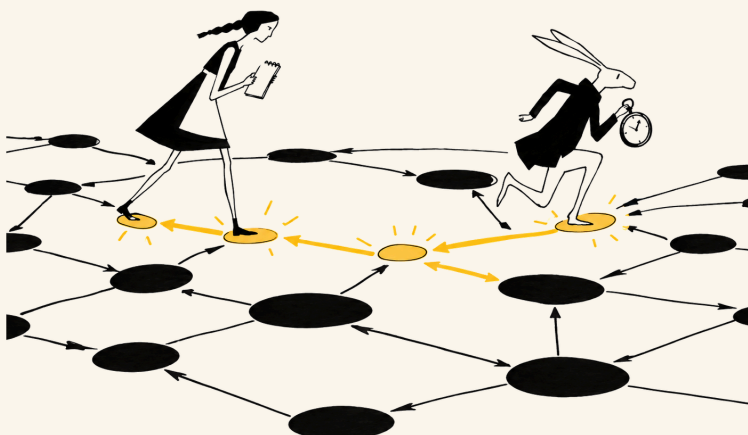


Las aventuras de Alicia en el País de las Maravillas Diferenciables



Simone Scardapane · 2024

Audiolecciones ilustradas

Lecturized

Las aventuras de Alicia en el País de las Maravillas Diferenciables

Audiolecciones ilustradas

Simone Scardapane · 2024

Contado como lecciones ilustradas por Lecturized

lecturized.com

Del original: Alice's Adventures in a Differentiable Wonderland, Volume I: A Tour of the
Land

Índice

Unas pocas piezas simples, hechas muy grandes

Aprender como una búsqueda entre perillas

Ideas viejas a la espera de datos suficientes

Cajas de números y sus formas

Qué es este libro y qué no es

Créditos, una licencia y cortar rebanadas

Flechas, longitudes y el producto punto

Tablas de números que actúan

Contar el trabajo

Pilas de tablas y una abreviatura con letras

La pendiente en un punto

Pendientes en todas las direcciones

Donde el terreno cae más rápido

Bajar la cuesta, y dónde termina uno

Una bola que recuerda

Del papel al código

Qué hace bueno a un montón de ejemplos

Aprender sin etiquetas, y reutilizar lo aprendido

Puntuar una predicción

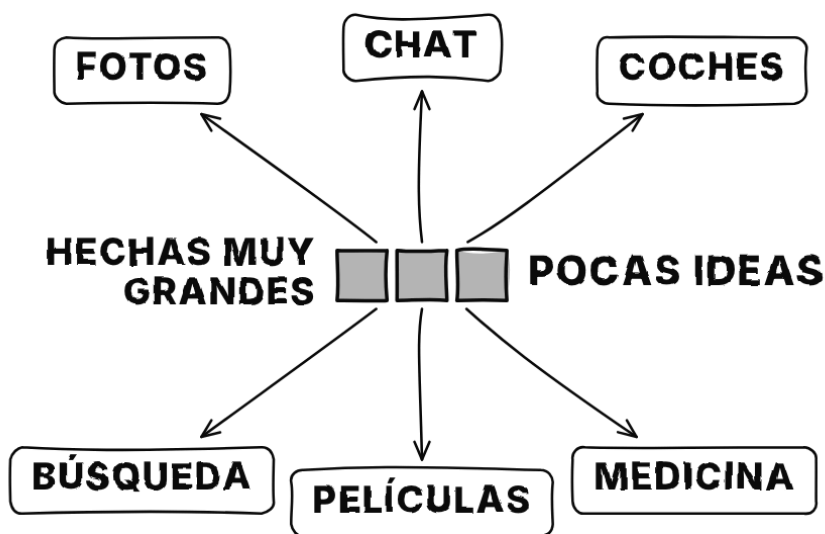
Memorizar no es aprender

La explicación más probable, y muchas explicaciones a la vez

Una penalización desde la prior, y el bloque más simple

Unas pocas piezas simples, hechas muy grandes

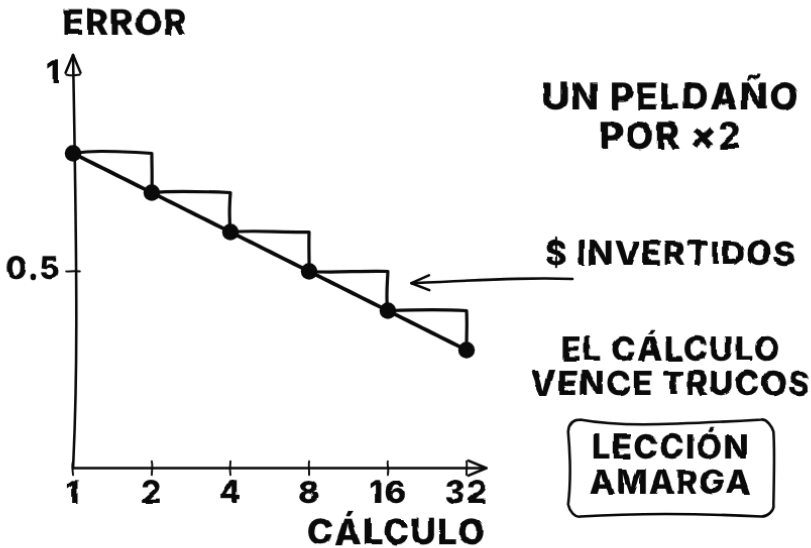
Abra la aplicación de fotos de su teléfono y escriba la palabra «perro» en el cuadro de búsqueda. En un segundo le muestra todos los perros que ha fotografiado en su vida, aunque usted nunca etiquetó una sola imagen. Pida a un asistente de chat que le explique una receta o corrija una frase, y responde en párrafos fluidos. Ambas hazañas las realizan redes neuronales, y también muchas cosas que usted nunca ve: la ayuda a la dirección en los coches, el orden de los resultados de búsqueda, las películas que sugiere un servicio de streaming, incluso el diseño de nuevas moléculas para medicamentos. La noticia sorprendente con la que abre el libro es que todo esto descansa sobre un conjunto muy pequeño de ideas guía y bloques de construcción.



Pocos bloques, muchos usos

Lo que cambió en los últimos años no son tanto las ideas como su tamaño. Los investigadores notaron un patrón llamado leyes de escalado neuronal: para casi cualquier tarea, si aumenta tres cosas a la vez, la cantidad de datos, la potencia de cálculo y el tamaño del modelo, la precisión mejora, y mejora de forma predecible. Dicho al revés, la potencia de cálculo necesaria para alcanzar una precisión dada sigue cayendo por un factor constante con el paso de los años. Imagine una escalera que puede ver de antemano: cada vez que duplica el presupuesto, sabe más o menos cuántos peldaños subirá. Esa previsibilidad es lo que convenció a las empresas de invertir sumas enormes en inteligencia artificial. El investigador Richard Sutton le puso nombre a la lección, la lección amarga: herramientas

simples y generales combinadas con cada vez más potencia de cálculo han vencido una y otra vez a los trucos ingeniosos hechos a mano.



La escalera de la ley de escalado

Tome las leyes de escalado como dadas, y le queda una herramienta casi mágica. En una frase, una red neuronal se ajusta para que imite el patrón escondido en una gran colección de ejemplos; en el lenguaje del libro, aproxima una distribución de probabilidad a partir de datos extraídos de ella. Esto podría salir mal con facilidad. Las redes modernas son tan grandes que podrían simplemente memorizar cada ejemplo que se les muestra, como un alumno que aprende de memoria las respuestas del examen del año pasado y se pierde cuando cambian las preguntas. Una red que hiciera eso no sería más que una tabla de consulta. Sin embargo, las redes entrenadas hacen algo

mejor: generalizan, es decir, manejan casos que nunca estuvieron en sus datos de entrenamiento. Y a medida que las colecciones de ejemplos se vuelven enormes, la línea entre casos familiares y desconocidos se difumina, y los modelos más grandes se apoyan sorprendentemente poco en la memoria mecánica.

MEMORIZAR

ENTRADA	SALIDA
2	4
3	6
5	10
4	?

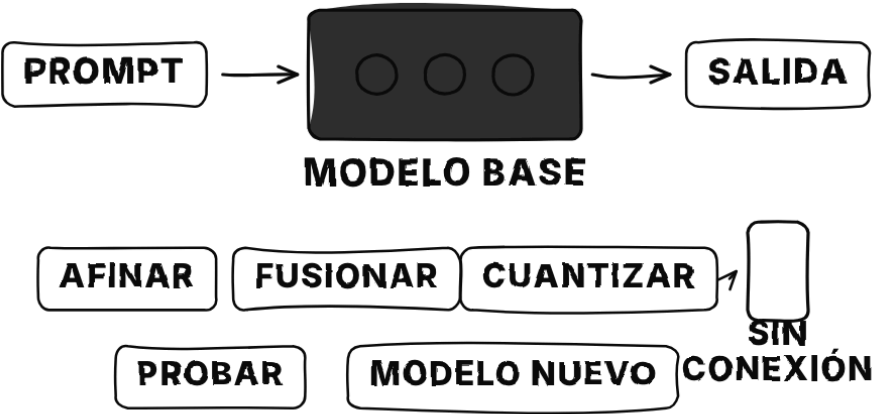
TABLA



Memorizar o generalizar

Estos modelos muy grandes, entrenados una vez y luego usados para muchos trabajos distintos, a veces se llaman modelos fundacionales. Junto con una animada comunidad de código abierto, han cambiado la forma en que la gente usa las redes. Muchas tareas ya no necesitan programación alguna: usted simplemente da un prompt al modelo, es decir, le da instrucciones en texto o imágenes, casi como escribe una petición a un asistente de chat. El modelo en sí sigue siendo una caja

negra, una máquina cerrada cuyo interior usted nunca mira. El libro compara esto con el software corriente: en lugar de escribir usted mismo cada biblioteca en un lenguaje como C++, se apoya en programas escritos por otros, a menudo sin ver su código. La comparación es acertada, porque solo unos pocos equipos en el mundo tienen la potencia de cálculo y la destreza para construir modelos verdaderamente grandes, como la familia Llama de modelos de lenguaje, igual que solo unas pocas empresas pueden construir el gran software empresarial con el que funcionan las compañías.



Una caja negra que puede adaptar

Pero la comparación también apunta en la otra dirección. El software de código abierto permite a cualquiera adaptar un programa o escribir uno nuevo desde cero, y del mismo modo un ordenador corriente y

algo de ingenio abren muchas puertas con estos modelos. Puede afinar un modelo, entrenándolo un poco más con sus propios ejemplos; fusionar dos modelos en uno; cuantizar un modelo, guardando sus números de forma más gruesa para que funcione en un dispositivo pequeño y de bajo consumo; probar qué tan robusto es frente a entradas engañosas; o inventar tipos de modelos completamente nuevos. El asistente de voz que le responde sin conexión en su teléfono es a menudo exactamente un modelo así, encogido y cuantizado. Para todo esto hay que mirar bajo el capó: entender cómo estos modelos reciben y transforman datos, junto con los trucos y las rarezas que los practicantes aprendieron con la experiencia y la depuración. El libro se presenta como la puerta de entrada a ese mundo, y su autor espera que un lector curioso por naturaleza, como Alicia, disfrute el viaje.

Pruébelo

Abra la aplicación de fotos de su teléfono y busque un objeto que nunca haya etiquetado, como «bicicleta», «playa» o «pastel de cumpleaños». Observe que encuentra la mayoría, y observe también el ocasional error extraño, una lámpara tomada por un pastel o un perro tomado por un oso de peluche. Esos errores son un atisbo de un modelo que aprendió un patrón a partir de ejemplos y no de una lista de reglas, que es exactamente el tipo de máquina que este libro desarma.

Para recordar

Las redes neuronales hoy impulsan asistentes de chat, aplicaciones de fotos, coches, búsqueda, recomendaciones y ciencia, y sin embargo están construidas a partir de un conjunto muy pequeño de principios. Las leyes de escalado dicen que más datos, más potencia de cálculo y modelos más grandes dan juntos una ganancia predecible en precisión, lo que impulsó una inversión enorme. Una red se ajusta para imitar el patrón detrás de sus ejemplos; podría memorizarlos, pero las redes grandes entrenadas generalizan a casos que nunca vieron. Los modelos fundacionales se usan a menudo como cajas negras mediante prompts, pero adaptar, encoger o inventar modelos exige entender qué ocurre en su interior. Esa comprensión del funcionamiento interno es lo que el libro se propone dar.

Preguntas

1. ¿Qué dicen las leyes de escalado neuronal que ocurre cuando los datos, la potencia de cálculo y el tamaño del modelo aumentan todos a la vez?
 - a. La precisión sube, y en una cantidad que se puede predecir de antemano
 - b. La precisión sube al principio y luego siempre cae porque los modelos memorizan
 - c. La precisión se queda igual, pero el modelo funciona más rápido

- d. La precisión sube solo en tareas de lenguaje, no en imágenes
2. ¿Por qué una red que memorizara todos sus ejemplos de entrenamiento sería de poca utilidad?
 - a. Necesitaría mucha más potencia de cálculo para responder
 - b. Sería una tabla de consulta, perdida ante cualquier caso que no hubiera visto
 - c. Ya no se le podrían dar prompts de texto
 - d. Olvidaría los ejemplos al cabo de un tiempo
 3. En la comparación del libro con el software, ¿a qué corresponde dar un prompt a un modelo fundacional?
 - a. A escribir usted mismo cada biblioteca en C++
 - b. A construir su propio software empresarial
 - c. A usar el programa de otra persona sin ver su código fuente
 - d. A leer el código fuente de una biblioteca de código abierto
 4. ¿Cuál de estos trabajos exige mirar bajo el capó de un modelo en lugar de solo darle prompts?
 - a. Pedir a un asistente de chat que corrija una frase
 - b. Buscar en sus fotos la palabra perro
 - c. Escribir una pregunta en un buscador
 - d. Cuantizar un modelo para que funcione en un dispositivo pequeño

Respuestas

1. a) Las leyes de escalado describen una ganancia predecible en precisión para casi cualquier tarea cuando las tres crecen juntas; esa previsibilidad es lo que impulsó la inversión.

2. **b)** Memorizar convierte la red en una tabla de consulta, como un alumno que sabe de memoria las respuestas del año pasado; lo que hace útiles a las redes entrenadas es que generalizan a casos nuevos.
3. **c)** Dar un prompt usa el modelo como una caja negra, igual que uno se apoya en programas escritos por otros sin acceso a su código.
4. **d)** Afinar, fusionar, cuantizar o probar un modelo requieren entender cómo procesa los datos internamente, que es lo que ofrece el libro.

Ejercicios

1. Un amigo dice que un modelo entrenado con millones de fotos simplemente debe estar recordándolas. Usando lo que dijo esta parte, ¿cómo le respondería?

Pista: Piense en qué haría un modelo que solo recordara con una foto que nunca hubiera visto.

Respuesta: Un modelo que solo recordara sería una tabla de consulta y fallaría con fotos nuevas, y sin embargo las redes entrenadas reconocen objetos en imágenes que nunca se les mostraron. Generalizan, y los modelos más grandes se apoyan sorprendentemente poco en la pura memorización.

2. Explique con sus propias palabras por qué las leyes de escalado convencieron a las empresas de gastar tanto en inteligencia artificial.

Pista: ¿Qué significa que una ganancia sea predecible antes de pagar por ella?

Respuesta: Las leyes de escalado dicen que más datos, potencia de cálculo y tamaño del modelo dan una subida predecible de la precisión. Una empresa puede por tanto estimar de antemano qué comprará un presupuesto mayor, lo que hace que una gran inversión parezca una apuesta segura y no un juego de azar.

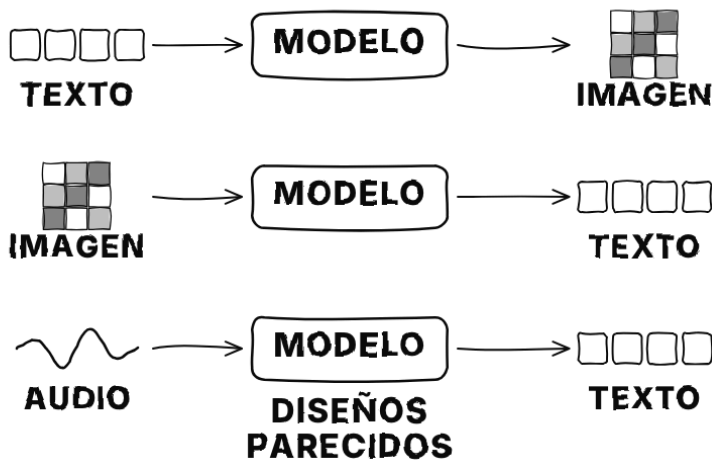
3. Nombre dos cosas que podría hacer con un modelo existente en un ordenador corriente, y diga por qué cada una necesita entender el interior del modelo.

Pista: Mire la lista de opciones que dio esta parte después de la comparación con el código abierto.

Respuesta: Por ejemplo, afinarlo con sus propios ejemplos, lo que significa cambiar sus números internos con sentido, y cuantizarlo para que funcione en un dispositivo pequeño, lo que significa saber qué números se pueden guardar de forma más gruesa sin romperlo. Ambas actúan sobre el interior del modelo, no solo sobre sus respuestas.

Aprender como una búsqueda entre perillas

Imagine que quiere un programa que distinga fotos de gatos de fotos de perros. Podría intentar escribir reglas: orejas puntiagudas, bigotes, cierta forma de nariz. En una hora estaría atascado, porque siempre hay un gato que rompe las reglas y un perro que las cumple. El libro supone que usted ya conoce el otro camino, el que funciona: reunir miles de fotos, cada una etiquetada «gato» o «perro», y dejar que un sistema se ajuste hasta que sus respuestas coincidan con las etiquetas. Esto es el aprendizaje supervisado, el núcleo de lo que se llama aprendizaje automático: reunir ejemplos del comportamiento que quiere y luego entrenar, que simplemente significa optimizar, un sistema hasta que imite ese comportamiento.

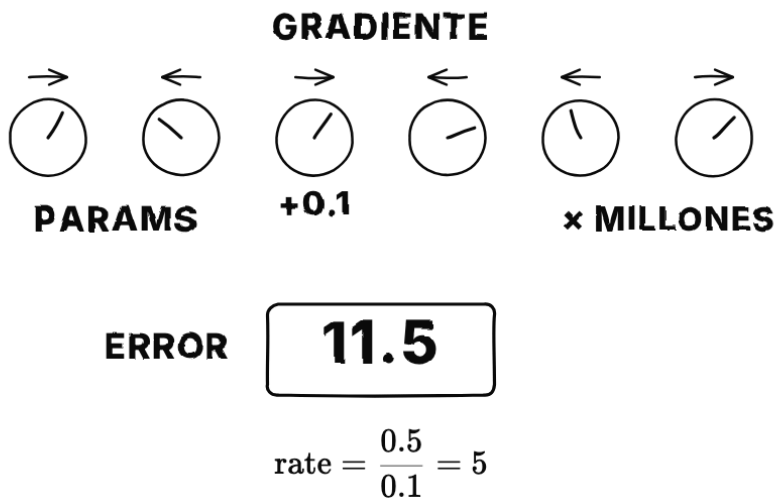


Qué entra, qué sale

La idea suena casi demasiado simple, y sin embargo llega notablemente lejos. Generar imágenes a partir de una descripción se convierte en la tarea de reunir un conjunto enorme de imágenes, cada una con su pie de foto. Imitar el idioma inglés, la tarea detrás de un asistente de chat, se convierte en la tarea de reunir una cantidad vasta de texto y aprender a predecir cada frase a partir de las anteriores. Leer una radiografía se convierte en la tarea de guardar un gran archivo de imágenes junto con la decisión que un médico tomó sobre cada una. La figura del libro ordena esas tareas según lo que entra y lo que sale. Generar una imagen toma un texto, una fila de caracteres, y devuelve una imagen, una cuadrícula ordenada de puntos de color llamados píxeles. Poner pies de foto hace lo contrario, de imagen a texto, y responder una pregunta hablada va del audio, otra fila

ordenada, esta vez de números, al texto. Lo fascinante, señala el libro, es que los modelos para todas estas tareas se diseñan con líneas muy parecidas.

En el fondo, dice el libro, aprender es buscar. Usted empieza con un programa que tiene una gran cantidad de números ajustables, como una mesa de mezclas con millones de perillas, y esas perillas se llaman parámetros. Luego gira las perillas hasta que el programa funciona lo bastante bien. La dificultad es el número de perillas: millones en un modelo pequeño, miles de millones o incluso billones en los más grandes, demasiadas para probar ajustes al azar. La aplicación de fotos de su teléfono se ajustó exactamente así, con sus perillas giradas una y otra vez sobre montones enormes de imágenes etiquetadas.



AUTODIFF DESCENSO DE GRADIENTE

Girar perillas cuesta abajo

Los modelos diferenciables hacen posible la búsqueda al permitir solo un tipo de bloque de construcción: funciones matemáticas que se pueden derivar, lo que significa que para cada una podemos calcular cómo cambia su salida cuando su entrada cambia un poco. Para una función con muchas entradas, el conjunto de estas tasas de cambio se llama gradiente. Conocer el gradiente significa saber, para todas las perillas a la vez, qué pasa con el rendimiento del programa si esa perilla se mueve ligeramente. Aquí va una pequeña imagen de ello. Suponga que el error del programa es doce, y usted descubre que subir una perilla concreta una décima bajaría el error media unidad: la tasa es cinco unidades de error por unidad de perilla, así que sabe que debe subirla, y más o menos cuánto. Con una tasa así para cada perilla, girar las perillas se convierte en una rutina automática, y las dos rutinas sobre las que se construye el libro se llaman diferenciación automática, que calcula las tasas, y descenso de gradiente, que las usa. La primera parte del libro, llamada Brújula y aguja, capítulos dos a seis, monta exactamente esta maquinaria.

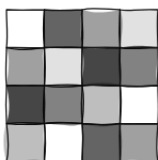
A B C x
SALTOS



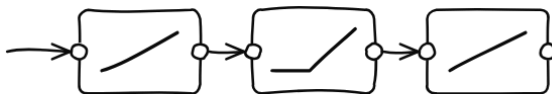
SONIDOIMAGEN



TEXTO



TENSOR



CONVOLUCIÓN

FORMAS
ENTRADA/SALIDA

Bloques que encajan como enchufes

Ver una red como una cadena de bloques diferenciables plantea dos preguntas, y la segunda figura del libro las ilustra. Primero, ¿qué tipos de datos pueden entrar y salir? Segundo, ¿qué tipos de bloques podemos usar? Ser diferenciable es una exigencia fuerte. Las letras y los números enteros saltan de un valor al siguiente sin nada en medio, y una cantidad que salta no se puede mover un poco, así que esos datos no se pueden manejar directamente. Lo que los modelos diferenciables manejan con facilidad son datos escritos como grandes matrices de números, que el libro llama tensores, como una imagen, una cuadrícula de valores de brillo. Estas matrices se transforman encadenando pasos simples, algunos rectos y otros curvados, que el libro llama transformaciones lineales y no lineales. En la figura, cada bloque se describe por lo que recibe y lo que entrega, y esas

descripciones deciden qué bloques se pueden unir con cuáles, igual que las formas de los enchufes y las tomas.

La segunda parte del libro se dedica a un bloque típico, la convolución, en los capítulos siete a nueve. Una convolución encaja con cualquier dato que venga como una fila ordenada de elementos: sonido, imágenes, texto y vídeo. Por el camino el libro presenta formas de construir modelos profundos, donde profundo significa hecho de muchos pasos en fila, y varias ideas que sostienen los grandes modelos de lenguaje de hoy: partir el texto en piezas llamadas tokens, generar una secuencia pieza a pieza, cada una a partir de las anteriores, e impedir que el modelo espíe lo que viene después. La tercera parte, llamada Por la madriguera del conejo, pasa a otros tipos de datos: conjuntos, manejados por las capas de atención y los modelos transformadores de los capítulos diez y once; grafos, redes de puntos conectados, en el capítulo doce; y capas recurrentes para secuencias en el tiempo, en el capítulo trece.

Fuera de las páginas impresas, el libro va acompañado de un sitio web donde el autor planea reunir más capítulos sobre temas no ligados a un tipo de dato: generar datos nuevos, cálculo que enciende y apaga partes de un modelo, transferir lo aprendido de una tarea a otra, y explicar qué ha aprendido un modelo. Estos capítulos se inclinan más hacia la investigación actual y pueden leerse en cualquier orden. También hay sesiones de laboratorio guiadas en forma de cuadernos, que cubren buena parte del libro así como temas avanzados como el aprendizaje contrastivo y la fusión de modelos, la combinación de dos modelos entrenados en uno.

Para recordar

El libro asume el aprendizaje supervisado: reunir ejemplos del comportamiento deseado y entrenar un sistema hasta que los imite, una idea que cubre por igual la generación de imágenes, los asistentes de chat y el diagnóstico por radiografía. Las tareas se pueden ordenar por lo que entra y lo que sale, y los modelos para ellas se diseñan de formas sorprendentemente parecidas. Aprender es una búsqueda entre una cantidad enorme de números ajustables, los parámetros, y se vuelve práctico construyendo modelos solo con bloques diferenciables. El gradiente dice, para todos los parámetros a la vez, cómo cambia el rendimiento cuando se mueve cada uno, lo que permite la diferenciación automática y el descenso de gradiente. Las tres partes del libro cubren esta maquinaria, luego las convoluciones, luego la atención, los transformadores, los grafos y las capas recurrentes, con capítulos extra y cuadernos de laboratorio en línea.

Preguntas

1. ¿Por qué el libro describe el aprendizaje como una búsqueda?
 - a. Porque el modelo busca cada respuesta en una base de datos de ejemplos
 - b. Porque el entrenamiento gira una cantidad enorme de números ajustables hasta que el programa funciona bien
 - c. Porque el programador busca en internet reglas ya hechas

- d. Porque el modelo prueba una por una todas las posiciones posibles de sus perillas
2. ¿Por qué un modelo diferenciable no puede trabajar directamente con letras o números enteros?
- a. Porque ocupan demasiada memoria
 - b. Porque no se pueden meter en un tensor de ninguna manera
 - c. Porque saltan de un valor al siguiente, así que no se pueden mover un poco
 - d. Porque los ordenadores los guardan como texto y no como números
3. Un error de doce bajaría media unidad si una perilla se subiera una décima. ¿Qué le dice eso?
- a. La tasa es cinco unidades de error por unidad de perilla, así que subirla ayuda
 - b. La tasa es media unidad, así que la perilla no importa
 - c. El error llegará a cero tras veinticuatro movimientos
 - d. La perilla debería bajarse para reducir el error
4. ¿A qué tipo de datos se adapta una convolución, según el libro?
- a. Datos sin ningún orden, como un conjunto
 - b. Datos dispuestos como una red de puntos conectados
 - c. Solo fotografías
 - d. Datos que vienen como una fila ordenada de elementos, como sonido, imágenes, texto y vídeo

Respuestas

1. **b)** El aprendizaje parte de un programa con muchísimos parámetros y los ajusta hasta que el rendimiento es suficiente; con millones de perillas, probar ajustes al azar no tiene esperanza, y por eso se necesitan los gradientes.
2. **c)** Un bloque diferenciable necesita saber cómo cambia su salida ante un cambio pequeño de la entrada, y una cantidad que salta no tiene cambios pequeños.
3. **a)** Media unidad de cambio por un movimiento de una décima es una tasa de cinco, y como el error baja, la perilla debe subirse; el gradiente da una tasa así para todas las perillas a la vez.
4. **d)** Las convoluciones se aplican siempre que los datos se puedan escribir como una secuencia ordenada de elementos; los conjuntos y los grafos quedan para las capas de atención y los modelos de grafos en la tercera parte.

Ejemplos resueltos

Leer una tasa de cambio El error de un programa es doce. Subir una perilla una décima bajaría el error media unidad. ¿Cuál es la tasa de cambio del error respecto a esa perilla, y hacia dónde hay que girarla?

El cambio en la perilla es una décima, y el cambio en el error es una bajada de media unidad.

La tasa es el cambio del error dividido por el cambio de la perilla: media entre una décima es cinco.

Como el error baja cuando la perilla sube, la tasa dice que el error cae cinco unidades por unidad de perilla en esa dirección.

Así que la perilla debe subirse, y un giro pequeño de una décima vale alrededor de media unidad de error.

Respuesta: Una tasa de cinco unidades de error por unidad de perilla; subir la perilla.

Ejercicios

1. Usando el marco de entrada y salida del libro, describa la tarea «convertir una foto en un pie de foto» y la tarea «responder una pregunta hablada».

Pista: Para cada una, diga qué tipo de dato entra y cuál sale.

Respuesta: Poner un pie de foto toma una imagen, una cuadrícula ordenada de píxeles, y devuelve un texto, una fila de caracteres.

Responder una pregunta hablada toma audio, una fila ordenada de números, y devuelve un texto. El libro señala que los modelos para esas tareas se diseñan con líneas muy parecidas.

2. Explique con sus propias palabras por qué un asistente de chat se puede construir con aprendizaje supervisado.

Pista: ¿Qué ejemplos del comportamiento deseado se pueden reunir para el lenguaje?

Respuesta: Imitar el lenguaje se puede convertir en predecir cada frase a partir de las anteriores, y cantidades enormes de texto ofrecen ejemplos infinitos de exactamente eso. Entrenar el sistema para hacer bien esas predicciones es aprendizaje supervisado sobre texto.

3. Un modelo tiene mil millones de parámetros. ¿Por qué no tiene esperanza mejorarlo probando ajustes al azar, y qué ofrece el gradiente en su lugar?

Pista: Piense en cuántas combinaciones tienen mil millones de perillas, y qué le dice un solo gradiente.

Respuesta: El número de ajustes posibles es astronómicamente grande, así que los intentos al azar casi nunca darían con uno bueno. El gradiente dice, para todos los parámetros a la vez, hacia dónde y con qué fuerza un

pequeño movimiento cambia el rendimiento, de modo que las perillas se pueden girar automáticamente en una dirección útil.

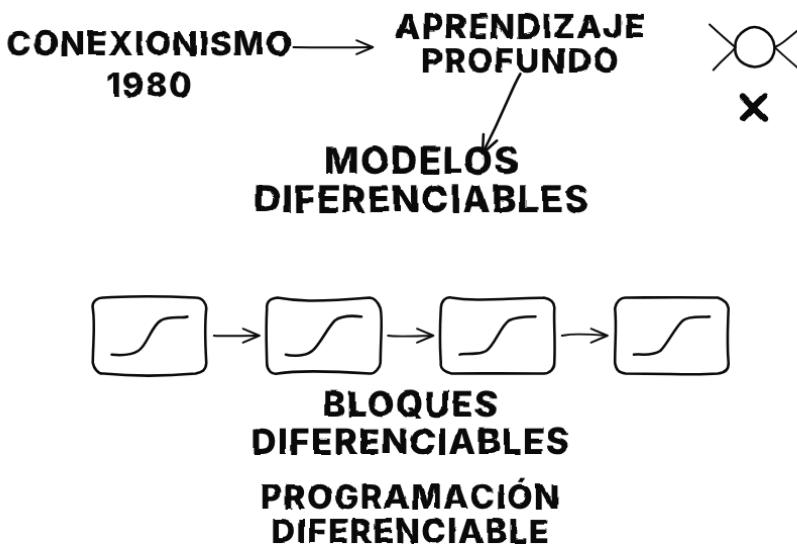
Ideas viejas a la espera de datos suficientes

Imagine una página de periódico de 1958. Un psicólogo, Frank Rosenblatt, está en los titulares por una máquina llamada perceptrón, uno de los primeros prototipos funcionales de una red neuronal, y los periodistas ya prometen máquinas que caminarán, hablarán y pensarán. El libro muestra una imagen de ese momento con un pie irónico: bombo de la inteligencia artificial, salvo que es 1958. Si eso le suena familiar, es porque el mismo tipo de titular recibe hoy a cada nuevo asistente de chat. La escena recuerda que las ideas detrás de las redes de hoy son mucho más viejas que la emoción que las rodea.

El propio nombre de red neuronal es un resto de aquellos primeros intentos del siglo veinte de imitar las neuronas biológicas, las células nerviosas del cerebro. El vocabulario se ha quedado: para ser coherente con el software que la gente realmente usa, el libro también hablará de neuronas, capas y activaciones. El interés por estas redes llegó en varias oleadas. Entre 2012 y 2017 las redes crecieron en complejidad como nunca antes, impulsadas por grandes concursos públicos, sobre todo el desafío ImageNet, en el que los programas competían por reconocer los objetos de más de un millón de fotos; el libro vuelve a él en el capítulo nueve. Una segunda gran oleada empezó en 2017 con el transformador, tema del capítulo diez. Igual que los modelos convolucionales se habían apoderado de la visión por computador unos años antes, los transformadores se apoderaron del procesamiento del lenguaje humano en muy poco tiempo. Siguiéron avances en vídeo, en grafos y en sonido, que llevaron a la emoción actual por los grandes modelos de lenguaje, los modelos que mezclan

varios tipos de datos y los modelos que generan imágenes, texto y sonido nuevos.

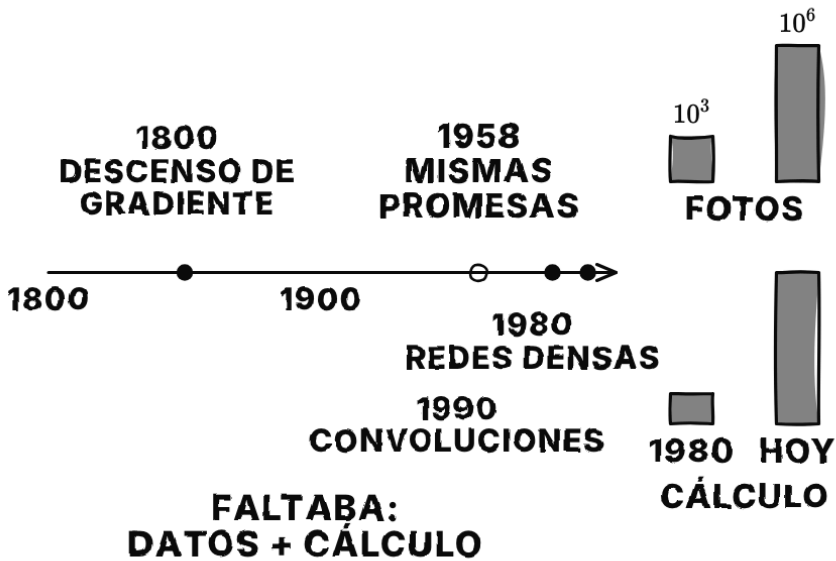
Las palabras cambiaron junto con las oleadas. En los años ochenta se hablaba de conexionismo; más tarde, aprendizaje profundo pasó a significar las redes modernas, frente a los modelos más pequeños y menos profundos del pasado. Sin embargo, todos estos términos siguen siendo vagos, porque las redes artificiales modernas apenas se parecen a las neuronas biológicas. Lo que de verdad las define, sostiene el libro, es que están construidas con bloques diferenciables, piezas cuya salida cambia suavemente cuando su entrada se mueve un poco. Por eso el libro prefiere el nombre de modelos diferenciables siempre que puede. Vistas así, las redes neuronales pertenecen a un campo más amplio llamado programación diferenciable, una disciplina joven que mezcla la informática y la optimización para estudiar programas de computador que se pueden derivar. El libro atribuye esta visión a un breve manifiesto de Yann LeCun, publicado en 2018, y a un segundo de Colin Raffel en 2021, sobre construir modelos como se construye el software de código abierto.



Los nombres cambian, los bloques quedan

Viajar por la tierra de los modelos diferenciables es también un viaje por la historia, y vale la pena oír las fechas. Ajustar un modelo simple de línea recta por descenso de gradiente, el tema del capítulo cuatro, se conocía al menos desde el siglo diecinueve. Las redes totalmente conectadas del tipo que el libro usa más adelante se remontan a los años ochenta. Los modelos convolucionales ya se usaban a finales de los años noventa. Así que las piezas principales existían hace décadas, y lo que faltaba era otra cosa: datos suficientes y potencia de cálculo suficiente para mostrar lo bien que funcionan estos modelos cuando tienen mucho de ambos, y muchos parámetros. Una forma de sentir la brecha: un teléfono en su bolsillo hoy hace más aritmética en un segundo que los computadores de los años ochenta en muchas horas, y las colecciones de fotos con que se entrenan sus aplicaciones

contienen millones de imágenes donde los primeros investigadores tenían unos pocos miles.



Piezas viejas, combustible nuevo

Pruébalo

Busque en la web los reportajes de periódico de 1958 sobre el perceptrón de Rosenblatt, y luego busque los titulares sobre un asistente de chat reciente. Póngalos uno al lado del otro y compare las promesas: máquinas que aprenden, que entienden, que pronto pensarán. Verá lo mucho que se repite el lenguaje de la emoción, mientras que lo que cambió de verdad por debajo fue sobre todo la cantidad de datos y de potencia de cálculo.

El libro cierra esta breve historia con una promesa modesta. La investigación avanza tan rápido que ningún libro puede cubrir todos los temas en profundidad, pero la esperanza es dar al lector base suficiente para orientarse por su cuenta en los artículos de investigación más nuevos.

Para recordar

Las redes neuronales son viejas: el nombre viene de los intentos del siglo veinte de imitar las células del cerebro, y en 1958 el perceptrón ya provocó una oleada de bombo. El interés llegó en oleadas, con un salto en el tamaño de las redes entre 2012 y 2017 impulsado por concursos como ImageNet, y un segundo salto con los transformadores desde 2017, que llevó a los modelos de lenguaje y generativos de hoy. Las redes modernas apenas se parecen al cerebro; su rasgo definitorio real es que están construidas con bloques diferenciables, por eso el libro las llama modelos diferenciables, parte de la programación diferenciable. El descenso de gradiente se remonta al siglo diecinueve, las redes totalmente conectadas a los años ochenta y los modelos convolucionales a los noventa; lo que faltó durante décadas fueron datos y potencia de cálculo suficientes.

Preguntas

1. ¿Por qué el libro prefiere el nombre de modelos diferenciables al de redes neuronales?

- a. Porque las redes modernas apenas se parecen a las células del cerebro, y su rasgo definitorio real es estar construidas con bloques diferenciables
 - b. Porque la palabra neuronal está protegida por una marca registrada
 - c. Porque los modelos diferenciables son un invento más nuevo que las redes neuronales
 - d. Porque las redes neuronales no se pueden usar para el lenguaje
2. Las redes totalmente conectadas datan de los años ochenta y los modelos convolucionales de los noventa. ¿Por qué su éxito llegó tanto después?
- a. Porque las ideas tuvieron que reinventarse desde cero
 - b. Porque los datos y la potencia de cálculo suficientes para mostrar su fuerza llegaron solo mucho más tarde
 - c. Porque los programadores se negaron a usarlas hasta 2017
 - d. Porque solo funcionan con fotografías
3. ¿Qué pasó con el procesamiento del lenguaje después de que aparecieran los transformadores en 2017?
- a. Abandonó poco a poco las redes neuronales
 - b. Volvió a las reglas escritas a mano
 - c. Los transformadores se apoderaron de él en muy poco tiempo, como los modelos convolucionales habían hecho con la visión
 - d. Quedó limitado solo a la traducción

4. ¿Qué sugiere la imagen de 1958 del perceptrón que muestra el libro sobre la emoción actual por los asistentes de chat?
- a. Que el perceptrón era en secreto un asistente de chat
 - b. Que nada ha mejorado desde 1958
 - c. Que el bombo es un fenómeno nuevo de la era de internet
 - d. Que las oleadas de bombo alrededor de las redes neuronales no son nuevas; las ideas son más viejas que la emoción

Respuestas

1. **a)** Los nombres viejos siguen siendo vagos, ya que las redes artificiales apenas se parecen a las neuronas biológicas; lo que comparten todas las redes modernas es que están construidas con bloques diferenciables.
2. **b)** Las piezas principales existían hace décadas; lo que faltaba eran datos, potencia de cálculo y parámetros suficientes para revelar lo bien que funcionan.
3. **c)** El libro traza el paralelo: los modelos convolucionales se adueñaron de la visión por computador, y unos años después los transformadores se adueñaron muy rápido del procesamiento del lenguaje natural.
4. **d)** El pie lo llama bombo de la IA, salvo que en 1958, un recordatorio de que esas oleadas ya han ocurrido antes y de que las ideas preceden a la atención de hoy.

Ejercicios

1. Ponga estos en el orden en que el libro dice que aparecieron: transformadores, descenso de gradiente para modelos lineales, modelos convolucionales, redes totalmente conectadas.

Pista: Uno se remonta al siglo diecinueve; otro es de 2017.

Respuesta: Descenso de gradiente para modelos lineales (siglo diecinueve), redes totalmente conectadas (años ochenta), modelos convolucionales (finales de los noventa), transformadores (2017).

2. Con sus propias palabras, ¿qué cambió entre los años noventa y los 2010 para que los viejos diseños de redes triunfaran de repente?

Pista: El libro nombra dos ingredientes, más el tamaño de los modelos.

Respuesta: La cantidad de datos disponibles y la potencia de cálculo crecieron enormemente, lo que hizo posible entrenar modelos con muchos parámetros y mostrar lo bien que funcionan los viejos diseños cuando tienen suficiente de todo.

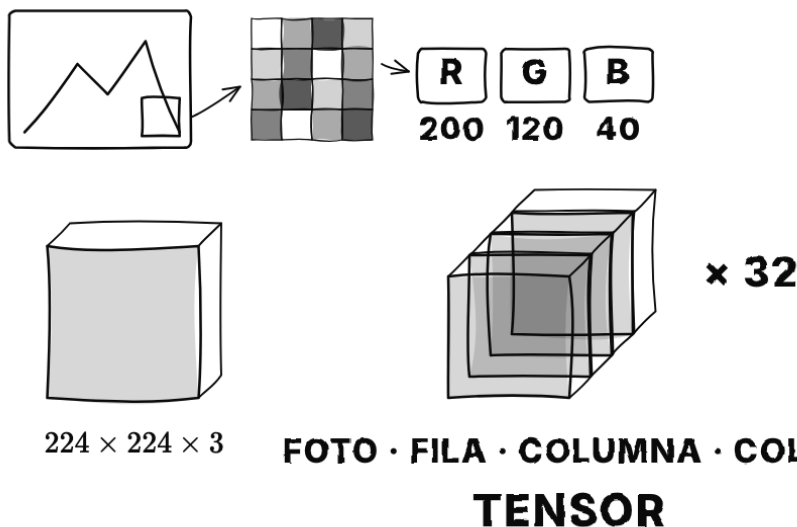
3. Explique qué es la programación diferenciable, usando la idea de la parte anterior de que un gradiente dice cómo responde una cantidad a pequeños movimientos.

Pista: ¿Qué debe permitir cada bloque de un programa así?

Respuesta: Es el estudio de programas de computador contruidos de modo que cada paso se pueda derivar, es decir, que se pueda calcular cómo responde la salida del programa a pequeños movimientos de sus entradas y parámetros. Las redes neuronales son un ejemplo, y por eso el libro las trata como modelos diferenciables.

Cajas de números y sus formas

Tome una foto de su teléfono y pregúntese qué es realmente para un computador. Acérquese lo suficiente y se rompe en cuadraditos diminutos, cada uno con tres números: cuánto rojo, cuánto verde y cuánto azul. Una foto de 224 cuadrados por 224 cuadrados es por tanto un bloque de números de 224 de alto, 224 de ancho y 3 de profundidad. Cuando la aplicación de fotos aprende de sus imágenes, toma un montón de esos bloques a la vez, digamos treinta y dos, y ese montón es una caja de números con cuatro direcciones: qué foto, qué fila, qué columna, qué color. El libro llama tensor a cada caja así, y es la cosa básica que manejan los modelos diferenciables.

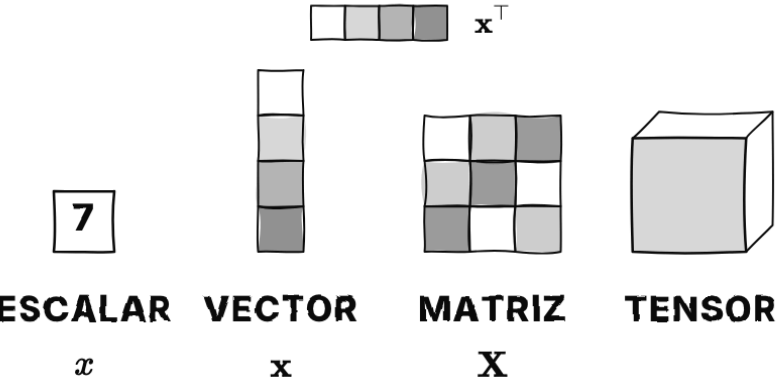


Una foto es una caja de números

Un tensor, en este libro, es simplemente una matriz de números dispuestos en cierto número de direcciones, y ese número de direcciones se llama el rango del tensor. El libro admite que los matemáticos usan la palabra tensor para algo más refinado, una clase de operador, y se disculpa por tomarla prestada, pero el significado prestado está tan extendido que lo conserva. Un solo número, rango cero, se llama escalar, y el libro lo escribe con una letra minúscula corriente. Una columna de números, rango uno, es un vector, escrito con una minúscula en negrita; cuando los mismos números se disponen en fila, el libro lo marca con una pequeña T elevada, llamada transpuesta, aunque omite la marca cuando el significado está claro. La figura del libro dibuja el vector como fila solo por comodidad; en el texto todo vector es una columna. Un rectángulo de números, rango

dos, es una matriz, escrita con una mayúscula en negrita. Para rangos por encima de dos no hay letras especiales, y las mayúsculas caligráficas, con rizos, se reservan para los conjuntos y las distribuciones de probabilidad.

RANGO 0 RANGO 1 RANGO 2 RANGO 3



LA FORMA IMPORTA

Rango: cero, uno, dos, tres

El libro hace una observación que importará en todo momento: entender una operación suele reducirse a entender la forma de cada tensor implicado. Así que usa una manera breve de indicar una forma: el nombre del tensor, un signo ondulado y la lista de sus tamaños entre paréntesis. El montón de fotos de antes se escribiría como un tensor cuya forma es el tamaño del lote, la altura, la anchura y tres. Algunos tamaños son números fijos, como los tres colores, y otros se dejan como letras, como el tamaño del lote, porque pueden cambiar

de un uso al siguiente. Contemos lo que significa esa forma: treinta y dos fotos, cada una de 224 por 224 cuadrados con 3 colores, es 224 por 224, que da 50 176 cuadrados, por 3, que da 150 528 números por foto, por 32, que da 4 816 896 números en todo el montón. Son casi cinco millones de números para un lote pequeño, y por eso las formas merecen atención.

$$X \sim (b, 224, 224, 3)$$

PUEDE CAMBIAR FIJO

$$224 \times 224 = 50,176$$

$$\times 3 = 150,528$$

$$\times 32 = 4,816,896$$

≈ 5 MILLONES

$$x \sim \mathcal{N}(0, 1)$$



Escribir y contar una forma

El mismo signo ondulado se usa también, rara vez, para decir que un número se extrae al azar de una distribución de probabilidad, por ejemplo un número extraído de la conocida curva de campana centrada en cero. El contexto siempre dice qué significado se quiere. El libro usa la notación de forma en lugar de la manera matemática

más habitual de decir que un vector es una lista de cierto número de números reales; es más corta y mantiene la vista en la forma.

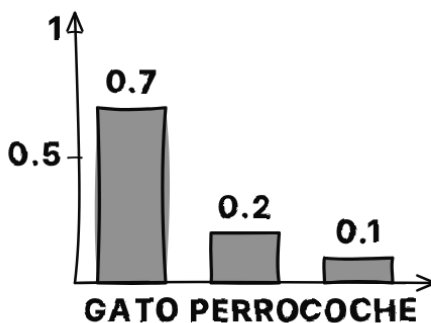
Dos etiquetas más dicen qué valores puede contener un tensor. Una marca un tensor cuyas entradas son solo ceros y unos, llamado binario, como una fila de interruptores que están apagados o encendidos. La otra marca un vector que está en el simplex: sus entradas nunca son negativas y suman exactamente uno. Eso es precisamente lo que produce un asistente de chat cuando adivina la siguiente palabra: probabilidades como siete décimas para «gato», dos décimas para «perro» y una décima para «coche», que suman uno. Para un tensor de rango mayor, como una tabla con una fila por ejemplo, el libro aplica la regla a lo largo de la última dirección, de modo que cada fila por sí sola suma uno.

BINARIO



OFF / ON

SÍMPLEX



$$0.7 + 0.2 + 0.1 = 1$$

☐	☐	☐	= 1
☐	☐	☐	= 1

Binario y símplex

Por último, el libro coloca tres pequeñas imágenes en sus márgenes para guiar al lector.

Para recordar

Un tensor significa aquí una matriz de números en cierto número de direcciones, el rango: un escalar tiene rango cero, un vector rango uno, una matriz rango dos. Los escalares son minúsculas, los vectores minúsculas en negrita (columnas, con una T elevada para la forma de fila), las matrices mayúsculas en negrita; las mayúsculas caligráficas se reservan para conjuntos y distribuciones. Entender una operación significa sobre todo conocer la forma de cada tensor, escrita con un signo ondulado y una lista de tamaños, algunos fijos y otros dejados como letras; un lote de treinta y dos fotos en color de 224 por 224 contiene casi cinco millones de números. Etiquetas adicionales marcan los tensores de solo ceros y unos, y los vectores en el símplex , cuyas entradas no son negativas y suman uno.

Preguntas

1. Un lote de fotos se guarda con cuatro direcciones: qué foto, qué fila, qué columna, qué color. ¿Cuál es su rango?
 - a. Tres, porque el color no cuenta
 - b. Cuatro, uno por cada dirección
 - c. Uno, porque es un solo lote
 - d. Dos, porque una foto es plana
2. ¿Cuál de estas listas de números está en el símplex ?
 - a. Siete décimas, dos décimas y una décima

- b. Un medio, un medio y un medio
 - c. Uno, cero y menos uno... no, menos uno no está permitido:
cero, uno y uno
 - d. Seis décimas, tres décimas y dos décimas
3. ¿Por qué el libro deja algunos tamaños de una forma como letras, como el tamaño del lote, mientras escribe otros como números?
- a. Porque las letras ocupan menos memoria
 - b. Porque los tamaños escritos como letras siempre son mayores
 - c. Porque esos tamaños pueden cambiar de un uso al siguiente, mientras que otros, como los tres colores, son fijos
 - d. Porque las letras marcan tamaños que nadie conoce todavía
4. En una tabla con una fila por ejemplo y la etiqueta de *simplex*, ¿qué debe sumar uno?
- a. Todos los números de la tabla entera juntos
 - b. Cada columna por sí sola
 - c. Solo la primera fila
 - d. Cada fila por sí sola, a lo largo de la última dirección

Respuestas

1. **b)** El rango es el número de direcciones de la matriz; foto, fila, columna y color hacen cuatro.
2. **a)** Un vector está en el *simplex* cuando ninguna entrada es negativa y las entradas suman exactamente uno; siete décimas, dos décimas y una décima suman uno, como las probabilidades de la siguiente palabra de un asistente de chat.

3. c) Los tamaños fijos como los tres colores se escriben como números; los tamaños que varían entre usos, como cuántas fotos hay en un lote, se dejan como letras.

4. d) Para tensores de rango mayor el libro aplica la regla a lo largo de la última dirección, de modo que cada fila por separado debe sumar uno.

Ejemplos resueltos

Contar los números de una forma Un lote contiene 8 fotos en color, cada una de 64 cuadrados de alto y 64 de ancho, con 3 valores de color por cuadrado. ¿Cuántos números contiene el tensor y cuál es su rango?

Las direcciones son foto, fila, columna y color, así que el rango es cuatro.

Una foto tiene 64 por 64 cuadrados, que son 4096 cuadrados.

Cada cuadrado tiene 3 valores de color, así que una foto contiene 4096 por 3, que son 12 288 números.

Ocho fotos contienen 12 288 por 8, que son 98 304 números.

Respuesta: 98 304 números en un tensor de rango cuatro.

Ejercicios

1. Dé el rango de cada uno: la temperatura exterior ahora mismo; las temperaturas diarias de una semana; una tabla de temperaturas diarias para cada semana de un año.

Pista: Cuente cuántas direcciones necesita para localizar un número.

Respuesta: Un solo número es rango cero, un escalar. Siete valores diarios en una columna son rango uno, un vector. Una tabla con una fila por semana y una columna por día es rango dos, una matriz.

2. Un modelo da las probabilidades cinco décimas, tres décimas y tres décimas para tres palabras posibles. ¿Está este vector en el simplex?

Explique.

Pista: Compruebe las dos condiciones: ninguna entrada negativa, y el total.

Respuesta: No. Ninguna de las entradas es negativa, pero suman once décimas, no uno, así que el vector no está en el simplex.

3. Una sola foto en color de 100 por 200 cuadrados se guarda como un tensor. ¿Cuántos números contiene?

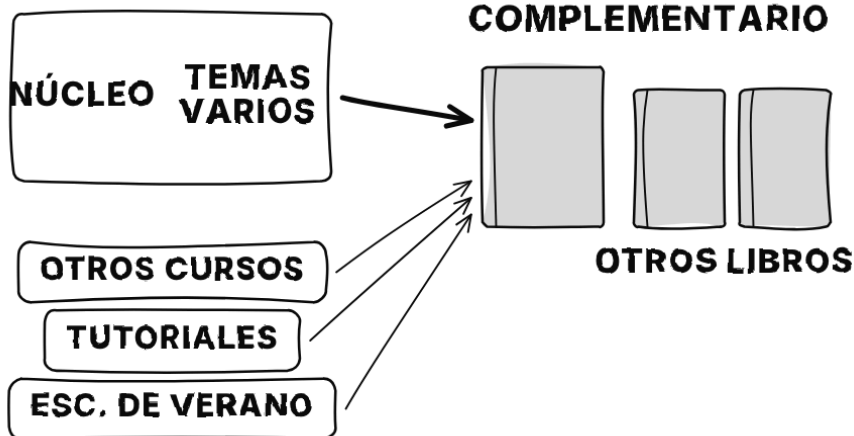
Pista: Multiplique los tamaños de todas las direcciones, incluidos los colores.

Respuesta: 100 por 200 son 20 000 cuadrados, y con 3 valores de color cada uno son 60 000 números, en un tensor de rango tres.

Qué es este libro y qué no es

Antes de una larga caminata, un buen guía le dice tres cosas: de dónde viene el camino, qué verá y qué no verá. El libro hace exactamente eso antes de empezar. Nació de un curso que el autor ha impartido durante muchos años en el máster de ciencia de datos de la Universidad Sapienza de Roma, llamado Redes neuronales para aplicaciones de ciencia de datos. Los capítulos centrales son el corazón de ese curso; los demás capítulos son temas que van y vienen de un año a otro. También se añadieron piezas de otras enseñanzas: un curso de redes neuronales para ingenieros informáticos, una introducción al aprendizaje automático para ingenieros de telecomunicaciones, tutoriales, cursos de doctorado y escuelas de verano.

CURSO SAPIENZA

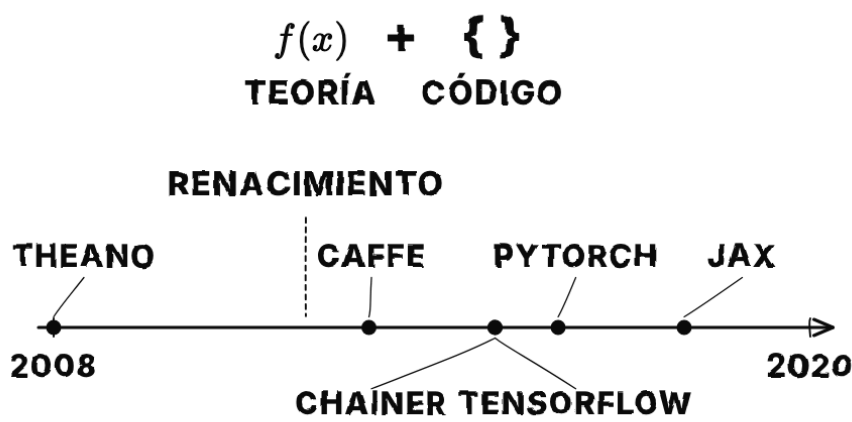


De dónde salió el libro

Ya existen varios libros recientes excelentes sobre las redes profundas modernas, y el libro lo dice abiertamente. Al principio cubre en gran medida el mismo terreno que ellos; más adelante, la forma de explicar las cosas y algunas secciones adicionales se apartan de ellos, siguiendo los intereses de investigación del propio autor. El objetivo es ofrecer un punto de vista adicional y complementario, no reemplazar a los demás.

La elección del nombre programas diferenciables lleva un mensaje: entenderlos exige tanto teoría como código. Cómo se diseña un modelo y cómo se programa se moldean constantemente uno al otro, y la diferenciación automática, el cálculo automático de tasas de cambio que vimos antes, es el mejor ejemplo. De hecho, el

renacimiento de las redes neuronales desde alrededor de 2012 debe mucho al software: bibliotecas como Theano, Caff  y Chainer, y luego las actuales TensorFlow, PyTorch y JAX, que hicieron pr ctico para cualquiera construir y entrenar redes. La aplicaci n de fotos de su tel fono y el asistente de chat que usa casi con seguridad se construyeron con herramientas de esta familia. Siempre que puede, el libro conecta sus ideas con estas herramientas, sobre todo PyTorch y JAX, pero no es un manual de programaci n, y para aprender las bibliotecas en s  remite a su propia documentaci n.



El software trajo de vuelta a las redes

Luego viene una lista honesta de lo que el libro no es. No intenta ser completo. Escoge conceptos que hoy son comunes y lo bastante generales para seguir siendo  tiles un tiempo, pero nadie puede

prever el futuro, y algunas partes pueden estar incompletas o desfasadas cuando usted las lea. Para cada idea muestra unas pocas de las variantes que se encuentran en los artículos de investigación, por ejemplo dos formas distintas de normalizar los números dentro de una red, pero existen cientos más, y el libro le invita a hojear un gran catálogo en línea de artículos y su código para el resto.

Por último, el libro trata de los componentes básicos de los modelos diferenciables, no de ejecutarlos a gran escala. Hacer que un modelo grande funcione de verdad requiere destreza de ingeniería y algo de intuición, y el libro dice poco sobre el hardware. Para la intuición, dice, nada supera la experiencia, más las entradas de blog con opiniones firmes de los practicantes; recomienda una conocida receta para entrenar redes de Andrej Karpathy, y la serie de vídeos de Karpathy que construye redes desde cero. Piénselo como aprender a cocinar: el libro enseña qué hace cada ingrediente, mientras que la maña para cocinar para cien invitados viene de la práctica en una cocina ajetreada.

Para recordar

El libro nació de un curso de máster de larga tradición en la Universidad Sapienza de Roma, ampliado con material de otros cursos, tutoriales y escuelas de verano. Al principio se superpone con otros buenos libros, luego sigue su propio camino, con la intención de ser un punto de vista complementario. Entender los programas diferenciables significa unir teoría y código; el regreso de las redes neuronales desde 2012 debe mucho a bibliotecas como TensorFlow, PyTorch y JAX, a las que el libro se vincula sin ser un manual de programación. No es completo, puede envejecer, muestra solo unas pocas de las muchas variantes de cada idea, y deja la ingeniería a gran escala, el hardware y la intuición práctica a la experiencia y a los escritos de los practicantes.

Preguntas

1. ¿Por qué dice el libro que entender los programas diferenciables exige tanto teoría como código?
 - a. Porque el diseño del modelo y su programación se moldean constantemente uno al otro, como muestra la diferenciación automática
 - b. Porque cada capítulo es un ejercicio de programación
 - c. Porque la teoría sola es demasiado corta para llenar un libro
 - d. Porque las bibliotecas no se pueden usar sin leer el libro

2. ¿A qué atribuye esta parte, en gran medida, el renacimiento de las redes neuronales desde alrededor de 2012?
 - a. A una nueva teoría del cerebro
 - b. Al final del desafío ImageNet
 - c. A bibliotecas de software potentes como TensorFlow, PyTorch y JAX
 - d. A una prohibición de las reglas escritas a mano
3. ¿Cuál de estas cosas dice el libro que no intenta hacer?
 - a. Conectar sus ideas con PyTorch y JAX
 - b. Explicar los componentes básicos de los modelos diferenciables
 - c. Ofrecer un punto de vista complementario al de otros libros
 - d. Cubrir todas las variantes de cada idea y la ingeniería de modelos muy grandes
4. ¿Adónde envía el libro al lector en busca de intuición práctica sobre el entrenamiento de redes?
 - a. Solo a sus propios cuadernos de laboratorio
 - b. A la experiencia y a las entradas de blog con opiniones firmes de los practicantes, como la receta de Karpathy
 - c. A la documentación de TensorFlow
 - d. A la historia del perceptrón

Respuestas

1. a) Cómo se diseñan los modelos y cómo se implementan se influyen mutuamente todo el tiempo, y la diferenciación automática es el ejemplo más claro.

2. c) El renacimiento debe mucho a las bibliotecas de software, desde Theano, Caffe y Chainer hasta las actuales TensorFlow, PyTorch y JAX.

3. d) Muestra solo unas pocas variantes de cada idea, dice que puede estar incompleto o desfasado, y deja la ingeniería a gran escala, el hardware y la intuición a la experiencia.

4. b) Para la intuición, nada supera la experiencia, más las entradas de blog de los practicantes; el libro recomienda la receta de entrenamiento de Andrej Karpathy y su serie de vídeos.

Ejercicios

1. Explique con sus propias palabras la diferencia entre saber qué hacen los componentes de un modelo y ser capaz de entrenar un modelo muy grande que funcione.

Pista: Piense en la comparación con la cocina de esta parte.

Respuesta: Conocer los componentes es como saber qué hace cada ingrediente; eso lo enseña el libro. Entrenar un modelo grande que funcione es como cocinar para cien invitados: requiere destreza de ingeniería, conocimiento del hardware e intuición, que vienen sobre todo de la práctica.

2. ¿Por qué podrían estar desfasadas partes del libro cuando usted lo lea, y qué hace el libro para limitarlo?

Pista: Mire cómo escoge sus conceptos.

Respuesta: La investigación avanza rápido y nadie puede prever qué métodos durarán. El libro limita el riesgo escogiendo conceptos que son a la vez comunes hoy y lo bastante generales para seguir siendo útiles un tiempo.

3. El libro no es un manual de programación, y sin embargo menciona a menudo PyTorch y JAX. ¿Por qué ambas afirmaciones

tienen sentido juntas?

Pista: ¿Qué papel juegan las bibliotecas en las explicaciones?

Respuesta: El libro usa las bibliotecas para conectar las ideas con cómo se implementan realmente, ya que teoría y código se moldean uno al otro. No enseña las bibliotecas desde cero; para eso remite a su propia documentación.

Créditos, una licencia y cortar rebanadas

Todo libro tiene una página que pocos leen, y este tiene en ella algo que vale la pena oír. El resaltado en color de sus ecuaciones proviene de un paquete de software escrito por S. T. John. Las imágenes en color de Alicia y los pequeños símbolos en blanco y negro de los márgenes tienen licencia de una empresa de imágenes de archivo, Shutterstock, mientras que las imágenes de Alicia dentro de las figuras reproducen las ilustraciones originales de John Tenniel, obtenidas a través de Wikimedia. El autor agradece a Roberto Alma sus comentarios sobre un borrador anterior y su ánimo para publicar, y a Corrado Zoccolo, Emanuele Rodolà, Marcin Słaby, Konstantin Burlachenko y Diego Sandoval sus muchas correcciones, así como a todos los que enviaron comentarios por correo electrónico.

CC BY-SA



CRÉDITOS



COMPARTIR ✓

ADAPTAR ✓

COMERCIAL ✓

CRÉDITO

MISMA LICENCIA

ESTAS CLASES

CC BY-SA

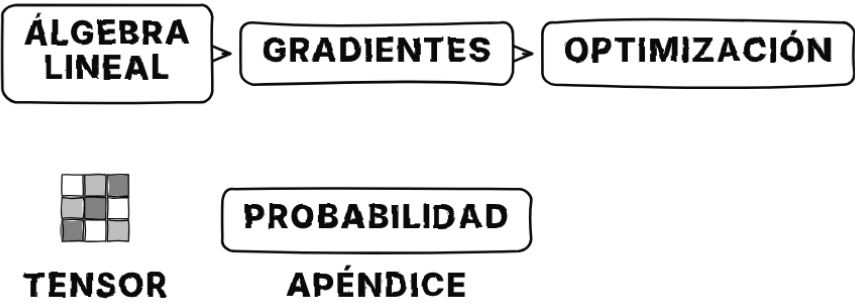
Una licencia que permite adaptar

Luego viene la licencia, y le concierne directamente. El libro se publica bajo una licencia Creative Commons llamada Attribution-ShareAlike. En sus propias palabras, permite a cualquiera «distribuir, remezclar, adaptar y construir sobre el material en cualquier medio o formato», siempre que se acredite al creador; el uso comercial está permitido; y lo que se construya sobre él debe publicarse con los mismos términos. Las clases que está escuchando son exactamente una adaptación así, y por eso también llevan esa licencia.

Con eso, el libro abre el capítulo dos, sobre los fundamentos matemáticos. Es un repaso comprimido, no un primer curso: supone que usted ya ha visto estos temas, y sobre todo fija la notación y muestra cómo encajan las piezas, con la vista en cómo se

implementan en la práctica. El capítulo tiene tres etapas, cada una construida sobre la anterior: álgebra lineal, luego gradientes para objetos con muchas dimensiones, luego cómo optimizar una función usando esos gradientes. La teoría de la probabilidad, con atención a un principio llamado máxima verosimilitud, se repasa por separado en un apéndice. El libro advierte, con buen humor, que el capítulo está lleno de definiciones y pide un poco de paciencia.

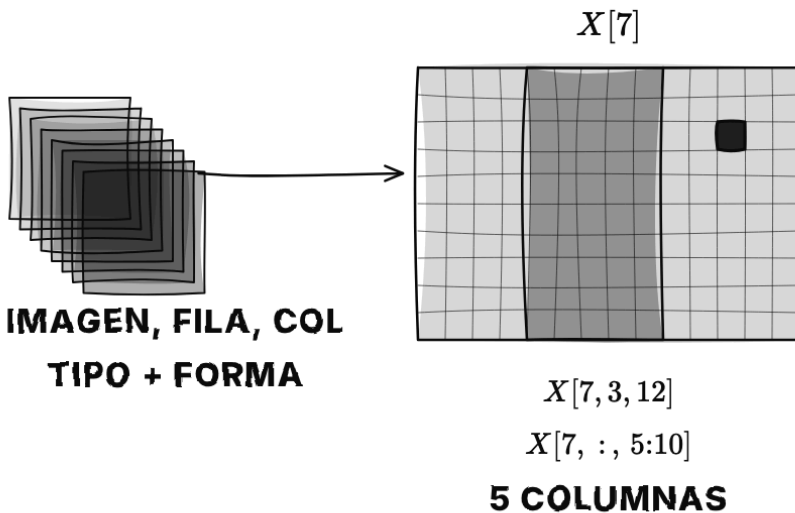
MUCHAS DEFINICIONES



El capítulo dos en tres etapas

La primera etapa, el álgebra lineal, empieza de nuevo por el tensor, la caja de números vista antes, ahora definida formalmente: una matriz en cierto número de direcciones cuyos elementos son todos del mismo tipo. Su forma se escribe con el signo ondulado y la lista de tamaños, un tamaño por dirección. Los tensores son centrales en el aprendizaje

profundo por una razón práctica: la misma operación simple aplicada a millones de números a la vez es exactamente lo que los chips gráficos hacen bien, y por eso la aplicación de fotos de su teléfono y los servidores detrás de un asistente de chat funcionan en esos chips, llamados GPU, o en otros aún más especializados. Un tensor se describe por dos cosas, el tipo de sus elementos y su forma. La mayoría de las veces los elementos son números con coma decimal, guardados en un formato que el libro trata más adelante, pero los tensores también pueden contener números enteros, por ejemplo etiquetas de clase, o cadenas de texto.



Cortar rebanadas de un tensor

A menudo solo necesita una parte de un tensor, una rebanada, y el libro sigue las convenciones de la popular biblioteca NumPy, con una

diferencia: por legibilidad cuenta las posiciones desde uno, no desde cero. Imagine un tensor que contiene un lote de imágenes en escala de gris: qué imagen, qué fila, qué columna. Fijar solo la primera posición, digamos la imagen número siete, da una rebanada que es una imagen entera, un rectángulo de filas y columnas. Fijar las tres posiciones, imagen siete, fila tres, columna doce, da un solo número, el brillo de un punto. Para rebanadas más complejas el libro separa las posiciones con comas y usa dos puntos para indicar todo a lo largo de una dirección, y un rango desde una posición hasta otra, sin incluirla. Tomemos una: imagen siete, todas las filas, y las columnas desde la cinco hasta la diez sin incluirla. Eso conserva todas las filas y diez menos cinco, es decir cinco, columnas, así que la rebanada es una franja de todas las filas por cinco columnas. Cuando la indexación recargaría una ecuación, el libro dibuja corchetes en gris claro alrededor de una expresión, con las posiciones escritas en pequeño abajo a la derecha, para que el ojo separe qué se indexa de cómo.

Para recordar

Se acreditan el coloreado de las ecuaciones, las imágenes de archivo y de Tenniel y los muchos lectores que enviaron correcciones, y el libro se publica bajo Creative Commons Attribution-ShareAlike, que permite la adaptación y el uso comercial con crédito y la misma licencia, como hacen estas clases. El capítulo dos es un repaso compacto de álgebra lineal, gradientes y optimización, con la probabilidad en un apéndice. Un tensor es una matriz en cierto número de direcciones con elementos de un solo tipo; se describe por ese tipo y su forma, y se adapta a los chips masivamente paralelos usados en el aprendizaje profundo. Las rebanadas siguen las convenciones de NumPy pero cuentan desde uno: fijar algunas posiciones da un tensor más pequeño, fijar todas da un número, y un rango como de cinco a diez conserva cinco posiciones.

Preguntas

1. Según la licencia del libro, ¿cuál de estas cosas está permitida?
 - a. Vender una versión adaptada, si se acredita al autor y la adaptación conserva la misma licencia
 - b. Vender una versión adaptada bajo una licencia propia más estricta
 - c. Adaptar el libro sin acreditar al autor
 - d. Nada más que leerlo en privado

2. Un tensor guarda imágenes en escala de gris por imagen, fila y columna. ¿Qué obtiene al fijar solo el número de imagen?
 - a. Un solo valor de brillo
 - b. Una imagen entera, un rectángulo de filas y columnas
 - c. Una columna a través de todas las imágenes
 - d. Un tensor con las mismas tres direcciones
3. ¿Por qué los tensores son tan centrales en el aprendizaje profundo, según esta parte?
 - a. Porque son la única forma de guardar texto
 - b. Porque hacen que las redes se parezcan al cerebro
 - c. Porque la misma operación sobre millones de números a la vez se adapta a las GPU y chips similares
 - d. Porque nunca usan números decimales
4. Una rebanada toma la imagen siete, todas las filas y las columnas desde la cinco hasta la diez sin incluirla. ¿Cuántas columnas conserva?
 - a. Seis
 - b. Diez
 - c. Cuatro
 - d. Cinco

Respuestas

1. a) Atribución, Compartir Igual permite la distribución, la remezcla y el uso comercial, siempre que se acredite al creador y el material adaptado lleve la misma licencia.

- 2. b)** Fijar una posición elimina esa dirección y deja una rebanada de las otras dos: una imagen completa.
- 3. c)** Los tensores se adaptan al hardware masivamente paralelo, como los chips gráficos, que aplican una operación a muchísimos números al mismo tiempo.
- 4. d)** Un rango hasta el final, sin incluirlo, conserva diez menos cinco, es decir cinco, posiciones.

Ejemplos resueltos

El tamaño de una rebanada Un tensor guarda 50 imágenes en escala de gris, cada una de 28 filas por 28 columnas. Usted toma la imagen tres, las filas desde la 1 hasta la 15 sin incluirla, y todas las columnas. ¿Cuántos números hay en la rebanada y qué forma tiene?

Fijar la imagen tres elimina la primera dirección, así que a la rebanada le quedan filas y columnas.

Las filas desde la 1 hasta la 15 sin incluirla conservan 15 menos 1, es decir 14 filas.

Todas las columnas conserva las 28 columnas.

La rebanada es de 14 filas por 28 columnas, que es 14 por 28, es decir 392 números.

Respuesta: Un rectángulo de 14 filas por 28 columnas, 392 números.

Ejercicios

1. En el mismo tensor de 50 imágenes de 28 por 28, ¿cuántos números obtiene al fijar la imagen, la fila y la columna?

Pista: ¿Qué queda cuando todas las direcciones están fijadas?

Respuesta: Un número: el brillo de un solo punto. Fijar todas las posiciones no deja ninguna dirección, así que el resultado es un escalar.

2. Usted toma las 50 imágenes, solo la fila 10, y las columnas desde la 3 hasta la 13 sin incluirla. ¿Cuál es la forma de la rebanada y cuántos números contiene?

Pista: Fijar la fila elimina esa dirección; cuente las columnas como fin menos inicio.

Respuesta: La dirección de las filas desaparece, y quedan 50 imágenes por 13 menos 3, es decir 10 columnas. La rebanada es de 50 por 10, que contiene 500 números.

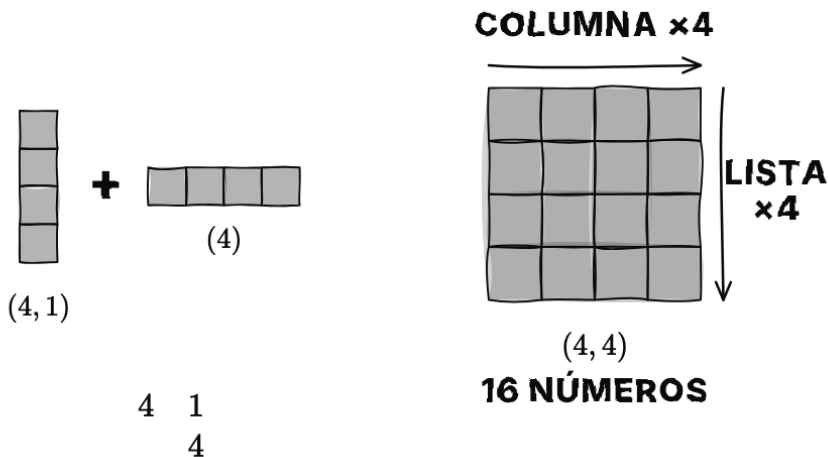
3. Alguien adapta este libro en un curso de vídeo de pago. Según la licencia, ¿qué dos cosas debe hacer?

Pista: Mire las dos condiciones que acompañan al permiso.

Respuesta: Debe acreditar al creador del libro, y debe publicar su material adaptado bajo los mismos términos de Atribución, Compartir Igual. El uso comercial en sí está permitido.

Flechas, longitudes y el producto punto

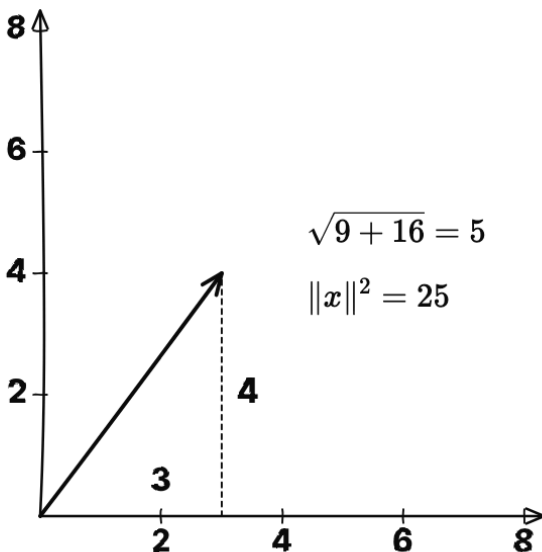
Aquí hay una pequeña trampa que atrapa a los programadores cada día. Tome cuatro números apilados en columna, y cuatro números puestos como una lista plana, y pida a una biblioteca como PyTorch que los sume. Usted espera cuatro sumas, una por cada posición. Lo que obtiene es una tabla de cuatro filas por cuatro columnas, dieciséis números, sin ningún mensaje de error. El libro muestra exactamente esto en un pequeño cuadro de código, y el culpable tiene nombre: broadcasting.



Broadcasting: cuatro más cuatro da dieciséis

El broadcasting es una regla, tomada de la biblioteca NumPy, por la que una biblioteca alinea las formas de dos tensores desde la derecha y repite en silencio uno de ellos allí donde eso hace coincidir las formas. Una columna de cuatro es, para la biblioteca, un tensor de dos direcciones de cuatro filas por una columna; una lista plana de cuatro tiene una sola dirección. Alineadas desde la derecha, la única columna se estira a lo largo de cuatro columnas y la lista se copia hacia abajo en cuatro filas, y cada par se suma. Los libros de matemáticas separan con cuidado los vectores columna de los vectores fila, y el libro sigue esa costumbre, pero en el código una columna, una fila y una lista plana son tres formas distintas, y mezclarlas es como ocurren estas sorpresas. El consejo del libro es simplemente tenerlo en mente, porque casi todos los frameworks se comportan igual.

La mayoría de los modelos del libro se construyen con ingredientes muy sencillos: sumas, multiplicaciones y unas pocas funciones curvadas como la exponencial, senos y cosenos, y raíces cuadradas. Los vectores, los tensores con una dirección, tienen su propia aritmética, que los matemáticos llaman espacio vectorial. Dos vectores cualesquiera de la misma forma se pueden combinar escalando cada uno y sumándolos, posición por posición, para dar un tercer vector. Por ejemplo, tome el vector dos, uno y súmele dos veces el vector uno, tres: la posición uno da dos más dos, que es cuatro, y la posición dos da uno más seis, que es siete, así que el resultado es cuatro, siete.



Vectores como flechas

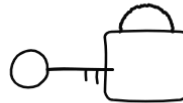
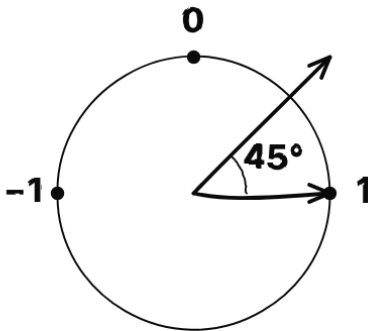
Ayuda ver un vector como una flecha desde el origen hasta un punto. Sumar dos flechas significa entonces ponerlas punta con cola, lo que completa un paralelogramo, y la longitud de una flecha, su distancia al origen, se llama su norma: eleve al cuadrado cada entrada, sume los cuadrados y saque la raíz cuadrada. Para la flecha tres, cuatro, los cuadrados son nueve y dieciséis, suman veinticinco, y la raíz cuadrada es cinco. La norma al cuadrado, simplemente la suma de las entradas al cuadrado, veinticinco aquí, resulta especialmente práctica.

La operación más importante sobre vectores es el producto interno, también llamado producto punto: multiplique los dos vectores posición por posición y sume los resultados. La respuesta es siempre un solo número. El propio ejemplo del libro multiplica el vector una

décima, cero, menos tres décimas por el vector menos cuatro, cinco centésimas, una décima. Los tres productos son menos cuatro décimas, cero y menos tres centésimas, y su suma es menos cuarenta y tres centésimas.

$$(0.1)(-4) + (0)(0.05) + (-0.3)(0.1) \\ = -0.4 + 0 - 0.03 = -0.43$$

$$a \cdot b = \|a\| \|b\| \cos \theta$$



PLANTILLA

$$\cos \theta = \frac{1}{1.41} \approx 0.71$$

El producto punto y el ángulo

El producto punto tiene un bello significado geométrico: es igual a la longitud de la primera flecha, por la longitud de la segunda, por el coseno del ángulo entre ellas. Así que si ambas flechas tienen longitud uno, el producto punto es simplemente el coseno de su ángulo, y entonces se llama similitud coseno. Vale uno cuando las flechas apuntan en la misma dirección, menos uno cuando apuntan en direcciones opuestas, y cero cuando forman un ángulo recto, lo que se llama ortogonal. Tome las flechas uno, cero y uno, uno: su producto

punto es uno, sus longitudes son uno y alrededor de uno coma cuarenta y uno, así que el coseno es alrededor de siete décimas, que es el coseno de cuarenta y cinco grados, exactamente el ángulo entre ellas.

Dele la vuelta a esto. Si una flecha de longitud uno está fija, ¿qué otra flecha de longitud uno da el mayor producto punto? La respuesta es la propia flecha: el coseno es máximo, uno, cuando el ángulo es cero. Esta es la razón por la que el producto punto está por todas partes en el libro. Más adelante el primer vector será una entrada y el segundo un conjunto de parámetros ajustables, y su producto punto es máximo cuando la entrada resuena con los parámetros, encajando con ellos como una llave en una cerradura. El libro llama a esto coincidencia de plantilla. Una forma común en que las funciones de búsqueda encuentran elementos relacionados, como fotos parecidas a una que usted elige, es convertir cada elemento en un vector y compararlos exactamente con esta similitud coseno.

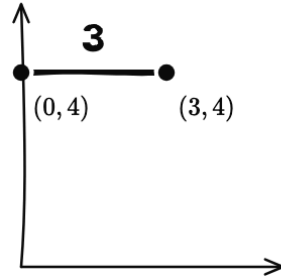
Dos últimas observaciones ahorrarán esfuerzo más adelante. La suma de todas las entradas de un vector es su producto punto con un vector hecho solo de unos: para dos, cinco, tres, multiplicar por uno, uno, uno y sumar da diez. Y la distancia al cuadrado entre dos vectores se puede escribir solo con productos punto: el producto punto del primero consigo mismo, más el del segundo consigo mismo, menos dos veces el producto punto de ambos. Compruébelo con las flechas tres, cuatro y cero, cuatro: veinticinco más dieciséis menos dos veces dieciséis da nueve, y en efecto los dos puntos están a tres de distancia, y tres al cuadrado es nueve. Cuando el segundo vector es todo ceros, la regla dice que la longitud al cuadrado de un vector es su producto punto consigo mismo.

$$(2, 5, 3) \cdot (1, 1, 1) = 10$$

$$\|a - b\|^2 = a \cdot a + b \cdot b - 2a \cdot b$$

$$25 + 16 - 2 \cdot 16 = 9 = 3^2$$

$$b = 0, \quad \|a\|^2 = a \cdot a$$



Distancia a partir de productos punto

Pruébalo

En una hoja de papel cuadriculado, dibuje la flecha desde el origen hasta el punto tres, cuatro y médala con una regla: mide cinco cuadros. Luego dibuje las flechas uno, cero y uno, uno y mida el ángulo entre ellas con un transportador: cuarenta y cinco grados. Por último multiplique tres, cuatro por sí mismo, posición por posición, y sume: veinticinco, que es cinco al cuadrado, la longitud al cuadrado que midió.

Para recordar

Las bibliotecas hacen broadcasting: alinean las formas desde la derecha y repiten tensores para que coincidan, así que sumar una columna de cuatro a una lista plana de cuatro da en silencio una tabla de cuatro por cuatro. Los vectores se pueden escalar y sumar posición por posición; vistos como flechas, sumar completa un paralelogramo y la norma es la longitud, la raíz cuadrada de la suma de cuadrados. El producto punto multiplica dos vectores posición por posición y suma, dando un número; es igual a las dos longitudes por el coseno del ángulo, así que para vectores unitarios es la similitud coseno, de uno a menos uno, donde cero significa ortogonal. Para un vector unitario fijo el producto punto es máximo con el propio vector, la coincidencia de plantilla en la que se apoyarán entradas y parámetros. La suma de las entradas es el producto punto con un vector de unos, y las distancias al cuadrado se pueden escribir solo con productos punto.

Preguntas

1. ¿Por qué sumar una columna de cuatro números a una lista plana de cuatro números da dieciséis números en una biblioteca como PyTorch?
 - a. Porque la biblioteca alinea las formas desde la derecha y repite cada tensor para que coincidan, produciendo una tabla de cuatro por cuatro

- b. Porque la biblioteca multiplica en lugar de sumar
 - c. Porque una columna siempre contiene dieciséis números
 - d. Porque la biblioteca informa de un error y rellena el hueco con ceros
2. Dos flechas de longitud uno tienen un producto punto de cero. ¿Qué le dice eso?
- a. Apuntan en la misma dirección
 - b. Apuntan en direcciones opuestas
 - c. Son ortogonales, forman un ángulo recto
 - d. Una de ellas es toda ceros
3. Con una flecha unitaria fija, ¿qué flecha unitaria da el mayor producto punto con ella?
- a. Cualquier flecha en ángulo recto con ella
 - b. La flecha que apunta en dirección opuesta
 - c. La flecha de todo unos
 - d. La propia flecha
4. ¿Cómo se puede escribir la suma de todas las entradas de un vector como un producto punto?
- a. Como su producto punto consigo mismo
 - b. Como su producto punto con un vector hecho solo de unos
 - c. Como su producto punto con un vector de ceros
 - d. Como la raíz cuadrada de su producto punto consigo mismo

Respuestas

1. a) El broadcasting alinea las formas desde la derecha y repite tensores en silencio; la columna de cuatro filas por una columna y la lista de cuatro se

combinan en cuatro filas por cuatro columnas.

2. c) Para vectores unitarios el producto punto es el coseno del ángulo, y un coseno de cero significa un ángulo recto.

3. d) El coseno es máximo, uno, cuando el ángulo es cero, así que la mejor coincidencia es la misma flecha; esta es la coincidencia de plantilla que usarán entradas y parámetros.

4. b) Multiplicar cada entrada por uno y sumar simplemente suma las entradas: para dos, cinco, tres esto da diez.

Ejemplos resueltos

El producto punto del libro Halle el producto punto del vector una décima, cero, menos tres décimas con el vector menos cuatro, cinco centésimas, una décima.

Multiplique las primeras entradas: una décima por menos cuatro es menos cuatro décimas.

Multiplique las segundas entradas: cero por cinco centésimas es cero.

Multiplique las terceras entradas: menos tres décimas por una décima es menos tres centésimas.

Sume los tres productos: menos cuatro décimas más cero menos tres centésimas es menos cuarenta y tres centésimas.

Respuesta: Menos cuarenta y tres centésimas, un solo número.

Ejercicios

1. ¿Cuál es la longitud, la norma, de la flecha seis, ocho?

Pista: Eleve al cuadrado cada entrada, sume los cuadrados y luego saque la raíz cuadrada.

Respuesta: Diez: seis al cuadrado es treinta y seis, ocho al cuadrado es sesenta y cuatro, suman cien, y la raíz cuadrada de cien es diez.

2. Use la regla del producto punto para distancias para hallar a qué distancia están los puntos uno, dos y cuatro, seis.

Pista: Calcule el producto punto de cada vector consigo mismo y el producto punto de los dos, luego combine: el primero más el segundo menos dos veces el mixto.

Respuesta: El primero consigo mismo da uno más cuatro, cinco; el segundo consigo mismo da dieciséis más treinta y seis, cincuenta y dos; el producto mixto es cuatro más doce, dieciséis. Cinco más cincuenta y dos menos treinta y dos es veinticinco, la distancia al cuadrado, así que los puntos están a cinco de distancia.

3. ¿Cuál es la similitud coseno de las flechas uno, cero y cero, uno, y qué ángulo significa?

Pista: Ambas flechas ya tienen longitud uno.

Respuesta: Su producto punto es uno por cero más cero por uno, que es cero. Como ambas tienen longitud uno, la similitud coseno es cero: son ortogonales, a noventa grados.

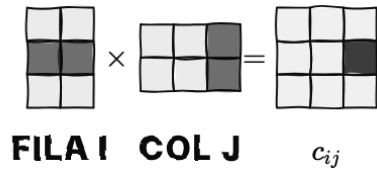
Tablas de números que actúan

Anote una tabla pequeña: tres pisos en venta, y para cada uno dos números, su superficie y su número de habitaciones. Tiene tres filas y dos columnas, un rectángulo de números, que los matemáticos llaman matriz. Léala fila por fila y aparece algo útil: cada fila es un piso, un pequeño vector por sí mismo, y la tabla es simplemente esos tres vectores apilados uno sobre otro. El libro llama a esa pila un lote de vectores de datos, y de ahora en adelante los modelos se escribirán, tanto en papel como en código, para trabajar con lotes enteros a la vez. Cuando la aplicación de fotos de su teléfono aprende, nunca mira una imagen a la vez; toma un lote, una pila de imágenes, y procesa toda la pila de una vez.

$$(a \times b)(b \times c) = (a \times c)$$

80	3
55	2
120	4

**UN LOTE DE
3 VECTORES**



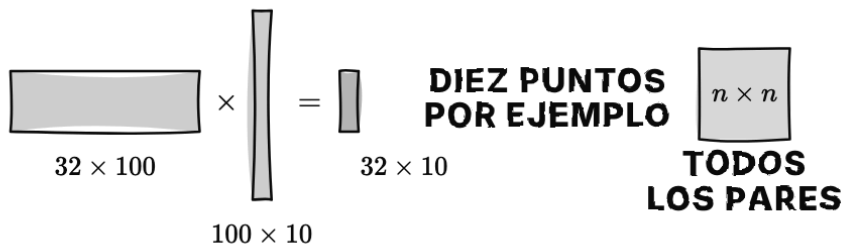
$$c_{ij} = \text{fila}_i \cdot \text{col}_j$$

Una tabla es una pila de vectores

La operación clave sobre matrices es la multiplicación, y tiene una regla sobre las formas. Puede multiplicar una tabla de a filas y b columnas por una tabla de b filas y c columnas, y el resultado tiene a filas y c columnas: los dos tamaños interiores deben coincidir, y desaparecen. Cada entrada del resultado es un producto punto: la entrada de la fila i y la columna j es el producto punto de la fila i de la primera tabla con la columna j de la segunda. Hagamos uno a mano. Tome la primera tabla con filas uno, dos y tres, cuatro, y la segunda con filas cinco, seis y siete, ocho. La entrada superior izquierda es la fila uno, dos por la columna cinco, siete: cinco más catorce, diecinueve. La entrada superior derecha es uno, dos por seis, ocho: seis más dieciséis, veintidós. La fila inferior da tres por cinco más cuatro por siete, cuarenta y tres, y tres por seis más cuatro por ocho,

cincuenta. Así que el producto es diecinueve, veintidós arriba y cuarenta y tres, cincuenta abajo.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$



Multiplicar a mano, luego en masa

Cuando el segundo factor es un solo vector, el producto es una matriz por un vector, que da un vector. Ahora la visión por lotes da fruto. Si cada fila de una tabla es un vector de datos y multiplica la tabla por la tabla volteada de unos pesos, calcula el producto punto de cada vector de datos con los pesos, todo en una sola operación. Un lote de treinta y dos ejemplos con cien números cada uno, multiplicado por pesos de forma cien por diez, da treinta y dos por diez: diez puntuaciones para cada uno de los treinta y dos ejemplos, a la vez. Y una tabla multiplicada por su propia versión volteada, que da una tabla

cuadrada de n filas y n columnas, contiene el producto punto de cada par de filas, así que cada par de ejemplos se compara de una vez.

Un segundo tipo de producto es mucho más simple: el producto de Hadamard, que multiplica dos tablas de la misma forma posición por posición. Carece de las ricas propiedades de la verdadera multiplicación de matrices, pero las redes lo usan todo el tiempo para enmascarar, es decir poner algunas entradas a cero, o para escalar entradas. Multiplique la tabla uno, dos sobre tres, cuatro posición por posición por la tabla uno, cero sobre cero, uno, y las entradas fuera de la diagonal desaparecen: uno, cero sobre cero, cuatro. Algunas familias recientes de modelos se apoyan mucho en estas interacciones multiplicativas, y el libro volverá a ellas.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$$

MÁSCARA

ENTRADA A ENTRADA

EXP DE MATRIZ

$$e^0 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\exp(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

TORCH.LINALG

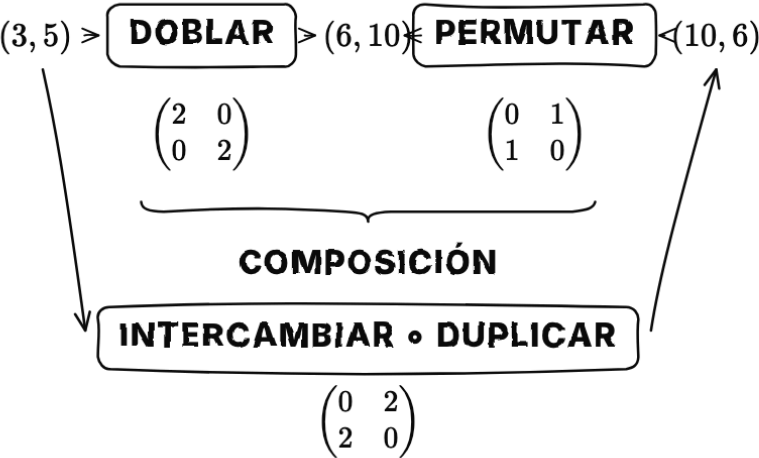
SUMA POR FILA → POR PISO
SUMA POR COLUMNA → TOTALES

El libro también advierte de una trampa de notación. Escribir la exponencial de una matriz suele significar aplicar la exponencial a cada entrada por separado, y eso funciona para tensores de cualquier forma. Pero el álgebra lineal también tiene una verdadera exponencial de matriz, definida solo para matrices cuadradas como una suma infinita de la matriz multiplicada por sí misma cada vez más veces, con cada término dividido por un factorial creciente. Las dos son muy distintas. Para una tabla llena de ceros, la exponencial entrada por entrada da una tabla llena de unos, mientras que la verdadera exponencial de matriz da unos solo en la diagonal y ceros en el resto. Por eso bibliotecas como PyTorch guardan las operaciones genuinamente matriciales, como la exponencial de matriz, las inversas y los determinantes, en un módulo separado de álgebra lineal, como muestra el cuadro de código del libro.

Por último, una reducción como una suma o un promedio recorre toda una dirección de una tabla, salvo que se den límites. En código, es un método del tensor con un argumento que nombra la dirección: sumar la tabla de pisos a lo largo de las columnas da, para cada piso, la superficie más las habitaciones; sumar a lo largo de las filas da los totales de los tres pisos.

¿Por qué la multiplicación de matrices se define de esta extraña manera fila por columna, y no posición por posición? El libro responde en un recuadro. Una función sobre vectores es lineal cuando escalar y sumar primero las entradas da el mismo resultado que escalar y sumar las salidas. Toda función así se puede escribir como una matriz, y aplicar la función a un vector es exactamente multiplicar la matriz por el vector. Multiplicar dos matrices corresponde entonces a hacer una función después de la otra, lo que se llama composición. Por ejemplo, si una matriz duplica cada entrada y otra intercambia las

dos entradas, su producto es una sola matriz que duplica y luego intercambia: de tres, cinco obtiene diez, seis, se calcule como se calcule.



Las matrices componen funciones

Para recordar

Una matriz es una tabla de filas y columnas, que se ve mejor como una pila de vectores de datos, un lote, y los modelos se escriben para trabajar con lotes enteros a la vez. Las matrices se multiplican cuando los tamaños interiores coinciden; cada entrada del resultado es el producto punto de una fila de la primera con una columna de la segunda, así que un producto calcula muchos productos punto a la vez. El producto de Hadamard multiplica posición por posición y se usa para enmascarar y escalar; una exponencial escrita sobre una matriz significa entrada por entrada, a diferencia de la verdadera exponencial de matriz de las matrices cuadradas. Las reducciones como las sumas recorren toda una dirección, elegida en código con un argumento de eje. La multiplicación de matrices se define así porque las matrices son funciones lineales: multiplicar por un vector aplica la función, y multiplicar dos matrices compone dos funciones.

Preguntas

1. Una tabla de 32 filas y 100 columnas se multiplica por una tabla de 100 filas y 10 columnas. ¿Cuál es la forma del resultado?
 - a. 100 filas por 100 columnas
 - b. 32 filas por 10 columnas
 - c. 10 filas por 32 columnas
 - d. Las tablas no se pueden multiplicar

2. ¿Qué contiene cada entrada de un producto de matrices?
 - a. El producto punto de una fila de la primera tabla con una columna de la segunda
 - b. El producto de las dos entradas en la misma posición
 - c. La suma de todas las entradas de ambas tablas
 - d. La entrada más grande de la fila
3. Para una tabla llena de ceros, ¿en qué difieren la exponencial entrada por entrada y la verdadera exponencial de matriz?
 - a. Ambas dan una tabla de ceros
 - b. Ambas dan una tabla de unos
 - c. La de entrada por entrada da unos solo en la diagonal
 - d. La de entrada por entrada da todo unos; la exponencial de matriz da unos en la diagonal y ceros en el resto
4. ¿Por qué la multiplicación de matrices se define fila por columna y no posición por posición?
 - a. Porque es más rápida en los chips gráficos
 - b. Porque los productos posición por posición no existen
 - c. Porque las matrices representan funciones lineales, y este producto corresponde a aplicar una función después de otra
 - d. Porque siempre da un resultado cuadrado

Respuestas

1. **b)** Los tamaños interiores, 100 y 100, coinciden y desaparecen; el resultado conserva los tamaños exteriores, 32 por 10, diez puntuaciones para cada uno de los treinta y dos ejemplos.

- 2. a)** La entrada de la fila i y la columna j es el producto punto de la fila i de la primera tabla con la columna j de la segunda.
- 3. d)** La exponencial de cero en cada entrada es uno, mientras que la verdadera exponencial de matriz de la matriz cero deja solo los unos de la diagonal; las dos operaciones son muy distintas.
- 4. c)** Toda función lineal es una matriz; multiplicar por un vector la aplica, y multiplicar dos matrices compone las dos funciones.

Ejemplos resueltos

Multiplicar dos tablas pequeñas Multiplique la tabla con filas dos, cero y uno, tres por la tabla con filas uno, cuatro y dos, uno.

Arriba a la izquierda: la fila dos, cero con la columna uno, dos da dos por uno más cero por dos, que es dos.

Arriba a la derecha: la fila dos, cero con la columna cuatro, uno da ocho más cero, que es ocho.

Abajo a la izquierda: la fila uno, tres con la columna uno, dos da uno más seis, que es siete.

Abajo a la derecha: la fila uno, tres con la columna cuatro, uno da cuatro más tres, que es siete.

Respuesta: El producto tiene filas dos, ocho y siete, siete.

Ejercicios

1. ¿Se puede multiplicar una tabla de 3 filas por 2 columnas por otra tabla de 3 filas por 2 columnas? ¿Y por su propia versión volteada?

Pista: Compruebe si los tamaños interiores coinciden.

Respuesta: No: los tamaños interiores serían 2 y 3, que difieren. Sí con su versión volteada, que es de 2 por 3: los tamaños interiores son ambos 2, y el resultado es de 3 por 3, los productos punto de cada par de filas.

2. Multiplique la tabla cinco, seis sobre siete, ocho posición por posición, el producto de Hadamard, por la máscara cero, uno sobre uno, cero.

Pista: Cada entrada se multiplica solo por la entrada en la misma posición.

Respuesta: Cero, seis sobre siete, cero: la máscara conserva las entradas fuera de la diagonal y pone a cero las de la diagonal.

3. Una matriz duplica ambas entradas de un vector; otra intercambia sus dos entradas. Aplique «duplicar, luego intercambiar» al vector tres, cinco, y compruebe que «intercambiar, luego duplicar» da aquí el mismo resultado.

Pista: Haga cada paso por turno.

Respuesta: Duplicar da seis, diez, e intercambiar da diez, seis.

Intercambiar primero da cinco, tres, y duplicar da diez, seis. Ambas composiciones dan diez, seis, como lo haría su única matriz combinada.

Contar el trabajo

Multiplique dos tablas a mano, una de dos filas y tres columnas y otra de tres filas y cuatro columnas. El resultado tiene dos filas y cuatro columnas, ocho entradas, y cada entrada es un producto punto de tres pares, así que hace ocho por tres, veinticuatro multiplicaciones. Ahora duplique cada tamaño: cuatro filas por seis, por seis por ocho. El resultado tiene treinta y dos entradas de seis multiplicaciones cada una, ciento noventa y dos en total, ocho veces más que antes, aunque cada tamaño solo se duplicó. Cómo crece el trabajo a medida que crecen los tamaños es lo que los informáticos llaman complejidad, y el libro usa la multiplicación de matrices para presentarla.

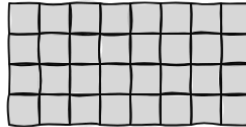
$$(2 \times 3)(3 \times 4)$$



$$8 \times 3 = 24$$

× 8

$$(4 \times 6)(6 \times 8)$$



$$32 \times 6 = 192$$

$$(a \times b)(b \times c)$$

A·C PRODUCTOS PUNTO, LONGITUD B

$$\text{tiempo} \sim a b c$$

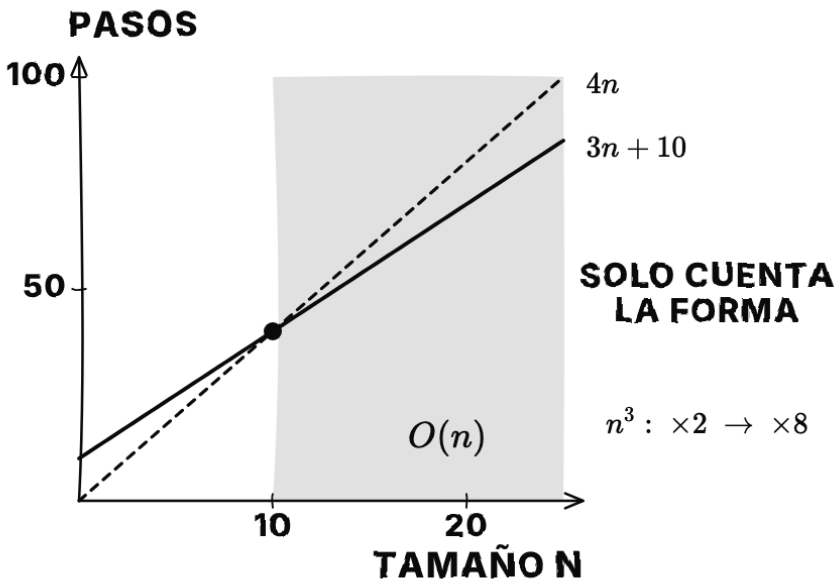
$$\text{memoria} \sim a c$$

El doble de tamaño, ocho veces el trabajo

Aplicando la definición directamente, una tabla de a filas y b columnas por una de b filas y c columnas necesita a por c productos punto, cada uno de longitud b . El número de operaciones se llama complejidad temporal. La memoria que necesita el cálculo, para un programa sencillo de un paso a la vez, es simplemente el tamaño de la tabla resultado, y eso se llama complejidad espacial.

Para hablar del crecimiento sin perderse en los detalles de máquinas concretas, la informática usa la notación O grande, donde la O viene de la palabra alemana *Ordnung*, orden, como en orden de aproximación. Decir que el trabajo de un programa es de orden g significa esto: a partir de cierto tamaño, el trabajo nunca supera algún múltiplo fijo de g . Por ejemplo, un programa que necesita tres veces el

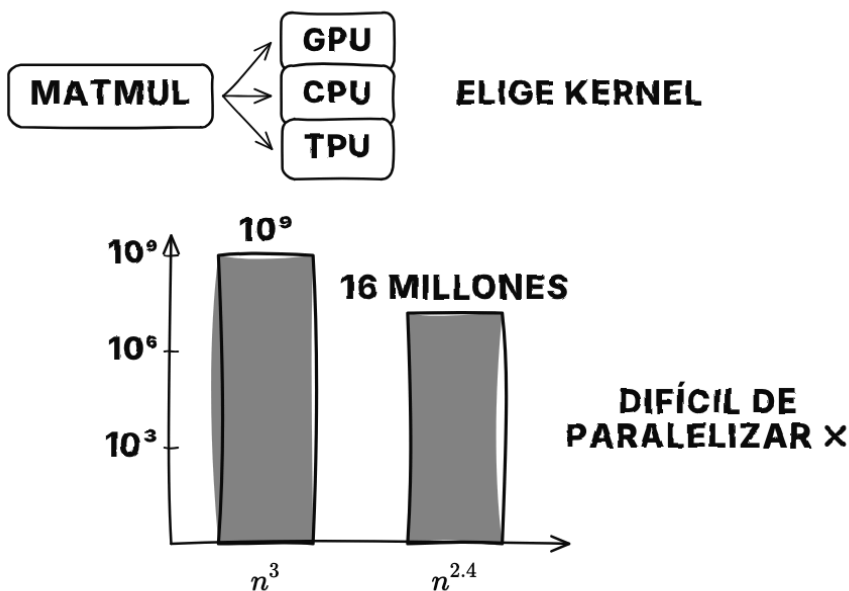
tamaño más diez pasos es de orden el tamaño, porque una vez que el tamaño llega a diez, tres veces el tamaño más diez nunca es más que cuatro veces el tamaño. Más allá de ese punto, las constantes y los pequeños términos extra dejan de importar, y solo cuenta la forma general del crecimiento; esto se llama análisis asintótico. En este lenguaje, la multiplicación de matrices ingenua es de orden a por b por c , y crece en proporción a cada uno de los tres tamaños. Para dos tablas cuadradas de n filas y n columnas eso es n por n por n , y decimos que la multiplicación es cúbica: duplique n y el trabajo crece ocho veces, exactamente como en la cuenta a mano.



O grande: acotado a partir de cierto tamaño

El razonamiento con O grande es elegante e importante, pero el libro advierte que no decide qué método es realmente más rápido. Eso

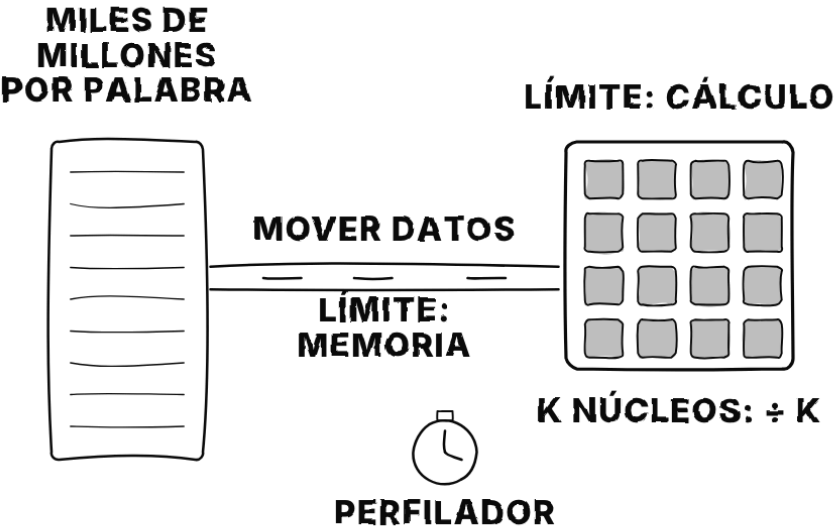
depende de detalles como el hardware y el tipo de trabajo en paralelo que admite. Cuando pide a una biblioteca como PyTorch que multiplique dos matrices, la biblioteca escoge una rutina concreta de bajo nivel, llamada kernel, para el hardware en que corre su programa, sea un chip gráfico, un procesador corriente o un chip especializado, en un paso llamado despacho; compiladores como XLA van más allá y fusionan varias operaciones en una sola ajustada a ese hardware. Un ejemplo llamativo: los mejores métodos conocidos para multiplicar dos tablas cuadradas tienen un trabajo de orden n elevado a una potencia por debajo de dos coma cuatro, mucho mejor que el cubo del método ingenuo. Para tablas de mil por mil, el cubo son mil millones de operaciones, mientras que mil elevado a dos coma cuatro son solo unos dieciséis millones. Sin embargo, estos métodos ingeniosos son difíciles de repartir con eficiencia entre los miles de núcleos pequeños de un chip gráfico, así que en la práctica rara vez se usan.



Mejor crecimiento, rara vez usado

Dos límites más de la O grande. Primero, tener k procesadores trabajando en paralelo no cambia nada en el orden de crecimiento: en el mejor de los casos divide el tiempo por k , un factor constante, así que un método cúbico sigue siendo cúbico. Segundo, la O grande ignora el tiempo gastado en mover datos de un lugar a otro, y eso puede convertirse en el cuello de botella principal. Un asistente de chat que le responde suele estar limitado menos por la aritmética que por la rapidez con que sus miles de millones de números pueden sacarse de la memoria para cada palabra que escribe. Un programa limitado por el movimiento de datos se llama limitado por memoria, frente a limitado por cálculo, y la única forma fiable de saber cuál tiene es ejecutar un perfilador, una herramienta que mide a dónde va realmente el tiempo. El libro concluye que juzgar el coste de un

algoritmo está lejos de ser trivial, porque el crecimiento teórico y la velocidad observada interactúan.



Limitado por cálculo o por memoria

Para recordar

La complejidad describe cómo crece el trabajo con el tamaño: multiplicar una tabla de a por b por una de b por c toma a por c productos punto de longitud b , su complejidad temporal, y necesita memoria para el resultado, su complejidad espacial. La O grande conserva solo la forma del crecimiento para tamaños grandes, ignorando constantes: la multiplicación de matrices ingenua es de orden a por b por c , cúbica para tablas cuadradas, así que duplicar el tamaño multiplica por ocho el trabajo. La velocidad real también depende del hardware, del kernel que despacha la biblioteca y de los compiladores; los métodos con mejor crecimiento, por debajo de la potencia dos coma cuatro, rara vez se usan porque se paralelizan mal. Los procesadores en paralelo dan solo una aceleración constante, y mover datos puede dominar, haciendo que un programa esté limitado por memoria y no por cálculo, lo que solo un perfilador puede revelar.

Preguntas

1. La multiplicación ingenua de dos tablas cuadradas es cúbica. ¿Qué pasa con el trabajo cuando el tamaño de las tablas se duplica?
 - a. Se duplica
 - b. Se cuadruplica
 - c. Crece ocho veces

- d. Se queda igual en un chip gráfico
2. ¿Por qué los métodos de multiplicación de matrices con crecimiento por debajo de la potencia dos coma cuatro rara vez se usan en la práctica?
- a. Son difíciles de repartir con eficiencia en hardware muy paralelo como los chips gráficos
 - b. Dan respuestas erróneas para tablas grandes
 - c. Necesitan más memoria que la tabla resultado
 - d. Solo funcionan para tablas con menos de mil filas
3. Un programa corre en 8 procesadores en lugar de 1. ¿Qué dice la O grande sobre su orden de crecimiento?
- a. Baja de cúbico a cuadrático
 - b. Se vuelve ocho veces menor en orden
 - c. Ya no se puede calcular
 - d. No cambia: el paralelismo da a lo sumo una aceleración constante
4. ¿Qué significa que un programa esté limitado por memoria?
- a. Usa más memoria de la que tiene el computador
 - b. Su velocidad está limitada sobre todo por mover datos y no por la aritmética
 - c. Guarda cada resultado intermedio en disco
 - d. Tiene una complejidad espacial de orden cero

Respuestas

1. c) El crecimiento cúbico significa que el trabajo va como el tamaño multiplicado por sí mismo tres veces, así que duplicar el tamaño multiplica

el trabajo por dos por dos por dos, ocho.

2. a) Su mejor crecimiento asintótico no se traduce en velocidad, porque se paralelizan mal en las GPU.

3. d) Con k procesadores el tiempo se divide a lo sumo por k , un factor constante, que la O grande ignora.

4. b) La O grande ignora el movimiento de datos, que puede convertirse en el cuello de botella principal; entonces el programa está limitado por memoria y no por cálculo, como puede mostrar un perfilador.

Ejemplos resueltos

Contar multiplicaciones ¿Cuántas multiplicaciones necesita el método ingenuo para multiplicar una tabla de 10 filas por 20 columnas con una tabla de 20 filas por 5 columnas, y cuál es la forma del resultado?

Los tamaños interiores son ambos 20, así que el producto existe y tiene 10 filas y 5 columnas.

Eso es 10 por 5, que son 50 entradas.

Cada entrada es un producto punto de longitud 20, así que necesita 20 multiplicaciones.

En total, 50 por 20 da 1000 multiplicaciones, que es a por b por c con a 10, b 20 y c 5.

Respuesta: 1000 multiplicaciones; el resultado es de 10 por 5.

Ejercicios

1. Una multiplicación ingenua de dos tablas cuadradas tarda un segundo. ¿Cuánto tardarían aproximadamente tablas tres veces más grandes?

Pista: El trabajo es cúbico en el tamaño.

Respuesta: Unos 27 segundos: tres por tres por tres es veintisiete, así que el trabajo, y aproximadamente el tiempo, crece veintisiete veces.

2. Muestre que un programa que necesita cinco veces el tamaño más cien pasos es de orden el tamaño.

Pista: Encuentre un múltiplo del tamaño que nunca se supere más allá de cierto punto.

Respuesta: Una vez que el tamaño llega a cien, cinco veces el tamaño más cien nunca es más que seis veces el tamaño, así que más allá de ese punto el trabajo es a lo sumo un múltiplo fijo del tamaño: es de orden el tamaño.

3. Para tablas cuadradas de mil filas, compare el trabajo del método ingenuo con el de un método cuyo trabajo crece como el tamaño elevado a dos coma cuatro.

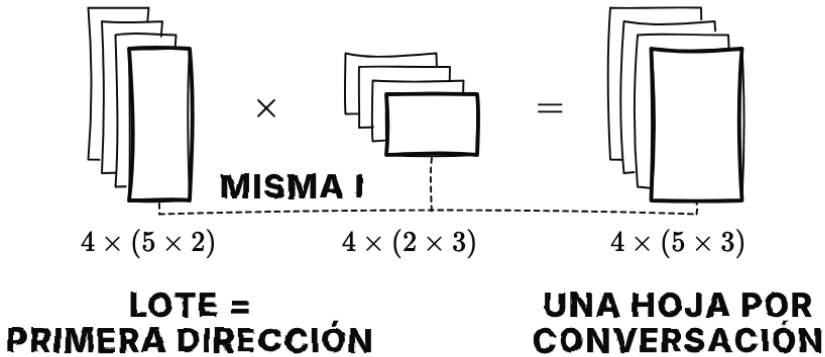
Pista: Mil al cubo, y mil elevado a dos coma cuatro, que son unos dieciséis millones.

Respuesta: El método ingenuo necesita unos mil millones de operaciones, el ingenioso unos dieciséis millones, aproximadamente sesenta veces menos; sin embargo el ingenioso rara vez se usa porque no se paraleliza bien en los chips gráficos.

Pilas de tablas y una abreviatura con letras

Imagine cuatro alumnos, cada uno con una hoja de deberes que les pide multiplicar una tabla de cinco por dos por una tabla de dos por tres. Cada alumno trabaja solo con su propio par y entrega un resultado de cinco por tres. Reunidas, las cuatro respuestas forman una pila: cuatro resultados de cinco por tres. Dentro no ocurrió nada nuevo, la misma multiplicación simplemente se hizo cuatro veces, una al lado de otra. Así funcionan la mayoría de las operaciones sobre tensores con más de dos direcciones, y el libro llama a esta multiplicación de matrices por lotes.

MATMUL POR LOTES

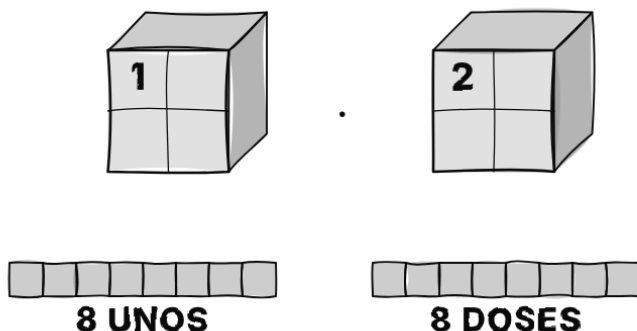


Multiplicación de matrices por lotes

Los vectores y las matrices son especiales porque muchas operaciones tienen sentido para ellos y se vuelven incómodas o indefinidas más allá: la multiplicación de matrices, la verdadera exponencial de matriz, los determinantes. Así que cuando los tensores tienen más direcciones, las operaciones que necesitamos son sobre todo de dos tipos: versiones por lotes de operaciones matriciales, o combinaciones de operaciones matriciales con reducciones como las sumas. Para la multiplicación por lotes, tome una primera pila de n tablas, cada una de a por b , y una segunda pila de n tablas, cada una de b por c . El resultado es una pila de n tablas, cada una de a por c , donde la tabla número i es simplemente la tabla i de la primera pila por la tabla i de la segunda. Las bibliotecas tratan la dirección inicial, la primera, como el lote y la manejan en silencio: en PyTorch, el comando de

multiplicación corriente aplicado a una pila de cuatro tablas de cinco por dos y una pila de cuatro tablas de dos por tres devuelve una pila de cuatro tablas de cinco por tres, exactamente como muestra el cuadro de código del libro. Así es como un asistente de chat responde a los mensajes de muchas personas a la vez: cada conversación es una hoja de la pila, y las mismas multiplicaciones corren sobre todas las hojas juntas.

Para el segundo tipo, el libro da un producto punto generalizado. Tome dos tensores de la misma forma, con tres direcciones, multiplíquelos posición por posición y sume todos y cada uno de los productos. No es más que el producto punto corriente de los dos tensores aplanados en listas largas. Por ejemplo, dos cubitos de dos por dos por dos, el primero lleno de unos y el segundo de doses, tienen ocho posiciones, cada producto es dos, y el producto punto generalizado es dieciséis. El libro añade que este breve repaso cubre la mayoría de las operaciones con tensores usadas en el resto del libro, y que cualquier cosa nueva se presentará cuando haga falta.



$$8 \times (1 \cdot 2) = 16$$

EINSUM

Un producto punto de tensores enteros

Luego viene una sección opcional sobre una abreviatura ingeniosa llamada einsum, por la convención de suma de Einstein, que puede describir prácticamente cualquier operación con tensores con una cadena corta de letras. La idea es dar una letra a cada dirección de cada tensor. Para la multiplicación por lotes, la primera pila tiene direcciones i, j, z, la segunda tiene i, z, k, y el resultado tiene i, j, k. Para el producto punto generalizado, ambos tensores tienen direcciones i, j, k, y el resultado no tiene ninguna letra, ya que es un solo número. La regla, en la forma simplificada que usa el libro, es esta: toda letra que aparece en las entradas pero no en el resultado se suma. En la multiplicación por lotes la letra z aparece en ambas entradas pero no en el resultado, así que es la dirección que se multiplica y se suma, que es exactamente el producto punto dentro de

cada multiplicación de tablas. En el producto punto generalizado las tres letras desaparecen del resultado, así que se suma todo. La operación completa se escribe entonces como una sola cadena: las letras de las entradas separadas por una coma, una flecha y las letras del resultado.

EN ENTRADAS, NO EN RESULTADO → SUMA SE SUMA



ijk, ijk → **EINOPS**
UN NÚMERO

ijz, ikz → ijk **VOLTEADO**

UN LOTE DE 2, A MANO

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

Einsum: letras para las direcciones

La mayoría de los frameworks ofrecen una operación einsum que toma exactamente una cadena así. Su primera ventaja es que no necesita recordar los comandos propios de cada biblioteca: solo PyTorch tiene varias funciones de multiplicación con distintas reglas de forma y comportamiento de broadcasting, mientras que einsum las describe todas de la misma manera. Su segunda ventaja es la portabilidad: la cadena para la multiplicación por lotes es idéntica en

PyTorch y en JAX, así que el código pasa de una biblioteca a otra sin cambios. Dar la vuelta a una dirección también es fácil: si la segunda pila tiene sus tablas en el otro sentido, con direcciones i, k, z, simplemente escribe esas letras, y einsum multiplica por las tablas volteadas sin ningún paso extra. Por estas razones einsum, y paquetes contruidos sobre la misma idea como einops, se han vuelto muy populares.

Pruébalo

Tome dos hojas de papel y en cada una dibuje una pila de dos tablas pequeñas, de dos por dos. En la primera hoja escriba las tablas uno, dos sobre tres, cuatro y uno, cero sobre cero, uno; en la segunda escriba dos, cero sobre cero, dos y cinco, seis sobre siete, ocho. Multiplique la tabla uno por la tabla uno y la tabla dos por la tabla dos: debería obtener dos, cuatro sobre seis, ocho, y luego cinco, seis sobre siete, ocho. Eso es una multiplicación por lotes con un lote de dos, hecha a mano.

Para recordar

Más allá de dos direcciones, la mayoría de las operaciones son operaciones matriciales por lotes u operaciones matriciales combinadas con reducciones. La multiplicación de matrices por lotes multiplica la tabla i de una pila por la tabla i de otra, dando una pila de resultados, y las bibliotecas manejan la dirección inicial del lote automáticamente. El producto punto generalizado multiplica dos tensores de la misma forma posición por posición y suma todo, el producto punto de sus formas aplanadas. Einsum da una letra a cada dirección y suma sobre las letras que aparecen en las entradas pero no en el resultado, así que una cadena describe la operación de la misma manera en PyTorch y JAX, lo que lo hizo popular, junto con einops.

Preguntas

1. Una pila de 4 tablas de 5 por 2 se multiplica, por lotes, por una pila de 4 tablas de 2 por 3. ¿Qué sale?
 - a. Una tabla de 20 por 3
 - b. Una pila de 4 tablas de 5 por 3
 - c. Una pila de 4 tablas de 2 por 2
 - d. Nada, porque las formas no coinciden
2. En la regla de einsum que usa el libro, ¿qué direcciones se suman?
 - a. Todas las letras que aparecen en el resultado

- b. Solo la primera letra de cada entrada
 - c. Las letras que aparecen en las entradas pero no en el resultado
 - d. Las letras que aparecen en una sola entrada
3. ¿Qué es el producto punto generalizado de dos tensores de la misma forma?
- a. La suma de todos sus productos posición por posición, el producto punto de sus formas aplanadas
 - b. Una pila de productos de matrices
 - c. El producto de sus entradas más grandes
 - d. Un tensor con la misma forma que las entradas
4. ¿Por qué es útil einsum al pasar código de PyTorch a JAX?
- a. Porque JAX no puede multiplicar matrices de otra manera
 - b. Porque einsum corre más rápido que cualquier otro comando
 - c. Porque einsum elimina la necesidad de lotes
 - d. Porque la misma cadena de letras describe la operación de forma idéntica en ambas bibliotecas

Respuestas

1. **b)** La multiplicación por lotes multiplica la tabla i por la tabla i; cada 5 por 2 por 2 por 3 da 5 por 3, y hay cuatro de ellas.
2. **c)** Una letra que está en las entradas pero falta en el resultado se suma; en la multiplicación por lotes esa es z, la dirección de los productos punto interiores.
3. **a)** Multiplicar posición por posición y sumar todo es lo mismo que un producto punto corriente de los dos tensores escritos como listas largas.

4. d) La cadena einsum para la multiplicación por lotes es idéntica en PyTorch y JAX, así que el código pasa sin cambios, sin recordar los comandos de cada biblioteca.

Ejemplos resueltos

Formas en una multiplicación por lotes En el cuadro de código del libro, una pila de forma 4 por 5 por 2 se multiplica por una pila de forma 4 por 2 por 3. Halle la forma del resultado.

La dirección inicial, 4, es el lote: hay cuatro pares de tablas.

Cada par es una tabla de 5 por 2 por una tabla de 2 por 3; los tamaños interiores, 2 y 2, coinciden.

Cada producto es por tanto de 5 por 3.

Apilar los cuatro productos da una pila de forma 4 por 5 por 3.

Respuesta: Una pila de forma 4 por 5 por 3.

Ejercicios

1. Dos tensores de forma 2 por 2, aplanados, se leen uno, dos, tres, cuatro y dos, cero, uno, uno. ¿Cuál es su producto punto generalizado?

Pista: Multiplique posición por posición, luego sume todo.

Respuesta: Nueve: uno por dos es dos, dos por cero es cero, tres por uno es tres, cuatro por uno es cuatro, y dos más cero más tres más cuatro es nueve.

2. Usando la regla de las letras, describa la multiplicación de matrices corriente de una tabla de i por z con una tabla de z por k .

Pista: ¿Qué letra debe desaparecer en el resultado?

Respuesta: Las entradas llevan las letras i , z y z , k , y el resultado conserva i , k . La letra z aparece en las entradas pero no en el resultado, así

que se suma: cada entrada es el producto punto a lo largo de z , que es la multiplicación de matrices.

3. Una pila contiene 32 conversaciones, cada una una tabla de 10 por 64, y cada una debe multiplicarse por su propia tabla de 64 por 64. ¿Cuál es la forma del resultado, y cómo se calcula?

Pista: Piense en cada conversación como una hoja de un lote.

Respuesta: Una pila de 32 tablas de 10 por 64: la tabla de 10 por 64 de cada conversación por su tabla de 64 por 64 da 10 por 64, y los 32 productos se hacen uno al lado de otro como una sola multiplicación por lotes.

La pendiente en un punto

Tome una calculadora y eleve al cuadrado el número tres: nueve. Ahora eleve al cuadrado tres coma cero uno: nueve coma cero seis cero uno. La entrada se movió una centésima y la salida se movió seis centésimas y un poquito más, así que cerca de tres el cuadrado cambia unas seis veces más rápido que su entrada. Pruebe con tres coma cero cero uno y la razón se acerca aún más a seis. Esa razón límite es la derivada, y es la primera herramienta que el libro necesita, porque los gradientes, sus primos de muchas dimensiones, son lo que impulsa el entrenamiento de todos los modelos del libro.

x	x^2	RAZÓN
3	9	
3.01	9.0601	6.01
3.001	9.006001	6.001 $\longrightarrow$ 6

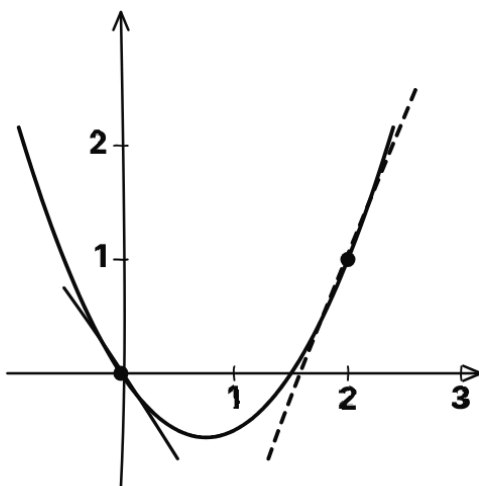
$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

se escribe $\frac{\partial f}{\partial x}$ o $f'(x)$

Elevar al cuadrado cerca de tres

Para una función con un número de entrada y un número de salida, la derivada en un punto se define exactamente como acaba de hacerlo: mueva la entrada una cantidad pequeña, vea cuánto cambia la salida, divida el cambio de la salida por el movimiento, y deje que el movimiento se encoja hacia cero. El libro usa varias maneras de escribirla: una d rizada para derivadas y gradientes de cualquier tamaño, el mismo signo con la entrada escrita en pequeño al lado para decir qué entrada se varía, y para funciones de un número una marca prima tras el nombre de la función, conocida como notación de Lagrange. No se preocupa de si una derivada siempre existe, lo que no está garantizado ni siquiera para una curva continua de aspecto suave; simplemente lo supone, y vuelve a los casos incómodos, como la esquina aguda del valor absoluto en cero, en el capítulo seis.

Para funciones familiares la definición da reglas familiares. Una potencia de la entrada tiene una derivada igual a esa potencia por la entrada elevada a uno menos: el cubo tiene derivada tres veces el cuadrado, así que en dos es tres por cuatro, doce. El logaritmo tiene derivada uno dividido por la entrada, así que en cuatro su pendiente es un cuarto. El seno tiene derivada igual al coseno, así que en cero su pendiente es uno.



$$(x^3)' = 3x^2 \rightarrow 12$$

$$(\log x)' = \frac{1}{x} \rightarrow \frac{1}{4}$$

$$(\sin x)' = \cos x \rightarrow 1$$

$$f'(x) = 2x - 1.5$$

**EN 0: -1.5,
BAJA**

**EN 2: +2.5,
SUBE**

Pendientes de tangentes en un valle

Geométricamente, la derivada es la pendiente de la tangente, la recta que apenas toca la curva en un punto, y es la mejor aproximación por una recta de la función ahí. La figura del libro dibuja la curva $x^2 - 1.5x$, un valle poco profundo, con su tangente trazada en dos puntos. Su derivada es $2x - 1.5$. En cero la pendiente es menos uno coma cinco: la curva baja hacia la derecha, así que ahí es decreciente. En dos la pendiente es más dos coma cinco: la curva sube hacia la derecha. Esa es la lección duradera: el signo de la pendiente le dice, en un pequeño entorno, hacia dónde va la función, y la misma idea se trasladará a funciones de muchas entradas. Cuando se entrena la aplicación de fotos, es exactamente esta clase de pendiente, calculada para millones de perillas a la vez, la que le dice a cada perilla hacia dónde girar.

Tres reglas nos permiten derivar funciones construidas a partir de otras más simples, y las tres se extienden a muchas dimensiones.

Linealidad: la derivada de una suma es la suma de las derivadas; para $x^2 + 3x$ en uno, eso es dos más tres, cinco. La regla del producto: la derivada de un producto es la derivada del primero por el segundo, más el primero por la derivada del segundo; para $x^2 \cdot x^3$ esto da dos equis a la cuarta más tres equis a la cuarta, cinco equis a la cuarta, que es en efecto la derivada de equis a la quinta. La regla de la cadena: la derivada de una función aplicada al resultado de otra es la derivada exterior, tomada en el valor interior, por la derivada interior. Para $(2x + 1)^2$ en uno: el valor interior es tres, la derivada exterior ahí es seis, la derivada interior es dos, así que la pendiente es doce; desarrollar primero el cuadrado y derivar da el mismo doce. La regla de la cadena es la que hay que recordar: una pendiente a través de una cadena de pasos es el producto de las pendientes de los pasos, y esa idea sostendrá todo el libro.

SUMA $(f + g)' = f' + g'$ $(x^2 + 3x)'|_1 = 2 + 3 = 5$

PRODUCTO $(fg)' = f'g + fg'$ $(x^2x^3)' = 2x^4 + 3x^4 = 5x^4$

CADENA

$$(f(g(x)))' = f'(g(x))g'(x)$$

$$((2x + 1)^2)'|_1 = \underbrace{6}_{\text{exterior}} \cdot \underbrace{2}_{\text{interior}} = 12$$

Reglas de la suma, el producto y la cadena

Para recordar

La derivada de una función de un número es el límite del cambio de la salida dividido por un movimiento cada vez menor de la entrada; elevar al cuadrado cerca de tres da seis. Potencias, logaritmos y senos tienen derivadas familiares: el cubo da tres veces el cuadrado, el logaritmo uno sobre la entrada, el seno el coseno. La derivada es la pendiente de la tangente y la mejor aproximación por una recta; su signo dice si la función sube o baja cerca, como muestra el valle de la figura. Tres reglas construyen las derivadas de funciones combinadas: las sumas se suman, los productos siguen la regla del producto, y las cadenas multiplican sus pendientes, la regla de la cadena.

Preguntas

1. La pendiente de una función en un punto es negativa. ¿Qué dice eso sobre la función cerca de ese punto?
 - a. Baja cuando la entrada aumenta, y sube hacia la izquierda
 - b. Sube cuando la entrada aumenta
 - c. Ha alcanzado su valor más bajo
 - d. No es derivable ahí
2. ¿Qué regla da la pendiente de una función aplicada al resultado de otra?
 - a. La linealidad
 - b. La regla del producto

- c. La regla de la cadena: pendiente exterior en el valor interior por la pendiente interior
 - d. La regla de la potencia
3. ¿Por qué el libro se preocupa tanto por las derivadas?
- a. Porque todo modelo debe ser un polinomio
 - b. Porque las derivadas sustituyen la necesidad de datos
 - c. Porque solo importan para senos y logaritmos
 - d. Porque los gradientes, su forma de muchas entradas, impulsan el entrenamiento de todos los modelos del libro
4. ¿Cuál es la pendiente del logaritmo en la entrada cuatro?
- a. Cuatro
 - b. Un cuarto
 - c. Cero
 - d. Menos un cuarto

Respuestas

- 1. a)** Una pendiente negativa significa que la curva baja hacia la derecha en un pequeño entorno, como hace el valle de la figura en cero.
- 2. c)** La regla de la cadena multiplica las pendientes de los pasos; para dos equis más uno, al cuadrado, en uno da seis por dos, doce.
- 3. d)** Los gradientes, los primos de muchas dimensiones de la derivada, le dicen a cada número ajustable hacia dónde girar durante el entrenamiento.
- 4. b)** La derivada del logaritmo es uno dividido por la entrada, así que en cuatro es un cuarto.

Ejemplos resueltos

Leer la pendiente de la curva del libro La figura del libro muestra la curva equis al cuadrado menos uno coma cinco equis. Halle su pendiente en uno, y el punto donde la curva es plana.

La regla de la potencia da dos equis para equis al cuadrado, y la linealidad da menos uno coma cinco para el segundo término, así que la derivada es dos equis menos uno coma cinco.

En uno la pendiente es dos menos uno coma cinco, que es un medio: la curva sube suavemente ahí.

La curva es plana donde la pendiente es cero: dos equis menos uno coma cinco igual a cero.

Eso ocurre en equis igual a tres cuartos, el fondo del valle.

Respuesta: Pendiente un medio en uno; plana en tres cuartos, el punto más bajo.

Ejercicios

1. Estime la pendiente del cuadrado en cinco moviendo la entrada una centésima, y compárela con la regla de la potencia.

Pista: Eleve al cuadrado cinco coma cero uno, reste veinticinco, divida por una centésima.

Respuesta: Cinco coma cero uno al cuadrado es veinticinco coma uno cero cero uno, así que el cambio es de alrededor de una décima, que dividido por una centésima da diez coma cero uno. La regla de la potencia da dos por cinco, diez, el valor al que se acerca la razón cuando el movimiento se encoge.

2. Use la regla de la cadena para hallar la pendiente de tres equis menos uno, al cubo, en equis igual a uno.

Pista: Halle primero el valor interior, luego la pendiente exterior ahí, luego multiplique por la pendiente interior.

Respuesta: El valor interior es dos. La función exterior es el cubo, con pendiente tres veces el cuadrado, así que en dos es doce. La pendiente interior es tres, así que la pendiente es doce por tres, treinta y seis.

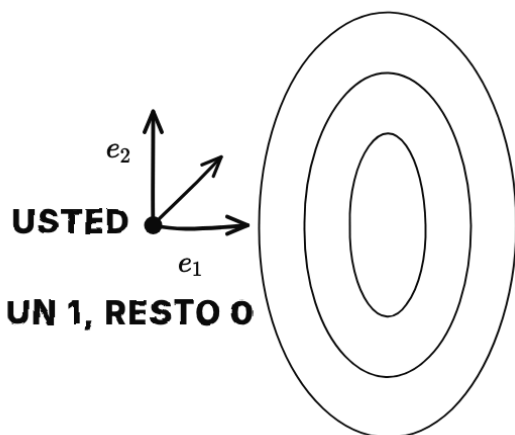
3. Use la regla del producto para $e^{\ln x}$ por el logaritmo de $e^{\ln x}$ en $e^{\ln x}$ igual a uno.

Pista: Derivada del primero por el segundo, más el primero por la derivada del segundo; el logaritmo de uno es cero.

Respuesta: El primero da uno por el logaritmo de uno, que es cero; el segundo da uno por uno sobre uno, que es uno. La pendiente es cero más uno, uno.

Pendientes en todas las direcciones

Póngase en la ladera de una colina con un mapa en la mano. Pregunte qué tan empinado es el terreno, y la respuesta honesta es: depende de hacia dónde camine. Dé un paso hacia el este y quizá suba bruscamente, dé un paso hacia el norte y quizá apenas suba, dé un paso hacia el noreste y obtendrá algo intermedio. En una línea solo había dos formas de moverse, izquierda y derecha, pero en un mapa, y más aún en muchas dimensiones, hay infinitas direcciones, así que una pendiente no significa nada hasta que diga hacia dónde. Esta parte muestra cómo una sola lista de números, el gradiente, responde la pregunta para todas las direcciones a la vez.



$$\frac{\partial f}{\partial x_1}$$

$$\frac{\partial f}{\partial x_2}$$

$$\nabla f = \begin{pmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{pmatrix}$$

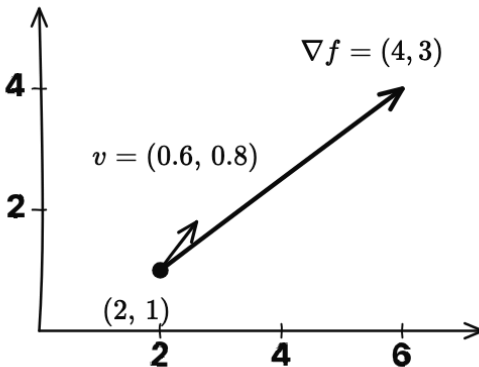
Una pendiente necesita una dirección

Tome una función que convierte un vector de d números en un número, como la altura del terreno en un punto del mapa. Las direcciones más simples son los ejes: muévase solo a lo largo de un eje, manteniendo fijas todas las demás coordenadas, y mida la pendiente exactamente como en el caso de un número. Moverse a lo largo del eje i significa sumar un pequeño múltiplo de un vector especial que tiene un uno en la posición i y ceros en el resto, una fila de la tabla identidad, llamado vector de la base. La pendiente obtenida así se llama derivada parcial respecto a esa coordenada. Apile las d derivadas parciales en un vector y tendrá el gradiente, tan importante que el libro le da su propio signo, un triángulo invertido llamado nabla.

Aquí va un ejemplo. Sea la altura la primera coordenada al cuadrado más tres veces la segunda, y párese en el punto dos, uno. Moviéndose solo a lo largo del primer eje, la pendiente es el doble de la primera coordenada, cuatro; moviéndose solo a lo largo del segundo, la pendiente es tres. Así que el gradiente ahí es el vector cuatro, tres.

$$f(x) = x_1^2 + 3x_2, \quad x = (2, 1)$$

$$\partial_1 f = 2x_1 = 4, \quad \partial_2 f = 3$$



pendiente en $v = \nabla f \cdot v$

$$(4, 3) \cdot (1, 1) = 7$$

$$(4, 3) \cdot (0.6, 0.8) = 4.8$$

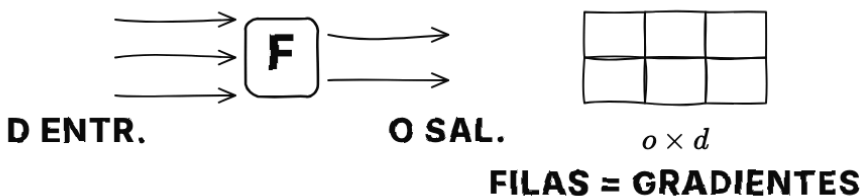
**MILLONES DE
PERILLAS,
UNA LISTA**

Gradiente (4, 3) y pendientes direccionales

¿Y una dirección que no es un eje, digamos cualquier flecha v ? La pendiente a lo largo de ella se llama derivada direccional, definida de nuevo por un pequeño paso, esta vez a lo largo de v . Como cualquier movimiento se puede dividir en pequeños movimientos a lo largo de cada eje, la derivada direccional resulta ser simplemente el producto punto del gradiente con la dirección. En nuestro ejemplo, caminar a lo largo de uno, uno da cuatro más tres, siete; caminar a lo largo de la

flecha seis décimas, ocho décimas, que tiene longitud uno, da cuatro por seis décimas más tres por ocho décimas, dos coma cuatro más dos coma cuatro, cuatro coma ocho. Así que conocer el gradiente basta para conocer la pendiente en todas las direcciones posibles. Cuando se entrena la aplicación de fotos, su error depende de millones de perillas, y su gradiente es una sola lista de millones de pendientes que dice cómo cambia el error hacia donde sea que se muevan las perillas juntas.

Ahora deje que la función devuelva varios números en lugar de uno, digamos o salidas a partir de d entradas. Cada salida tiene su propio gradiente, y apilar estos gradientes como filas da una tabla de o filas y d columnas, llamada jacobiano. Con una salida el jacobiano es simplemente el gradiente, y con una entrada y una salida es la derivada corriente. Por ejemplo, con dos entradas, la primera salida su producto y la segunda su suma, en el punto dos, uno, la primera fila es uno, dos y la segunda fila es uno, uno. El jacobiano hereda las propiedades de la derivada, y la más importante es la regla de la cadena: el jacobiano de una función aplicada después de otra es el producto de sus dos jacobianos, el exterior tomado en el punto producido por la interior. Esta es la cadena de pendientes de la parte anterior, ahora como una multiplicación de matrices, y es el corazón matemático de cómo se entrenan las redes.



$$(x_1 x_2, x_1 + x_2) \text{ en } (2, 1) : J = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}$$

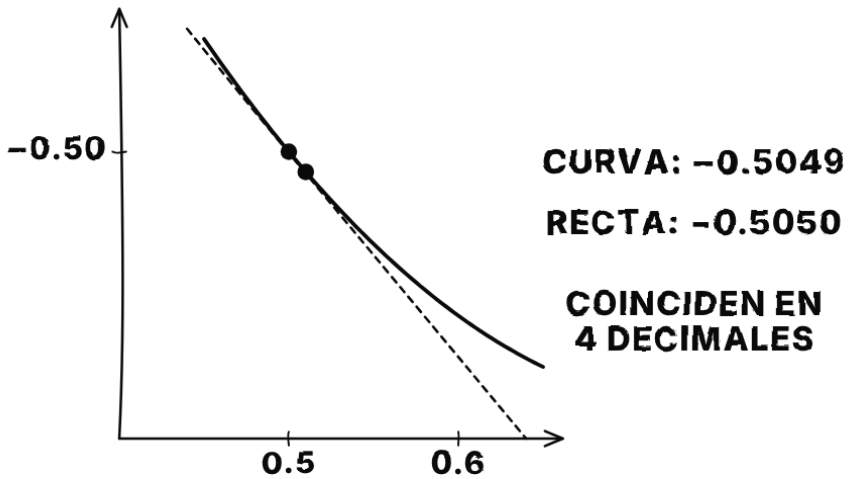
$$J_{f \circ g}(x) = J_f(g(x)) J_g(x)$$

REGLA DE LA CADENA = PRODUCTO DE MATRICES

El jacobiano y la regla de la cadena

Como la derivada, el gradiente también da la mejor aproximación recta de una función cerca de un punto, un enunciado conocido como teorema de Taylor: cerca de un punto elegido, la función es aproximadamente su valor ahí más el producto punto del gradiente con el desplazamiento desde ese punto. El libro lo comprueba en su curva anterior, $x^2 - 1,5x$, en el punto un medio. Ahí el valor es menos un medio y la pendiente es menos un medio. Moviéndose una centésima a la derecha, el valor verdadero es menos cero coma cinco cero cuatro nueve, mientras que la aproximación recta da menos un medio menos la mitad de una centésima, menos cero coma cinco cero cinco cero. Los dos coinciden hasta la cuarta cifra decimal, que es lo que significa una buena aproximación de primer orden.

$$f(x+h) \approx f(x) + \nabla f(x) \cdot h$$



Aproximación de primer orden, comprobada

Para recordar

Para una función de muchas entradas, una pendiente necesita una dirección; a lo largo de un eje es una derivada parcial, y el conjunto de todas las derivadas parciales es el gradiente. La pendiente en cualquier dirección, la derivada direccional, es el producto punto del gradiente con esa dirección, así que el gradiente solo da todas las pendientes direccionales: cuatro, tres a lo largo de uno, uno da siete. Para una función con varias salidas, los gradientes de las salidas apilados como filas forman el jacobiano, una tabla de salidas por entradas. El jacobiano de una cadena de funciones es el producto de sus jacobianos, la regla de la cadena en forma de matriz. Cerca de un punto, el valor ahí más el producto punto del gradiente con el desplazamiento es la mejor aproximación recta, como muestra la comprobación en un medio hasta cuatro decimales.

Preguntas

1. El gradiente de una función en un punto es cuatro, tres. ¿Cuál es su pendiente a lo largo de la dirección uno, uno?
 - a. Siete, el producto punto del gradiente con la dirección
 - b. Doce, el producto de cuatro y tres
 - c. Cinco, la longitud del gradiente
 - d. Uno, porque la dirección tiene dos unos

2. Una función recibe 5 números y da 3 números. ¿Cuál es la forma de su jacobiano?
 - a. 5 filas por 3 columnas
 - b. 5 filas por 5 columnas
 - c. 3 filas por 5 columnas, una fila por salida
 - d. Un solo vector de 15 números
3. ¿Cómo se obtiene el jacobiano de una función aplicada después de otra?
 - a. Sumando los dos jacobianos
 - b. Tomando el mayor de los dos
 - c. Multiplicándolos posición por posición
 - d. Multiplicando los dos jacobianos, el exterior tomado en el resultado de la función interior
4. ¿Por qué conocer el gradiente basta para conocer la pendiente en todas las direcciones?
 - a. Porque toda función tiene la misma pendiente en todas las direcciones
 - b. Porque cualquier movimiento se divide en movimientos a lo largo de los ejes, así que la pendiente es el producto punto del gradiente con la dirección
 - c. Porque el gradiente ya enumera todas las direcciones posibles
 - d. Porque en la práctica solo importan las direcciones de los ejes

Respuestas

1. **a)** La derivada direccional es el producto punto del gradiente con la dirección: cuatro por uno más tres por uno, siete.
2. **c)** Cada salida tiene un gradiente sobre las 5 entradas, y los 3 gradientes apilados como filas dan una tabla de 3 por 5.
3. **d)** La regla de la cadena en forma de matriz: el jacobiano de una composición es el producto matricial de los jacobianos individuales.
4. **b)** Dividir un movimiento en sus partes a lo largo de los ejes convierte la derivada direccional en un producto punto del gradiente con la dirección.

Ejemplos resueltos

La comprobación con recta del libro Para la curva equis al cuadrado menos uno coma cinco equis, use la aproximación recta en un medio para predecir el valor en un medio más una centésima, y compárelo con el valor verdadero.

En un medio el valor es un cuarto menos tres cuartos, que es menos un medio.

La pendiente es dos por un medio menos uno coma cinco, que es menos un medio.

La aproximación suma la pendiente por el paso: menos un medio más menos un medio por una centésima da menos cero coma cinco cero cinco.

El valor verdadero en cero coma cinco uno es cero coma dos seis cero uno menos cero coma siete seis cinco, que es menos cero coma cinco cero cuatro nueve, coincidiendo hasta cuatro decimales.

Respuesta: Aproximación menos cero coma cinco cero cinco cero; valor verdadero menos cero coma cinco cero cuatro nueve.

Ejercicios

1. Para la función que multiplica dos entradas, halle el gradiente en el punto tres, dos y la pendiente a lo largo de la dirección uno, menos

uno.

Pista: Cada derivada parcial es la otra entrada.

Respuesta: La derivada parcial respecto a la primera entrada es la segunda, dos; respecto a la segunda es la primera, tres. El gradiente es dos, tres, y la pendiente a lo largo de uno, menos uno es dos menos tres, menos uno.

2. Una función tiene dos salidas: el doble de la primera entrada más la segunda, y la primera entrada menos la segunda. Escriba su jacobiano.

Pista: Una fila por salida, una columna por entrada.

Respuesta: La primera fila es dos, uno, y la segunda fila es uno, menos uno: una tabla de dos filas por dos columnas, la misma en todos los puntos porque la función es lineal.

3. Use la aproximación recta del cuadrado en tres para estimar tres coma uno al cuadrado, y compare con el valor verdadero.

Pista: El valor en tres es nueve y la pendiente ahí es seis.

Respuesta: Nueve más seis por una décima da nueve coma seis. El valor verdadero es nueve coma seis uno, así que la aproximación se desvía solo una centésima en este paso.

Donde el terreno cae más rápido

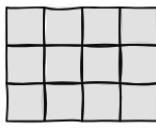
De vuelta en la ladera de la parte anterior, ahora con niebla espesa. Quiere llegar al fondo del valle pero solo puede ver el suelo a sus pies. Un plan sensato: sentir hacia dónde baja el terreno con más pendiente, dar un paso en esa dirección y repetir. Ese plan, hecho preciso, es el descenso de gradiente, y esta parte lo prepara en dos movimientos: primero una mirada cuidadosa a los tamaños de los jacobianos, luego la razón por la que el camino más empinado hacia abajo es exactamente el opuesto al gradiente.



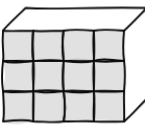
$$y = W x$$

D ENTR. O SAL.

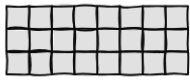
SENTIR, PISAR, REPETIR



$$\partial y / \partial x = W$$



$$o \times o \times d$$



$$o \times (o d)$$

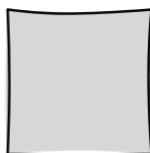
APLANAR

El libro empieza con lo que llama una nota pedante sobre tamaños. Tome la operación básica de una capa de red: una tabla de pesos multiplica un vector de entrada para dar un vector de salida, con d entradas y o salidas. Vista como función de la entrada, su jacobiano es simplemente la propia tabla de pesos, o filas por d columnas, lo que tiene sentido: cada peso dice con qué fuerza una salida responde a una entrada. Vista como función de los pesos, el panorama cambia, porque ahora lo que se varía es en sí una tabla. El «jacobiano» tiene entonces tres direcciones: una para la salida, y dos para la fila y la columna del peso que se mueve, o por o por d . Siempre se puede aplanar la tabla de pesos en un solo vector largo de o por d números, y entonces el jacobiano es una tabla corriente de o filas y o por d columnas.

El libro lo desarrolla en un recuadro. Cada salida es el producto punto de una fila de pesos con la entrada, así que mover un peso cambia solo la salida de su propia fila, y a una tasa igual al valor de entrada que ese peso multiplica. Probémoslo con dos salidas y tres entradas, siendo la entrada uno, dos, tres. La primera salida responde a los tres pesos de la primera fila con tasas uno, dos y tres, y no responde en absoluto a los pesos de la segunda fila. La segunda salida responde a los pesos de la segunda fila con las mismas tasas, uno, dos, tres, y no a los de la primera. Así que el jacobiano aplanado es una tabla de dos filas por seis columnas: uno, dos, tres, luego tres ceros en la primera fila; tres ceros, luego uno, dos, tres en la segunda. La mitad son ceros, y el resto es la entrada copiada una y otra vez. Guardar una tabla así explícitamente, materializarla, desperdicia memoria en valores repetidos, y el capítulo seis mostrará cómo evitarlo, porque en la práctica solo necesitamos la tabla aplicada a otro tensor.

$$x = (1, 2, 3)$$

1	2	3	0	0	0
0	0	0	1	2	3



$$3200 \times 3200$$

> 10 MILLONES DE ENTRADAS

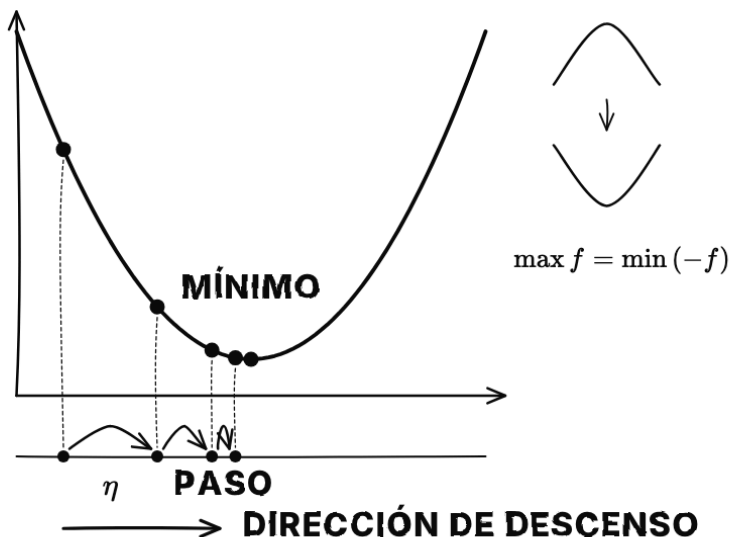
$$32 \times 100 = 3200$$

PRODUCTOS VECTOR-JACOBIANO

Sobre todo ceros y copias

Esa es la advertencia que hay que tener en mente. El libro escribirá a menudo los jacobianos como tablas simples para mantener limpia la notación, pero sus tamaños verdaderos se esconden en las formas de las entradas y las salidas, tamaños de lote incluidos. Para un lote de treinta y dos ejemplos con cien números cada uno, la entrada de una capa ya contiene tres mil doscientos números, y un jacobiano de esa entrada a una salida del mismo tamaño sería una tabla de tres mil doscientos por tres mil doscientos, más de diez millones de entradas, para una capa y un lote. El capítulo seis evita esto con lo que llama productos vector-jacobiano, y el libro remite a un tratamiento más formal que ve los jacobianos como aplicaciones lineales abstractas.

Ahora al descenso de gradiente. Suponga que queremos la entrada que hace una función de muchos números tan pequeña como sea posible, su mínimo. Más adelante la entrada serán las perillas de un modelo, los parámetros, y la función medirá qué tan mal lo hace el modelo con sus datos; para la aplicación de fotos, cuántas imágenes etiqueta mal. Minimizar no pierde nada frente a maximizar: hacer una función tan grande como sea posible es lo mismo que hacer su negativo tan pequeño como sea posible, igual que poner una colina boca abajo convierte su cima en el punto más bajo de un valle. Solo rara vez el mejor punto se puede escribir directamente con una fórmula, como ocurrirá con los mínimos cuadrados en el capítulo cuatro. Normalmente hay que buscar paso a paso: empezar desde una suposición al azar, y en cada paso moverse una cierta longitud en una cierta dirección. La longitud se llama tamaño de paso o, en aprendizaje automático, tasa de aprendizaje. Una dirección se llama dirección de descenso si algún paso a lo largo de ella baja la función. Escoja una dirección de descenso cada vez, elija el tamaño de paso con cuidado, y el procedimiento se acerca a un mínimo, en un sentido que la siguiente parte hace preciso.

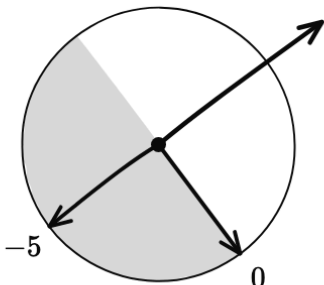


Paso a paso cuesta abajo

¿Qué direcciones van cuesta abajo? Exactamente aquellas cuya derivada direccional es negativa, y por la parte anterior la derivada direccional es el producto punto del gradiente con la dirección. Escribiendo ese producto punto como la longitud del gradiente, por la longitud de la dirección, por el coseno del ángulo entre ellos, y tomando direcciones de longitud uno, solo queda por elegir el coseno. El coseno es negativo cuando el ángulo está entre un cuarto de vuelta y tres cuartos de vuelta, así que todas esas direcciones llevan hacia abajo. El coseno más negativo, menos uno, se da a media vuelta, apuntando exactamente en sentido opuesto al gradiente: esa es la dirección de máximo descenso. Con el gradiente cuatro, tres de antes, cuya longitud es cinco, dar un paso en la dirección opuesta, menos ocho décimas, menos seis décimas, da una pendiente de menos cinco,

la más empinada posible; dar un paso en ángulo recto con él, a lo largo de seis décimas, menos ocho décimas, da una pendiente de cero, un camino a nivel por la ladera.

$\nabla f = (4, 3), \quad \|\nabla f\| = 5$



ABAJO: $\cos < 0$

$(-0.8, -0.6) : \text{pendiente } -5$

$(0.6, -0.8) : \text{pendiente } 0$

Opuesto al gradiente es lo más empinado

Para recordar

Para una capa que multiplica una entrada por una tabla de pesos, el jacobiano respecto a la entrada es la propia tabla de pesos, mientras que respecto a los pesos tiene tres direcciones o, aplanado, o filas por o por d columnas. Ese jacobiano de los pesos es sobre todo ceros y copias de la entrada, y los tamaños reales, lotes incluidos, hacen enormes los jacobianos explícitos, lo que el capítulo seis evita con productos vector-jacobiano. El descenso de gradiente busca un mínimo paso a paso, moviéndose un tamaño de paso, la tasa de aprendizaje, a lo largo de una dirección de descenso, una cuya derivada direccional es negativa. Las direcciones a más de un cuarto de vuelta del gradiente van cuesta abajo, y la dirección exactamente opuesta al gradiente es la dirección de máximo descenso.

Preguntas

1. Para una capa en la que una tabla de pesos multiplica una entrada, ¿cuál es el jacobiano respecto a la entrada?
 - a. La propia tabla de pesos
 - b. La entrada copiada o veces
 - c. Una tabla de ceros
 - d. Un tensor con tres direcciones
2. ¿Por qué el libro advierte contra materializar el jacobiano respecto a los pesos?

- a. Porque no se puede calcular de ninguna manera
 - b. Porque es sobre todo ceros y copias de la entrada, desperdicia memoria, y los tamaños de lote reales lo hacen enorme
 - c. Porque cambia cada vez que cambia la entrada
 - d. Porque es lo mismo que el gradiente
3. Una dirección forma un ángulo de un cuarto de vuelta, un ángulo recto, con el gradiente. ¿Qué pasa si da un paso a lo largo de ella?
- a. La función cae más rápido
 - b. La función sube más rápido
 - c. A primer orden la función no cambia: el camino es a nivel
 - d. El tamaño de paso se vuelve cero
4. ¿Por qué siempre podemos hablar de minimizar en lugar de maximizar?
- a. Porque toda función tiene un mínimo pero no un máximo
 - b. Porque maximizar exige una fórmula cerrada
 - c. Porque los gradientes solo apuntan hacia abajo
 - d. Porque hacer una función tan grande como sea posible es lo mismo que hacer su negativo tan pequeño como sea posible

Respuestas

1. **a)** Cada peso dice con qué fuerza una salida responde a una entrada, así que el jacobiano respecto a la entrada es exactamente la tabla de pesos, o filas por d columnas.
2. **b)** Sus entradas son los valores de la entrada y ceros; con lotes su tamaño explota, y el capítulo seis evita construirlo con productos vector-jacobiano.

- 3. c)** El coseno de un ángulo recto es cero, así que la derivada direccional es cero; solo los ángulos más allá de un cuarto de vuelta van cuesta abajo.
- 4. d)** Poner una función boca abajo convierte sus cimas en puntos más bajos, así que maximizar f es minimizar menos f .

Ejemplos resueltos

El tamaño de un jacobiano de pesos Una capa tiene 4 entradas y 3 salidas. ¿Cuántas entradas tiene su jacobiano respecto a la tabla de pesos, y qué forma tiene cuando los pesos se aplanan?

La tabla de pesos tiene 3 filas por 4 columnas, 12 pesos.

Respecto a los pesos, el jacobiano tiene una dirección para las 3 salidas y dos para los pesos de 3 por 4: 3 por 3 por 4.

Eso es 3 por 3 por 4, que son 36 entradas.

Aplanar los 12 pesos en un vector da una tabla de 3 filas por 12 columnas, todavía 36 entradas, la mayoría ceros.

Respuesta: 36 entradas; aplanado, 3 filas por 12 columnas.

Ejercicios

1. El gradiente en su suposición actual es seis, ocho. ¿Cuál es la dirección de máximo descenso de longitud uno, y qué pendiente da?

Pista: La longitud de seis, ocho es diez; apunte exactamente en sentido contrario.

Respuesta: La dirección es menos seis décimas, menos ocho décimas, opuesta al gradiente. Su pendiente es el producto punto: menos tres coma seis menos seis coma cuatro, que es menos diez, la longitud del gradiente con signo menos.

2. Con el mismo gradiente seis, ocho, ¿es la dirección cero, uno una dirección de descenso?

Pista: Calcule el producto punto del gradiente con la dirección y mire su signo.

Respuesta: No. El producto punto es ocho, que es positivo, así que la función sube a lo largo de esa dirección; está a menos de un cuarto de vuelta del gradiente.

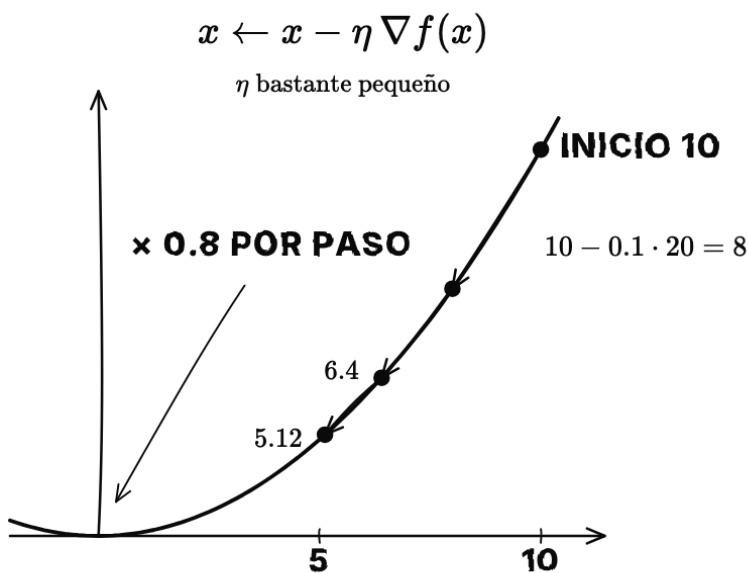
3. Con pesos en una tabla de 2 filas y la entrada uno, dos, tres, ¿a qué tasa responde la segunda salida al peso de la fila dos, columna tres, y al peso de la fila uno, columna tres?

Pista: Cada salida depende solo de su propia fila de pesos, a una tasa igual al valor de entrada que el peso multiplica.

Respuesta: Al peso de la fila dos, columna tres responde a una tasa de tres, el tercer valor de entrada. Al peso de la fila uno no responde en absoluto: la tasa es cero.

Bajar la cuesta, y dónde termina uno

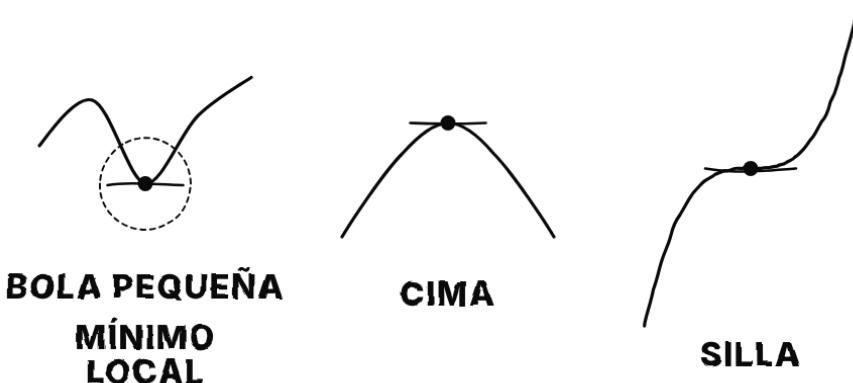
Ejecutemos el plan de la ladera con niebla en el valle más simple que existe: la altura es el cuadrado de su posición, y usted empieza en diez. La pendiente ahí es el doble de la posición, veinte, así que da un paso en contra, con un tamaño de paso de una décima: diez menos una décima de veinte lo lleva a ocho. En ocho la pendiente es dieciséis, y la misma regla lo lleva a seis coma cuatro; luego a cinco coma uno dos, y así sucesivamente. Cada paso conserva cuatro quintos de la distancia al fondo, y usted se desliza hacia cero, el punto más bajo. Ese bucle simple, moverse contra el gradiente una y otra vez, es el descenso de gradiente de máxima pendiente.



Descenso en x al cuadrado

En palabras, la regla es: la nueva suposición es la suposición anterior menos el tamaño de paso por el gradiente en la suposición anterior. El libro no se detiene en cómo elegir el tamaño de paso; supone que es lo bastante pequeño para que cada paso baje la función. También señala que el descenso de gradiente es solo tan rápido como el método usado para calcular el gradiente, y el capítulo seis presenta un método general para eso. La pregunta de esta parte es distinta: ¿dónde termina la caminata?

$$\nabla f = 0 : \text{estacionario}$$



$$f(-0.1) = -0.001$$

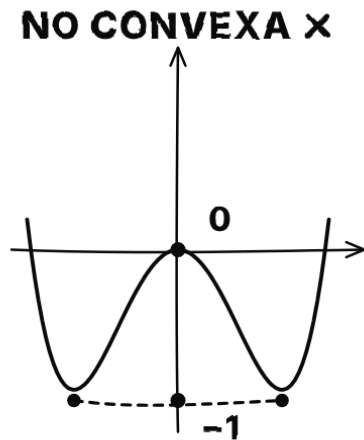
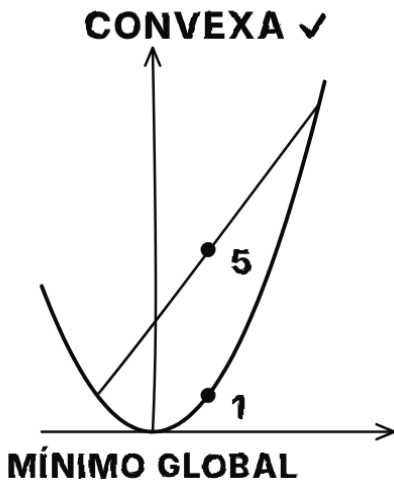
CUÁL DEPENDE DEL INICIO

Lugares planos: mínimo, cima, silla

El libro primero define sus objetivos con cuidado, señalando que un lector confiado puede saltar adelante. Un mínimo local es un punto cuyo valor no es mayor que los valores de todos los puntos cercanos, dentro de alguna pequeña bola a su alrededor, como el fondo de un hoyo en un paisaje con baches. En un punto así la tangente es plana, y a su alrededor el terreno sube. La planitud la captura un punto estacionario: un punto donde el gradiente es cero, así que no hay pendiente en ninguna dirección. Pero un lugar plano no siempre es un fondo. Puede ser una cima, que es simplemente el fondo de la función boca abajo, o un punto de silla, donde la curva cambia su manera de curvarse. La figura del libro muestra un punto así; un caso simple es el cubo de x en cero, donde la pendiente es cero pero la curva sigue bajando hacia la izquierda, con menos una milésima en menos una

décima. En general, sin suposiciones sobre la función, lo más que se puede demostrar es que el descenso de gradiente llega a algún punto estacionario, y cuál depende de dónde empezó.

¿Podemos hacerlo mejor? Imagine una parábola en forma de cuenco: no tiene puntos de silla y solo un mínimo, y ese mínimo es el valor más bajo sobre todo el rango de entradas, llamado mínimo global. En una parábola, desde cualquier punto de partida, todo gradiente apunta de vuelta hacia ese fondo, así que el descenso de gradiente siempre lo encuentra, como lo hizo nuestra caminata desde diez. La propiedad que generaliza la parábola se llama convexidad. Una función es convexa si, siempre que elija dos puntos y tienda una cuerda recta entre sus valores en la gráfica, la gráfica entre ellos nunca sube por encima de la cuerda. Compruébelo en el cuadrado: tome los puntos menos uno y tres, cuyos valores son uno y nueve; a medio camino, en uno, la cuerda está en el promedio, cinco, mientras que la gráfica está solo en uno, por debajo de la cuerda. Ahora pruebe un paisaje con dos valles, la cuarta potencia menos dos veces el cuadrado: sus dos fondos están en menos uno y más uno, ambos con valor menos uno, y a medio camino, en cero, la gráfica está en cero, por encima de la cuerda en menos uno, así que esta función no es convexa.



Convexa o no: la prueba de la cuerda

La convexidad separa los problemas de optimización en fáciles y difíciles. Para una función no convexa, el descenso de gradiente llega a un punto estacionario y nada más se puede decir sin mirar derivadas de derivadas. Para una función convexa, llega a un mínimo global sea cual sea el punto de partida. Y si la cuerda está siempre estrictamente por encima de la gráfica, lo que se llama convexidad estricta, ese mínimo global es el único. Hallar el mínimo global de una función no convexa solo con descenso de gradiente significaría reiniciar desde cada punto de partida posible, sin fin, lo que coloca la tarea entre los problemas más difíciles que conoce la informática, llamados NP-difíciles.

Esto tiene un peso histórico real. El capítulo cinco mostrará que toda red no trivial es no convexa, así que su entrenamiento puede tener muchos puntos estacionarios. Métodos más antiguos de aprendizaje supervisado, como las máquinas de vectores de soporte, conservaban un comportamiento no lineal y aun así disfrutaban de un problema convexo, y durante años eso pareció una ventaja decisiva. Sin embargo, las redes detrás de su aplicación de fotos y su asistente de chat son no convexas, y funcionan. Curiosamente, observa el libro, cuando esos modelos se inician desde una suposición inicial razonable, su entrenamiento tiende a asentarse en puntos que funcionan bien en la práctica.

Para recordar

El descenso de gradiente repite una regla: la nueva suposición es la anterior menos el tamaño de paso por el gradiente; en el cuadrado, desde diez con paso una décima, va a ocho, luego seis coma cuatro, luego cinco coma uno dos. Un mínimo local es el más bajo dentro de un pequeño entorno; un punto estacionario tiene gradiente cero y puede ser un mínimo, un máximo o un punto de silla, y en general el descenso de gradiente solo llega a algún punto estacionario. Un mínimo global es el más bajo en todas partes; para una función convexa, cuya gráfica nunca sube por encima de la cuerda entre dos de sus puntos, el descenso de gradiente lo alcanza desde cualquier inicio, y la convexidad estricta lo hace único. Las redes son no convexas, a diferencia de las máquinas de vectores de soporte, pero desde un inicio razonable su entrenamiento llega a puntos que funcionan bien.

Preguntas

1. Para una función convexa, ¿dónde termina el descenso de gradiente, con un paso lo bastante pequeño?
 - a. En un mínimo global, sea cual sea el punto de partida
 - b. En el punto de silla más cercano
 - c. En un punto distinto para cada inicio, sin garantía
 - d. Solo en un máximo
2. ¿Por qué llegar a un punto estacionario no es lo mismo que llegar a un mínimo?
 - a. Porque los puntos estacionarios siempre tienen un gradiente grande
 - b. Porque un punto plano también puede ser una cima o un punto de silla
 - c. Porque los puntos estacionarios solo existen en funciones convexas
 - d. Porque un mínimo debe tener pendiente negativa
3. A medio camino entre los puntos menos uno y más uno, la función con dos valles vale cero, mientras que la cuerda entre sus valores está en menos uno. ¿Qué muestra esto?
 - a. La función es convexa
 - b. La función tiene un único mínimo global
 - c. La función no es convexa, ya que su gráfica sube por encima de la cuerda
 - d. La función no tiene puntos estacionarios

4. ¿Por qué importa históricamente que las redes sean no convexas?
- a. Porque hace imposible entrenarlas
 - b. Porque significa que sus gradientes siempre son cero
 - c. Porque los métodos convexos siempre son más lentos
 - d. Porque métodos más antiguos como las máquinas de vectores de soporte conservaban un problema convexo, y aun así las redes se entrenan bien desde un inicio razonable

Respuestas

1. **a)** En una función convexa todo camino cuesta abajo lleva al mínimo global, como en una parábola en forma de cuenco.
2. **b)** Un gradiente cero significa un lugar plano, que puede ser un mínimo, un máximo o un punto de silla como el cubo en cero.
3. **c)** Una gráfica convexa nunca sube por encima de la cuerda entre dos de sus puntos; aquí lo hace, así que la función no es convexa.
4. **d)** La no convexidad significa muchos puntos estacionarios, una supuesta debilidad frente a los métodos convexos, pero las redes iniciadas razonablemente llegan a puntos que funcionan bien.

Ejemplos resueltos

Tres pasos de descenso de gradiente Minimice el cuadrado de x con descenso de gradiente, empezando en cinco, con un tamaño de paso de un cuarto. ¿Dónde está después de tres pasos?

El gradiente del cuadrado es dos veces x , así que cada paso resta un cuarto de dos veces x , que es la mitad de x .

Paso uno: de cinco, reste dos y medio, llegando a dos y medio.

Paso dos: de dos y medio, reste uno y cuarto, llegando a uno y cuarto.

Paso tres: reste cinco octavos, llegando a cinco octavos, reduciendo a la mitad la distancia al fondo cada vez.

Respuesta: Cinco octavos, acercándose al mínimo global en cero.

Ejercicios

1. Halle el punto estacionario de la función x al cuadrado menos cuatro x , y diga si es un mínimo global.

Pista: Iguale la pendiente, dos x menos cuatro, a cero; la función es una parábola en forma de cuenco.

Respuesta: La pendiente dos x menos cuatro es cero en x igual a dos, donde el valor es cuatro menos ocho, menos cuatro. La función es una parábola convexa, así que este punto estacionario es su único mínimo global.

2. Compruebe la condición de convexidad para el cuadrado entre los puntos cero y cuatro, en el punto a un cuarto del camino desde cuatro hacia cero, es decir en tres.

Pista: Compare el valor de la gráfica en tres con el valor de la cuerda ahí, que es un cuarto del valor en cero más tres cuartos del valor en cuatro.

Respuesta: La gráfica en tres es nueve. La cuerda ahí es un cuarto de cero más tres cuartos de dieciséis, que es doce. Nueve está por debajo de doce, como exige la convexidad.

3. Ejecute el descenso de gradiente en el cuadrado desde cinco con un tamaño de paso de uno. ¿Qué pasa, y qué sugiere sobre el tamaño de paso?

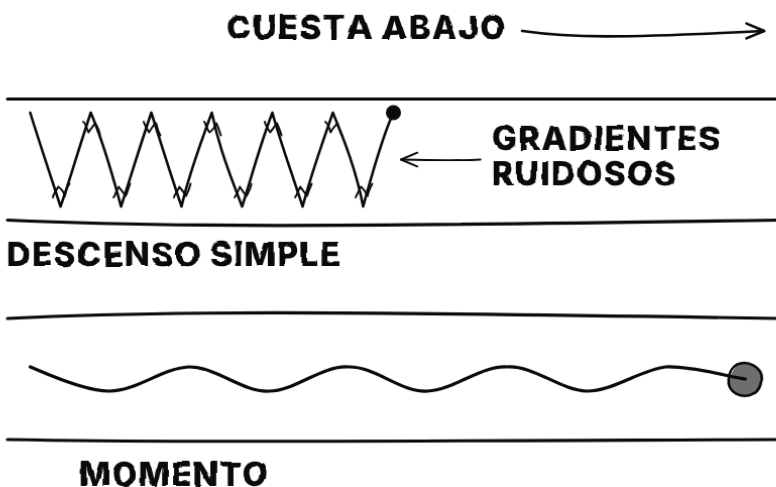
Pista: Cada paso resta uno por dos veces x .

Respuesta: Cada paso lleva x a x menos dos x , es decir menos x : cinco, menos cinco, cinco, y así sucesivamente, sin acercarse nunca a cero. El

paso es demasiado grande; el libro supone un tamaño de paso lo bastante pequeño para que cada paso baje la función.

Una bola que recuerda

Deje rodar una canica por un canalón largo y estrecho, ligeramente inclinado a lo largo. No se desliza recta por el medio; rebota de una pared lateral a la otra, mientras avanza despacio por el canalón. Ahora deje rodar una bola pesada por el mismo canalón. Su peso la lleva hacia adelante: los bandazos de lado a lado se cancelan en parte, y el tirón constante cuesta abajo se acumula, así que baja mucho más rápido. Esa diferencia es la idea detrás del momento.



Canica y bola pesada en un canalón

El gradiente negativo da el camino más empinado hacia abajo, pero solo en un entorno diminuto del punto actual. El descenso de gradiente simple se comporta como la canica ligera, un camino errático, porque cada nuevo gradiente puede apuntar a un lugar bastante distinto; el libro señala que, con un paso perfectamente elegido en una función convexa, dos gradientes sucesivos están incluso en ángulo recto entre sí. El capítulo cinco mostrará que en los modelos grandes estas direcciones son además muy ruidosas. Cuando se entrena un asistente de chat, el gradiente de cada paso viene de un pequeño puñado de ejemplos, así que tiembla mucho de un paso al siguiente. Muchas técnicas intentan acelerar las cosas eligiendo mejores direcciones, y el libro prefiere las que no necesitan derivadas de derivadas, como la tabla llamada hessiano, ni evaluaciones extra de la función, ya que ambas cosas son caras.

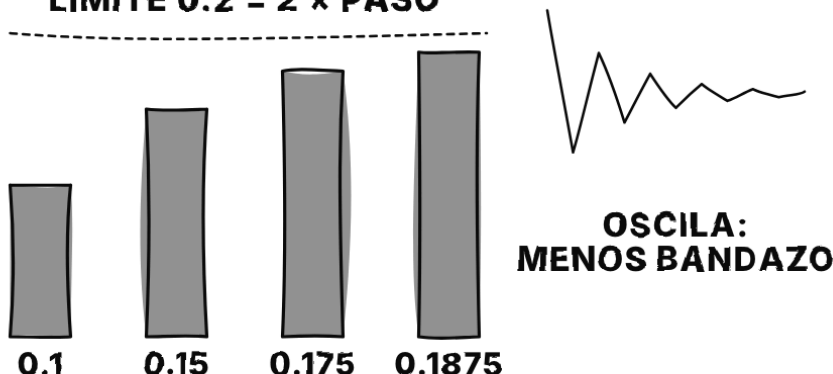
El momento es una de esas técnicas. En lugar de moverse con el paso simple cuesta abajo, cada movimiento es el paso simple cuesta abajo más una fracción del movimiento anterior. La fracción, un número entre cero y uno, es el coeficiente de momento, y el primerísimo movimiento anterior se toma como cero. Aquí va un pequeño cálculo con una perilla, un tamaño de paso de una décima, un gradiente que se mantiene en uno y un coeficiente de un medio. El primer movimiento es menos una décima. El segundo es menos una décima más la mitad de menos una décima, menos quince centésimas. El tercero es menos una décima más la mitad de eso, menos ciento setenta y cinco milésimas, y los movimientos siguen creciendo hacia menos dos décimas, el doble del paso simple, porque una dirección constante sigue acumulándose. Cuando en cambio el gradiente va y viene, la parte recordada trabaja contra la nueva y el bandazo se encoge.

$$\text{movimiento} = -\eta \nabla f + \beta \cdot \text{movimiento anterior}$$

$$\beta \in (0, 1), \quad \text{primer movimiento anterior} = 0$$

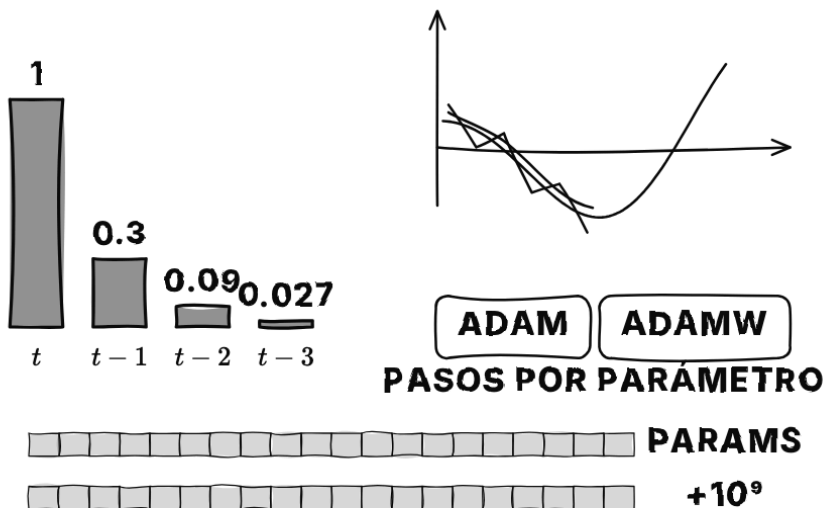
$$\eta = 0.1, \quad \nabla f = 1, \quad \beta = 0.5$$

LÍMITE 0.2 = 2 × PASO



Movimientos que se acumulan

Desenrollar la regla muestra cómo se desvanece la memoria. El gradiente más nuevo cuenta por completo; el anterior se escala por el coeficiente; el anterior a ese por el coeficiente al cuadrado, y así sucesivamente, de modo que un gradiente de n pasos atrás se escala por el coeficiente elevado a n menos uno. Con el coeficiente tres décimas, usado en la figura del libro, el gradiente de dos pasos atrás cuenta tres décimas y el de tres pasos atrás solo nueve centésimas. La figura compara el descenso de gradiente simple y el descenso de gradiente con momento en una curva ondulada, $x \cdot \sin(2x)$, empezando justo pasado el uno, y el camino con momento es el más suave de los dos. En efecto se puede demostrar que el momento acelera el entrenamiento al suavizar el camino.



Memoria que se desvanece, Adam y su coste

Otro truco común da a cada parámetro su propio tamaño de paso, adaptado a lo grandes que han sido sus gradientes. Un método popular llamado Adam combina varias de estas ideas, y se valora por ser bastante insensible a sus propios ajustes, los hiperparámetros, es decir números elegidos por el usuario en lugar de aprendidos por descenso de gradiente; los valores por defecto de la biblioteca son un buen comienzo en la mayoría de los casos. Adam y sus variantes, como AdamW, siguen siendo la elección estándar en aprendizaje profundo, y diseñar un optimizador que claramente los supere sigue siendo un problema de investigación abierto.

El precio de la aceleración es la memoria. El momento debe guardar el movimiento anterior, un número extra por cada parámetro, lo que

duplica la memoria que el propio optimizador necesita: para un modelo con mil millones de parámetros, mil millones de números más. En la mayoría de los casos, sin embargo, añade el libro, la memoria necesaria para calcular el gradiente importa mucho más, como mostrará el capítulo seis.

Para recordar

El gradiente negativo es la mejor dirección solo muy localmente, y en los modelos grandes es ruidoso, así que el descenso de gradiente simple sigue un camino errático. El momento añade a cada paso una fracción del movimiento anterior, empezando desde cero, así que las direcciones constantes se acumulan y los bandazos de ida y vuelta se cancelan; los gradientes viejos se desvanecen por potencias del coeficiente. Con paso una décima, un gradiente constante de uno y un coeficiente de un medio, los movimientos crecen de una décima hacia dos décimas. Adam combina ideas tipo momento con tamaños de paso por parámetro, es robusto a sus hiperparámetros y es el optimizador por defecto, con AdamW entre sus variantes. El momento guarda el movimiento anterior de cada parámetro, duplicando la memoria del optimizador, aunque calcular el gradiente suele necesitar más.

Preguntas

1. ¿Qué añade el momento a cada paso del descenso de gradiente?

- a. Una fracción del movimiento anterior
 - b. Un segundo gradiente calculado en un punto al azar
 - c. La derivada del gradiente, el hessiano
 - d. Una cantidad fija en la dirección del primer paso
2. ¿Por qué el momento reduce el bamboleo del descenso de gradiente simple?
- a. Porque usa un tamaño de paso mayor
 - b. Porque ignora el gradiente después del primer paso
 - c. Porque cuando el gradiente va y viene, el movimiento recordado trabaja contra el nuevo
 - d. Porque calcula el gradiente con más ejemplos
3. Con un coeficiente de momento de tres décimas, ¿cuánto cuenta el gradiente de tres pasos atrás?
- a. Tres décimas
 - b. Nada en absoluto
 - c. Nueve décimas
 - d. Nueve centésimas
4. ¿Cuál es el coste en memoria del momento?
- a. Guarda el movimiento anterior de cada parámetro, duplicando la memoria del optimizador
 - b. No necesita memoria alguna
 - c. Guarda todos los gradientes desde el inicio del entrenamiento
 - d. Guarda la tabla del hessiano

Respuestas

1. **a)** Cada movimiento es el paso simple cuesta abajo más el coeficiente de momento por el movimiento anterior, empezando desde un movimiento anterior de cero.
2. **c)** Las direcciones constantes se acumulan, mientras que las alternantes se cancelan en parte contra la parte recordada, suavizando el camino.
3. **d)** Un gradiente de n pasos atrás se escala por el coeficiente elevado a n menos uno; para tres pasos atrás eso es tres décimas al cuadrado, nueve centésimas.
4. **a)** Conservar un movimiento anterior por parámetro duplica lo que necesita el optimizador, aunque calcular el gradiente suele necesitar aún más memoria.

Ejemplos resueltos

El momento contra un gradiente que oscila Una perilla, tamaño de paso una décima, coeficiente de momento un medio. El gradiente es uno, luego menos uno, luego uno. Calcule los tres movimientos y compárelos con el descenso de gradiente simple.

El primer movimiento es menos una décima por uno, que es menos una décima, ya que el movimiento anterior es cero.

El segundo movimiento es menos una décima por menos uno, más una décima, más la mitad de menos una décima: una décima menos cinco centésimas, que es más cinco centésimas.

El tercer movimiento es menos una décima más la mitad de cinco centésimas: menos setenta y cinco milésimas.

El descenso de gradiente simple se movería menos una décima, más una décima, menos una décima; con momento los movimientos de ida y vuelta son más pequeños.

Respuesta: Movimientos de menos una décima, más cinco centésimas y menos setenta y cinco milésimas: el bandazo se encoge.

Ejercicios

1. Con un tamaño de paso de una décima, un gradiente constante de uno y un coeficiente de momento de nueve décimas, ¿a qué tamaño se acercan los movimientos?

Pista: Los movimientos dejan de crecer cuando un movimiento es igual a una décima más nueve décimas de sí mismo.

Respuesta: Uno: si un movimiento de tamaño m cumple que m es igual a una décima más nueve décimas de m , entonces una décima de m es igual a una décima, así que m es uno. Los movimientos se acercan a diez veces el paso simple.

2. Un modelo tiene siete mil millones de parámetros. ¿Cuántos números extra tiene que guardar el momento?

Pista: Un movimiento anterior por parámetro.

Respuesta: Siete mil millones: un movimiento recordado por cada parámetro, duplicando la memoria del optimizador, aunque calcular el gradiente suele ocupar aún más memoria.

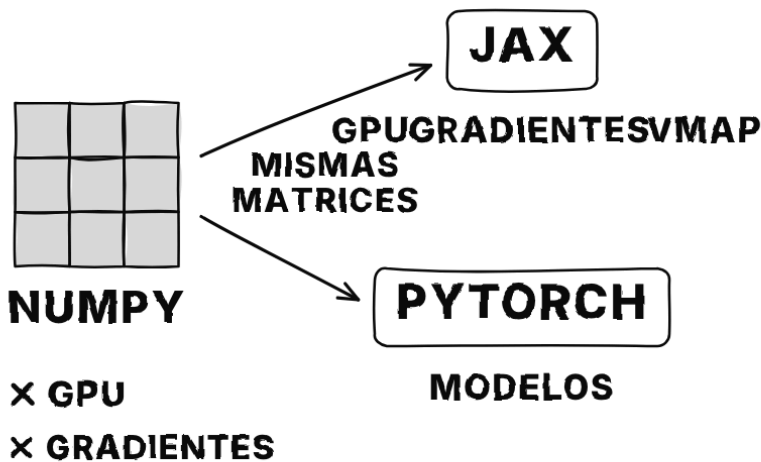
3. Explique con sus propias palabras por qué el libro prefiere métodos de aceleración sin segundas derivadas ni llamadas extra a la función.

Pista: Piense en el tamaño de una tabla de segundas derivadas para un modelo con millones de parámetros.

Respuesta: Para un modelo con millones de parámetros, una tabla de segundas derivadas tendría millones por millones de entradas, y cada evaluación extra de la función cuesta una pasada completa por los datos. Métodos como el momento mejoran la dirección usando solo gradientes ya calculados.

Del papel al código

Leer sobre una bicicleta nunca enseñó a nadie a montarla. El libro lo sabe, y al final del capítulo le entrega no una lista de ejercicios de manual sino un breve camino de autoaprendizaje a través de dos herramientas de programación reales, JAX y PyTorch, para seguirlo junto con la lectura. Las soluciones se publicarán en el sitio web del libro, y el libro advierte que muchos de sus enlaces web pueden haberse movido o caducado cuando usted los busque.



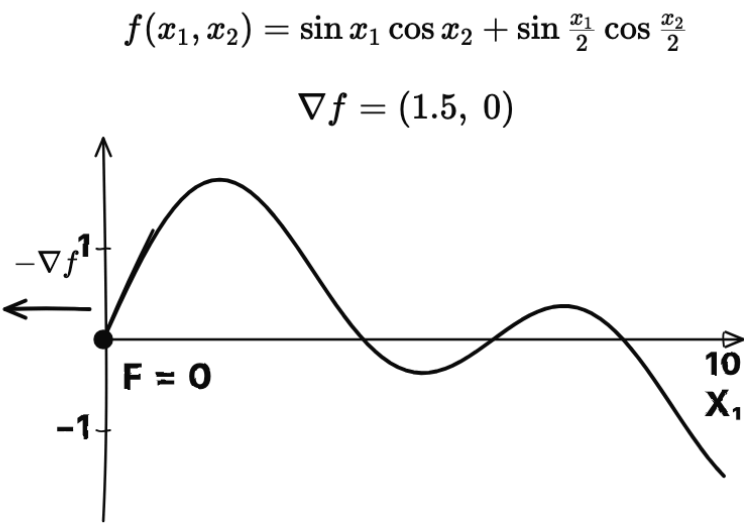
NumPy, luego JAX y PyTorch

El camino empieza con NumPy, la biblioteca clásica para trabajar con matrices multidimensionales, los tensores de este libro, y para indexar y transformar su contenido. El libro sugiere leer la guía de inicio rápido de NumPy hasta que manejar matrices se sienta natural, sobre todo rebanarlas, y practicar con una conocida colección, de ritmo pausado, de cien ejercicios de NumPy.

NumPy, con toda su influencia, tiene dos límites. Por sí solo aprovecha poco el hardware paralelo como los chips gráficos, y no puede calcular gradientes automáticamente, el tema del capítulo seis. JAX copia la manera de escribir de NumPy y añade amplio soporte de hardware, gradientes automáticos y transformaciones extra, como una que toma una función escrita para un solo ejemplo y la ejecuta sobre un lote entero, llamada mapa vectorizado. PyTorch también está construido alrededor de un núcleo parecido a NumPy, con pequeñas diferencias de nombres y comportamiento, más herramientas de nivel superior para construir modelos diferenciables. El libro sugiere hojear cómo crea matrices cada biblioteca, para ver cuánto comparten con NumPy, e ignorar por ahora las partes de nivel superior para construir modelos; el capítulo seis explica cómo están diseñados estos frameworks. El asistente de chat y la aplicación de fotos que usted usa fueron muy probablemente entrenados con exactamente estas herramientas.

Luego viene el ejercicio real, que hay que hacer tres veces, una en NumPy, una en JAX y una en PyTorch, cada una cuestión de minutos una vez que la sintaxis es familiar. La función a explorar toma dos números, cada uno entre cero y diez, y devuelve el seno del primero por el coseno del segundo, más el seno de la mitad del primero por el coseno de la mitad del segundo. Es un paisaje ondulado con varias colinas y hoyos. El libro añade una nota al pie notable: esta función se

generó pidiéndole a ChatGPT una función bonita con varios mínimos y máximos, y nada más en el libro fue generado por un modelo de lenguaje, una aclaración que el autor siente que se está volviendo importante hacer.



Una función ondulada y su pendiente

En la esquina de partida, ambas entradas cero, todos los senos son cero y la función es cero. Su pendiente a lo largo de la primera entrada ahí es el coseno de cero por el coseno de cero, uno, más un medio por el mismo tipo de producto, un medio: uno y medio en total. A lo largo de la segunda entrada la pendiente involucra senos de cero, así que es cero. El gradiente en la esquina es por tanto uno y medio, cero, y el descenso de gradiente daría un paso desde ahí en la dirección opuesta, hacia donde la función baja.

Los cuatro pasos del ejercicio son estos. Primero, escriba la función de modo que tome una tabla entera de n puntos, uno por fila, y devuelva n valores a la vez, lo que se llama vectorizada. Segundo, escriba a mano una segunda función que devuelva su gradiente, ya que la diferenciación automática aún no se ha presentado. Tercero, ejecute el descenso de gradiente desde varios puntos de partida y dibuje los caminos que sigue cada uno sobre el paisaje; en JAX o PyTorch, el mapa vectorizado es una buena herramienta para probar aquí. Cuarto, añada un término de momento y grafique la longitud del gradiente a medida que pasan los pasos: debería encogerse hacia cero conforme la caminata se acerca a un punto estacionario.

Pruébelo

Incluso sin programar, puede probar el espíritu del ejercicio con una hoja de cálculo. Ponga las dos entradas en dos celdas, calcule la función en una tercera, y las dos pendientes en otras dos celdas. Luego haga que cada fila nueva sea igual a las entradas anteriores menos una décima por las pendientes, copie la fila hacia abajo unas cuantas docenas de veces, y vea cómo el valor baja y las pendientes se encogen hacia cero.

Para recordar

El capítulo termina con un camino de autoaprendizaje en lugar de ejercicios clásicos, con soluciones previstas en el sitio web del libro. Empieza con las matrices y la indexación de NumPy, luego pasa a JAX, que añade soporte de hardware, gradientes automáticos y el mapa vectorizado, y a PyTorch, que añade herramientas para construir modelos. El ejercicio usa una función ondulada de dos entradas, generada con ChatGPT, la única salida de un modelo de lenguaje en el libro. En cada biblioteca: calcular la función en muchos puntos a la vez, programar su gradiente a mano, dibujar caminos de descenso de gradiente desde varios inicios, luego añadir momento y ver cómo la longitud del gradiente se encoge a cero.

Preguntas

1. ¿Qué añade JAX a la manera de NumPy de trabajar con matrices?
 - a. Amplio soporte de hardware, gradientes automáticos y transformaciones como el mapa vectorizado
 - b. Una nueva forma de escribir matrices que no tiene nada en común con NumPy
 - c. Solo una impresión más rápida de los resultados
 - d. Construcción de modelos de alto nivel, pero sin gradientes

2. En el ejercicio, ¿por qué hay que escribir el gradiente a mano por ahora?
 - a. Porque la función no tiene gradiente
 - b. Porque los gradientes escritos a mano siempre son más rápidos
 - c. Porque la diferenciación automática aún no se ha presentado; llega en el capítulo seis
 - d. Porque NumPy prohíbe los gradientes automáticos
3. ¿Qué debería pasar con la longitud del gradiente cuando el descenso de gradiente con momento corre sobre la función del ejercicio?
 - a. Debería crecer de forma constante
 - b. Debería mantenerse constante
 - c. Debería saltar entre cero y uno
 - d. Debería encogerse hacia cero conforme la caminata se acerca a un punto estacionario
4. ¿Qué significa escribir la función de forma vectorizada?
 - a. Dibujarla como flechas
 - b. Hacer que tome una tabla entera de puntos, uno por fila, y devuelva todos sus valores a la vez
 - c. Guardarla como un solo vector largo
 - d. Calcular solo su gradiente

Respuestas

1. a) JAX copia la interfaz de NumPy y añade soporte de hardware, cálculo automático de gradientes y transformaciones extra como el mapa vectorizado.

2. **c)** El ejercicio viene antes de la diferenciación automática, así que las funciones de pendiente se escriben a mano.
3. **d)** En un punto estacionario el gradiente es cero, así que su longitud cae hacia cero a medida que la caminata se asienta.
4. **b)** Una función vectorizada toma n puntos a la vez y devuelve n valores, en lugar de manejar un punto a la vez.

Ejemplos resueltos

Un paso en el paisaje del ejercicio En el punto donde la primera entrada es π , unos tres coma uno cuatro, y la segunda es cero, la función es cero más uno, que es uno, y su gradiente es menos uno, cero. Dé un paso de descenso de gradiente con tamaño de paso una décima y compruebe que la función baja.

El descenso de gradiente resta el tamaño de paso por el gradiente: la primera entrada pasa a ser π menos una décima por menos uno, π más una décima, unos tres coma dos cuatro.

La segunda entrada se queda en cero, ya que su pendiente es cero.

En el nuevo punto el primer término es el seno de unos tres coma dos cuatro, aproximadamente menos una décima, y el segundo término es el seno de unos uno coma seis dos, aproximadamente cero coma nueve nueve nueve.

El nuevo valor es de unos cero coma nueve, más bajo que el valor inicial de uno, como debe ser en un paso de descenso.

Respuesta: El paso lleva a unos tres coma dos cuatro, cero, y el valor baja de uno a unos cero coma nueve.

Ejercicios

1. Evalúe la función del ejercicio en la esquina de partida, ambas entradas cero, y dé su gradiente ahí.

Pista: Todo seno de cero es cero y todo coseno de cero es uno.

Respuesta: La función es cero, ya que ambos términos contienen el seno de cero. La pendiente a lo largo de la primera entrada es uno más un medio, uno y medio; a lo largo de la segunda es cero. El gradiente es uno y medio, cero.

2. ¿Por qué el libro pide el mismo ejercicio en tres bibliotecas distintas?

Pista: Piense en qué cambia y qué se mantiene igual entre NumPy, JAX y PyTorch.

Respuesta: Las tres bibliotecas comparten un núcleo parecido a NumPy, así que repetir un ejercicio corto en cada una le da fluidez en todas y muestra lo poco que difieren entre sí, a la vez que revela lo que añaden JAX y PyTorch.

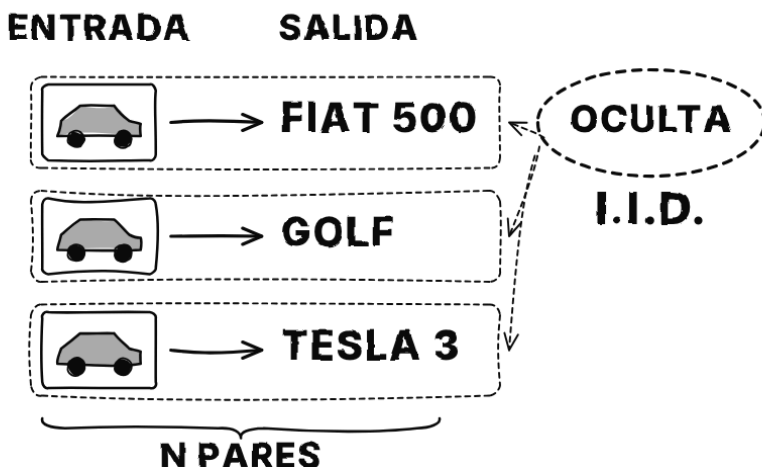
3. Describa cómo seguiría el descenso de gradiente en una hoja de cálculo para esta función.

Pista: Cada fila nueva se calcula a partir de la anterior.

Respuesta: Ponga las dos entradas en una fila con la función y sus dos pendientes escritas a mano. Haga que las entradas de cada fila siguiente sean iguales a las entradas anteriores menos una décima por las pendientes, recalcule el valor y las pendientes, y copie la fila hacia abajo; el valor debería bajar y las pendientes encogerse hacia cero.

Qué hace bueno a un montón de ejemplos

Suponga que quiere una aplicación que diga el modelo de un coche a partir de una foto. Escribir las reglas a mano no tiene esperanza: nadie puede describir en código cada curva de cada coche desde cada ángulo. Pero reunir ejemplos es fácil: recorra las calles con una cámara y escriba el modelo debajo de cada imagen. Esta es la situación de la que parte todo el aprendizaje supervisado, y el capítulo tres, breve pero descrito por el libro como la columna vertebral de lo que sigue, la hace precisa. Cubre los conjuntos de datos, las funciones de pérdida, qué significa minimizar el error promedio en los ejemplos, las suposiciones detrás de todo ello y, por último, una versión en el lenguaje de la probabilidad construida sobre la máxima verosimilitud.



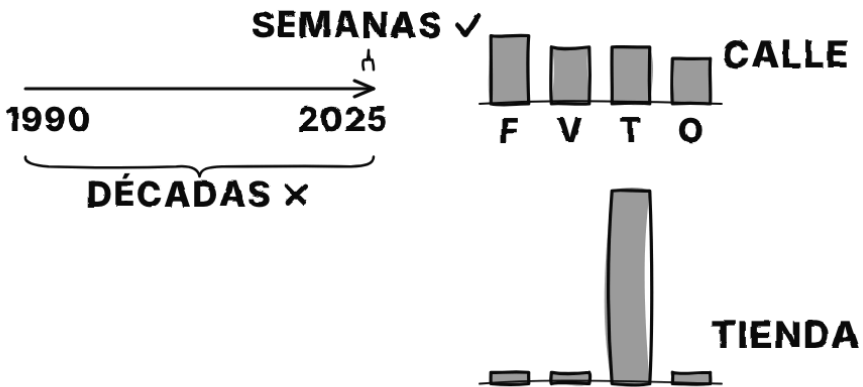
Un conjunto de datos de pares

Un conjunto de datos supervisado es simplemente una colección de n pares, cada par una entrada junto con la salida que queremos para ella: una foto y el modelo de coche que aparece en ella. Estos problemas están en todas partes, desde convertir el habla en texto hasta dirigir un robot. La aplicación de fotos de su teléfono se entrenó exactamente con una colección así: millones de imágenes, cada una emparejada con una descripción de lo que contiene. La definición viene con una suposición extra que suena técnica: cada par se extrae de forma independiente de la misma distribución de probabilidad desconocida, y esa distribución nunca se puede conocer con exactitud. Esto se llama la suposición i.i.d., abreviatura de independientes e idénticamente distribuidos.

Idénticamente distribuidos significa que lo que intentamos imitar se mantiene estable en el tiempo. Para la aplicación de coches, fotos reunidas a lo largo de unas semanas cumplen esto. Fotos reunidas a lo largo de las últimas décadas no, porque los coches en la carretera han cambiado: un modelo entrenado con ellas no reconocería los modelos nuevos, o rendiría peor de lo que debería.

IDÉNTICOS

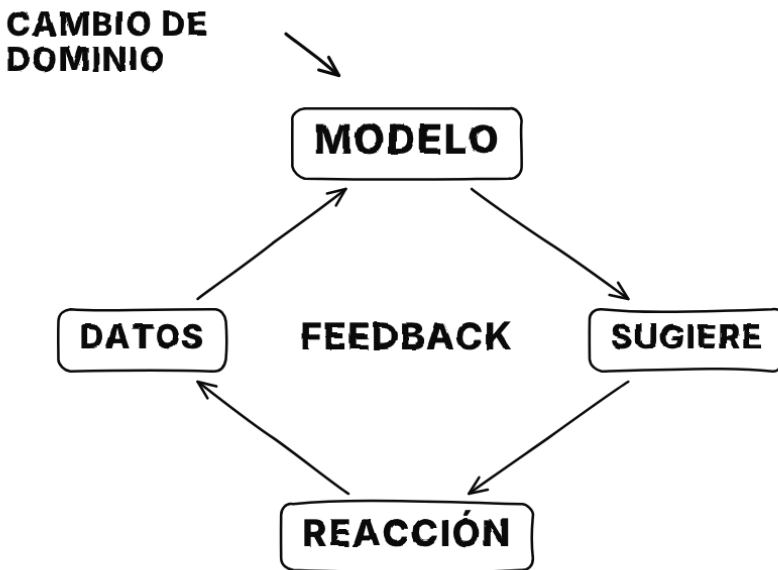
INDEPENDIENTES



Estable en el tiempo, justo en la muestra

Independientes significa que la colección no tiene sesgo y representa con justicia todo el abanico de casos. Si tomara todas sus fotos frente a un concesionario de Tesla, tendría demasiados Tesla y demasiado pocos coches de otras marcas. Que la suposición se cumpla también depende de dónde se usa el modelo. Una colección de coches reunida en Italia puede servir perfectamente en Roma o Milán, y mal en Tokio

o Taiwán, donde las calles se llenan de coches distintos. Así que la suposición debe comprobarse siempre antes de confiar en las herramientas del aprendizaje supervisado. El libro añade que los grandes modelos de lenguaje de hoy se entrenan con colecciones tan vastas y variadas que se vuelve difícil decir qué tareas están dentro de su distribución de entrenamiento y cuáles fuera, y hasta dónde generalizan de verdad.

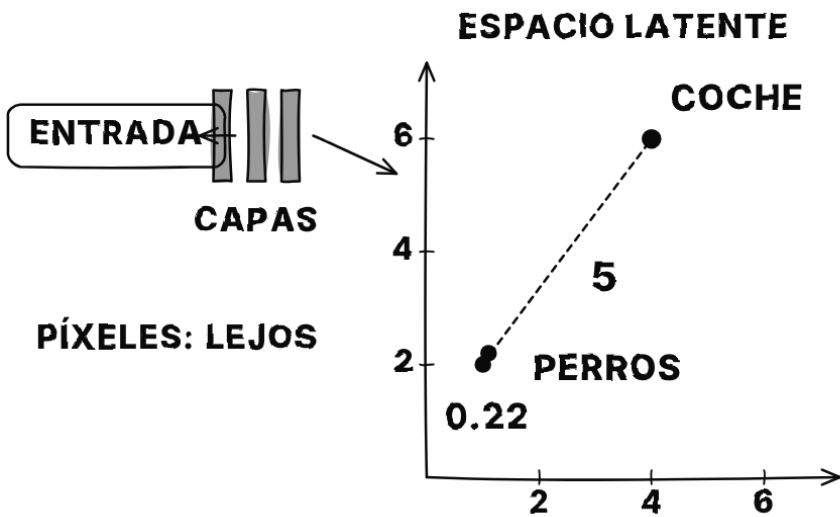


Un recomendador se alimenta a sí mismo

Esta comprobación no se hace una vez y se olvida, como insiste un recuadro del libro; tiene que continuar durante toda la vida de un modelo. Si nadie vigila, la mezcla de coches en la carretera cambia lentamente y la aplicación de coches empeora en silencio, un problema llamado cambio de dominio. Un caso más sutil es un

sistema de recomendación, la parte de una aplicación que sugiere qué ver o comprar. Una vez que empieza a sugerir, la gente reacciona a sus sugerencias, así que los datos de los que aprende después están en parte moldeados por sus propias elecciones anteriores. Eso es un bucle de retroalimentación, y obliga a comprobar constantemente qué tan bien funcionan el sistema y la aplicación.

La figura de apertura del capítulo anticipa algo notable. Un modelo diferenciable transforma su entrada paso a paso con operaciones de álgebra lineal. Después del entrenamiento, las versiones intermedias de los datos dentro de un modelo así, en lo que entonces se llama un modelo preentrenado, suelen tener una geometría útil: imágenes con significado parecido terminan como puntos cercanos entre sí en este espacio oculto, o latente. Las imágenes en bruto no tienen una noción sensata de distancia; dos fotos del mismo perro, tomadas a un paso de distancia, pueden diferir en casi todos los píxeles. Llevar los datos de un espacio así a un espacio con distancias significativas, un espacio métrico, se llama hacer un embedding de los datos. Midámoslo en un ejemplo de juguete: si una foto de perro cae en el punto uno, dos y otra en uno coma uno, dos coma dos, mientras que una foto de coche cae en cuatro, seis, entonces los dos perros están a unas dos décimas de distancia y el coche a unos cinco.



Significados cercanos, puntos cercanos

Para recordar

Un conjunto de datos supervisado es una colección de pares entrada-salida que muestran el comportamiento que queremos, usado cuando reunir ejemplos es fácil pero escribir las reglas no. Se supone que los pares se extraen de forma independiente de una única distribución desconocida e inmutable, la suposición i.i.d.. Idénticamente distribuidos significa que el objetivo se mantiene estable en el tiempo; independientes significa que la colección no tiene sesgo y es representativa, lo que también depende de dónde se usará el modelo. La suposición debe vigilarse durante toda la vida de un modelo, por el cambio de dominio y por bucles de retroalimentación como un recomendador que reacciona a sus propios efectos. Los modelos entrenados suelen hacer un embedding de los datos en un espacio latente donde los significados parecidos quedan cerca, un espacio con distancias significativas.

Preguntas

1. Un conjunto de datos de reconocimiento de coches se construye con fotos tomadas durante los últimos cuarenta años. ¿Qué parte de la suposición i.i.d. rompe?
 - a. Idénticamente distribuidos: los coches en la carretera cambiaron con el tiempo
 - b. Independientes: las fotos están demasiado alejadas
 - c. Ninguna: las fotos antiguas siempre son mejores

- d. Ambas, porque las fotos no pueden ser independientes
2. Todas las fotos se toman frente a un concesionario de Tesla.
¿Qué tiene de malo el conjunto de datos?
- a. Cambia demasiado rápido con el tiempo
 - b. Está sesgado y no es representativo, rompiendo la suposición de independencia
 - c. Contiene demasiadas marcas de coche distintas
 - d. Es demasiado pequeño para hacer un embedding
3. ¿Por qué hay que comprobar la suposición i.i.d. durante toda la vida de un modelo?
- a. Porque los conjuntos de datos crecen automáticamente
 - b. Porque el modelo olvida su entrenamiento
 - c. Porque los datos pueden derivar, como en el cambio de dominio, o ser remodelados por el propio sistema, como en los bucles de retroalimentación
 - d. Porque la suposición solo se necesita después del entrenamiento
4. ¿Qué es hacer un embedding de los datos?
- a. Guardar los datos dentro de los pesos de un modelo
 - b. Comprimir una foto en menos píxeles
 - c. Añadir etiquetas a ejemplos en bruto
 - d. Llevar los datos a un espacio con distancias significativas, donde los elementos parecidos quedan cerca

Respuestas

1. **a)** Idénticamente distribuidos significa que el objetivo se mantiene estable en el tiempo; décadas de modelos de coche cambiantes lo violan, así que el modelo fallaría con los coches nuevos.
2. **b)** El muestreo independiente significa que no hay sesgo en la recolección; un concesionario da demasiados Tesla y demasiado pocos coches de otras marcas.
3. **c)** Los cambios lentos en el mundo degradan el rendimiento, y sistemas como los recomendadores influyen en los datos de los que aprenden después.
4. **d)** Las imágenes en bruto no tienen una distancia sensata; el embedding las lleva a un espacio métrico donde los elementos semánticamente parecidos están cerca unos de otros.

Ejemplos resueltos

Distancias en un embedding En un espacio latente, una foto de perro cae en el punto uno, dos, otra foto de perro en uno coma uno, dos coma dos, y una foto de coche en cuatro, seis. ¿Qué fotos están cerca?

Entre los dos perros, las diferencias son una décima y dos décimas.

Sus cuadrados son una centésima y cuatro centésimas, que suman cinco centésimas, y la raíz cuadrada es de unas dos décimas.

Entre el primer perro y el coche, las diferencias son tres y cuatro, con cuadrados nueve y dieciséis, que suman veinticinco.

La raíz cuadrada de veinticinco es cinco, así que el coche está lejos de los perros, mientras que los perros están cerca entre sí.

Respuesta: Los dos perros están a unas dos décimas de distancia; el coche está a cinco del primer perro.

Ejercicios

1. Un conjunto de datos de coches reunido en Italia se usa para una aplicación en Milán y otra en Tokio. ¿Se cumplirá la suposición i.i.d. en cada caso? Explique.

Pista: Pregúntese si los coches que se ven en uso se parecen a los coches de la colección.

Respuesta: En Milán plausiblemente se cumple, ya que los coches en la carretera se parecen a los de la colección italiana. En Tokio puede fallar, porque la mezcla de marcas y modelos es distinta, así que la colección no es representativa de lo que verá la aplicación.

2. Explique con sus propias palabras cómo un sistema de recomendación puede crear un bucle de retroalimentación.

Pista: ¿De dónde vienen los datos de los que aprende después?

Respuesta: Una vez que el recomendador sugiere elementos, la gente hace clic en parte por esas sugerencias, así que los siguientes datos de los que aprende reflejan sus propias elecciones anteriores y no los gustos independientes de la gente. Por eso su rendimiento y el de la aplicación deben comprobarse de nuevo constantemente.

3. ¿Por qué una imagen en bruto es un mal espacio para medir la similitud, y qué ofrece en cambio un embedding?

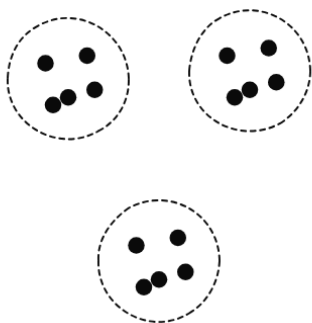
Pista: Piense en dos fotos del mismo perro tomadas a un paso de distancia.

Respuesta: Dos fotos del mismo perro pueden diferir en casi todos los píxeles, así que la distancia píxel a píxel dice poco sobre el significado. Un embedding de un modelo entrenado coloca las imágenes parecidas cerca unas de otras, de modo que la distancia en ese espacio refleja la similitud del contenido.

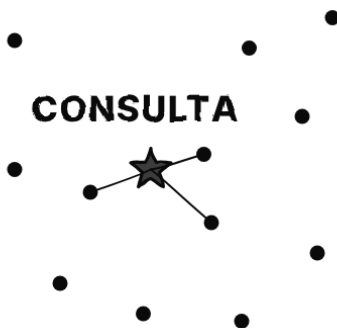
Aprender sin etiquetas, y reutilizar lo aprendido

Abra la aplicación de fotos y mire el álbum que junta las imágenes de la misma persona, aunque usted nunca le dijo quién es nadie. Nadie etiquetó esas caras; la aplicación simplemente puso las imágenes parecidas en el mismo montón. Eso no es aprendizaje supervisado en sentido estricto, porque no había salidas objetivo que imitar. La mayoría de las aplicaciones exitosas usan aprendizaje supervisado de una forma u otra, pero el libro repasa ahora sus muchos parientes.

SIN ETIQUETAS



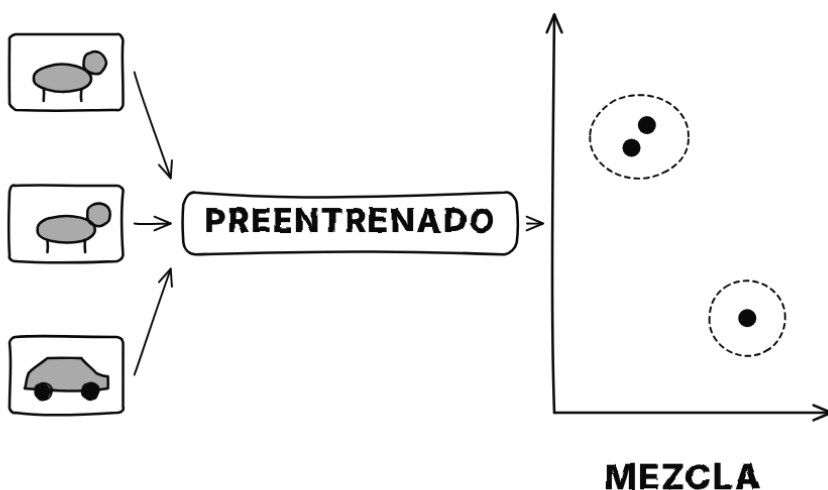
GRUPOS



K MÁS CERCANOS

Montones y vecinos más cercanos

Cuando un conjunto de datos tiene entradas pero no objetivos, el escenario se llama aprendizaje no supervisado. Una tarea típica es el agrupamiento: dividir los datos en montones de modo que los elementos dentro de un montón sean parecidos y los de montones distintos no lo sean. Otra es la recuperación: dada una consulta de un usuario, encontrar en una gran colección los k elementos más parecidos a ella, digamos los diez más cercanos. Con imágenes ambas tareas son complicadas, porque la distancia entre imágenes en bruto no significa nada: un desplazamiento diminuto puede cambiar millones de píxeles.

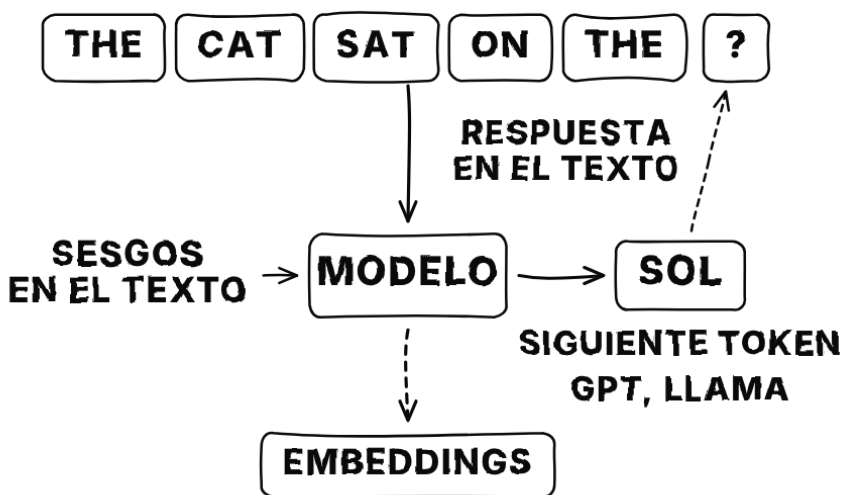


Hacer el embedding, luego agrupar

El truco es tomar prestado un modelo ya entrenado en alguna tarea amplia, como nombrar objetos en fotos, llamado modelo

preentrenado. Sus estados internos se pueden leer como vectores en un espacio de muchas dimensiones, y en muchos casos estos vectores tienen una geometría útil: las cosas que significan algo parecido quedan en puntos cercanos. Así que se pueden dar estos vectores a un método de agrupamiento corriente, como un modelo de mezcla de gaussianas, que describe los datos como una mezcla de unas pocas nubes en forma de campana, una por montón. La figura del libro muestra exactamente esto: las imágenes pasan por el modelo preentrenado, y sus vectores se reúnen en grupos por significado.

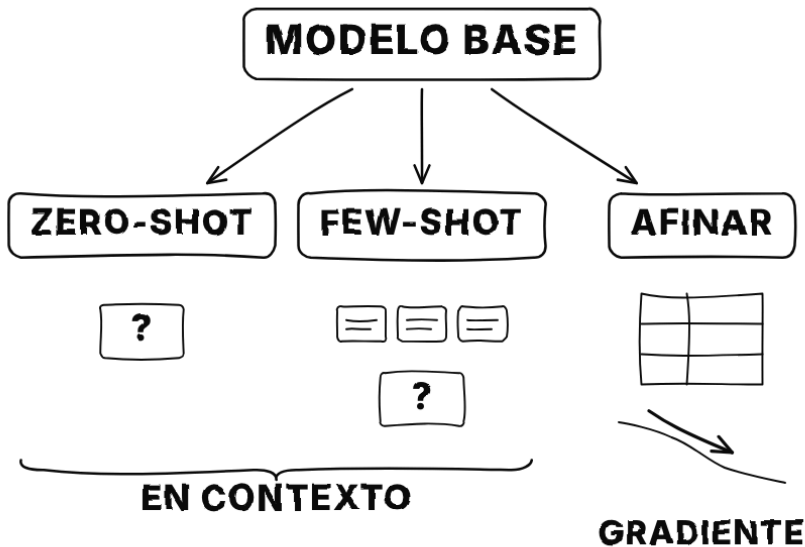
¿Y si no hay ningún modelo preentrenado disponible? Entonces una variante muy popular llamada aprendizaje autosupervisado inventa una tarea supervisada a partir de datos sin etiquetar, para entrenar un modelo que luego pueda servir a muchas otras tareas. Con un gran cuerpo de texto, por ejemplo, siempre se puede entrenar un programa para predecir cómo continúa un fragmento corto de texto, porque la continuación verdadera está ahí mismo, en el texto. El descubrimiento de que las redes entrenadas así también producen excelentes embeddings de texto tuvo, en palabras del libro, un impacto profundo en la comunidad. Una nota al pie añade un punto sobrio: los grandes conjuntos de datos extraídos de la web están llenos de sesgos, groserías y contenido vulgar, los modelos entrenados con ellos absorben esos sesgos, y esta es una de las principales críticas a los modelos fundacionales de código cerrado.



Predecir lo que viene después

Los grandes modelos de lenguaje como GPT y Llama son versiones modernas de esta idea: su entrenamiento siempre empieza con la predicción autosupervisada del siguiente token, adivinando el siguiente trozo de texto una y otra vez. A estos modelos a veces se les llama modelos fundacionales, y la segunda figura del libro muestra tres formas de usarlos. En el uso sin ejemplos, zero-shot, usted simplemente le hace al modelo una pregunta sobre una tarea nueva. Pedir a un asistente de chat que resuma un correo, sin ningún ejemplo, es uso sin ejemplos. En el prompting con pocos ejemplos, few-shot, usted incluye unos pocos ejemplos de la tarea nueva en el propio prompt, por ejemplo tres correos con sus resúmenes, seguidos del que quiere resumir. Ambos funcionan solo si el modelo generaliza muy bien. Cuando no hace falta entrenamiento de esta manera, se

dice que el modelo aprende en contexto. En el afinado, por último, usted toma el modelo preentrenado y sigue entrenando sus parámetros con descenso de gradiente sobre un pequeño conjunto de datos para la tarea nueva, de la misma manera que al entrenar desde cero. El libro se centra en construir modelos desde cero, pero señala que el afinado usa los mismos medios.



Tres formas de usar un modelo

El afinado se facilita gracias a las grandes colecciones de modelos de código abierto en línea, y no tiene por qué tocar todos los parámetros. Entrenar solo un pequeño subconjunto de los parámetros, o un pequeño número de parámetros añadidos, se llama afinado eficiente en parámetros; el libro lo deja para su segundo volumen. Para ver por qué importa, imagine ajustar solo el uno por ciento de un modelo con

siete mil millones de parámetros: son setenta millones de números que entrenar en lugar de siete mil millones.

La lista de variantes continúa, y el libro solo las insinúa. Si solo una parte de un conjunto de datos lleva etiquetas, el escenario es semisupervisado, con ejemplos en el capítulo doce. Con varios conjuntos de datos de distribuciones parecidas, o de la misma distribución en momentos distintos, aparece toda una familia de problemas, según el orden en que llegan las tareas y los datos: adaptarse a un dominio nuevo, aprender a aprender, aprender de forma continua, aprender distancias e incluso desaprender; algunos de ellos se prometen para el siguiente volumen.

Para recordar

Sin objetivos, el aprendizaje es no supervisado: el agrupamiento junta elementos parecidos y la recuperación encuentra los elementos más cercanos a una consulta, ambos difíciles con imágenes en bruto. Un modelo preentrenado hace un embedding de los datos de modo que los significados parecidos quedan cerca, lo que permite que un agrupamiento corriente como las mezclas de gaussianas funcione sobre sus vectores. El aprendizaje autosupervisado inventa una tarea a partir de datos sin etiquetar, como predecir cómo continúa un texto, y así es como empiezan los grandes modelos de lenguaje como GPT y Llama; los datos de la web también les transmiten sesgos. Los modelos fundacionales se pueden usar sin ejemplos, con unos pocos ejemplos en el prompt, o afinarse por descenso de gradiente, por completo o de forma eficiente en parámetros. El aprendizaje semisupervisado y muchos otros escenarios, como el aprendizaje continuo o el desaprendizaje, amplían el panorama.

Preguntas

1. ¿Por qué es difícil agrupar imágenes en bruto sin un modelo preentrenado?
 - a. Porque las imágenes no tienen píxeles
 - b. Porque la distancia entre imágenes en bruto no significa nada: un desplazamiento diminuto cambia millones de píxeles

- c. Porque el agrupamiento solo funciona con texto
 - d. Porque las imágenes no se pueden guardar como vectores
2. ¿Cómo obtiene el aprendizaje autosupervisado una tarea de entrenamiento a partir de texto sin etiquetar?
- a. Pidiendo a personas que etiqueten cada frase
 - b. Agrupando las palabras en montones
 - c. Prediciendo cómo continúa un fragmento corto de texto, ya que la continuación verdadera está en el texto
 - d. Afinando un modelo que ya existe
3. Usted da a un asistente de chat tres correos con sus resúmenes y luego un cuarto correo para resumir, sin ningún entrenamiento. ¿Cómo se llama esto?
- a. Afinado
 - b. Uso sin ejemplos
 - c. Aprendizaje semisupervisado
 - d. Prompting con pocos ejemplos, una forma de aprendizaje en contexto
4. ¿Qué distingue el afinado del uso sin ejemplos y con pocos ejemplos?
- a. Los parámetros del modelo se entrenan más con descenso de gradiente sobre un pequeño conjunto de datos
 - b. El modelo no recibe ninguna entrada
 - c. El modelo se entrena desde cero con la web
 - d. El modelo se usa solo para agrupar

Respuestas

1. **b)** Las distancias entre píxeles no reflejan el significado; el embedding de un modelo preentrenado coloca las imágenes parecidas cerca, así que el agrupamiento puede funcionar sobre esos vectores.
2. **c)** Los datos aportan sus propios objetivos: la continuación real de cada fragmento de texto es la respuesta que hay que predecir.
3. **d)** Incluir unos pocos ejemplos de la tarea en el prompt es prompting con pocos ejemplos; ningún parámetro cambia, así que el modelo aprende en contexto.
4. **a)** El afinado sigue entrenando los parámetros preentrenados con descenso de gradiente, mientras que el uso sin ejemplos y con pocos ejemplos los deja sin cambios.

Ejemplos resueltos

Cuánto ahorra el afinado eficiente en parámetros Un modelo tiene siete mil millones de parámetros. Si el afinado ajusta solo el uno por ciento de ellos, ¿cuántos números se entrenan?

Uno por ciento significa una centésima.

Siete mil millones dividido por cien es setenta millones.

Así que se ajustan setenta millones de números, mientras que los otros seis mil novecientos treinta millones se quedan como estaban.

Son cien veces menos números que entrenar que en el afinado completo.

Respuesta: Setenta millones de parámetros en lugar de siete mil millones.

Ejercicios

1. Clasifique cada situación: agrupar clientes por hábitos de compra sin etiquetas; un conjunto de datos donde solo una décima parte de

las fotos está etiquetada; entrenar un modelo para rellenar palabras ocultas de frases.

Pista: No supervisado, semisupervisado o autosupervisado.

Respuesta: Agrupar sin etiquetas es aprendizaje no supervisado, concretamente agrupamiento. Un conjunto de datos parcialmente etiquetado es semisupervisado. Rellenar palabras ocultas inventa objetivos a partir de los propios datos, lo que es aprendizaje autosupervisado.

2. ¿Por qué el libro conecta el aprendizaje autosupervisado con los grandes modelos de lenguaje como GPT y Llama?

Pista: ¿Cómo empieza su entrenamiento?

Respuesta: Su entrenamiento siempre empieza con la predicción autosupervisada del siguiente token sobre cantidades enormes de texto, lo que les da capacidades generales y buenos embeddings; luego se usan sin ejemplos, con pocos ejemplos o tras un afinado.

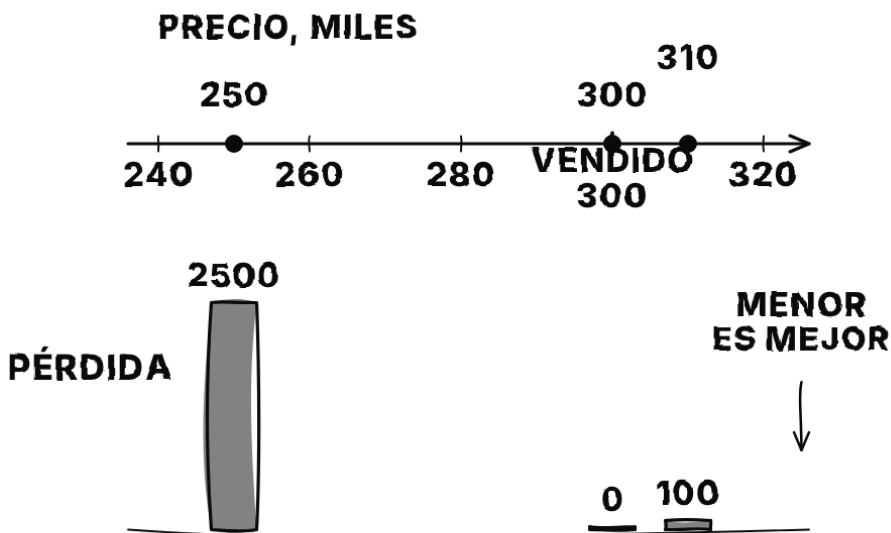
3. ¿Qué crítica plantea el libro sobre los modelos entrenados con grandes conjuntos de datos de la web?

Pista: Piense en lo que contienen esos conjuntos de datos.

Respuesta: Los datos de la web están llenos de sesgos, groserías y contenido vulgar, y los modelos entrenados con ellos absorben esos sesgos; esta es una de las principales críticas a los modelos fundacionales de código cerrado.

Puntuar una predicción

Pida a tres amigos que adivinen el precio de un piso que en realidad se vendió por trescientos mil. Uno dice trescientos diez, otro dice doscientos cincuenta, otro dice exactamente trescientos. Quién lo hizo mejor es obvio, pero para que un computador mejore sus suposiciones automáticamente, «mejor» tiene que convertirse en un número que se hace más pequeño a medida que las suposiciones mejoran. Ese número es la pérdida, y una vez reunidos los datos, elegirla es la forma de decir con precisión qué entendemos por imitar el comportamiento deseado.



Tres suposiciones, tres pérdidas

Una función de pérdida toma el objetivo y la predicción del modelo y devuelve un solo número, y debe ser diferenciable. Su valor sigue la calidad: una pérdida menor para una predicción que para otra significa que la primera predicción es mejor, medida frente a ese objetivo. Una elección común, por ejemplo, es la diferencia al cuadrado: las suposiciones de arriba puntúan diez al cuadrado, cien; cincuenta al cuadrado, dos mil quinientos; y cero. La pérdida lleva nuestra comprensión de la tarea y nuestras preferencias entre soluciones, escritas en una escala de números reales con la que un optimizador puede trabajar. Como es diferenciable, aprender se convierte en un problema matemático de optimización que el descenso de gradiente puede resolver.

El objetivo natural es la menor pérdida promedio posible sobre todo el conjunto de datos. Tome cada ejemplo, compare la predicción del modelo con el objetivo, calcule la pérdida, sume las n pérdidas y divida por n ; luego busque el modelo que haga este promedio tan pequeño como pueda. Con nuestras tres suposiciones como tres predicciones sobre tres ejemplos, la pérdida cuadrática promedio sería cien más dos mil quinientos más cero, entre tres, unos ochocientos sesenta y siete. Por razones históricas este objetivo se llama minimización del riesgo empírico, donde riesgo es simplemente otra palabra para pérdida, y empírico significa medido sobre los ejemplos que realmente tenemos.

$$\frac{100 + 2500 + 0}{3} \approx 867$$

RIESGO EMPÍRICO

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(x_i, \theta))$$

θ **PARAMS**

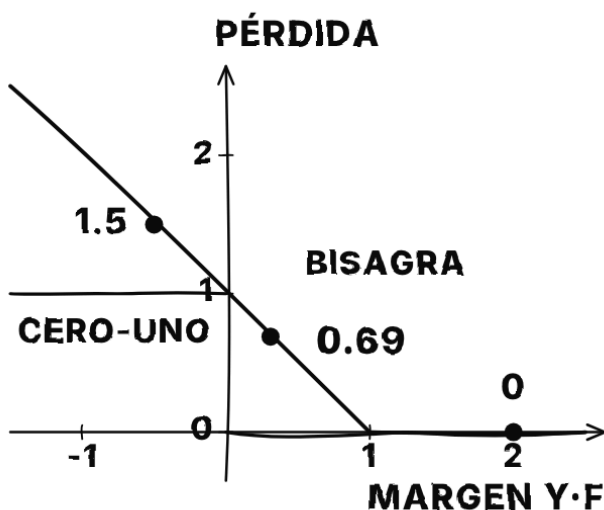
$$\theta \leftarrow \theta - \eta \nabla_{\theta} L(\theta)$$

DESCENSO DE GRADIENTE

Promediar las pérdidas

Escrita así, la búsqueda parece recorrer todas las funciones posibles de la entrada. En la práctica, explica el libro, un modelo siempre se describe mediante un conjunto de tensores, sus parámetros, así que el modelo se convierte en una función tanto de la entrada como de los parámetros. Entrenar significa entonces buscar los valores de los parámetros que minimizan la pérdida promedio, y la búsqueda se hace con descenso de gradiente sobre los parámetros. Entrenar la aplicación de fotos es exactamente esto: millones de imágenes etiquetadas, una pérdida que es pequeña cuando la aplicación nombra una imagen correctamente, y el descenso de gradiente girando los parámetros para encoger el promedio.

El libro subraya luego por qué la pérdida debe ser diferenciable. Considere distinguir gatos de perros, con objetivos más uno para gato y menos uno para perro, y un modelo que da como salida cualquier número real cuyo signo es su decisión. La puntuación más natural es la pérdida cero-uno: cero cuando el signo es correcto, uno cuando es incorrecto. Coincide perfectamente con nuestra idea de acertar, pero es inútil para entrenar, porque su pendiente es cero casi en todas partes: mover la salida de tres décimas a treinta y una centésimas no cambia nada, así que el descenso de gradiente nunca se mueve de donde empieza. Una cantidad menos intuitiva pero mejor es el margen, el objetivo por la predicción, positivo cuando el signo es correcto y negativo cuando es incorrecto, y que cambia suavemente. Una pérdida construida sobre él es la pérdida de bisagra: cero una vez que el margen llega a uno, de lo contrario uno menos el margen. Para un gato, una predicción de dos tiene margen dos y no cuesta nada; una predicción de tres décimas es correcta pero tímida y cuesta siete décimas; una predicción de menos un medio es incorrecta y cuesta uno y medio. Y ahora un pequeño movimiento de tres décimas a treinta y una centésimas baja el coste a sesenta y nueve centésimas, lo que el descenso de gradiente puede sentir. La pérdida de bisagra es la que se usa para entrenar las máquinas de vectores de soporte, y el ejemplo muestra la tensión inherente entre las pérdidas que expresan lo que queremos y las pérdidas que los optimizadores pueden usar.



Pérdida cero-uno frente a bisagra

Un recuadro da el contexto: la minimización del riesgo empírico y del riesgo esperado se asocian al informático ruso Vladimir Vapnik, cuyo trabajo fundó la teoría del aprendizaje estadístico. Esa teoría estudia qué tan bien la pérdida promedio sobre un conjunto finito de ejemplos sustituye al promedio verdadero sobre todos los datos posibles, para tipos restringidos de modelos. Las redes modernas, que siguen generalizando bien mucho después de cuando uno esperaría que sobreajustaran, le han abierto muchas preguntas nuevas.

Para recordar

Una función de pérdida compara una predicción con su objetivo y devuelve un número diferenciable que es menor para mejores predicciones; codifica lo que queremos del modelo. El entrenamiento minimiza la pérdida promedio sobre el conjunto de datos, llamada minimización del riesgo empírico, buscando sobre los parámetros del modelo con descenso de gradiente. La pérdida cero-uno coincide con «acertar» pero tiene pendiente cero casi en todas partes, así que el descenso de gradiente no puede usarla. El margen, objetivo por predicción, varía suavemente, y la pérdida de bisagra construida sobre él, usada para las máquinas de vectores de soporte, da al descenso de gradiente algo que seguir. La minimización del riesgo viene de la teoría del aprendizaje estadístico de Vapnik, a la que la sorprendente generalización de las redes modernas sigue desafiando.

Preguntas

1. ¿Por qué la pérdida cero-uno es inútil para el descenso de gradiente?
 - a. Su pendiente es cero casi en todas partes, así que los pequeños movimientos nunca la cambian y los parámetros nunca se mueven
 - b. Siempre es igual a uno
 - c. No se puede promediar sobre un conjunto de datos

- d. Da valores mayores a las mejores predicciones
2. ¿Qué minimiza la minimización del riesgo empírico?
- a. La mayor pérdida en cualquier ejemplo individual
 - b. La pérdida en el único ejemplo más difícil
 - c. La pérdida promedio sobre los ejemplos del conjunto de datos
 - d. El número de parámetros del modelo
3. Para un gato (objetivo más uno), el modelo da tres décimas. ¿Cuál es su pérdida de bisagra?
- a. Cero, porque el signo es correcto
 - b. Uno coma tres
 - c. Tres décimas
 - d. Siete décimas, porque el margen está por debajo de uno
4. ¿Qué le dice el margen, el objetivo por la predicción?
- a. El tamaño del conjunto de datos
 - b. Si el signo es correcto (positivo) o incorrecto (negativo), cambiando suavemente
 - c. El número de parámetros necesarios
 - d. La probabilidad exacta de acertar

Respuestas

1. a) La pérdida cero-uno solo salta cuando cambia el signo de la predicción; en el resto un cambio pequeño no tiene efecto, así que el descenso de gradiente se queda atascado en el inicio.
2. c) Promedia la pérdida sobre los n ejemplos y busca los parámetros que hacen ese promedio lo más pequeño posible.

3. d) El margen es uno por tres décimas; la pérdida de bisagra es uno menos el margen mientras el margen está por debajo de uno, así que siete décimas.

4. b) El margen es positivo cuando el signo de la predicción coincide con el objetivo y negativo en caso contrario, y a diferencia de la pérdida cero-uno varía de forma continua.

Ejemplos resueltos

Pérdida de bisagra promedio para tres fotos de perros Tres fotos de perros tienen objetivo menos uno. El modelo da menos dos, menos cuatro décimas y un medio. Calcule las pérdidas de bisagra, su promedio, y compare con la pérdida cero-uno.

Los márgenes son el objetivo por la salida: dos, cuatro décimas y menos un medio.

Pérdidas de bisagra: el primer margen es al menos uno, así que cero; el segundo es uno menos cuatro décimas, seis décimas; el tercero es uno menos menos un medio, uno y medio.

La pérdida de bisagra promedio es cero más seis décimas más uno y medio, dos coma uno, dividido por tres, siete décimas.

Las pérdidas cero-uno son cero, cero y uno, promedio un tercio; ignora que la segunda predicción fue correcta pero tímida.

Respuesta: Pérdidas de bisagra cero, seis décimas y uno y medio, promedio siete décimas; promedio cero-uno un tercio.

Ejercicios

1. El piso se vendió por trescientos mil. Usando la diferencia al cuadrado medida en miles, ¿qué suposición tiene la menor pérdida, doscientos noventa o trescientos veinte, y por cuánto?

Pista: Eleve al cuadrado las diferencias diez y veinte.

Respuesta: Doscientos noventa cuesta diez al cuadrado, cien; trescientos veinte cuesta veinte al cuadrado, cuatrocientos. La primera suposición es mejor, con una pérdida trescientos menor.

2. Para un gato, la salida del modelo pasa de un medio a seis décimas. ¿Cómo cambian la pérdida cero-uno y la pérdida de bisagra?

Pista: Ambas salidas tienen el signo correcto; calcule uno menos el margen para la pérdida de bisagra.

Respuesta: La pérdida cero-uno se queda en cero, ya que el signo ya era correcto. La pérdida de bisagra baja de un medio a cuatro décimas, así que el descenso de gradiente recibe una señal para empujar más.

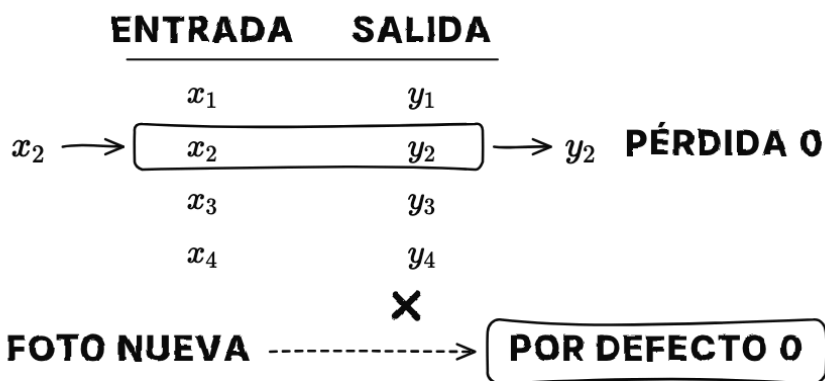
3. Explique con sus propias palabras la tensión que describe el libro al elegir una pérdida.

Pista: Compare lo que queremos medir con lo que un optimizador puede usar.

Respuesta: La pérdida debería expresar lo que de verdad nos importa, como acertar, pero también debe ser diferenciable e informativa para el descenso de gradiente. La medida más natural, la pérdida cero-uno, no da pendiente, así que usamos sustitutos como la pérdida de bisagra que siguen el rendimiento mientras cambian suavemente.

Memorizar no es aprender

Aquí hay un modelo que nunca se equivoca en sus datos de entrenamiento. Guarde todo el conjunto de datos en una tabla; cuando llega una entrada que está en la tabla, devuelva su respuesta guardada; cuando llega cualquier otra cosa, devuelva algún valor fijo por defecto, digamos cero. En los ejemplos de entrenamiento sus predicciones son exactamente los objetivos, así que, mientras la pérdida sea mínima cuando la predicción es igual al objetivo, su pérdida promedio en el conjunto de datos es la más baja posible. Y no vale nada: muéstrole una foto nueva y responde cero. El libro usa esta tabla de consulta para señalar algo que atraviesa todo el aprendizaje automático: minimizar la pérdida en los ejemplos no es el objetivo real.

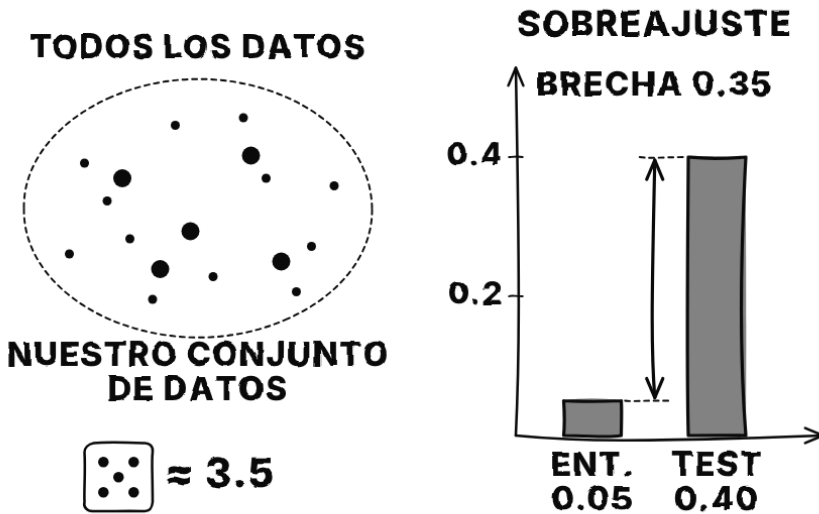


MEMORIZAR $\neq$ APRENDER

Una tabla de consulta no aprende nada

Lo que de verdad queremos es una pérdida pequeña en entradas futuras que aún no hemos visto; los ejemplos de entrenamiento son solo un sustituto de ellas. El libro lo formaliza como el riesgo esperado: la pérdida promedio sobre todos los pares entrada-salida que el modelo podría encontrar jamás, ponderados por lo probable que es cada uno, por ejemplo sobre todos los correos que podrían escribirse. Un modelo con un riesgo esperado bajo tendría garantizado funcionar. Pero no se puede calcular: nunca podemos enumerar y etiquetar todas las entradas posibles. El riesgo empírico, el promedio sobre nuestro conjunto de datos, es una estimación de él, lo que los matemáticos llaman una aproximación de Monte Carlo, un promedio sobre muestras aleatorias que sustituye a un promedio sobre todo. Funciona como lanzar un dado muchas veces: el promedio

de sus lanzamientos se acerca a tres y medio, el promedio verdadero, sin enumerar todos los resultados posibles.

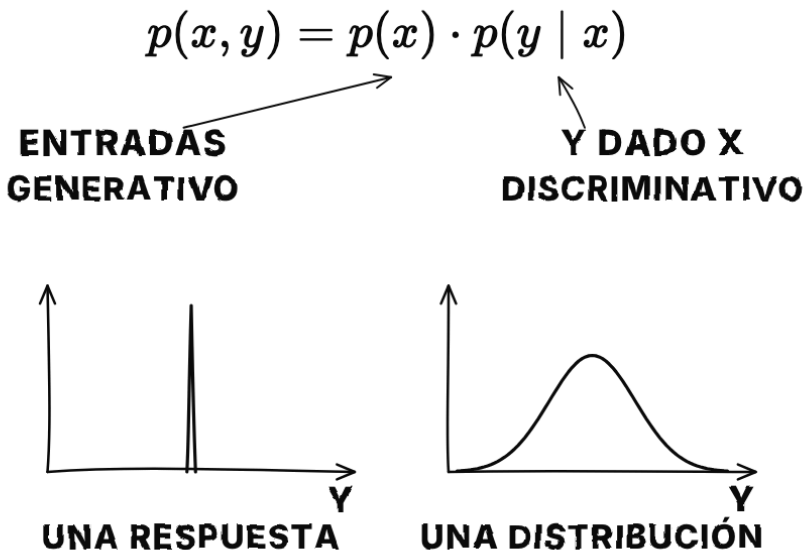


La brecha de generalización

La diferencia entre el riesgo esperado y el riesgo empírico se llama brecha de generalización. Un memorizador puro, como la tabla de consulta, tiene una brecha enorme: se ajusta a sus datos de entrenamiento particulares y falla más allá, que es lo que significa sobreajuste. En la práctica la brecha se mide con un conjunto de prueba separado, ejemplos apartados y nunca usados durante el entrenamiento. Si la pérdida promedio es cinco centésimas en el conjunto de entrenamiento y cuarenta centésimas en el de prueba, la brecha de treinta y cinco centésimas es una medida aproximada de cuánto sobreajusta el modelo. Antes de lanzar una nueva versión de la

aplicación de fotos, sus creadores la comprueban con imágenes que nunca vio durante el entrenamiento, exactamente por esta razón.

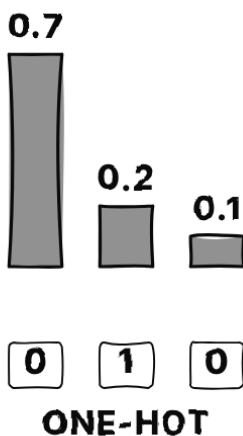
¿Cómo debería elegirse la propia pérdida? El libro sugiere que este es un buen momento para leer su apéndice sobre probabilidad, sobre todo distribuciones, estadísticos suficientes y máxima verosimilitud. La pérdida codifica lo que sabemos de antemano sobre la tarea y afecta fuertemente al rendimiento. A veces basta el sentido común, como con la pérdida de bisagra de la parte anterior. Pero hay una ruta más principista que reformula todo el entrenamiento en el lenguaje de la probabilidad, un punto de vista que muchos libros prefieren, y el libro expone ahora su idea básica.



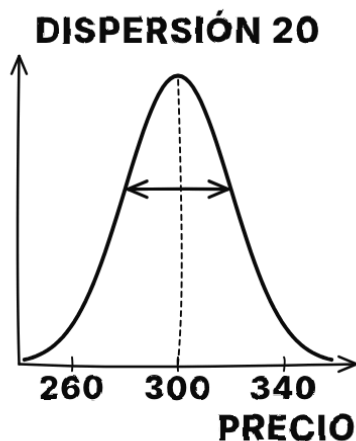
Dividir la distribución conjunta

Supusimos que los ejemplos vienen de una distribución conjunta sobre entradas y salidas. Por la regla del producto de la probabilidad, se divide en dos piezas: lo probable que es cada entrada, y lo probable que es cada salida una vez conocida la entrada, la distribución condicional de la salida dada la entrada. Una nota al pie añade que también podría dividirse al revés; los métodos que modelan las entradas se llaman generativos, los que modelan solo la salida dada la entrada se llaman discriminativos, y aparte de los modelos de lenguaje, el libro se concentra en el segundo tipo. Usar un modelo que devuelve una sola predicción para cada entrada tiene sentido cuando, para cada entrada, casi toda la probabilidad se concentra en una salida, un pico que los matemáticos llaman delta de Dirac. Eso lo simplifica todo, pero se puede relajar.

En lugar de producir la predicción en sí, el modelo puede producir los números que describen una distribución completa sobre las salidas posibles, sus estadísticos suficientes. Tome una clasificación con tres clases. El modelo devuelve tres probabilidades que suman uno, un vector en el simplex visto antes, digamos siete décimas, dos décimas y una décima. La clase verdadera se escribe como un vector one-hot: todo ceros salvo un uno en la posición de la clase. La probabilidad que el modelo da a la clase verdadera es entonces el producto de cada probabilidad predicha elevada a la entrada correspondiente del vector one-hot, lo que simplemente selecciona la probabilidad predicha de la clase verdadera. Si la clase verdadera es la segunda, el vector one-hot es cero, uno, cero, y el producto es dos décimas elevado a uno, por las otras elevadas a cero, que es dos décimas.



$$0.7^0 \cdot 0.2^1 \cdot 0.1^0 = 0.2$$



Predecir una distribución

Para predecir un solo número, lo que se llama regresión, el modelo puede devolver en cambio dos números: el centro y la dispersión de una curva de campana, una gaussiana, sobre las salidas posibles. Para el ejemplo del precio del piso, el modelo podría decir: alrededor de trescientos mil, veinte mil más o menos. Así el modelo expresa no solo una suposición sino cuánta seguridad tiene, y la siguiente parte muestra cómo esas predicciones probabilísticas llevan a pérdidas de una manera principista.

Para recordar

Una tabla de consulta de los datos de entrenamiento alcanza el riesgo empírico más bajo posible y es inútil, lo que separa memorizar de aprender. El objetivo real es el riesgo esperado, la pérdida promedio sobre todos los datos posibles, que no se puede calcular; el riesgo empírico es una estimación de Monte Carlo de él. La diferencia entre ellos es la brecha de generalización; el sobreajuste significa una brecha grande, medida en la práctica con un conjunto de prueba separado nunca usado en el entrenamiento. En la visión probabilística, la distribución conjunta se divide en una distribución sobre las entradas y una distribución condicional de las salidas dadas las entradas, que es la que aprenden los modelos discriminativos. Un modelo puede dar como salida la descripción de una distribución, como probabilidades de clase puntuadas frente a un objetivo one-hot, o el centro y la dispersión de una gaussiana para la regresión.

Preguntas

1. ¿Por qué el modelo de tabla de consulta no vale nada aunque su riesgo empírico sea el más bajo posible?
 - a. Porque solo reproduce las respuestas de entrenamiento guardadas y devuelve un valor por defecto para toda entrada nueva
 - b. Porque su pérdida en los datos de entrenamiento es muy alta

- c. Porque no puede guardar todo el conjunto de datos
 - d. Porque su gradiente es demasiado grande
2. ¿Por qué no se puede calcular directamente el riesgo esperado?
- a. Porque la función de pérdida no es diferenciable
 - b. Porque exigiría enumerar y etiquetar todas las entradas posibles
 - c. Porque siempre es cero
 - d. Porque el conjunto de prueba es demasiado pequeño
3. La pérdida promedio de un modelo es cinco centésimas en los datos de entrenamiento y cuarenta centésimas en un conjunto de prueba apartado. ¿Qué sugiere esto?
- a. El modelo generaliza perfectamente
 - b. El conjunto de prueba se usó para entrenar
 - c. El modelo sobreajusta: la brecha de generalización es de unas treinta y cinco centésimas
 - d. La función de pérdida es incorrecta
4. Un modelo da las probabilidades siete décimas, dos décimas y una décima a tres clases, y la clase verdadera es la tercera. ¿Qué probabilidad da a la clase verdadera?
- a. Siete décimas
 - b. Dos décimas
 - c. Uno
 - d. Una décima

Respuestas

1. **a)** Memoriza perfectamente los pares de entrenamiento pero no ha aprendido nada sobre las entradas nuevas, así que su riesgo esperado, el objetivo real, es malo.
2. **b)** El riesgo esperado promedia sobre todos los pares entrada-salida posibles, que nunca se pueden enumerar y etiquetar; el riesgo empírico lo estima en su lugar.
3. **c)** La diferencia entre la pérdida de prueba y la de entrenamiento aproxima la brecha de generalización; una brecha grande señala sobreajuste.
4. **d)** Con el vector one-hot cero, cero, uno, el producto selecciona la probabilidad predicha de la tercera clase, una décima.

Ejemplos resueltos

Seleccionar la clase verdadera Un modelo predice las probabilidades de clase un quinto, tres quintos y un quinto para una foto cuya clase verdadera es la segunda. Escriba el vector one-hot y calcule la probabilidad que el modelo asigna a la clase verdadera.

La clase verdadera es la segunda, así que el vector one-hot es cero, uno, cero.

Eleve cada probabilidad predicha a la entrada correspondiente: un quinto elevado a cero es uno, tres quintos elevado a uno es tres quintos, un quinto elevado a cero es uno.

Multiplique los tres resultados: uno por tres quintos por uno.

El producto es tres quintos, simplemente la probabilidad predicha de la clase verdadera.

Respuesta: Tres quintos.

Ejercicios

1. Explique la diferencia entre memorizar y aprender, usando la aplicación de fotos como ejemplo.

Pista: Piense en qué pasa con una foto que la aplicación nunca ha visto.

Respuesta: Una aplicación que memoriza recordaría las etiquetas de sus fotos de entrenamiento y fallaría con cualquier foto nueva. Una aplicación que aprende captura los patrones detrás de las fotos y etiqueta correctamente las nuevas; por eso sus creadores la prueban con fotos nunca usadas en el entrenamiento.

2. ¿Por qué un promedio sobre un conjunto de datos es una estimación razonable del riesgo esperado, y cuándo fallaría?

Pista: Recuerde la comparación con el lanzamiento del dado y la suposición i.i.d.

Respuesta: Igual que promediar muchos lanzamientos de dado para estimar tres y medio, promediar la pérdida sobre ejemplos extraídos de la distribución verdadera estima el promedio sobre todos los datos. Falla cuando los ejemplos no son representativos, por ejemplo cuando se rompe la suposición i.i.d., o cuando el modelo simplemente los ha memorizado.

3. Para predecir el precio de un piso, el modelo da un centro y una dispersión. ¿Qué expresa «trescientos mil, veinte mil más o menos» que un solo número no expresa?

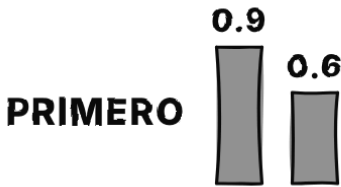
Pista: Piense en la curva de campana.

Respuesta: Describe una distribución gaussiana de precios posibles: el valor más probable y cuánta incertidumbre tiene el modelo. Un solo número da la suposición pero no la confianza del modelo en ella.

La explicación más probable, y muchas explicaciones a la vez

Dos pronosticadores del tiempo llevan una semana prediciendo lluvia. Los días que llovió, el primero había dado a la lluvia una probabilidad de nueve sobre diez y de seis sobre diez; el segundo había dicho un medio ambas veces. ¿En qué pronosticador confía más? Multiplique las probabilidades que cada uno dio a lo que realmente ocurrió: nueve décimas por seis décimas es cincuenta y cuatro centésimas para el primero, un medio por un medio es un cuarto para el segundo. El primer pronosticador hizo que lo ocurrido pareciera mucho más probable, y esa idea simple, llamada máxima verosimilitud, es como se entrenan los modelos probabilísticos.

$$\text{dispersión} = s^2$$



$$0.9 \times 0.6 = 0.54$$



$$0.5 \times 0.5 = 0.25$$

MÁXIMA VEROSIMILITUD

Dos pronosticadores, dos productos

Primero, un detalle pendiente de la parte anterior: cuando un modelo predice una curva de campana, su segunda salida se eleva al cuadrado para dar la dispersión, de modo que la dispersión nunca pueda volverse negativa. Este planteamiento probabilístico es muy general e incluye todo lo dicho sobre pérdidas hasta ahora. Da al diseñador más libertad, ya que puede ser más fácil elegir una forma para la distribución de las salidas dada una entrada que inventar directamente una función de pérdida, y expresa la incertidumbre de forma natural, a través de la dispersión.

¿Cómo se entrena un modelo así? Como se supone que los ejemplos son independientes y extraídos de la misma distribución, la probabilidad que un modelo asigna a todo el conjunto de datos es el

producto de las probabilidades que asigna a la salida verdadera de cada ejemplo. Ese producto se llama la verosimilitud del conjunto de datos. Un modelo elegido al azar reparte sus probabilidades más o menos al azar y da a nuestro conjunto de datos particular una verosimilitud pequeña. Así que la elección sensata es el modelo que hace la verosimilitud lo más grande posible: la solución de máxima verosimilitud. La diferencia respecto a antes es que el objetivo de entrenamiento ahora se deduce de las reglas de la probabilidad una vez elegidas las distribuciones, en lugar de diseñarse a mano. Para la aplicación de fotos, esto significa preferir los ajustes bajo los cuales las etiquetas verdaderas de sus imágenes de entrenamiento parecen lo más probables posible.

$$\text{verosimilitud} = p_1 p_2 \cdots p_n$$

HACERLA MÁXIMA

$$0.9^{1000} \approx 1.7 \times 10^{-46}$$

↓

$$-\log \text{verosimilitud} = \sum_{i=1}^n -\log p_i$$

MINIMIZAR = UNA PÉRDIDA

$$-\log 0.9 \approx 0.105, \quad -\log 0.6 \approx 0.511$$

$$0.105 + 0.511 \approx 0.616 = -\log 0.54$$

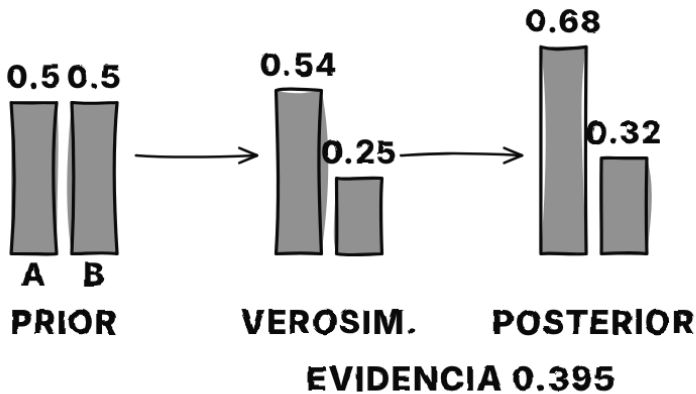
Del producto a la suma de logaritmos

Los dos puntos de vista están estrechamente ligados. Los productos de muchas probabilidades se vuelven absurdamente pequeños: mil ejemplos, cada uno con nueve probabilidades sobre diez, tienen una verosimilitud conjunta de unos dos entre cien mil billones de billones de billones, una coma decimal seguida de cuarenta y cinco ceros. Tomar el logaritmo convierte el producto en una suma, y cambiar el signo convierte maximizar en minimizar. Así que maximizar la verosimilitud es lo mismo que minimizar la suma, sobre los ejemplos, de menos el logaritmo de la probabilidad dada a cada salida verdadera. Con el primer pronosticador, menos el logaritmo de nueve décimas es alrededor de una décima y menos el logaritmo de seis décimas es alrededor de un medio, unas seis décimas en total, que es exactamente menos el logaritmo de cincuenta y cuatro centésimas. Ese término de menos logaritmo hace el papel de una pérdida, y el libro anuncia que toda pérdida usada en la práctica se puede derivar de este modo a partir de una elección particular de distribución. Insta al lector a tener en mente ambos puntos de vista.

La sección tres punto tres da un paso más, hacia el aprendizaje bayesiano y las redes neuronales bayesianas, que el libro solo esboza, remitiendo a tutoriales para profundizar. Un modelo probabilístico ya puede permitir varias salidas plausibles para una entrada. Pero el entrenamiento sigue eligiendo una única función, mientras que varios ajustes bastante distintos del modelo podrían explicar los datos igual de bien, y conservarlos todos podría dar predicciones más fieles. Así que ponemos una distribución de probabilidad sobre las propias funciones. Empezamos con una prior, una distribución que expresa qué funciones favorecemos antes de ver ningún dato; como un modelo se describe por sus parámetros, la prior es en realidad una distribución sobre los pesos. Por ejemplo, las funciones con una

norma menor suelen preferirse porque son más estables, así que la prior puede darles más probabilidad.

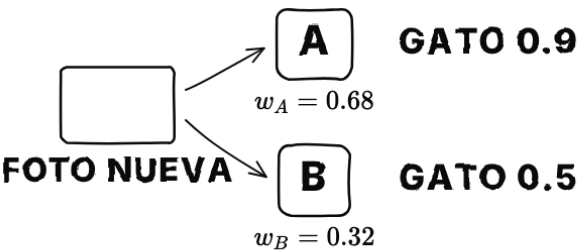
$$\text{posterior} = \frac{\text{verosim.} \times \text{prior}}{\text{evidencia}}$$



Prior por verosimilitud

Una vez observado el conjunto de datos, el teorema de Bayes actualiza la prior: la posterior, la distribución después de ver los datos, es la verosimilitud por la prior, dividida por un número normalizador llamado la evidencia, que hace que todo sume uno. Aquí va un caso diminuto con solo dos modelos candidatos, cada uno con una prior de un medio, y datos cuya verosimilitud es cincuenta y cuatro centésimas bajo el primero y un cuarto bajo el segundo. Verosimilitud por prior da veintisiete centésimas y doce centésimas y media; su suma, unas cuarenta centésimas, es la evidencia; dividiendo, la posterior da al primer modelo unos sesenta y ocho sobre cien y al segundo unos

treinta y dos. Para predecir, promediamos las predicciones de todos los modelos, cada una ponderada por su posterior. Si el primer modelo da a una foto nueva una probabilidad de nueve décimas de ser un gato y el segundo un medio, la predicción promediada es de unos setenta y siete sobre cien. Con modelos reales hay infinitos candidatos, y el promedio se aproxima extrayendo un puñado de modelos de la posterior, de nuevo un promedio de Monte Carlo.



$$0.68 \cdot 0.9 + 0.32 \cdot 0.5 \approx 0.77$$



Promedio sobre los modelos

La pega, admite el libro, es que la posterior casi nunca se puede calcular con exactitud, salvo para elecciones muy especiales de prior y verosimilitud. Hay que aproximarla, bien por Monte Carlo con cadenas de Markov, que extrae muestras mediante un largo paseo aleatorio, bien por inferencia variacional, que ajusta en su lugar una

distribución más simple; la sección nueve punto tres mostrará un ejemplo práctico llamado Monte Carlo dropout.

Para recordar

Un modelo probabilístico se entrena por máxima verosimilitud: elegir el modelo que da al conjunto de datos observado la probabilidad más alta, el producto de las probabilidades de cada salida verdadera. Tomar logaritmos convierte el producto en una suma, así que maximizar la verosimilitud equivale a minimizar la suma de menos los logaritmos de las probabilidades, que actúa como una pérdida; todas las pérdidas prácticas surgen así. El aprendizaje bayesiano conserva muchos modelos a la vez: una prior sobre los modelos, actualizada por el teorema de Bayes en una posterior, verosimilitud por prior dividida por la evidencia. Las predicciones promedian todos los modelos ponderados por su posterior, en la práctica muestreando unos pocos; con dos modelos ponderados sesenta y ocho y treinta y dos sobre cien, probabilidades de gato de nueve décimas y un medio se promedian a unos setenta y siete sobre cien. La posterior rara vez es calculable con exactitud, así que se aproxima por Monte Carlo con cadenas de Markov o por inferencia variacional.

Preguntas

1. ¿Qué elige la máxima verosimilitud?

- a. El modelo bajo el cual el conjunto de datos observado es más probable
 - b. El modelo con menos parámetros
 - c. El modelo con la prior más grande
 - d. El modelo que predice un medio para todo
2. ¿Por qué se toman logaritmos antes de optimizar la verosimilitud?
- a. Porque las probabilidades pueden ser negativas
 - b. Porque los logaritmos hacen bayesiano al modelo
 - c. Porque el producto de muchas probabilidades se vuelve diminuto, y el logaritmo lo convierte en una suma manejable
 - d. Porque los logaritmos eliminan la necesidad de datos
3. En el teorema de Bayes, ¿qué es la posterior?
- a. La distribución sobre los modelos antes de ver los datos
 - b. La probabilidad del conjunto de datos bajo un modelo
 - c. Una función de pérdida fija
 - d. La distribución sobre los modelos después de ver los datos, verosimilitud por prior dividida por la evidencia
4. ¿Por qué el aprendizaje bayesiano es difícil en la práctica?
- a. Porque las priors no se pueden definir
 - b. Porque la posterior casi nunca se puede calcular con exactitud y hay que aproximarla
 - c. Porque usa un solo modelo
 - d. Porque no puede hacer predicciones

Respuestas

1. **a)** La verosimilitud es el producto de las probabilidades dadas a cada salida verdadera; la máxima verosimilitud escoge el modelo que hace este producto más grande.
2. **c)** Un producto sobre miles de ejemplos es ínfimo; su logaritmo es una suma, y minimizar menos esa suma es lo mismo que maximizar la verosimilitud.
3. **d)** La posterior actualiza la prior con la verosimilitud de los datos observados, normalizada por la evidencia.
4. **b)** Salvo para elecciones especiales de prior y verosimilitud, la posterior no tiene forma cerrada, así que se usan Monte Carlo con cadenas de Markov o la inferencia variacional.

Ejemplos resueltos

Actualizar dos modelos con el teorema de Bayes Dos modelos candidatos tienen priors de un quinto y cuatro quintos. Los datos tienen verosimilitud cincuenta y cuatro centésimas bajo el primero y un cuarto bajo el segundo. Halle la posterior de cada uno.

Multiplique verosimilitud por prior: cincuenta y cuatro centésimas por un quinto es alrededor de once centésimas, exactamente ciento ocho milésimas; un cuarto por cuatro quintos es dos décimas.

Súmelos para obtener la evidencia: ciento ocho milésimas más doscientas milésimas es trescientas ocho milésimas.

Divida cada producto por la evidencia: ciento ocho entre trescientos ocho es alrededor de treinta y cinco sobre cien.

El segundo recibe doscientos entre trescientos ocho, alrededor de sesenta y cinco sobre cien.

Respuesta: Posterior de unos treinta y cinco sobre cien para el primer modelo y sesenta y cinco para el segundo.

Ejercicios

1. Un modelo da a las salidas verdaderas de tres ejemplos las probabilidades ocho décimas, un medio y nueve décimas. Calcule la verosimilitud y la suma de menos los logaritmos naturales.

Pista: Multiplique para la verosimilitud; menos los logaritmos naturales son alrededor de veintidós centésimas, sesenta y nueve centésimas y once centésimas.

Respuesta: La verosimilitud es ocho décimas por un medio por nueve décimas, treinta y seis centésimas. La suma de menos los logaritmos es alrededor de veintidós más sesenta y nueve más once centésimas, unos uno coma cero dos, que es igual a menos el logaritmo de treinta y seis centésimas.

2. Dos modelos tienen posteriors de siete décimas y tres décimas. Para una foto nueva, el primero dice gato con probabilidad ocho décimas y el segundo con probabilidad cuatro décimas. ¿Cuál es la predicción promediada?

Pista: Pondere cada predicción por su posterior y sume.

Respuesta: Siete décimas por ocho décimas es cincuenta y seis centésimas, y tres décimas por cuatro décimas es doce centésimas; juntas, sesenta y ocho centésimas.

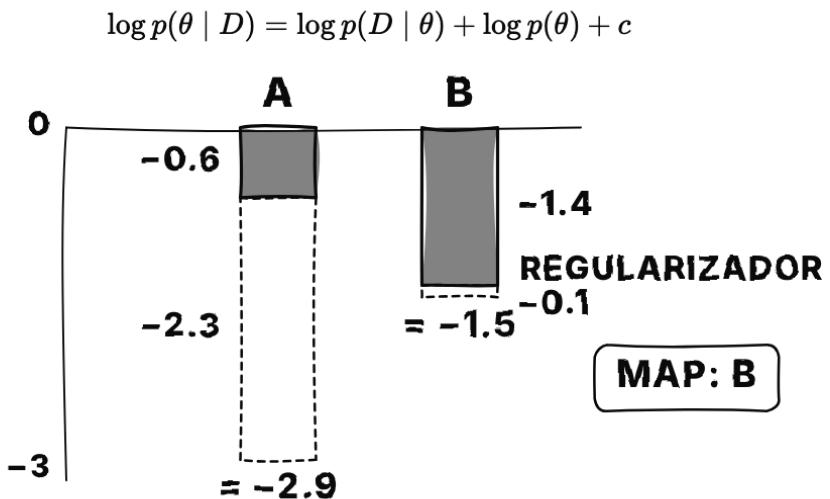
3. ¿Por qué una prior podría favorecer funciones con una norma menor?

Pista: Piense en con qué fuerza reacciona una función así a cambios pequeños.

Respuesta: Las funciones con norma menor son en general más estables, reaccionan de forma menos brusca a cambios pequeños de la entrada, así que antes de ver ningún dato es razonable considerarlas más plausibles.

Una penalización desde la prior, y el bloque más simple

Suponga que debe elegir un solo modelo, no toda una multitud. El pensamiento bayesiano sigue ayudando: elija el único modelo con la posterior más alta. Como la evidencia es la misma para todos los modelos, se puede ignorar, y tomar logaritmos divide el objetivo en dos partes que se suman: la log-verosimilitud, qué tan bien explica el modelo los datos, y el logaritmo de la prior, qué tan plausible parecía el modelo de antemano. Esto se llama la solución de máximo a posteriori.



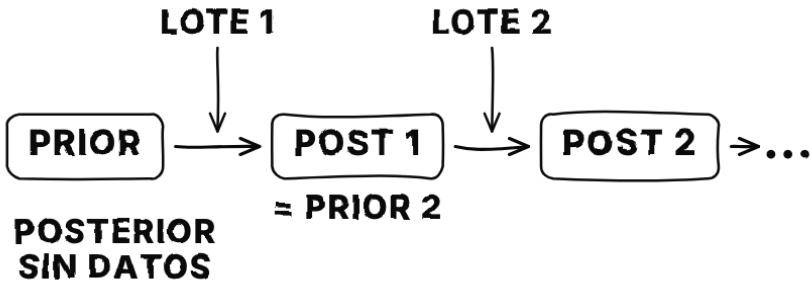
La prior puede dar la vuelta a la elección

Así es como la prior puede cambiar el veredicto. Digamos que el modelo A explica los datos con verosimilitud cincuenta y cuatro centésimas, un logaritmo de unas menos seis décimas, y el modelo B con un cuarto, un logaritmo de unos menos uno coma cuatro. La máxima verosimilitud elige A. Pero si la prior daba a A solo una probabilidad de uno sobre diez, un logaritmo de unos menos dos coma tres, y a B nueve sobre diez, un logaritmo de unas menos una décima, los totales pasan a ser unos menos dos coma nueve para A y menos uno coma cinco para B, y la elección de máximo a posteriori es B. Si todos los modelos tuvieran la misma prior, la segunda parte sería una constante y volveríamos a la máxima verosimilitud simple. En general, sin embargo, la prior penaliza los modelos que se alejan demasiado de lo que esperábamos, lo que el libro llama un regularizador: tira de la solución hacia la región que la prior favorece, un arma útil contra el sobreajuste y una forma de imponer restricciones.

Una nota al pie vincula esto vagamente con un viejo debate sobre qué significa la probabilidad. Para los frequentistas, la probabilidad es la frecuencia con que ocurren los eventos, y la máxima verosimilitud trata los parámetros como incógnitas fijas y los datos como aleatorios. Para los bayesianos, la probabilidad mide la incertidumbre, y los datos son fijos mientras que los parámetros son aleatorios.

FRECUENTISTA
 θ FIJO

BAYESIANO
 θ ALEATORIO

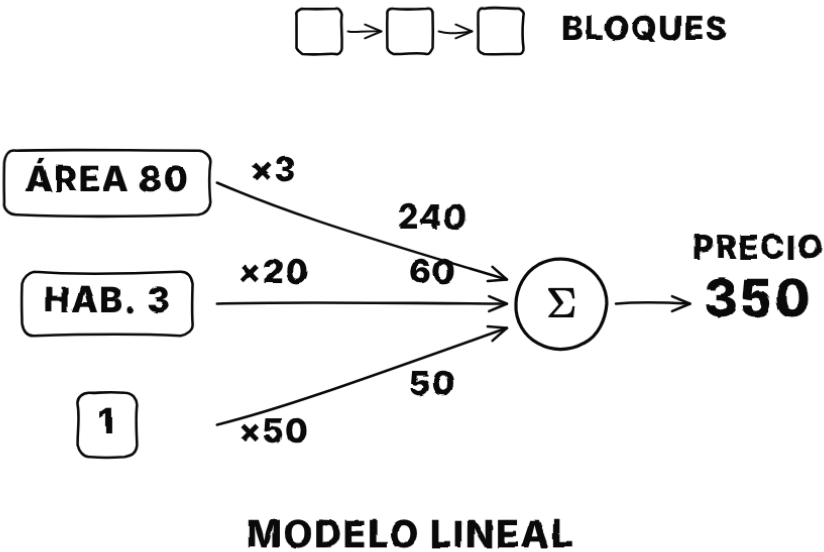


ENTRENAMIENTO EN LÍNEA SIN OLVIDO

La posterior de hoy, la prior de mañana

El tratamiento bayesiano completo tiene una segunda ventaja: los datos nuevos se pueden incorporar con facilidad. Cuando llega un nuevo lote de datos de la misma distribución, la posterior calculada con el primer lote simplemente se convierte en la prior para el segundo. La nota al pie del libro lo expresa con elegancia: la prior original es simplemente la posterior tras no observar nada. Imagine la aplicación de fotos aprendiendo de sus fotos nuevas cada semana: llevar la posterior hacia adelante de esta manera la ayuda a aprender caras nuevas sin olvidar las viejas. Entrenar a medida que llegan los datos se llama entrenamiento en línea, y este enfoque puede aliviar su problema más conocido, el olvido catastrófico, en el que un modelo pierde conocimiento viejo mientras aprende cosas nuevas.

Con esto empieza el capítulo cuatro, sobre modelos lineales. Programar, nos recuerda el libro, es elegir una buena secuencia de operaciones básicas; construir un modelo es elegir una buena secuencia de bloques diferenciiables. El bloque más simple es el modelo lineal, donde cada entrada añade su propia contribución ponderada a la salida. Para un piso, un modelo lineal podría decir: el precio es igual a tres veces la superficie, más veinte veces el número de habitaciones, más cincuenta; un piso de ochenta metros cuadrados con tres habitaciones obtiene entonces doscientos cuarenta más sesenta más cincuenta, trescientos cincuenta. En cierto sentido, dice el libro, todo modelo diferenciable es una variación o combinación ingeniosa de estos bloques lineales.



El bloque más simple: lineal

Para recordar

La solución de máximo a posteriori elige el único modelo con la posterior más alta: la log-verosimilitud más el log-prior, siendo irrelevante la evidencia. Con una prior uniforme se reduce a la máxima verosimilitud; de lo contrario el log-prior actúa como regularizador, penalizando los modelos improbables y combatiendo el sobreajuste, y puede revertir la elección de la verosimilitud. La máxima verosimilitud encaja con la visión frecuentista, con parámetros fijos y datos aleatorios; la visión bayesiana trata los parámetros como aleatorios. En un tratamiento bayesiano la posterior se convierte en la siguiente prior cuando llegan datos nuevos, lo que ayuda al entrenamiento en línea contra el olvido catastrófico. El capítulo cuatro empieza con los modelos lineales, donde las entradas añaden contribuciones ponderadas, el bloque básico de todos los modelos diferenciables.

Preguntas

1. ¿Cuándo coincide la solución de máximo a posteriori con la máxima verosimilitud?
 - a. Cuando todos los modelos tienen la misma prior, de modo que el log-prior es una constante
 - b. Cuando el conjunto de datos está vacío
 - c. Cuando la verosimilitud es uno
 - d. Nunca

2. ¿Por qué el libro llama regularizador al log-prior?
 - a. Porque hace más grande la verosimilitud
 - b. Porque penaliza los modelos que se alejan de lo que la prior favorece, tirando de la solución hacia ellos
 - c. Porque elimina la necesidad del descenso de gradiente
 - d. Porque normaliza la posterior
3. ¿Cómo absorbe un tratamiento bayesiano un nuevo lote de datos?
 - a. Descartando la posterior antigua
 - b. Reentrenando desde la prior original solo con el lote nuevo
 - c. Usando la posterior del primer lote como prior para el segundo
 - d. Promediando las verosimilitudes de los dos conjuntos de datos
4. Un modelo lineal valora un piso como tres veces la superficie más veinte veces las habitaciones más cincuenta. ¿Qué da para sesenta metros cuadrados y dos habitaciones?
 - a. Trescientos cincuenta
 - b. Doscientos treinta
 - c. Doscientos setenta
 - d. Doscientos setenta, menos cincuenta

Respuestas

1. a) Con una prior uniforme el segundo término añade la misma cantidad a todos los modelos, así que solo decide la verosimilitud.

2. **b)** Añadir el log-prior penaliza los modelos improbables, lo que ayuda contra el sobreajuste e impone restricciones.
3. **c)** La posterior tras el primer lote se convierte en la creencia de partida, actualizada por los datos nuevos, lo que ayuda contra el olvido catastrófico.
4. **c)** Tres por sesenta es ciento ochenta, veinte por dos es cuarenta, más cincuenta da doscientos setenta.

Ejemplos resueltos

Cuando la prior revierte la verosimilitud El modelo A tiene log-verosimilitud de unas menos seis décimas y el modelo B de unos menos uno coma cuatro. Sus priors son uno sobre diez y nueve sobre diez, con logaritmos de unos menos dos coma tres y menos una décima. ¿Qué modelo elige la máxima verosimilitud, y cuál elige el máximo a posteriori?

La máxima verosimilitud compara solo las log-verosimilitudes: menos seis décimas gana a menos uno coma cuatro, así que elige A.

El máximo a posteriori añade el log-prior: para A, menos seis décimas más menos dos coma tres es alrededor de menos dos coma nueve.

Para B, menos uno coma cuatro más menos una décima es alrededor de menos uno coma cinco.

El total mayor es menos uno coma cinco, así que el máximo a posteriori elige B.

Respuesta: La máxima verosimilitud elige A; el máximo a posteriori elige B.

Ejercicios

1. Explique con sus propias palabras la diferencia entre la visión frecuentista y la bayesiana tal como la esboza el libro.

Pista: ¿Qué se trata como aleatorio: los parámetros o los datos?

Respuesta: En la visión frecuentista, la probabilidad es la frecuencia de los eventos; los parámetros son fijos pero desconocidos y los datos son una muestra aleatoria. En la visión bayesiana, la probabilidad mide la incertidumbre; los datos son fijos y los parámetros son variables aleatorias con una distribución.

2. ¿Qué es el olvido catastrófico, y por qué puede ayudar llevar la posterior hacia adelante?

Pista: Piense en un modelo entrenado semana tras semana con datos nuevos.

Respuesta: Es cuando un modelo entrenado con datos nuevos pierde lo que aprendió de los datos viejos. Usar la posterior anterior como nueva prior conserva el conocimiento viejo como creencia de partida que los datos nuevos solo ajustan, en lugar de sobrescribir.

3. ¿Por qué dice el libro que todos los modelos diferenciables son, en cierto sentido, variaciones de bloques lineales?

Pista: Compare construir un modelo con escribir un programa.

Respuesta: Igual que los programas son secuencias de operaciones básicas, los modelos son secuencias de bloques diferenciables, y el bloque más básico es el lineal, donde las entradas añaden contribuciones ponderadas; los modelos más potentes surgen de variaciones y composiciones ingeniosas de esos bloques.