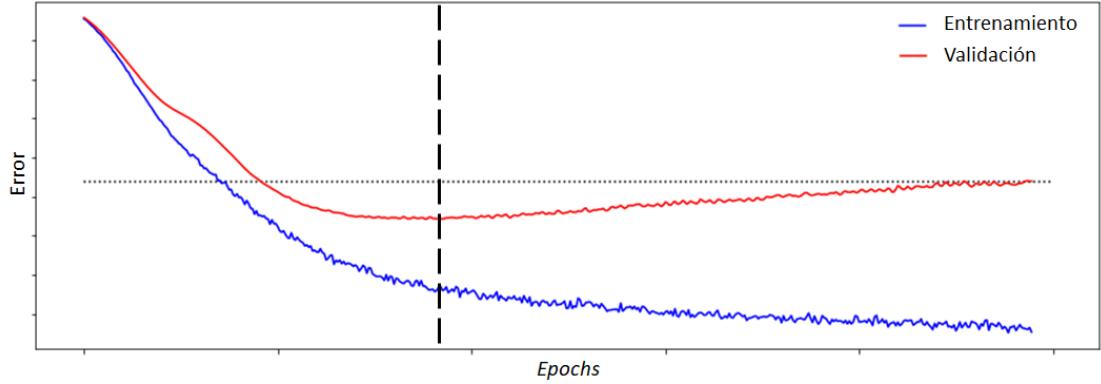


Figura 8.19: Error de entrenamiento y validación durante el aprendizaje. La linea de puntos horizontal marca el error de validación final. La linea segmentada vertical indica el punto donde el error de validación es mínimo.

Penalización Otra estrategia es imponer restricciones adicionales al modelo de aprendizaje automático. Típicamente se agregan restricciones a los valores de los parámetros como términos adicionales en la función objetivo. Si se eligen con cuidado, estas restricciones o penalizaciones adicionales pueden conducir a un mejor desempeño en los datos de validación. En general, se puede regularizar un modelo que aprende la función $f(\theta, \mathbf{x})$ agregando una penalización $\Omega(\mathbf{w})$ llamada “regularizador” a la función error. En la oración anterior, θ es el vector de parámetros que incluye al *bias* (término independiente), mientras que \mathbf{w} es el vector de pesos sin *bias*, es decir, \mathbf{w} contiene solo los pesos que se multiplican por las entradas. Esto se debe a que los *bias* típicamente requieren menos datos para ajustarse correctamente porque solo controlan una variable, no la interacción entre dos variables como los pesos de \mathbf{w} . Regularizar el *bias* podría inducir a un infra-ajuste [GBC16].

Las penalizaciones más utilizadas son:

- Regularización L1

$$\Omega(\mathbf{w}) = \beta \sum_{i=1}^N |w_i|$$

donde β es una constante que pondera la contribución relativa de la penalidad.

- Regularización L2

$$\Omega(\mathbf{w}) = \beta \sqrt{\sum_{i=1}^N |w_i|^2}$$

- Regularización L1 y L2

$$\Omega(\mathbf{w}) = \beta \sum_{i=1}^N |w_i| + \gamma \sqrt{\sum_{i=1}^N |w_i|^2}$$

Dropout Es una estrategia potente de regularización, donde se eliminan al azar algunas neuronas de las capas ocultas durante el entrenamiento. En cada ciclo, con una probabilidad previamente definida (generalmente por capas), se eligen ciertas neuronas y se anulan sus salidas. En la figura 8.20 se muestra un ejemplo.

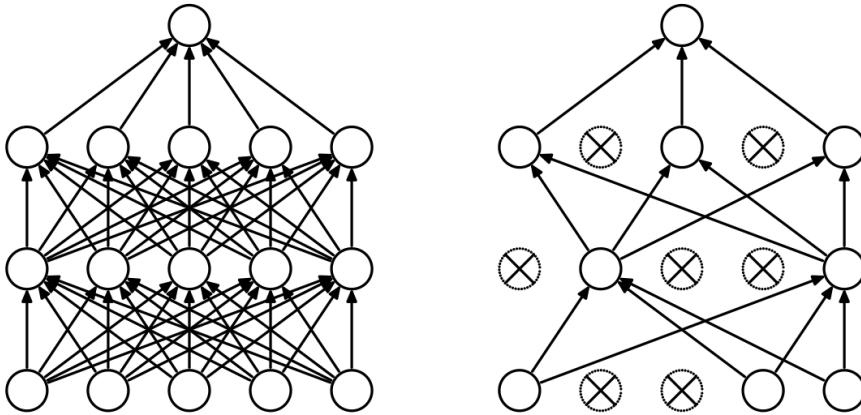


Figura 8.20: Ejemplo de aplicación de *dropout*. Izquierda: Red neuronal original con dos capas ocultas. Derecha: Red neuronal obtenida al aplicar *dropout* sobre la red de la izquierda. Imagen tomada del artículo de Srivastava *et al.* [Sri+14].

Srivastava *et al.* [Sri+14] demostraron que *dropout* es más efectivo que otras estrategias de regularización de bajo costo, como por ejemplo la penalización.

Dropout se puede aplicar a una amplia familia de modelos y también se puede combinar con otras técnicas de regularización reforzando la mejora obtenida [Sri+14].

Aumentación de datos Goodfellow *et al.* aseguran que, si bien son necesarias ciertas habilidades para obtener un buen rendimiento con un modelo de aprendizaje profundo, “afortunadamente” la cantidad de habilidades se reduce en la medida que la cantidad de datos se incrementa. La mejor manera de aumentar el grado de generalización de un modelo de aprendizaje automático es entrenarlo con más datos. Por supuesto, en la práctica, se cuenta con una cantidad limitada de datos. Una forma de solucionar este problema es crear datos nuevos [GBC16].

Para algunas tareas de aprendizaje automático, es relativamente sencillo crear datos

8 Aprendizaje automático

nuevos. En los casos de clasificación de imágenes y audio, un clasificador necesita tomar una entrada \mathbf{x} compleja y de alta dimensión y resumirla a una categoría única y . Esto significa que la tarea principal a la que se enfrenta un clasificador es ser invariante ante una amplia variedad de transformaciones. Podemos generar nuevos pares (\mathbf{x}, y) simplemente transformando las entradas \mathbf{x} de los datos existentes con las mismas transformaciones a las que se pretende ser invariante. En el caso del reconocimiento de patrones en imágenes, algunas de las transformaciones más comunes son la rotación, el *flipping* (espejado horizontal o vertical), los cambios de color, cambio de escala, el *cropping* (recortes), la traslación y la adición de ruido [SK19]. En la figura 8.21 se muestra un ejemplo de aplicación sobre imágenes.



Figura 8.21: Ejemplo de transformaciones de aumentación de datos en imágenes.

Algunos ejemplos de transformaciones utilizadas para la comprensión del habla en audio son cambios de tono, estiramiento del tiempo (*time stretching*) y filtrado de algunas frecuencias [SG15] o transformaciones que simulan perturbaciones por variación de la longitud del tracto vocal [CGK15]. Estas transformaciones no alteran la información importante, es decir, las características invariantes que son útiles para comprender el habla.

Referencias

- [But63] Samuel Butler. *Darwin Among The Machines*. E. P. Dutton & Company, 1863.
- [Wie61] N. Wiener. *Cybernetics: or Control and Communication in the Animal and the Machine*. MIT Press, 1961.
- [LeC+89] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.
- [BFS93] Yoshua Bengio, Paolo Frasconi, and Patrice Simard. “The problem of learning long-term dependencies in recurrent networks”. In: *IEEE international conference on neural networks*. IEEE. 1993, pp. 1183–1188.
- [BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.
- [RKC94] E. Rich, K. Knight, and P.A.G. Calero. *Inteligencia artificial*. McGraw-Hill, 1994. ISBN: 9788448118587.
- [LeC+95] Yann LeCun et al. “Comparison of learning algorithms for handwritten digit recognition”. In: *International conference on artificial neural networks*. Vol. 60. 1. Perth, Australia. 1995, pp. 53–60.
- [Nil95] Nils J Nilsson. “Perspective on Artificial Intelligence: Present and Future”. In: *ECAI*. 1995.
- [Yar95] David Yarowsky. “Unsupervised word sense disambiguation rivaling supervised methods”. In: *33rd annual meeting of the association for computational linguistics*. 1995, pp. 189–196.
- [Tur96] Alan M Turing. *Computing machinery and intelligence*. Mind, 1996.
- [LeC+98] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [BB01] Michele Banko and Eric Brill. “Scaling to very very large corpora for natural language disambiguation”. In: *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*. 2001, pp. 26–33.
- [Min04] Marvin Minsky. “A framework for representing knowledge”. In: *In AAAI*. Vol. 4. 2004, pp. 1090–1094.

8 Referencias

- [RNR04] S.J. Russell, P. Norvig, and J.M.C. Rodriguez. *Inteligencia artificial: un enfoque moderno. 2da Edición.* Colección de Inteligencia Artificial de Prentice Hall. Pearson Educación, 2004. ISBN: 9788420540030. URL: <https://books.google.com.ar/books?id=yZCVPwAACAAJ>.
- [Bis06] C.M. Bishop. *Pattern Recognition and Machine Learning.* Information Science and Statistics. Springer, 2006. ISBN: 9780387310732. URL: <https://books.google.com.ar/books?id=qWPwnQEACAAJ>.
- [Thr+06] Sebastian Thrun et al. “Stanley: The robot that won the DARPA Grand Challenge”. In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.
- [GP07] Ben Goertzel and Cassio Pennachin. *Artificial general intelligence.* Springer Science & Business Media, 2007.
- [HE07] James Hays and Alexei A Efros. “Scene completion using millions of photographs”. In: *ACM Transactions on Graphics (ToG)* 26.3 (2007), 4–es.
- [McC07] John McCarthy. “What is Artificial Intelligence?” In: *Stanford University* (2007).
- [Min07] Marvin Minsky. “Conversations on the Society of Mind”. In: *AI magazine* 28.1 (2007), pp. 25–40.
- [Omo08] Stephen Omohundro. “The basic AI drives”. In: *Frontiers in Artificial Intelligence and Applications* 171 (2008), pp. 483–492.
- [Yud08] Eliezer Yudkowsky. “Artificial intelligence as a positive and negative factor in global risk”. In: *Global catastrophic risks* (2008), pp. 303–324.
- [BW09] Carole R Beal and Patrick H Winston. *Intelligent tutoring systems: using AI to improve training performance and the learning experience.* John Wiley & Sons, 2009.
- [Den+09] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition.* Ieee. 2009, pp. 248–255.
- [GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.
- [HTF13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer New York, 2013. ISBN: 9780387216065. URL: <https://books.google.com.ar/books?id=yPfZBwAAQBAJ>.
- [Bos14] N. Bostrom. *Superintelligence: Paths, Dangers, Strategies.* Oxford University Press, 2014.
- [Sri+14] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

- [YD14a] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology. Springer London, 2014. ISBN: 9781447157793. URL: <https://books.google.com.ar/books?id=rUBTBQAAQBAJ>.
- [YD14b] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology. Springer London, 2014. ISBN: 9781447157793. URL: <https://books.google.com.ar/books?id=rUBTBQAAQBAJ>.
- [CGK15] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. “Data augmentation for deep neural network acoustic modeling”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.9 (2015), pp. 1469–1477.
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.
- [SG15] Jan Schlüter and Thomas Grill. “Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks.” In: *ISMIR*. 2015, pp. 121–126.
- [GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN: 9780262035613. URL: <https://books.google.com.ar/books?id=Np9SDQAAQBAJ>.
- [Gul+16] Varun Gulshan et al. “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs”. In: *Jama* 316.22 (2016), pp. 2402–2410.
- [Sil+16] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.
- [Est+17] Andre Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *nature* 542.7639 (2017), pp. 115–118.
- [KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.
- [Liu+17] Yun Liu et al. “Detecting cancer metastases on gigapixel pathology images”. In: *arXiv preprint arXiv:1703.02442* (2017).
- [Sil+17] David Silver et al. “Mastering the game of go without human knowledge”. In: *nature* 550.7676 (2017), pp. 354–359.
- [Tet17] Philip E.. Tetlock. *Expert Political Judgment: How Good Is It? How Can We Know?* Princeton University Press, 2017.
- [For18] Martin Ford. *Architects of Intelligence: The truth about AI from the people building it*. Packt Publishing Ltd, 2018.
- [Gra+18] Katja Grace et al. “When will AI exceed human performance? Evidence from AI experts”. In: *Journal of Artificial Intelligence Research* 62 (2018), pp. 729–754.

8 Referencias

- [GS18] Barbara J Grosz and Peter Stone. “A century-long commitment to assessing artificial intelligence and its impact on society”. In: *Communications of the ACM* 61.12 (2018), pp. 68–73.
- [Sil+18] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.
- [Ste+18] David F Steiner et al. “Impact of deep learning assistance on the histopathologic review of lymph nodes for metastatic breast cancer”. In: *The American journal of surgical pathology* 42.12 (2018), p. 1636.
- [Din+19] Yiming Ding et al. “A deep learning model to predict a diagnosis of Alzheimer disease by using 18F-FDG PET of the brain”. In: *Radiology* 290.2 (2019), pp. 456–464.
- [Liu+19a] Xiaoxuan Liu et al. “A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis”. In: *The lancet digital health* 1.6 (2019), e271–e297.
- [Liu+19b] Yun Liu et al. “Artificial intelligence-based breast cancer nodal metastasis detection: Insights into the black box for pathologists”. In: *Archives of pathology & laboratory medicine* 143.7 (2019), pp. 859–868.
- [SK19] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of Big Data* 6.1 (2019), pp. 1–48.
- [Top19] Eric Topol. *Deep medicine: how artificial intelligence can make healthcare human again*. Hachette UK, 2019.
- [Cal20] Ovidiu Calin. *Deep Learning Architectures: A Mathematical Approach*. Springer Series in the Data Sciences. Springer International Publishing, 2020. ISBN: 9783030367213. URL: <https://books.google.com.ar/books?id=R3vQDwAAQBAJ>.
- [Des20a] Eduardo Destéfanis. *Inferencia y probabilidad. Lógica difusa*. Aula virtual de IAR, 2020.
- [Des20b] Eduardo Destéfanis. *Reconocimiento de patrones*. Aula virtual de IAR, 2020.
- [Liu+20] Yuan Liu et al. “A deep learning system for differential diagnosis of skin diseases”. In: *Nature medicine* 26.6 (2020), pp. 900–908.
- [Gar21] Mario Alejandro García. “Clasificación automática del grado general de disfonía”. Available at <https://ria.utn.edu.ar/xmlui/handle/20.500.12272/5956>. PhD thesis. UTN FRC, Dec. 2021.
- [RN21] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education, 2021. ISBN: 9781292401171. URL: <https://books.google.com.ar/books?id=cb0qEAAAQBAJ>.

- [Sto+22] Peter Stone et al. “Artificial intelligence and life in 2030: the one hundred year study on artificial intelligence”. In: *arXiv preprint arXiv:2211.06318* (2022).
- [Jam+23a] G. James et al. *An Introduction to Statistical Learning: with Applications in Python*. Springer Texts in Statistics. Springer International Publishing, 2023. ISBN: 9783031387470. URL: <https://www.statlearning.com>.
- [Jam+23b] G. James et al. *An Introduction to Statistical Learning: with Applications in R, Second Edition*. Springer Texts in Statistics. Springer International Publishing, 2023. URL: <https://www.statlearning.com>.