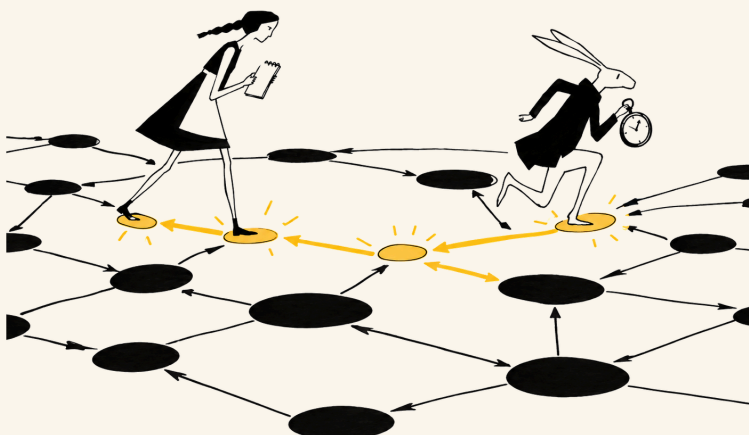


Las aventuras de Alicia en el País de las Maravillas Diferenciables



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

Por qué mirar bajo el capó

Aprender es buscar

Ideas viejas, poder nuevo

Cajas de números y sus formas

Qué es este libro, y qué no es

Cortar rebanadas de un tensor

Flechas, longitudes y coincidencia

Matrices como lotes y como funciones

Cómo crece el coste

Muchos productos a la vez, y una abreviatura para todos ellos

La pendiente en un punto

Pendientes en muchas direcciones

Tamaños ocultos, y el primer paso cuesta abajo

El descenso más pronunciado, y dónde termina

Cuencos, paisajes accidentados y un empujón de momento

El precio de la memoria, y dónde practicar

Un paisaje para practicar, y qué es un conjunto de datos

Aprender sin etiquetas, y reutilizar lo aprendido

Puntuar los errores: la pérdida

Pérdidas que pueden seguirse, y modelos que solo recuerdan

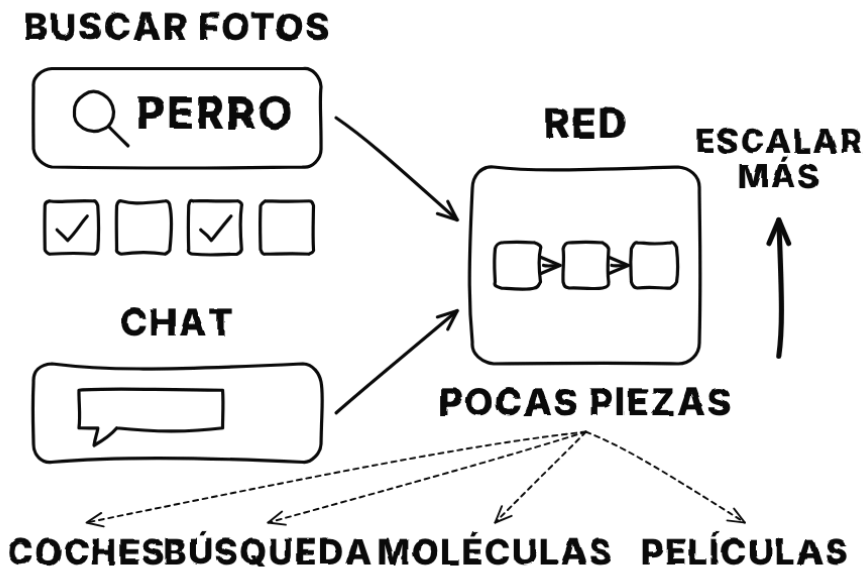
Predecir una distribución en vez de un número

Hacer probables los datos, y conservar todo modelo bueno

Aproximaciones, previas como penalizaciones y aprendizaje a plazos

Por qué mirar bajo el capó

Abra la aplicación de fotos de su teléfono y escriba una sola palabra en el buscador, digamos «perro». En un instante le muestra todas las fotos donde aparece un perro, aunque nadie etiquetó nunca sus fotos a mano. Ahora abra un asistente de chat y pídale que explique una receta con palabras más sencillas; responde con frases fluidas. Los dos trucos, nos dice el libro, los hacen redes neuronales, y ambas están construidas a partir de un conjunto sorprendentemente pequeño de principios y componentes. De esas pocas piezas trata todo este curso.

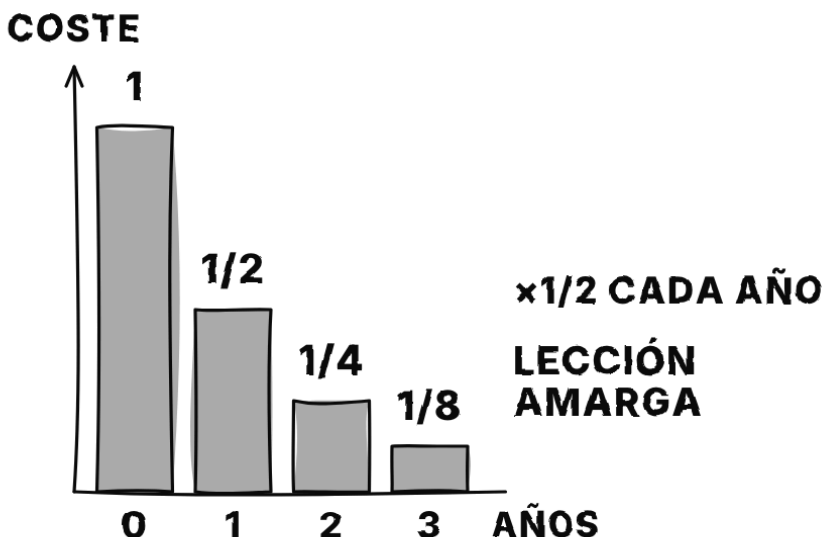


Dos aplicaciones, un mismo juego de piezas

Las redes neuronales se han vuelto parte de la vida cotidiana, a veces a la vista, como los grandes modelos de lenguaje, o LLM, y a veces escondidas. Mueven drones y coches, buscadores, el diseño de nuevas moléculas y las recomendaciones de qué ver a continuación. La primera afirmación del libro es que todo esto descansa sobre muy pocas ideas, y que el esfuerzo de investigación ha pasado de inventar ideas nuevas a escalar las viejas hasta los límites de lo físicamente posible.

Ese cambio tiene nombre: leyes de escalado neuronal. De manera informal, para prácticamente cualquier tarea, aumentar a la vez los datos, la potencia de cálculo y el tamaño del modelo casi siempre da una mejora predecible de la precisión. Dicho al revés, la potencia de cálculo necesaria para alcanzar una precisión dada se reduce por un factor constante cada cierto periodo de tiempo. Esto es lo que significa un factor constante con números que podemos comprobar: si el coste de alcanzar cierta precisión se reduce a la mitad cada año, al cabo de tres años es un medio por un medio por un medio, es decir, un octavo de lo que era. El investigador Richard Sutton llamó a esto la lección amarga: las herramientas sencillas y de propósito general, combinadas con una potencia de cálculo que crece exponencialmente, siguen ganando a los diseños especiales más ingeniosos.

LEYES ESCALADO

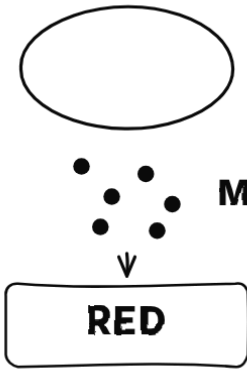


El coste se reduce a la mitad cada año

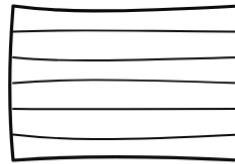
Si damos por sentadas las leyes de escalado, dice el libro, nos queda una herramienta casi mágica. En pocas palabras, una red neuronal se optimiza para aproximar alguna distribución de probabilidad a partir de datos extraídos de ella. Piense en la distribución como la regla oculta que produce todas las fotos de perros del mundo; la red ve una muestra de ellas e intenta captar la regla. En principio esto puede fallar: las redes modernas son tan grandes que podrían simplemente memorizar cada ejemplo que se les muestra y convertirse en una trivial tabla de consulta. En cambio, los modelos entrenados resultan generalizar bien, incluso a tareas que nunca estuvieron explícitamente en sus datos de entrenamiento. A medida que los conjuntos de datos crecen, la línea entre lo que está dentro de la distribución de entrenamiento y lo que está fuera se difumina, y los modelos muy

grandes muestran indicios de una fuerte generalización con una dependencia sorprendentemente pequeña de la pura memorización, que se llama sobreajuste.

REGLA OCULTA

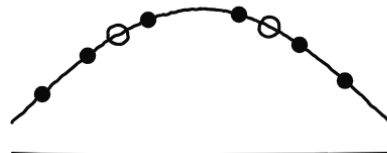


MUESTRA



COPIA

NUEVO

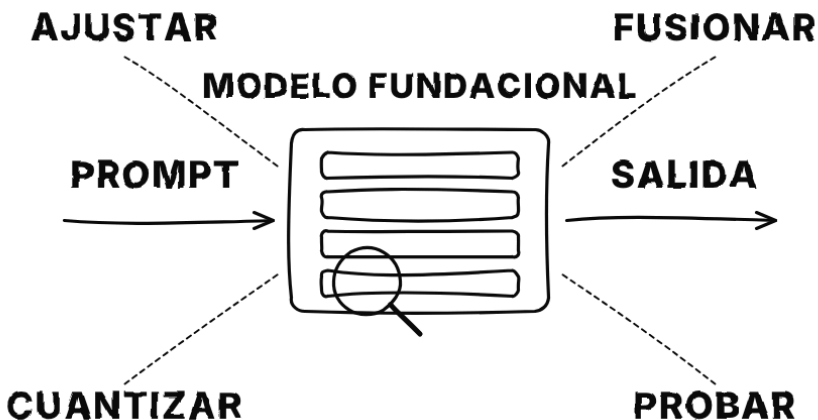


GENERALIZA

Muestras de una regla oculta

Los modelos extremadamente grandes que pueden reutilizarse para muchas tareas posteriores se llaman a veces modelos fundacionales. Junto con una animada comunidad de código abierto, han cambiado la forma en que usamos estos sistemas. Muchas tareas pueden resolverse ahora simplemente con prompting, es decir, dando instrucciones en texto o en imágenes a un modelo preentrenado encontrado en la web, mientras su interior sigue siendo una caja negra completa. El libro compara esto con el paso de escribir sus propias bibliotecas en un lenguaje como C++ a depender de software

terminado cuyo código fuente nunca ve. La comparación no es exagerada: hoy solo unos pocos equipos en el mundo tienen la potencia de cálculo y la experiencia para diseñar y publicar modelos realmente grandes, como la familia Llama de modelos de lenguaje, igual que solo unas pocas empresas pueden construir grandes programas empresariales.



Caja negra, luego abierta

Pero igual que el código abierto le permite personalizar programas o escribir otros nuevos desde cero, el hardware de consumo corriente y un poco de ingenio le dan un amplio abanico de cosas que probar con estos modelos. Puede ajustarlos finamente para su propia tarea, fusionar dos modelos en uno, cuantizarlos, es decir, guardar sus números con menos bits para que funcionen en hardware de bajo

consumo, probar su robustez o diseñar variantes completamente nuevas. Para todo eso necesita mirar bajo el capó y entender cómo estos modelos procesan y transforman los datos por dentro, con todos los trucos y rarezas que se aprendieron de la experiencia y la depuración. El libro se presenta como una puerta de entrada a ese mundo, y se dirige a su lector como alguien que, como Alicia, es curioso por naturaleza.

Pruébelo

Hágale al mismo asistente de chat la misma pregunta dos veces, una con una frase corta y sencilla y otra con unas pocas instrucciones extra sobre el tono y la longitud. Espere que las dos respuestas sean distintas aunque el modelo en sí no haya cambiado en absoluto. Lo único que cambió es el prompt; la red de dentro es la misma caja negra, que es exactamente la forma de trabajar que describe el libro, y la razón por la que quiere que usted abra la caja.

Para recordar

Las redes neuronales hoy mueven buscadores, fotos, asistentes de chat y mucho más, y sin embargo todas están construidas con un pequeño conjunto de principios y componentes. Las leyes de escalado neuronal dicen que hacer crecer a la vez los datos, la potencia de cálculo y el tamaño del modelo casi siempre sube la precisión de forma predecible, así que el coste de una precisión dada sigue reduciéndose por un factor constante; Sutton llamó a esto la lección amarga. Una red se optimiza para aproximar una distribución de probabilidad a partir de muestras, y aunque podría memorizar, los modelos grandes entrenados generalizan sorprendentemente bien. Los modelos fundacionales se usan a menudo solo mediante prompting, como cajas negras, pero personalizar, ajustar finamente, cuantizar o inventar modelos exige entender qué pasa dentro, y ese es el viaje que ofrece el libro.

Preguntas

1. Según las leyes de escalado neuronal, ¿qué ocurre cuando los datos, la potencia de cálculo y el tamaño del modelo se aumentan todos a la vez?
 - a. La precisión casi siempre sube de forma predecible
 - b. La precisión sube solo en las tareas para las que se diseñó el modelo

- c. La precisión deja de mejorar en cuanto el modelo puede memorizar los datos
2. ¿Por qué una red neuronal moderna podría, en principio, no aprender la regla que hay detrás de sus datos?
 - a. Porque nunca ve suficientes ejemplos para ajustarse a ellos
 - b. Porque es tan grande que puede memorizar cada ejemplo y convertirse en una tabla de consulta
 - c. Porque una distribución de probabilidad no puede aproximarse a partir de muestras
 3. ¿Con qué se compara usar un modelo fundacional solo mediante prompting?
 - a. Con escribir sus propias bibliotecas desde cero en un lenguaje como C++
 - b. Con reparar el código fuente de un programa que descargó
 - c. Con depender de software terminado cuyo código fuente no puede ver
 4. ¿Cuál de estas tareas exige mirar bajo el capó de un modelo en vez de simplemente darle un prompt?
 - a. Cuantizarlo para que funcione en hardware de bajo consumo
 - b. Pedirle que reescriba una receta con palabras más sencillas
 - c. Reformular una pregunta para obtener otra respuesta

Respuestas

1. a) Para prácticamente cualquier tarea, hacer crecer a la vez los datos, el cálculo y el tamaño del modelo casi siempre da una mejora predecible de la

precisión; dicho al revés, el cálculo necesario para una precisión dada sigue reduciéndose por un factor constante.

2. b) Las redes modernas son tan grandes que podrían simplemente guardar cada ejemplo que se les muestra; la sorpresa es que los modelos entrenados generalizan en vez de solo memorizar.

3. c) Cuando solo le da prompts a un modelo preentrenado, su interior sigue siendo una caja negra, igual que cuando depende de software cuyo código fuente no le es accesible.

4. a) Ajustar finamente, fusionar, cuantizar, probar la robustez y diseñar variantes nuevas requieren entender cómo el modelo procesa los datos por dentro, mientras que el prompting lo trata como una caja negra.

Ejercicios

1. Explique con sus propias palabras por qué el libro llama a una red neuronal entrenada bajo las leyes de escalado una herramienta casi mágica, y qué podría en principio estropearla.

Pista: Piense en qué intenta aproximar la red y en qué podría hacer en su lugar una red muy grande.

Respuesta: Una red se optimiza para aproximar la distribución de probabilidad que hay detrás de sus datos, y con el escalado su precisión sube de forma predecible. En principio una red enorme podría simplemente memorizar cada ejemplo y convertirse en una tabla de consulta, pero los modelos entrenados generalizan bien en su lugar, incluso a tareas que no estaban explícitamente en los datos de entrenamiento.

2. El libro compara dar prompts a un modelo fundacional con usar software sin su código fuente. Explique qué dice esa comparación sobre quién construye tales modelos y qué puede hacer todavía todo el mundo.

Pista: Pregúntese cuántos equipos pueden construir los modelos más grandes y qué permite el código abierto.

Respuesta: Solo unos pocos equipos tienen el cálculo y la experiencia para publicar modelos realmente grandes, igual que pocas empresas construyen grandes programas empresariales. Pero, como con el código abierto, el hardware corriente y algo de ingenio siguen permitiendo a cualquiera ajustar finamente, fusionar, cuantizar, probar o inventar modelos, siempre que entienda cómo funcionan por dentro.

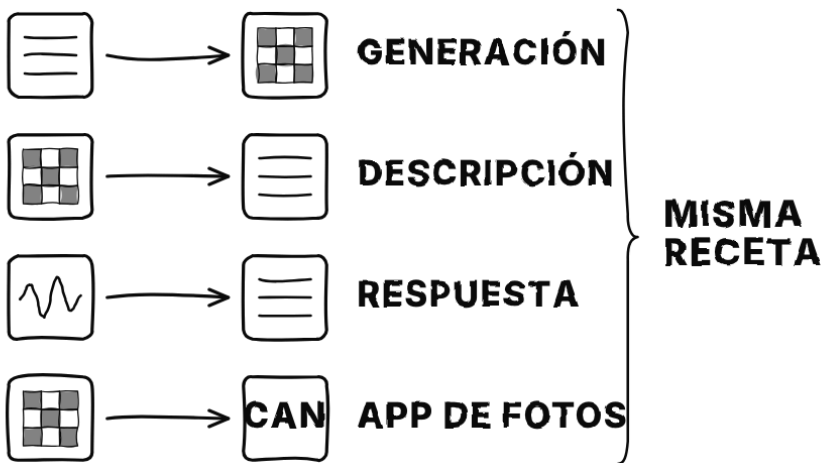
Aprender es buscar

Tome una radio con una vieja rueda de sintonía e intente encontrar una emisora que le guste. Gira un poco la rueda, escucha, y si el sonido se vuelve más claro sigue girando en el mismo sentido; si empeora, gira hacia atrás. Nunca necesita saber cómo funciona la radio por dentro, solo hacia qué lado un pequeño giro mejora las cosas. Guarde esa imagen, porque el libro describe casi toda la IA moderna como esta clase de girar ruedas, hecho con millones de



Girar, escuchar, girar otra vez

El libro da por supuesto que sus lectores conocen lo básico del aprendizaje automático, ML, y más concretamente del aprendizaje supervisado, SL. El aprendizaje supervisado resuelve una tarea reuniendo ejemplos del comportamiento que queremos y luego entrenando, es decir, optimizando, un sistema hasta que imita ese comportamiento. La idea suena sencilla, dice el libro, y es engañosamente poderosa. Generar imágenes se convierte en la tarea de reunir un conjunto muy grande de fotos con sus descripciones. Imitar la lengua inglesa se convierte en la tarea de reunir una gran cantidad de texto y aprender a predecir cada frase a partir de las anteriores. Leer una radiografía se convierte en la tarea de mantener una gran base de datos de imágenes junto con las decisiones de los médicos sobre ellas.



FRASE SIG. · RADIOGRAFÍA → DECISIÓN

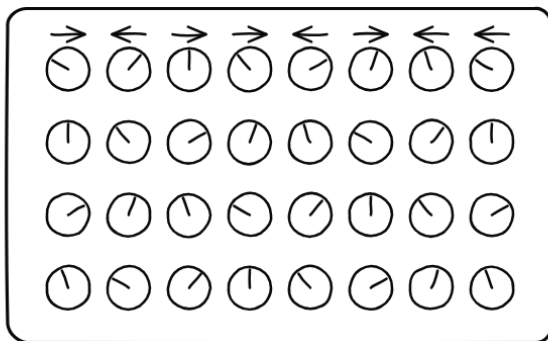
Qué entra, qué sale

La figura del libro ordena las tareas según lo que entra y lo que debe salir. La generación de imágenes quiere una imagen, una rejilla ordenada de píxeles, a partir de un texto, una secuencia de caracteres; describir una imagen es lo contrario, una descripción a partir de una imagen. Responder a una pregunta sobre una grabación quiere un texto a partir de un fragmento de audio, que es de nuevo una secuencia ordenada, esta vez de números. El punto fascinante que señala el libro es que los modelos para todos estos casos se diseñan siguiendo especificaciones muy parecidas. La aplicación de fotos de su teléfono es uno de ellos: entra una imagen, sale una palabra como «perro», aprendida de muchas imágenes que venían con sus etiquetas.

En general, dice el libro, aprender es un problema de búsqueda. Partimos de un programa con un número muy grande de grados de libertad, que se llaman parámetros; piense en ellos como las ruedas de nuestra radio. Luego cambiamos los parámetros hasta que el modelo funciona suficientemente bien. Para que eso sea práctico necesitamos maneras eficientes de buscar el mejor ajuste incluso cuando hay millones, miles de millones o hasta billones de parámetros, demasiados para probarlos uno a uno.

APRENDER = BUSCAR

PROGRAMA



$10^6 \dots 10^{12}$

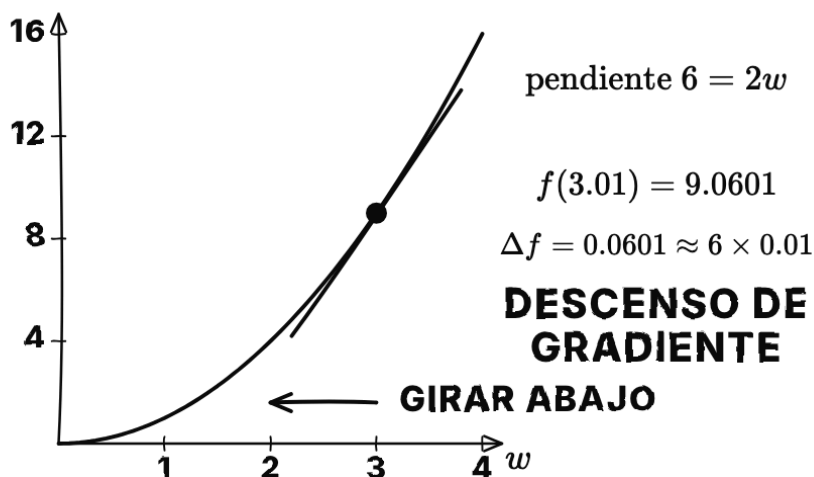
**UNO A
UNO X**

GRADIENTE

PARAMS

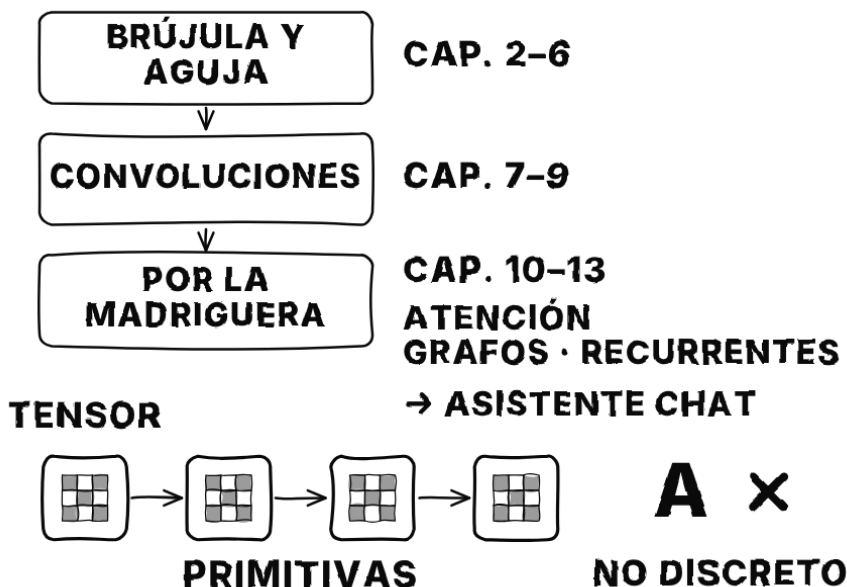
Un programa con muchas ruedas

Los modelos diferenciables lo resuelven permitiendo solo componentes diferenciables, funciones matemáticas cuya derivada podemos calcular. Conocer la derivada de una función de muchas entradas, que se llama gradiente, significa saber qué pasa cuando movemos ligeramente cada parámetro. Aquí está la idea con una sola rueda y números que podemos comprobar. Suponga que el error de un modelo es $f(w) = w^2$, y la rueda está en $w = 3$, así que el error es nueve. La derivada de w^2 es $2w$, que en tres vale seis: dice que un giro diminuto de la rueda cambia el error unas seis veces más que el propio giro. Compruébelo: gire la rueda hasta 3,01 y el error pasa a 9,0601, un cambio de 0,0601, casi exactamente seis veces el giro de 0,01. Así que sabemos, sin probar, que girar la rueda hacia abajo bajará el error, y aproximadamente cuánto.



Pendiente de w al cuadrado en tres

Ese conocimiento lleva a rutinas automáticas de optimización, sobre todo la diferenciación automática, que calcula estas derivadas para un programa entero, y el descenso de gradiente, que las usa para girar un poco cada rueda en la dirección correcta, una y otra vez. Montar todo esto es el tema de la primera parte del libro, llamada «Brújula y aguja», que va del capítulo 2 al capítulo 6.



El mapa del libro

Una vez que una red neuronal se ve como nada más que una composición de bloques diferenciables, llamados primitivas, el libro plantea dos preguntas básicas. Primera: ¿qué tipos de datos puede recibir y producir una red así? Segunda: ¿qué tipos de primitivas podemos usar? La diferenciable es una exigencia fuerte: descarta trabajar directamente con muchos tipos de datos corrientes, como caracteres o números enteros, porque saltan de un valor al siguiente y por tanto son discontinuos. Una letra no puede moverse un poquito hacia otra letra. Lo que los modelos diferenciables manejan con facilidad son grandes matrices de números, que el libro llamará tensores; una foto es una de ellas, una rejilla de valores de brillo, y esas matrices pueden transformarse con álgebra encadenando pasos sencillos lineales y no lineales. La segunda figura del libro muestra

exactamente esto: una red como una cadena de primitivas que trabajan sobre matrices estructuradas, cada primitiva clasificada según lo que recibe y lo que produce, lo que fija las reglas para unirlos.

La segunda parte del libro estudia a fondo un componente diferenciable típico, la convolución, en los capítulos 7 a 9. Las convoluciones pueden usarse siempre que los datos vengan como una secuencia ordenada de elementos: audio, imágenes, texto y vídeo, entre otros. Por el camino el libro presenta técnicas para construir modelos profundos, es decir, modelos hechos de muchos pasos en secuencia, e ideas como dividir el texto en tokens, generar una secuencia pieza a pieza y el modelado causal, que juntas son la base de los grandes modelos de lenguaje de hoy.

La tercera parte, llamada «Por la madriguera del conejo», explora otros diseños: para conjuntos, sobre todo las capas de atención y los modelos transformadores en los capítulos 10 y 11; para grafos en el capítulo 12; y por último las capas recurrentes para secuencias en el tiempo en el capítulo 13. El asistente de chat que usa cada día es un transformador, así que esa parte del viaje lleva directamente a él.

Más allá del libro, el autor promete una página web con capítulos adicionales, más orientados a la investigación, que pueden leerse en cualquier orden, sobre modelado generativo, cómputo condicional, transferencia de aprendizaje y explicabilidad. También hay una serie de sesiones de laboratorio guiadas en forma de cuadernos, que siguen buena parte del libro y continúan con temas avanzados como el aprendizaje contrastivo y la fusión de modelos.

Para recordar

El aprendizaje supervisado significa reunir ejemplos de un comportamiento deseado y optimizar un sistema para que lo imite, y esta receta sencilla cubre por igual la generación de imágenes, el lenguaje y las imágenes médicas. Aprender es una búsqueda, sobre los parámetros de un programa, sus ruedas, de un ajuste que funcione, incluso cuando hay miles de millones de ellos. Los modelos diferenciables hacen práctica la búsqueda: el gradiente dice cómo cada pequeño empujón cambia el error, como las seis veces el giro para doble u al cuadrado en tres, y esto impulsa la diferenciación automática y el descenso de gradiente. Tales modelos trabajan sobre tensores, grandes matrices de números, no sobre cosas discretas como caracteres; las tres partes del libro cubren lo básico, luego las convoluciones, luego la atención, los grafos y las capas recurrentes.

Preguntas

1. ¿Por qué el libro llama al aprendizaje un problema de búsqueda?
 - a. Porque un modelo busca el ejemplo guardado más parecido para cada nueva entrada
 - b. Porque el entrenamiento cambia los muchos parámetros de un programa hasta que funciona suficientemente bien
 - c. Porque el aprendizaje supervisado busca datos etiquetados en la web

2. ¿Qué nos da conocer el gradiente del error de un modelo?
 - a. Cómo cambia el error cuando cada parámetro se mueve ligeramente
 - b. El mejor ajuste posible de todos los parámetros a la vez
 - c. Una lista de los ejemplos de entrenamiento en los que el modelo se equivocó
3. ¿Por qué un modelo diferenciable no puede trabajar directamente con caracteres o números enteros?
 - a. Porque hay demasiados caracteres distintos para guardarlos
 - b. Porque son demasiado pequeños para multiplicarlos por matrices
 - c. Porque saltan de un valor al siguiente, así que no pueden moverse un poquito
4. ¿En qué se diferencian sobre todo la generación de imágenes y la descripción de imágenes?
 - a. Una necesita datos etiquetados y la otra no
 - b. En lo que entra y lo que debe salir: texto a imagen frente a imagen a texto
 - c. Una es aprendizaje supervisado y la otra es un problema de búsqueda

Respuestas

1. **b)** Partimos de un programa con un número muy grande de parámetros, sus ruedas, y los cambiamos hasta que el modelo funciona suficientemente bien; lo difícil es buscar con eficiencia entre miles de millones de ajustes.
2. **a)** El gradiente dice qué pasa cuando cada parámetro se mueve un poco, que es lo que permite a rutinas automáticas como el descenso de gradiente

mejorar el modelo paso a paso.

3. c) Los caracteres y los enteros son discretos y por tanto discontinuos; una letra no puede moverse ligeramente hacia otra, mientras que grandes matrices de números como las imágenes sí pueden.

4. b) El libro ordena las tareas por su entrada y su salida: la generación hace una rejilla de píxeles a partir de un texto, la descripción hace un texto a partir de una rejilla de píxeles, y los modelos siguen diseños parecidos en ambos casos.

Ejemplos resueltos

Una rueda, un empujón El error de un modelo es $f(w) = w^2$, y su único parámetro está en $w = 3$. Use la derivada para predecir cuánto cambia el error si w se sube 0,01, y luego compruébelo calculándolo directamente.

La derivada de w^2 es $2w$, así que en $w = 3$ vale 6: el error cambia unas seis veces más que el empujón.

Cambio predicho: $6 \times 0,01 = 0,06$.

Comprobación directa: $3,01^2 = 9,0601$, mientras que $3^2 = 9$, así que el cambio real es 0,0601.

La predicción coincide casi exactamente, y por eso un gradiente nos dice hacia dónde girar la rueda sin probar cada valor.

Respuesta: Unos 0,06 (el cambio exacto es 0,0601), así que bajar w reduciría el error.

Ejercicios

1. Con el mismo error $f(w) = w^2$, la rueda está ahora en $w = 5$. ¿Hacia qué lado hay que girarla para bajar el error, y cuánto cambia aproximadamente el error con un giro de 0,1?

Pista: Primero halle la derivada $2w$ en $w = 5$.

Respuesta: Girarla hacia abajo. La derivada es $2 \times 5 = 10$, así que un giro de 0,1 cambia el error en unos $10 \times 0,1 = 1$ (exactamente: $5,1^2 - 25 = 1,01$).

2. Elija una tarea de su teléfono, como buscar fotos de «perro» u obtener una respuesta de un asistente de chat, y descríbala como el libro describe las tareas: qué entra, qué sale y qué datos reuniría un enfoque de aprendizaje supervisado.

Pista: Fíjese en los ejemplos de generación de imágenes, lenguaje y radiografías.

Respuesta: La búsqueda de fotos recibe una imagen y da una etiqueta; se reunirían muchas fotos emparejadas con sus etiquetas y se optimizaría un modelo para reproducirlas. Un asistente de chat recibe texto y da texto; se reuniría una gran cantidad de texto y se aprendería a predecir cada fragmento a partir de lo que venía antes.

3. Explique por qué una foto puede darse a un modelo diferenciable, pero una palabra, escrita en letras, no puede dársele directamente.

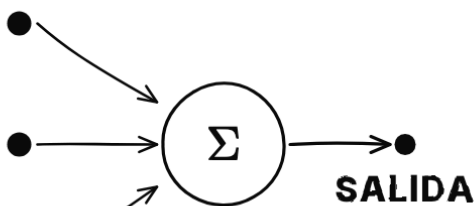
Pista: Pregúntese si cada una puede cambiarse en una cantidad diminuta.

Respuesta: Una foto es una matriz de números, un tensor de valores de brillo, que puede moverse ligeramente y transformarse con pasos lineales y no lineales. Las letras son discretas: no hay nada entre una letra y la siguiente, así que no hay empujón pequeño, y por tanto no hay derivada que seguir.

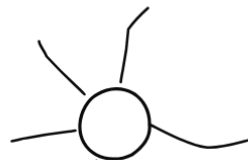
Ideas viejas, poder nuevo

Busque viejas noticias de periódico sobre una «máquina que aprende» y encontrará titulares entusiasmados que suenan como las noticias de esta semana sobre asistentes de chat. La sorpresa es la fecha que llevan. El libro muestra uno de esos momentos en una figura que rotula, con un guiño, «bombo de la IA»: el año es 1958, y el psicólogo estadounidense Frank Rosenblatt atrae una amplia atención de los medios con sus perceptrones, uno de los primeros prototipos funcionales de una red neuronal. Así que el entusiasmo no es nuevo; lo nuevo es lo bien que por fin funcionan estas máquinas.

BOMBO DE LA IA 1958 · PERCEPTRÓN



AHORA FUNCIONA

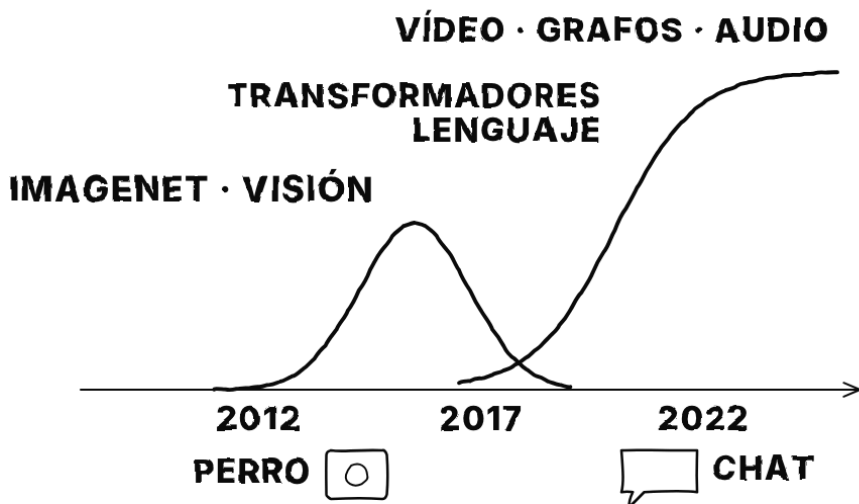


NEURONA

**NEURONAS
CAPAS
ACTIVACIONES**

Las redes neuronales tienen una historia larga y rica. Su propio nombre se remonta a los intentos de principios del siglo veinte de modelar las neuronas biológicas, las células nerviosas del cerebro. El vocabulario se quedó: para encajar con las bibliotecas de software de hoy, el libro seguirá hablando de neuronas, capas y activaciones, aunque la analogía con el cerebro se haya desdibujado.

El interés llegó en varias oleadas. Entre 2012 y 2017 las redes crecieron en complejidad como nunca antes, empujadas por grandes pruebas públicas y competiciones, sobre todo el ImageNet Large Scale Visual Recognition Challenge, ILSVRC, que el libro trata en el capítulo 9. Una segunda gran oleada llegó en 2017 con los transformadores, el tema del capítulo 10. Igual que la visión por computador había sido conquistada por los modelos convolucionales unos años antes, el procesamiento del lenguaje natural fue conquistado por los transformadores en muy poco tiempo. Siguieron más avances para el vídeo, para los grafos, que trata el capítulo 12, y para el audio, lo que llevó al entusiasmo actual por los grandes modelos de lenguaje, las redes multimodales que mezclan texto, imágenes y sonido, y los modelos generativos que producen contenido nuevo. La aplicación de fotos que encuentra a su perro pertenece a la primera oleada; el asistente de chat que le responde pertenece a la segunda.



Dos oleadas

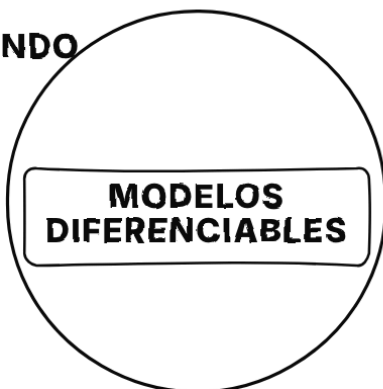
Las palabras cambiaron tan deprisa como los modelos. En los años ochenta se hablaba de conexionismo; más tarde «aprendizaje profundo» pasó a significar las redes modernas, en contraste con los modelos más pequeños y poco profundos del pasado. El libro encuentra todos estos términos irremediabilmente vagos, porque las redes artificiales modernas apenas conservan parecido con las redes biológicas o con la neurología. Lo que las redes modernas tienen de verdad en común es que están construidas con bloques diferenciables, piezas cuyas derivadas pueden calcularse, como vimos con la rueda que podía moverse un poquito. Por esa razón el libro prefiere llamarlas modelos diferenciables siempre que puede.

× **CONEXIONISMO**
(AÑOS 80)

PROGRAMACIÓN
DIFERENCIABLE

× **APRENDIZAJE PROFUNDO**

VAGO



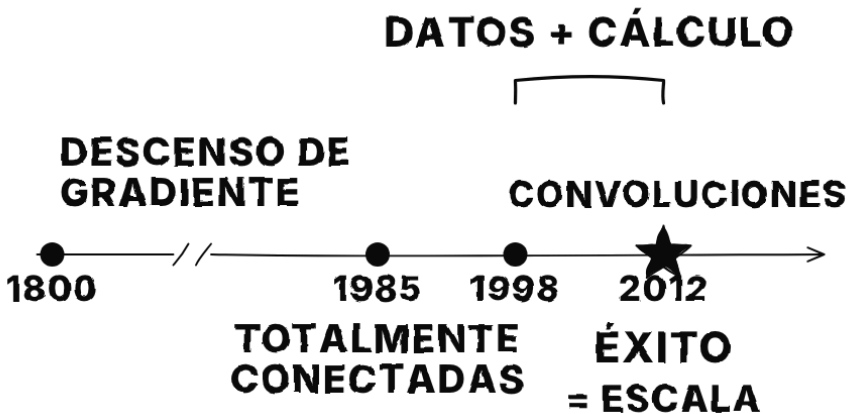
LECUN 2018 · RAFFEL 2021

Nombres, luego un nombre mejor

Esa visión se abre a un campo más amplio llamado programación diferenciable, una disciplina emergente que mezcla informática y optimización para estudiar programas de computador que pueden diferenciarse en general. El libro reconoce dos breves manifiestos que dieron forma a este enfoque: uno de Yann LeCun en 2018, que defiende esta visión más amplia de las redes neuronales, y uno de Colin Raffel en 2021, que pide que los modelos se construyan como se construye el software de código abierto.

Viajar por esta tierra de modelos diferenciables es también viajar por la historia, dice el libro. La optimización numérica de modelos lineales por descenso de gradiente, el tema del capítulo 4, se conocía al menos desde el siglo diecinueve. Las redes totalmente conectadas,

en la forma que el libro usa más adelante, se remontan a los años ochenta. Los modelos convolucionales ya se conocían y se usaban a finales de los años noventa. Ponga esas fechas en una línea y aparece un hueco: un método de los años 1800, otro de los ochenta, otro de los noventa, y sin embargo los grandes éxitos llegaron solo después de 2012.



Ideas viejas, éxito tardío

Lo que faltaba, explica el libro, no eran las ideas sino los medios: hicieron falta muchas décadas antes de que hubiera datos suficientes y potencia de cálculo suficiente para ver lo bien que rinden estos modelos cuando tienen mucho de ambas cosas, y muchos parámetros. Esto conecta con las leyes de escalado de la primera parte: las recetas

viejas, cocinadas a una escala mucho mayor, resultaron extraordinariamente buenas.

El libro es honesto sobre sus límites. No puede profundizar en cada tema, en parte porque la investigación avanza muy deprisa. Su objetivo, en cambio, es dar fundamentos suficientes para que el lector pueda orientarse en los artículos de investigación más recientes sin perderse. Para quienes quieran la historia completa, remite a un panorama de las redes neuronales modernas, a un libro de entrevistas con algunas de las figuras principales del campo y a una larga historia comentada, con opiniones propias, de Jürgen Schmidhuber.

Pruébalo

Escriba tres fechas en una tira de papel: 1800 para el descenso de gradiente en modelos lineales, 1985 para las redes totalmente conectadas, 1998 para los modelos convolucionales. Luego añada 2012, el comienzo de la gran oleada, y 2017, la llegada de los transformadores. Verá que la mayoría de las ideas estaban en la tira mucho antes de los éxitos, que es lo que señala el libro: el cambio vino de los datos y la potencia de cálculo, no de la invención de las ideas.

Para recordar

Las redes neuronales tienen una larga historia, desde los primeros modelos de neuronas biológicas, pasando por el perceptrón de Rosenblatt en 1958, hasta las oleadas de 2012 a 2017, impulsadas por ImageNet, y de los transformadores desde 2017. Nombres como conexionismo y aprendizaje profundo son vagos, porque las redes modernas apenas se parecen al cerebro; lo que comparten es estar construidas con bloques diferenciables, así que el libro las llama modelos diferenciables. Esta visión lleva a la programación diferenciable, una disciplina en el punto de encuentro de la informática y la optimización. El descenso de gradiente, las redes totalmente conectadas y las convoluciones se conocían hace décadas; lo que las hizo brillar fue tener datos, potencia de cálculo y parámetros suficientes.

Preguntas

1. El descenso de gradiente, las redes totalmente conectadas y las convoluciones se conocían mucho antes de 2012. ¿Por qué sus grandes éxitos llegaron solo después?
 - a. Porque los métodos viejos tenían errores que se corrigieron después de 2012
 - b. Porque hicieron falta décadas para tener datos, potencia de cálculo y parámetros suficientes para mostrar lo bien que funcionan

- c. Porque por fin se entendió el cerebro lo bastante bien para copiarlo
2. ¿Por qué el libro prefiere el nombre «modelos diferenciables» a «redes neuronales» o «aprendizaje profundo»?
- a. Porque las redes modernas apenas se parecen al cerebro, y lo que comparten es estar construidas con bloques diferenciables
 - b. Porque los nombres antiguos son marcas registradas de bibliotecas de software
 - c. Porque los modelos diferenciables son una clase de red distinta y más nueva
3. ¿Qué papel tuvieron los transformadores en la segunda oleada, comparados con los modelos convolucionales en la primera?
- a. Los transformadores sustituyeron a las convoluciones en el reconocimiento de imágenes durante el desafío ImageNet
 - b. Los transformadores eran una idea más antigua que los modelos convolucionales superaron después
 - c. Los transformadores conquistaron el procesamiento del lenguaje en poco tiempo, como los modelos convolucionales habían conquistado la visión por computador

Respuestas

1. **b)** Las ideas ya estaban ahí; lo que faltaba eran los medios, datos y potencia de cálculo suficientes, y muchos parámetros, para revelar lo bien que rinden estos modelos.
2. **a)** Los nombres antiguos son vagos: las redes artificiales modernas apenas conservan parecido con las redes biológicas, y su rasgo esencial es

estar compuestas de bloques diferenciables.

3. c) Igual que la visión por computador fue conquistada por los modelos convolucionales unos años antes, el procesamiento del lenguaje natural fue conquistado por los transformadores en muy poco tiempo después de 2017.

Ejercicios

1. Explique con sus propias palabras por qué el libro muestra un momento de periódico de 1958 sobre el perceptrón de Rosenblatt bajo el rótulo «bombo de la IA».

Pista: Compare el entusiasmo de entonces con el de ahora, y pregúntese qué ha cambiado realmente.

Respuesta: El perceptrón de Rosenblatt, una de las primeras redes neuronales que funcionaron, ya atrajo una amplia atención de los medios en 1958. El rótulo muestra que el entusiasmo por las máquinas que aprenden es viejo; lo nuevo hoy es lo bien que por fin funcionan esas máquinas, gracias a los datos y la potencia de cálculo.

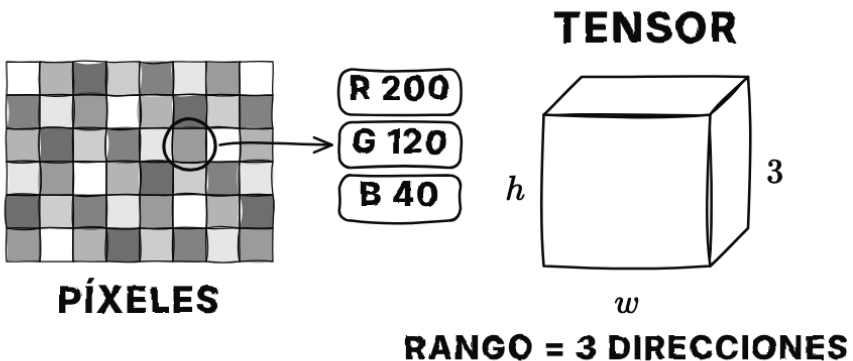
2. Coloque estos elementos en una línea de tiempo y diga qué enseña el hueco entre ellos: el descenso de gradiente para modelos lineales, las redes totalmente conectadas, los modelos convolucionales, la oleada de 2012 a 2017, los transformadores.

Pista: Use las fechas que da el libro: el siglo diecinueve, los años ochenta, finales de los noventa, 2012, 2017.

Respuesta: El descenso de gradiente en modelos lineales data al menos del siglo diecinueve, las redes totalmente conectadas de los años ochenta, los modelos convolucionales de finales de los noventa, la gran oleada de 2012 a 2017, los transformadores de 2017. La mayoría de las ideas existían mucho antes de los éxitos, así que el cambio vino de tener datos, potencia de cálculo y parámetros suficientes.

Cajas de números y sus formas

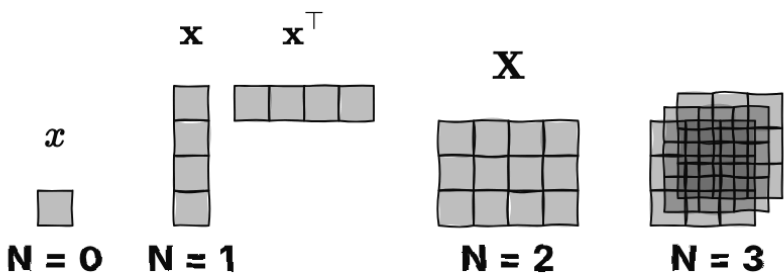
Abra cualquier foto en su teléfono y amplíela hasta ver los cuadraditos. Cada cuadradito es un píxel, y cada píxel se guarda como tres números: cuánto rojo, cuánto verde y cuánto azul. Así que para el computador la foto no es una imagen en absoluto; es una caja de números, tantos cuadraditos de alto, tantos de ancho y tres de fondo. Pregúntese cómo es esa caja para un álbum entero, y tendrá la primera idea de esta parte.



Una foto es una caja de números

El tipo de dato básico de los modelos diferenciables es el tensor, que el libro define como una matriz n -dimensional de objetos, normalmente números reales. El libro admite que el nombre es un pequeño abuso: en la literatura científica un tensor es algo más preciso, un operador multilineal, mientras que aquí significa simplemente una matriz multidimensional; el nombre está tan extendido que el libro lo conserva. Pidiendo perdón a los matemáticos, llama a n el rango del tensor, es decir, el número de direcciones en las que se extiende la caja.

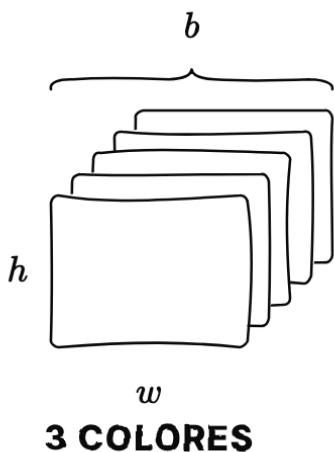
La notación cambia con el rango. Un tensor con $n = 0$ contiene un solo valor, un escalar, escrito con una letra minúscula como x o y . Los escalares se llaman así porque cualquiera de ellos puede escribirse como un múltiplo, un escalado, del número uno. Un tensor con $n = 1$ es una columna de valores, un vector, escrito en minúscula negrita, $\mathbf{x}$. Los mismos valores dispuestos como fila se escriben $\mathbf{x}^T$, donde la pequeña T significa que la columna se ha tumbado de lado; cuando el contexto lo deja claro, el libro omite la T para facilitar la lectura. La figura del libro dibuja los vectores como filas solo para ahorrar espacio; en el texto todo vector es una columna. Un tensor con $n = 2$ es un rectángulo de valores, una matriz, escrita en mayúscula negrita, $\mathbf{X}$ o $\mathbf{Y}$. Para rangos por encima de dos no hay letras especiales, y las letras caligráficas rizadas como $\mathcal{X}$ se reservan para conjuntos y distribuciones de probabilidad.



$\mathcal{X}$: conjuntos

Rango de cero a tres

La mayoría de las veces, dice el libro, entender un algoritmo o una operación se reduce a entender la forma de cada tensor implicado. Para escribir las formas de manera compacta usa el signo que parece una pequeña onda: $\mathbf{X} \sim (b, h, w, 3)$. En palabras, $\mathbf{X}$ es un tensor de rango cuatro cuyos cuatro tamaños son b , h , w y 3 . De dónde sale es de nuestro álbum de fotos: b es el número de fotos del lote, h la altura y w la anchura en píxeles, y 3 los tres canales de color. Algunos tamaños son números fijos, como el 3 , y otros son letras que pueden tomar cualquier valor.



$$X \sim (b, h, w, 3)$$

$$b = 32, h = w = 224$$

$$224 \times 224 \times 3 = 150,528$$

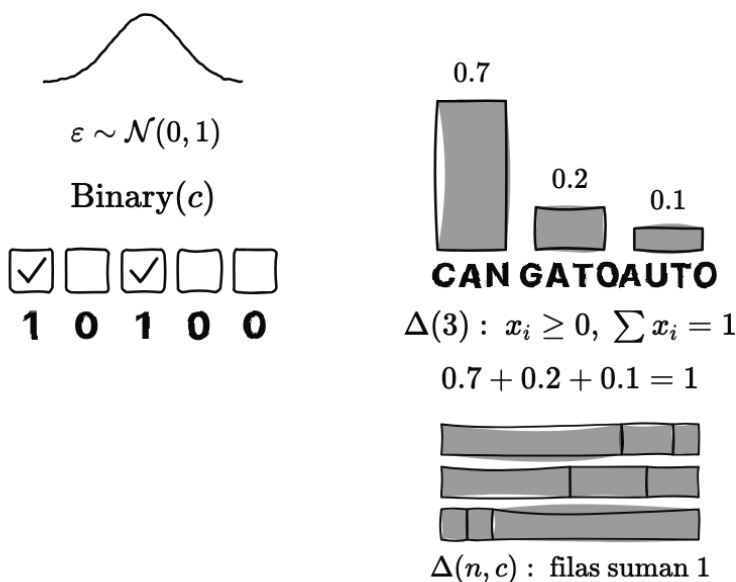
$$\times 32 = 4,816,896$$

$$\approx \mathbf{5 \text{ MILLONES}}$$

Un lote de fotos

Contemos en voz alta. Tome un lote de 32 fotos, cada una de 224 píxeles de alto y 224 de ancho. Una foto contiene 224 por 224, que son 50 176 píxeles, por 3 colores, lo que da 150 528 números. El lote entero contiene 32 veces eso: 4 816 896 números en un solo tensor de rango cuatro, casi cinco millones. Por eso importan tanto las formas: le dicen de un vistazo cuánto está fluyendo por un modelo.

El mismo signo de onda también significa extraer un valor al azar de una distribución de probabilidad, como en $\varepsilon \sim \mathcal{N}(0, 1)$, un número extraído de la curva de campana centrada en cero. El libro lo usa así solo rara vez, y el contexto siempre deja claro el significado. Así, $\mathbf{x} \sim (d)$ sustituye al más habitual $\mathbf{x} \in \mathbb{R}^d$, un vector de d números reales, y $\mathbf{X} \sim (n, d)$ sustituye a $\mathbf{X} \in \mathbb{R}^{n \times d}$, una matriz de n filas y d columnas.



Reglas para los valores

A veces los valores de un tensor deben obedecer una regla, y también hay notación para eso. $\mathbf{x} \sim \text{Binary}(c)$ es un tensor de tamaño c cuyos elementos son solo ceros y unos, por ejemplo una fila de casillas que dice cuáles de c etiquetas tiene una foto. $\mathbf{x} \sim \Delta(a)$ es un vector en el llamado simplex: cada elemento es al menos cero, $x_i \geq 0$, y todos suman uno, $\sum_i x_i = 1$. Eso es exactamente lo que parece una lista de probabilidades. Si su aplicación de fotos dice perro 0,7, gato 0,2, coche 0,1, los tres son al menos cero y 0,7 más 0,2 más 0,1 es 1, así que este vector está en el simplex de tamaño tres. Para rangos mayores, como en $\mathbf{X} \sim \Delta(n, c)$, la regla se aplica a lo largo de la última dimensión: cada una de las n filas es una lista de c probabilidades que suman uno, una fila por foto.

Más notación se introduce capítulo a capítulo según hace falta.

Pruébalo

Tome una hoja de papel cuadriculado y coloree un dibujo diminuto de tres cuadros de alto y cuatro de ancho. Ahora escriba, para cada cuadro, tres números para su rojo, verde y azul; tiene un tensor de forma tres por cuatro por tres, que contiene 36 números. Apile dos hojas así y la forma pasa a ser dos por tres por cuatro por tres, con 72 números: acaba de construir un pequeño lote, la misma clase de objeto que la X del libro.

Para recordar

Un tensor es una matriz multidimensional de números, y su rango es el número de direcciones en que se extiende: un escalar tiene rango cero, un vector rango uno, una matriz rango dos. Los escalares van en minúscula, los vectores en minúscula negrita y siempre como columnas, las matrices en mayúscula negrita; las letras rizadas se reservan para conjuntos y distribuciones. El signo de onda da la forma de un tensor, como en $X \sim (b, h, w, 3)$ para un lote de fotos en color, y un lote de 32 fotos de 224 por 224 píxeles contiene casi cinco millones de números. $\text{Binary}(c)$ significa solo ceros y unos, y $\Delta(a)$, el simplex, significa valores no negativos que suman uno, la forma de una lista de probabilidades.

Preguntas

1. Un lote de fotos en color se escribe $X \sim (b, h, w, 3)$. ¿Cuál es su rango, y por qué el último tamaño es un número fijo?
 - a. Rango tres, porque el tamaño del lote b no cuenta como dimensión
 - b. Rango cuatro, y el 3 es fijo porque cada píxel tiene tres valores de color
 - c. Rango cuatro, y el 3 es fijo porque cada lote contiene tres fotos
2. ¿Cuál de estos vectores está en el simplex $\Delta(3)$?
 - a. (0,5, 0,5, 0,5)
 - b. (0,9, 0,3, -0,2)
 - c. (0,6, 0,3, 0,1)
3. ¿Por qué dice el libro que entender una operación se reduce sobre todo a entender las formas?
 - a. Porque la forma de cada tensor dice de un vistazo qué entra, qué sale y cuánto fluye
 - b. Porque los valores dentro de los tensores nunca importan para cómo funciona un modelo
 - c. Porque las formas son lo único que un computador puede guardar
4. En la notación del libro, ¿cómo se distinguen en la página un vector y una matriz?

- a. Un vector va en mayúscula negrita y una matriz en minúscula negrita
- b. Un vector va en minúscula negrita, una matriz en mayúscula negrita
- c. Ambos se escriben con letras caligráficas rizadas

Respuestas

1. **b)** La forma lista cuatro tamaños, así que el rango es cuatro; b , h y w pueden variar, mientras que cada píxel siempre tiene rojo, verde y azul, por eso el último tamaño es 3.
2. **c)** Un vector del simplex tiene cada elemento al menos cero y elementos que suman uno; $0,6 + 0,3 + 0,1 = 1$, mientras que el primero suma 1,5 y el segundo tiene un elemento negativo.
3. **a)** Seguir la forma de cada tensor muestra qué recibe y qué da una operación, y lo grande que es, como muestran los casi cinco millones de números de un lote de 32 fotos.
4. **b)** Los vectores usan minúscula negrita, como x en negrita, las matrices mayúscula negrita, como X mayúscula en negrita, mientras que las letras rizadas se reservan para conjuntos y distribuciones de probabilidad.

Ejemplos resueltos

Cuántos números hay en un lote Un lote de fotos tiene forma $(b, h, w, 3)$ con $b = 32$, $h = 224$ y $w = 224$. ¿Cuántos números contiene el tensor?

Una foto tiene $h \times w = 224 \times 224 = 50\,176$ píxeles.

Cada píxel guarda 3 valores de color, así que una foto contiene $50\,176 \times 3 = 150\,528$ números.

El lote apila 32 fotos, así que contiene $32 \times 150\,528$ números.

$$32 \times 150\,528 = 4\,816\,896.$$

Respuesta: 4 816 896 números, casi cinco millones

Ejercicios

1. Un lote de 10 fotos, cada una de 100 píxeles de alto y 50 de ancho, en color. Escriba su forma en la notación del libro, dé su rango y cuente sus números.

Pista: La forma es (b, h, w, 3); multiplique los cuatro tamaños.

Respuesta: $X \sim (10, 100, 50, 3)$, rango cuatro. Contiene $10 \times 100 \times 50 \times 3 = 150\,000$ números.

2. Un clasificador da, para cada una de 2 fotos, probabilidades de 3 clases: (0,5, 0,3, 0,2) y (0,1, 0,1, 0,8). Escriba la forma de este resultado y compruebe que pertenece a $\Delta(2, 3)$.

Pista: La regla del simplex se aplica a lo largo de la última dimensión, fila por fila.

Respuesta: El resultado tiene forma (2, 3). Cada fila tiene elementos al menos cero y suma uno: $0,5 + 0,3 + 0,2 = 1$ y $0,1 + 0,1 + 0,8 = 1$, así que pertenece a $\Delta(2, 3)$.

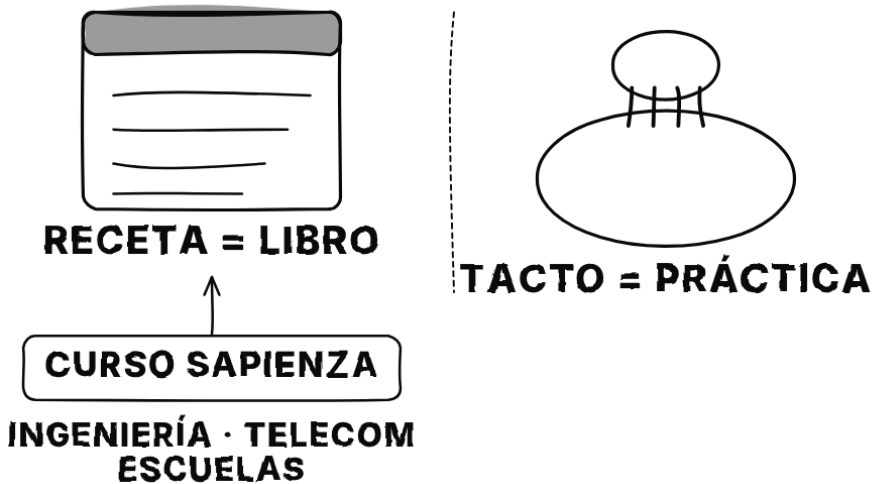
3. Reescriba $X \in \mathbb{R}^{n \times d}$ y $x \in \mathbb{R}^d$ en la notación de formas del libro, y diga qué es cada objeto.

Pista: El signo de onda sustituye a «pertenece a R elevado a ...».

Respuesta: $X \sim (n, d)$, una matriz de n filas y d columnas de números reales; $x \sim (d)$, un vector columna de d números reales.

Qué es este libro, y qué no es

Lea una receta de pan, luego intente hornearlo, y aprenderá dos cosas distintas. La receta le dice qué pasa con la harina, el agua y la levadura; sus manos aprenden lo húmeda que debe sentirse la masa, algo que ninguna receta puede escribir. El libro traza la misma línea para las redes neuronales: le da la receta completa, y es honesto en que el tacto solo llega con la práctica.



La receta y el tacto

El libro nació de unas clases. El autor ha impartido durante muchos años un curso llamado «Redes neuronales para aplicaciones de

ciencia de datos» en el máster de Ciencia de Datos de la Universidad Sapienza de Roma, y quería dar a esas clases una forma escrita coherente. Los capítulos centrales forman la parte principal de ese curso; los capítulos restantes cubren temas impartidos algunos años y otros no. Algunas partes también se apoyan en otra docencia: un curso de redes neuronales para ingeniería informática, una introducción al aprendizaje automático para ingeniería de telecomunicaciones y diversos tutoriales, cursos de doctorado y escuelas de verano.

Ya existen varios libros excelentes y recientes sobre las redes neuronales profundas modernas, y el libro lo dice abiertamente. Al principio cubre en gran medida el mismo terreno que ellos; más adelante la manera de explicar, algunas partes extra y unas pocas secciones de los capítulos avanzados se solapan menos, siguiendo los intereses de investigación del propio autor. Su esperanza declarada es ofrecer un punto de vista adicional y complementario sobre material que existe en otros sitios.

TEORÍA + CÓDIGO

DISEÑO $\rightleftharpoons$ CÓDIGO



NO ES MANUAL ×

Las bibliotecas tras el renacimiento

El nombre «programas diferenciables» se eligió a propósito: entenderlos viene tanto de la teoría como de la programación. Cómo se diseña un modelo y cómo se implementa se influyen mutuamente sin parar, y la diferenciación automática, que calcula gradientes para programas enteros, es el ejemplo más claro. El libro atribuye buena parte del renacimiento de las redes neuronales, más o menos desde 2012, a potentes bibliotecas de software: primero Theano, luego Caffe y Chainer, y después las actuales TensorFlow, PyTorch y JAX, entre otras. Siempre que puede, conecta sus explicaciones con ideas de estas bibliotecas, sobre todo PyTorch y JAX. Pero no es un manual de programación, y para una introducción completa a cada biblioteca remite a los lectores a la documentación de las propias bibliotecas.

Antes de ponerse en marcha, el libro enumera lo que no es. Primero, no intenta ser completo. Elige ideas que son a la vez comunes hoy y lo bastante generales para seguir siendo útiles un tiempo, pero no puede ver el futuro, y algunas partes pueden estar incompletas o desfasadas cuando usted las lea. En un campo donde el asistente de chat de su teléfono puede ser sustituido por una versión nueva cada pocos meses, esa advertencia es simplemente realista.

Segundo, para cada idea muestra unas pocas variaciones de la literatura de investigación, por ejemplo pasar de la normalización por lotes a la normalización por capas, dos maneras de reescalar los números dentro de una red. Existen cientos de variaciones más, y el libro invita a los lectores a explorarlas en la página web *Papers With Code*.

Tercero, y lo más importante, es un libro sobre los componentes fundamentales de los modelos diferenciables. Construirlos a gran escala y hacer que de verdad funcionen necesita además habilidad de ingeniería y cierta dosis de intuición. El libro dice poco sobre hardware, y para la intuición, dice, nada supera a la experiencia y a las entradas de blog con opiniones propias, señalando en particular la receta de Andrej Karpathy para entrenar redes neuronales y su serie de vídeos «Zero to Hero». Ahí está otra vez nuestro pan: el libro es la receta, y el tacto de la masa se aprende horneando.

Pruébalo

Abra la página de documentación de PyTorch o de JAX y mire cuántas funciones distintas enumera. Luego compárelo con las pocas ideas que este curso ha nombrado hasta ahora: tensores, gradientes, bloques diferenciables. Debería encontrar que las bibliotecas son enormes, mientras que las ideas centrales son pocas; el libro enseña las pocas, y la documentación cubre las muchas.

Para recordar

El libro convierte las clases de redes neuronales que el autor imparte desde hace años en la Universidad Sapienza de Roma en un texto coherente, con capítulos centrales y temas extra ocasionales. Se solapa con otros buenos libros al principio y luego sigue su propio camino, ofreciendo un punto de vista complementario. Teoría y código van juntos: el renacimiento de las redes neuronales desde alrededor de 2012 debe mucho a bibliotecas desde Theano hasta PyTorch y JAX, a las que el libro se refiere, sin ser un manual de programación. No es completo, muestra solo unas pocas de las muchas variantes y enseña los componentes fundamentales, mientras que construir a escala necesita además ingeniería e intuición que vienen de la experiencia.

Preguntas

1. ¿Por qué dice el libro que entender los programas diferenciables necesita tanto teoría como programación?
 - a. Porque la teoría es demasiado difícil de seguir sin ejecutar código
 - b. Porque cómo se diseñan los modelos y cómo se implementan se moldean mutuamente sin parar, como muestra la diferenciación automática
 - c. Porque cada capítulo está escrito como un manual de programación
2. ¿Qué dice el libro que deja sobre todo a la experiencia y a las entradas de blog?
 - a. La intuición y la ingeniería necesarias para construir modelos a gran escala y hacer que funcionen
 - b. Las definiciones de tensores y gradientes
 - c. La historia de las redes neuronales desde 1958
3. ¿Por qué advierte el libro de que algunas partes pueden estar desfasadas cuando usted las lea?
 - a. Porque sus ejemplos se escribieron para una versión antigua de PyTorch
 - b. Porque el curso del que procede ya no se imparte
 - c. Porque elige ideas comunes hoy y generales por un tiempo, pero el campo avanza deprisa y no pretende ser completo

Respuestas

1. **b)** Diseño e implementación interactúan constantemente, y la diferenciación automática, que calcula gradientes para programas enteros, es el ejemplo más claro; aun así, el libro no es un manual de programación.
2. **a)** Enseña los componentes fundamentales, pero construirlos a escala necesita habilidad de ingeniería e intuición, y para la intuición nada supera a la experiencia y a las entradas de blog con opiniones propias.
3. **c)** Elige ideas a la vez comunes ahora y lo bastante generales para durar, pero no puede prever el futuro y no aspira a ser completo.

Ejercicios

1. Explique con sus propias palabras la diferencia que traza el libro entre entender los componentes fundamentales de una red y hacer que una red grande funcione de verdad.

Pista: Piense en el pan: la receta y el tacto de la masa.

Respuesta: El libro explica los componentes por completo, como una receta. Construir y entrenar a gran escala necesita además habilidad de ingeniería, conocimiento del hardware y una intuición que viene de la experiencia, para la cual el libro remite a la práctica y a guías con opiniones propias como la de Karpathy.

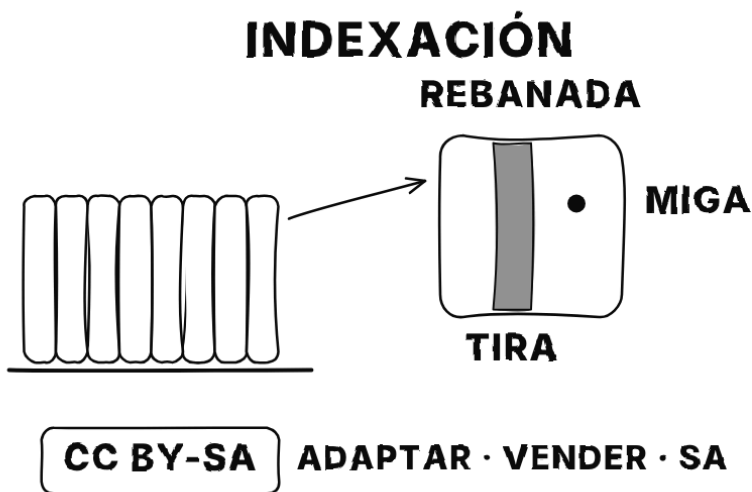
2. ¿Por qué un lector que ya tiene otro libro de aprendizaje profundo podría encontrar útil este, según el propio libro?

Pista: Fíjese en dónde se solapa el libro con otros y dónde no.

Respuesta: Al principio cubre en gran medida el mismo terreno, pero su manera de explicar, algunas partes extra y unas pocas secciones avanzadas siguen los intereses de investigación del autor, así que ofrece un punto de vista adicional y complementario sobre el material.

Cortar rebanadas de un tensor

Saque de su bolsa una barra de pan ya rebanado. Puede sacar una rebanada entera, puede cortar una tira del centro de esa rebanada, o puede pinzar una sola miga en un punto preciso. Cada una de estas es una manera de escoger una parte de la barra diciendo dónde está, y eso es exactamente lo que significa indexar un tensor. Antes de empezar a cortar, el libro cierra su introducción con agradecimientos y una licencia.



Rebanada, tira, miga

El libro agradece a quienes ayudaron a hacerlo: el coloreado de sus ecuaciones viene de un paquete de LaTeX de ST John; sus imágenes en color de Alicia y los símbolos de los márgenes están licenciados de una biblioteca de imágenes de archivo; y las imágenes de Alicia dentro de sus figuras reproducen las ilustraciones originales de John Tenniel, a través de Wikimedia. Agradece a Roberto Alma sus comentarios sobre un borrador anterior y su ánimo para publicar, y a Corrado Zoccolo, Emanuele Rodolà, Marcin Ślaby, Konstantin Burlachenko y Diego Sandoval sus extensas correcciones, junto con todos los que escribieron por correo electrónico.

El libro se publica bajo la licencia Creative Commons CC BY-SA. En palabras de la propia licencia, permite a otros distribuir, remezclar, adaptar y construir sobre el material en cualquier medio o formato, siempre que se dé crédito al creador; permite el uso comercial; y cualquier cosa construida a partir de él debe publicarse bajo los mismos términos. El curso que está escuchando es una adaptación de ese tipo, y lleva la misma licencia.

El capítulo 2 reúne las matemáticas que necesita el resto del libro. Da por supuesto que usted ya ha visto estos temas antes y se concentra en la notación y en una visión de conjunto coherente, enlazando ideas como los tensores con cómo se implementan en la práctica. Tiene tres partes, cada una construida sobre la anterior: álgebra lineal, luego gradientes para objetos con muchas dimensiones, luego cómo optimizar funciones usando esos gradientes. Un apéndice aparte da un resumen autocontenido de la teoría de la probabilidad, centrado en el principio de máxima verosimilitud. El libro advierte de que el capítulo está cargado de definiciones, y pide al lector que tenga paciencia.

**ÁLGEBRA
LINEAL**

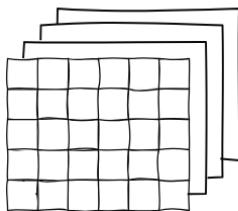
GRADIENTES

OPTIMIZACIÓN

**APÉNDICE:
PROBABILIDAD**

4 FOTOS

**5 DE
ALTO**



6 DE ANCHO

$$X \sim (s_1, s_2, \dots, s_n)$$

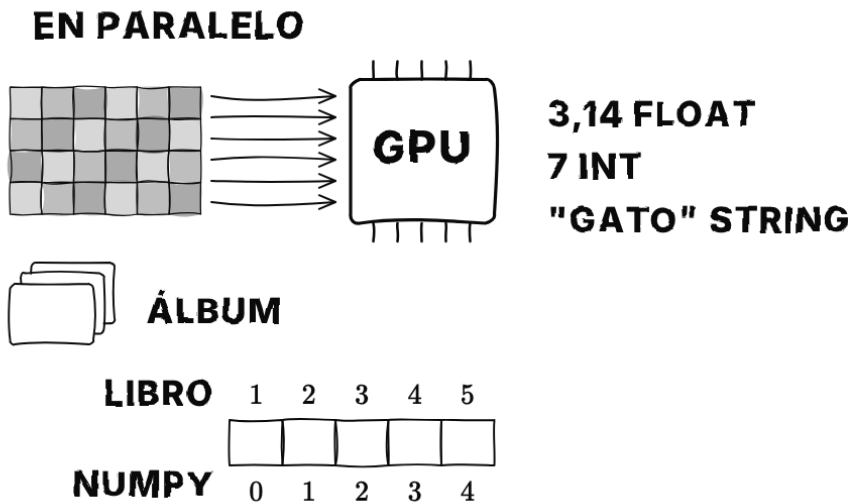
$$X \sim (4, 5, 6) : 4 \times 5 \times 6 = 120$$

RANGO 3

El capítulo 2 y la forma de un tensor

La primera parte recuerda el álgebra lineal, en parte para acordar una notación común, y empieza por el tensor. La definición: un tensor X es una matriz n -dimensional de elementos todos del mismo tipo, y su forma se escribe $X \sim (s_1, s_2, \dots, s_n)$. En palabras, la lista entre paréntesis da el tamaño de cada una de las n dimensiones. Por ejemplo, un lote de 4 fotos pequeñas en gris, cada una de 5 píxeles de alto y 6 de ancho, tiene forma $X \sim (4, 5, 6)$, y contiene 4 por 5 por 6, que son 120 números. Con n igual a cero obtenemos escalares, con uno vectores, con dos matrices, y por encima matrices de más dimensiones, manteniendo las letras que ya acordamos: minúscula para escalares, minúscula negrita para vectores, mayúscula negrita para matrices.

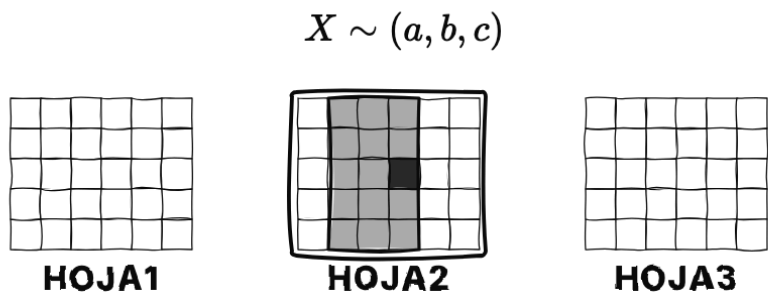
¿Por qué son los tensores tan centrales en el aprendizaje profundo? Porque se adaptan al cálculo masivamente paralelo: los procesadores gráficos, GPU, y chips más especializados como las TPU y las IPU pueden trabajar sobre todos los números de un tensor al mismo tiempo. Por eso el teléfono de su bolsillo puede etiquetar rápido un álbum entero de fotos: las trata como un gran tensor en vez de una foto tras otra.



Todos los números a la vez

Un tensor se describe con dos cosas: el tipo de sus elementos y su forma. El libro usa sobre todo tensores de números en coma flotante, números con decimales, cuyo formato exacto discute más adelante; pero los tensores también pueden contener enteros, por ejemplo números de clase en clasificación, o cadenas de texto.

Los tensores pueden indexarse para tomar rebanadas, es decir, subconjuntos de sus valores, y se aplican la mayoría de las convenciones de la biblioteca de Python NumPy. Una diferencia: para facilitar la lectura, el libro cuenta las posiciones desde 1, no desde 0 como hace NumPy. Para expresiones sencillas escribe los índices como pequeños subíndices. Tome un tensor tridimensional $X \sim (a, b, c)$. Entonces X_i es una rebanada de forma (b, c) : en nuestro lote de fotos, X_2 es la segunda foto entera, una rejilla de 5 por 6 con 30 números. Y $X_{i j k}$ es un solo escalar: X_{234} es el brillo del píxel en la fila 3, columna 4 de la segunda foto, una miga de la barra.



$$X_2 : 5 \times 6 = 30$$

$$X_{234} : \text{fila 3, columna 4}$$

$$X_{2,:;2:5} \rightarrow 5 \times 3 \qquad [X]_{ijk}$$

Cortando el lote de fotos

Para selecciones más complejas el libro separa los índices con comas, como en $X_{i,:;j:k}$. Los dos puntos solos significan «tomar todo a lo

largo de esta dimensión», y j dos puntos k significa «un rango de posiciones», así que el resultado tiene forma $(b, k - j)$. Con nuestras fotos, $X_{2,:;2:5}$ toma la segunda foto, las 5 filas y una banda de columnas; su forma es 5 por 5 menos 2, es decir, 5 por 3, una tira de 15 números. Cuando una expresión quedaría recargada, el libro escribe la indexación entre corchetes gris claro, $[X]_{i \ j \ k}$, para que los índices se distingan del resto, y lo que va dentro de los corchetes puede ser a su vez una expresión entera.

Pruébalo

Dibuje en papel una rejilla de 5 filas y 6 columnas y escriba un número en cada casilla; eso es una foto, y dos hojas más como esa forman un lote de forma 3 por 5 por 6. Señale la segunda hoja: eso es equis sub dos, 30 números. Señale en ella la fila 3, columna 4: eso es equis sub dos tres cuatro, un número. Luego sombree todas las filas de las columnas 2, 3 y 4 de esa hoja: ha cortado una tira de 5 por 3, la clase de rebanada que se escribe con dos puntos.

Para recordar

El libro agradece a sus colaboradores y se publica bajo CC BY-SA, que permite adaptar y vender siempre que se dé crédito al autor y el resultado conserve la misma licencia. El capítulo 2 reúne las matemáticas que harán falta después: álgebra lineal, gradientes en muchas dimensiones y optimización con gradientes, con la probabilidad en un apéndice. Un tensor es una matriz n -dimensional de elementos de un mismo tipo, con forma $X \sim (s_1, \dots, s_n)$; los tensores se adaptan a hardware paralelo como las GPU. La indexación sigue a NumPy pero cuenta desde 1: X_i es una rebanada, $X_{i j k}$ un solo número, los dos puntos escogen dimensiones enteras o rangos, y los corchetes grises separan los índices del resto.

Preguntas

1. Para un tensor $X \sim (a, b, c)$, ¿cuál es la diferencia entre X_i y $X_{i j k}$?
 - a. X_i es una rebanada entera de forma (b, c) , mientras que $X_{i j k}$ es un solo número
 - b. X_i es un solo número, mientras que $X_{i j k}$ es una rebanada de forma (a, b, c)
 - c. Ambos son rebanadas, pero $X_{i j k}$ tiene el triple de números
2. ¿Por qué los tensores se adaptan tan bien al aprendizaje profundo?

- a. Porque guardan los números con más precisión que otros formatos
 - b. Porque solo pueden contener números en coma flotante
 - c. Porque hardware como las GPU puede trabajar sobre todos sus números en paralelo
3. ¿Qué exige la licencia CC BY-SA del libro a una adaptación como este curso?
- a. Que nunca se venda
 - b. Que dé crédito al creador y se publique bajo la misma licencia
 - c. Que conserve el texto original sin cambios
4. ¿En qué se diferencia la indexación del libro de la de NumPy?
- a. Cuenta las posiciones desde 1 en vez de desde 0
 - b. Prohíbe los dos puntos para los rangos
 - c. Siempre escribe los índices entre corchetes grises

Respuestas

- 1. a)** Fijar un índice deja las otras dos dimensiones, una rebanada de forma (b, c); fijar los tres escoge un solo escalar, como el brillo de un píxel de una foto.
- 2. c)** Los tensores se adaptan a implementaciones masivamente paralelas en GPU y chips más especializados, que procesan muchos números al mismo tiempo.
- 3. b)** La licencia permite distribuir, remezclar, adaptar y el uso comercial, siempre que se dé crédito al creador y el material adaptado lleve términos idénticos.

4. a) Se aplican la mayoría de las convenciones de NumPy, pero para facilitar la lectura el libro indexa desde 1, no desde 0; los corchetes grises se usan solo cuando una expresión quedaría recargada.

Ejemplos resueltos

Formas de las rebanadas Un lote de fotos en gris tiene forma $X \sim (4, 5, 6)$. Dé la forma de X_2 , la forma de $X_{2,:,2:5}$ (que tiene forma $(b, k - j)$), y diga qué es X_{234} .

Aquí $a = 4$ fotos, $b = 5$ filas, $c = 6$ columnas.

X_2 fija el primer índice, dejando la forma $(b, c) = (5, 6)$: la segunda foto entera, 30 números.

$X_{2,:,2:5}$ fija la foto, conserva las $b = 5$ filas y toma las columnas de $j = 2$ a $k = 5$, así que su forma es $(b, k - j) = (5, 3)$, 15 números.

X_{234} fija los tres índices, así que es un solo número: fila 3, columna 4 de la segunda foto.

Respuesta: X_2 tiene forma $(5, 6)$; $X_{2,:,2:5}$ tiene forma $(5, 3)$; X_{234} es un solo escalar.

Ejercicios

1. Un tensor tiene forma $(10, 8, 12)$. ¿Cuántos números contiene, y cuántos contiene una rebanada X_i ?

Pista: Multiplique todos los tamaños para el tensor entero; quite el primero para una rebanada.

Respuesta: Contiene $10 \times 8 \times 12 = 960$ números; una rebanada X_i tiene forma $(8, 12)$ y contiene 96.

2. Para $X \sim (10, 8, 12)$, ¿cuál es la forma de $X_{3,:,4:10}$?

Pista: Use la regla del libro: la rebanada $X_{i,:,j:k}$ tiene forma $(b, k - j)$.

Respuesta: (8, 6): las 8 posiciones de la dimensión central, y $10 - 4 = 6$ posiciones de la última.

3. Explique en unas pocas frases cuáles son las dos cosas que describen un tensor, con un ejemplo de cada una tomado de la lección.

Pista: Una trata de qué son los elementos, la otra de cómo están dispuestos.

Respuesta: Un tensor se describe por el tipo de sus elementos, normalmente números en coma flotante pero posiblemente enteros como números de clase o cadenas para texto, y por su forma, la lista de tamaños de sus dimensiones, como (4, 5, 6) para cuatro fotos de 5 por 6 píxeles.

Flechas, longitudes y coincidencia

Póngase de pie en una habitación y señale la puerta con el brazo, y luego pida a un amigo que señale la ventana. Si ambos señalan en la misma dirección, sus brazos están completamente de acuerdo; si señalan en direcciones opuestas, están completamente en desacuerdo; si sus brazos forman un ángulo recto, no tienen nada en común. Un solo número puede medir ese acuerdo, y es la operación más importante de todo este curso: el producto escalar.

PRODUCTO ESCALAR



$$\begin{array}{|c|} \hline \square \\ \hline \square \\ \hline \square \\ \hline \square \\ \hline \end{array} \mathbf{x} \begin{pmatrix} d \\ 1 \end{pmatrix} \quad \mathbf{x}^\top \begin{pmatrix} 1 & d \end{pmatrix} \quad \begin{array}{|c|c|c|c|} \hline \square & \square & \square & \square \\ \hline \end{array} \quad (d) \neq (d, 1)$$

De acuerdo, en desacuerdo, nada en común

El libro empieza señalando que la mayoría de sus modelos son composiciones de operaciones diferenciables, y de hecho casi todas sencillas: sumas, multiplicaciones y unas pocas funciones no lineales como la exponencial $\exp(x)$, senos y cosenos, y raíces cuadradas. Los vectores, escritos $\mathbf{x} \sim (d)$, son tensores con una dimensión. Los libros de álgebra lineal distinguen con cuidado los vectores columna $\mathbf{x}$ de los vectores fila $\mathbf{x}^T$, y el libro intenta seguir esa convención. En el código es más delicado: una fila o una columna es un tensor bidimensional de forma $(1, d)$ o $(d, 1)$, que no es lo mismo que un tensor unidimensional de forma (d) .

Esa diferencia importa porque la mayoría de las bibliotecas siguen reglas de difusión tomadas de NumPy. En pocas palabras, la difusión alinea dos formas desde la derecha y repite un tensor allí donde puede, para que las formas coincidan. El recuadro de código del libro muestra un error muy común que resulta de ello: suma una «columna» de forma 4 por 1 a un vector simple de forma 4, esperando cuatro sumas, y obtendrá en cambio una matriz de 4 por 4. La columna se repite a lo largo de cuatro columnas y el vector a lo largo de cuatro filas, y se suma cada pareja, dando dieciséis números donde usted quería cuatro. Lo mismo ocurre en NumPy, TensorFlow, JAX y prácticamente cualquier biblioteca, así que hay que comprobar las formas.

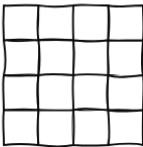
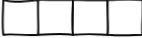
$$\mathbf{z} = a\mathbf{x} + b\mathbf{y}$$

(4, 1)



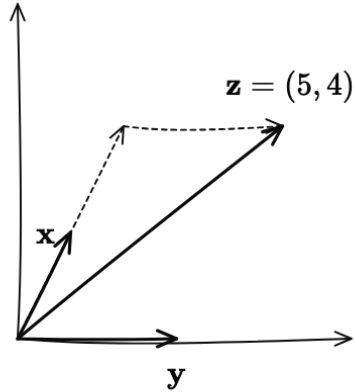
+

(4)



$4 \times 4 = 16$, no 4

REVISA FORMAS

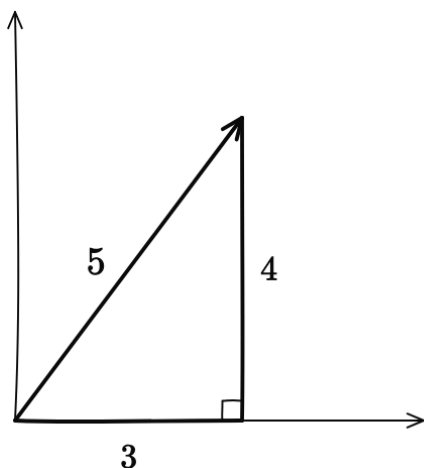


Difusión y una combinación lineal

Los vectores tienen su propia álgebra, llamada espacio vectorial: dos vectores cualesquiera de la misma forma pueden combinarse linealmente en un tercero, $\mathbf{z} = a\mathbf{x} + b\mathbf{y}$. Elemento a elemento, $z_i = ax_i + by_i$: cada entrada del resultado es a por la entrada de x más b por la entrada de y . De dónde viene es simplemente de hacer lo mismo en cada posición. Con $\mathbf{x} = (1, 2)$, $\mathbf{y} = (3, 0)$, a igual a 2 y b igual a 1, obtenemos 2 por 1 más 3, que es 5, y 2 por 2 más 0, que es 4: el vector (5, 4).

Si imaginamos un vector como un punto, o una flecha desde el origen, en un espacio de d dimensiones, sumar dos vectores significa completar un paralelogramo, y la distancia de un vector al origen es su norma euclídea, también llamada norma ℓ_2 : $\|\mathbf{x}\| = \sqrt{(\sum_i x_i^2)}$. En

palabras, eleve cada entrada al cuadrado, súmelas y saque la raíz cuadrada. Viene de Pitágoras: en dos dimensiones la flecha es el lado largo de un triángulo rectángulo cuyos otros lados son las dos entradas. Para $\mathbf{x} = (3, 4)$, 9 más 16 es 25, y su raíz es 5. La norma al cuadrado $\|\mathbf{x}\|^2$ es especialmente práctica, porque es simplemente la suma de las entradas al cuadrado, 25 aquí, sin raíz que sacar.



$$\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$$

$$\sqrt{9 + 16} = 5$$

$$\|\mathbf{x}\|^2 = 25$$

Longitud por Pitágoras

Ahora la operación central, el producto interno, también llamado producto escalar: multiplique dos vectores entrada a entrada y sume los resultados. La definición del libro dice $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \sum_i x_i y_i$. Los corchetes angulares vienen de la física, y el libro los usa de vez en cuando por claridad; lo importante es que el resultado es un solo número, un escalar. El propio ejemplo del libro: con $\mathbf{x} = [0.1, 0, -0.3]$

e $\mathbf{y} = [-4.0, 0.05, 0.1]$, los productos son menos 0,4, luego 0, luego menos 0,03, y suman menos 0,43.

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y} = \sum_i x_i y_i$$

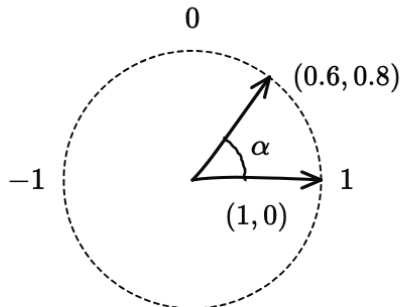
$$0.1 \times (-4.0) = -0.4$$

$$0 \times 0.05 = 0$$

$$-0.3 \times 0.1 = -0.03$$

$$\text{suma} = -0.43$$

$$\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos \alpha$$



$$\cos \alpha = 0.6$$

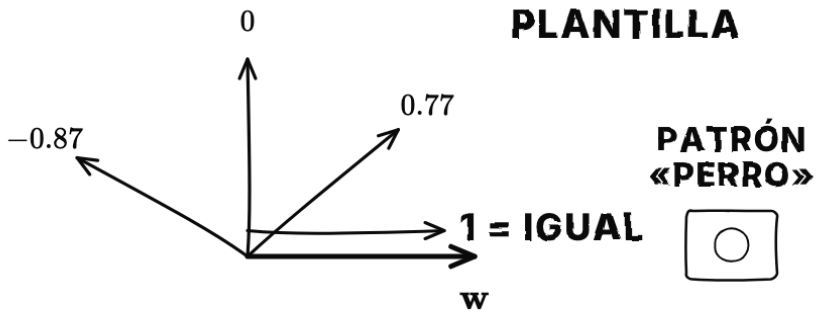
El producto escalar, de dos maneras

El producto escalar tiene un significado geométrico sencillo a través del ángulo α entre las dos flechas: $\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\alpha)$. En palabras, son las dos longitudes multiplicadas entre sí, por un factor que dice cuánto apuntan las flechas en la misma dirección. Para vectores normalizados a longitud uno, las longitudes desaparecen y el producto escalar es simplemente el coseno del ángulo, que entonces se llama similitud coseno. Va de 1, flechas que apuntan en la misma dirección, a menos 1, flechas que apuntan en direcciones opuestas, y es exactamente 0 cuando son ortogonales, en ángulo recto: de nuevo su brazo y el de su amigo. Compruebe un caso: las flechas unitarias (1, 0)

y (0,6, 0,8) tienen producto escalar 0,6, y en efecto la segunda tiene longitud 1, ya que 0,36 más 0,64 es 1, así que el coseno de su ángulo es 0,6.

Démosle la vuelta. Fije un vector unitario $\mathbf{x}$ y pregunte qué vector unitario $\mathbf{y}$ da el mayor producto escalar: $\mathbf{y}^* = \operatorname{argmax} \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}$. Aquí argmax significa «el valor de $\mathbf{y}$ que hace la salida lo más grande posible», y la respuesta es $\mathbf{y}$ igual al propio $\mathbf{x}$, ya que el coseno alcanza 1 solo cuando el ángulo es cero. Esta es la idea que el libro quiere sembrar: más adelante, $\mathbf{x}$ será una entrada y $\mathbf{w}$ un conjunto de parámetros ajustables, y el producto escalar es máximo cuando la entrada resuena con los parámetros, lo que se llama coincidencia de plantilla. Su aplicación de fotos funciona así en espíritu: compara los números de una foto con un patrón «perro» aprendido, y un producto escalar grande significa que la foto coincide con el patrón.

$$\mathbf{y}^* = \arg \max_y \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}$$



$$(2, 5, 1) \cdot (1, 1, 1) = 8 = \text{suma}$$

La mejor coincidencia es la plantilla

El libro cierra con una observación que será útil más adelante: la suma de las entradas de un vector puede escribirse a su vez como un producto escalar, con el vector **1** cuyas entradas son todas unos; multiplicar por uno y sumar simplemente suma las entradas.

Pruébalo

Tome dos lápices y sosténgalos ambos apuntando al techo: están de acuerdo, y su similitud coseno es 1. Gire uno para que apunte al suelo: coseno menos 1. Ponga uno tumbado mientras el otro sigue apuntando arriba: forman un ángulo recto, y el producto escalar de sus direcciones es 0. Cada ángulo intermedio da un número intermedio, que es todo lo que mide el producto escalar una vez que las longitudes se fijan en uno.

Para recordar

Los vectores pueden combinarse linealmente entrada a entrada, $z_i = ax_i + by_i$, y su longitud es la norma euclídea, la raíz cuadrada de la suma de las entradas al cuadrado, 5 para (3, 4). En el código, una fila o columna de forma (1, d) o (d, 1) difiere de un vector simple de forma (d), y la difusión puede convertir en silencio una suma de dos vectores en una matriz de 4 por 4. El producto escalar multiplica las entradas y las suma, dando un número, menos 0,43 en el ejemplo del libro, y es igual a las dos longitudes por el coseno del ángulo entre los vectores. Para vectores unitarios el producto escalar es la similitud coseno, de 1 a menos 1 con 0 para ángulos rectos, y es máximo cuando los dos vectores coinciden, por eso un vector de parámetros w actúa como una plantilla con la que coinciden las entradas.

Preguntas

1. En una biblioteca, suma un tensor de forma $(4, 1)$ a uno de forma (4) . ¿Por qué obtiene un resultado de 4 por 4?
 - a. Porque la difusión repite la columna a lo largo de las columnas y el vector a lo largo de las filas, y suma cada pareja
 - b. Porque la biblioteca siempre convierte las sumas de vectores en matrices
 - c. Porque un tensor de forma (4) es secretamente una matriz
2. Dos vectores de longitud uno tienen un producto escalar de 0. ¿Qué dice eso de ellos?
 - a. Apuntan en la misma dirección
 - b. Apuntan en direcciones opuestas
 - c. Están en ángulo recto entre sí
3. Fijado un vector unitario x , ¿qué vector unitario y hace el producto escalar lo más grande posible, y por qué importa eso más adelante?
 - a. Cualquier vector ortogonal a x , porque los vectores ortogonales son independientes
 - b. y igual a x , así que un vector de parámetros w actúa como una plantilla con la que resuenan las entradas que coinciden
 - c. y igual a menos x , así que se detectan las entradas opuestas a los parámetros
4. ¿Por qué la norma al cuadrado es a menudo más práctica que la propia norma?

- a. Porque siempre es menor que la norma
- b. Porque es la simple suma de las entradas al cuadrado, sin raíz cuadrada
- c. Porque no depende de las entradas del vector

Respuestas

1. **a)** La difusión alinea las formas desde la derecha y repite tensores para que coincidan, así que una columna más un vector unidimensional se convierte en una matriz de 4 por 4 en vez de cuatro sumas.
2. **c)** Para vectores unitarios el producto escalar es el coseno del ángulo entre ellos; es 0 exactamente cuando son ortogonales, en ángulo recto.
3. **b)** El coseno alcanza 1 solo con ángulo cero, así que el máximo es $y = x$; con x una entrada y w los parámetros, el producto escalar es máximo cuando la entrada coincide con la plantilla w .
4. **b)** La norma es la raíz cuadrada de la suma de las entradas al cuadrado, así que su cuadrado es simplemente esa suma, 25 para el vector (3, 4).

Ejemplos resueltos

El producto escalar del libro Calcule el producto interno de $x = [0.1, 0, -0.3]$ e $y = [-4.0, 0.05, 0.1]$.

Multiplique entrada a entrada: $0.1 \times (-4.0) = -0.4$.

$0 \times 0.05 = 0$.

$-0.3 \times 0.1 = -0.03$.

Sume los productos: $-0.4 + 0 - 0.03 = -0.43$, un solo número.

Respuesta: -0.43

Una sorpresa de la difusión x tiene forma (4, 1), una «columna», e y tiene forma (4), un tensor unidimensional. ¿Cuál es la forma de x

+ y, y cuántos números contiene?

La difusión alinea las formas desde la derecha: (4, 1) frente a (4).

El tamaño 1 de x se repite hasta 4, e y se repite a lo largo de una nueva primera dimensión de tamaño 4.

Cada entrada de x se suma a cada entrada de y.

Respuesta: Forma (4, 4), 16 números, en vez de las 4 sumas que probablemente se querían.

Ejercicios

1. Halle la norma de $x = (6, 8)$ y su norma al cuadrado.

Pista: Eleve cada entrada al cuadrado y sume.

Respuesta: $36 + 64 = 100$, así que la norma al cuadrado es 100 y la norma es $\sqrt{100} = 10$.

2. Calcule $z = 3x - 2y$ para $x = (1, 4)$ e $y = (2, 1)$.

Pista: Use $z_i = ax_i + by_i$ con $a = 3$, $b = -2$.

Respuesta: $z_1 = 3 \times 1 - 2 \times 2 = -1$ y $z_2 = 3 \times 4 - 2 \times 1 = 10$, así que $z = (-1, 10)$.

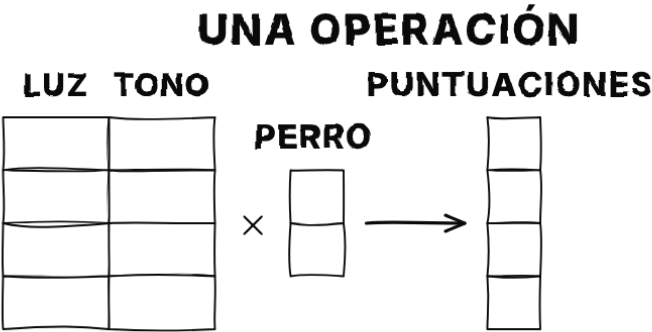
3. Los vectores unitarios (1, 0), (0,8, 0,6) y (0, 1) se comparan con $x = (1, 0)$. Ordénelos por similitud coseno con x.

Pista: Para vectores unitarios la similitud coseno es simplemente el producto escalar.

Respuesta: (1, 0) da 1, (0,8, 0,6) da 0,8, (0, 1) da 0; así que el orden es (1, 0), luego (0,8, 0,6), luego (0, 1), que es ortogonal a x.

Matrices como lotes y como funciones

Abra una hoja de cálculo y escriba en ella las características de cuatro fotos, una foto por fila: brillo, colorido, y así por las columnas. Ahora quiere puntuar cada foto frente al mismo patrón «perro»: multiplicar cada fila por el patrón y sumar. Podría hacerlo fila por fila, cuatro veces, o con una sola operación sobre toda la hoja. Esa única operación es la multiplicación de matrices, y esta parte muestra por qué es el caballo de batalla de todo modelo.

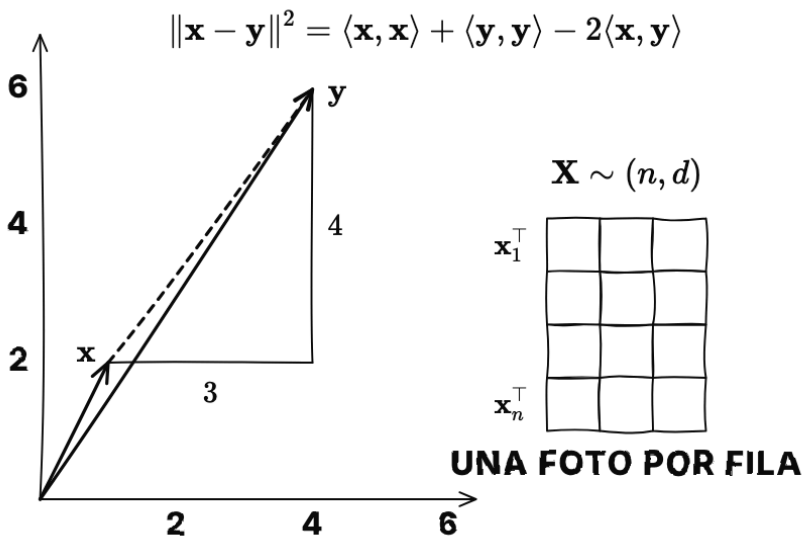


$$\langle \mathbf{x}, \mathbf{1} \rangle = (2, 5, 3) \cdot (1, 1, 1) = 10$$

Puntuar todas las filas a la vez

Primero, dos últimos hechos sobre vectores que el libro usará a menudo. La suma de las entradas de un vector es su producto escalar con el vector de todos unos: $\langle \mathbf{x}, \mathbf{1} \rangle = \sum_{i=1}^d x_i$. En palabras, multiplicar cada entrada por uno y sumarlas simplemente las suma. Para $\mathbf{x} = (2, 5, 3)$, eso es 2 más 5 más 3, que es 10.

Segundo, la distancia al cuadrado entre dos vectores puede escribirse solo con productos escalares: $\|\mathbf{x} - \mathbf{y}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle - 2\langle \mathbf{x}, \mathbf{y} \rangle$. Viene de desarrollar un cuadrado, igual que la regla escolar de que a menos b , al cuadrado, es a al cuadrado más b al cuadrado menos dos veces a por b . Compruébelo con $\mathbf{x} = (1, 2)$ e $\mathbf{y} = (4, 6)$: directamente, las diferencias son 3 y 4, y 9 más 16 es 25. Con productos escalares, $\mathbf{x}$ consigo mismo es 5, $\mathbf{y}$ consigo mismo es 52, $\mathbf{x}$ con $\mathbf{y}$ es 16, y 5 más 52 menos 32 es otra vez 25. Poniendo $\mathbf{y}$ igual al vector cero se obtiene el caso especial $\|\mathbf{x}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle$: la longitud al cuadrado de un vector es su producto escalar consigo mismo.



Distancia por productos escalares, luego un lote

Ahora a dos dimensiones. Una matriz $\mathbf{X} \sim (n, d)$ tiene n filas y d columnas, con entradas desde X_{11} en la esquina superior izquierda hasta X_{nd} en la inferior derecha. Para lo que sigue, la manera más útil de verla es como una pila de n vectores $\mathbf{x}_1, \dots, \mathbf{x}_n$, dispuestos como filas, siendo cada fila $\mathbf{x}_i^\top$. El libro dice entonces que $\mathbf{X}$ representa un lote de vectores de datos: nuestra hoja de cálculo, una foto por fila. Los modelos se definen habitualmente, tanto en papel como en código, para trabajar sobre lotes como este.

La operación fundamental es la multiplicación de matrices. Para matrices de formas compatibles, $\mathbf{X} \sim (a, b)$ e $\mathbf{Y} \sim (b, c)$, el producto $\mathbf{Z} = \mathbf{XY}$ tiene forma (a, c) , y cada entrada es $Z_{ij} = \langle \mathbf{X}_i, \mathbf{Y}_j^\top \rangle$. En palabras: la entrada en la fila i y la columna j es el producto escalar de la fila i de

la primera matriz con la columna j de la segunda, y por eso los tamaños interiores, ambos b , deben coincidir. Tome las filas (1, 2) y (3, 4) como $\mathbf{X}$ y las filas (5, 6) y (7, 8) como $\mathbf{Y}$. La entrada superior izquierda es 1 por 5 más 2 por 7, que es 19; la superior derecha, 1 por 6 más 2 por 8, que es 22; la inferior izquierda, 3 por 5 más 4 por 7, que es 43; la inferior derecha, 3 por 6 más 4 por 8, que es 50.

$$(a, b) (b, c) \rightarrow (a, c)$$

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

$$1 \cdot 5 + 2 \cdot 7 = 19$$

$$\mathbf{W}\mathbf{x} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 10 \end{pmatrix} \quad \mathbf{X}\mathbf{X}^T = \begin{pmatrix} 5 & 11 \\ 11 & 25 \end{pmatrix}$$

Fila por columna

Cuando el segundo factor es un solo vector obtenemos el producto matriz-vector, $\mathbf{z} = \mathbf{W}\mathbf{x}$: cada entrada de $\mathbf{z}$ es el producto escalar de una fila de $\mathbf{W}$ con $\mathbf{x}$. Con $\mathbf{W}$ de filas (1, 0) y (2, 1), y $\mathbf{x}$ igual a (3, 4), obtenemos 3 y 10. Vistas como lotes, $\mathbf{X}\mathbf{W}^T$ calcula, en una sola operación de álgebra lineal, el producto escalar de cada fila de $\mathbf{X}$ con cada patrón de $\mathbf{W}$: nuestra hoja de cálculo puntuada frente al patrón

perro, todas las fotos a la vez. Y $\mathbf{X}\mathbf{X}^T \sim (n, n)$ calcula de una vez todos los productos escalares entre pares de filas: para las filas (1, 2) y (3, 4) da 5 y 11 en la primera fila y 11 y 25 en la segunda, cada foto comparada con todas las demás.

Importan unas pocas operaciones más. El producto de Hadamard multiplica dos matrices de la misma forma entrada a entrada: $[\mathbf{X} \odot \mathbf{Y}]_{ij} = X_{ij} Y_{ij}$. Carece del álgebra rica de la verdadera multiplicación de matrices, pero los modelos diferenciables lo usan constantemente para enmascarar, poner algunas entradas a cero, y para escalar. Multiplique las filas (1, 2) y (3, 4) entrada a entrada con la máscara cuyas filas son (1, 0) y (0, 1), y conservará el 1 y el 4 y anulará el resto. Las interacciones multiplicativas de este tipo también se han hecho populares en algunas familias de modelos recientes, a las que el libro vuelve más adelante.

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

MÁSCARA

$$[\exp(\mathbf{X})]_{ij} = \exp(X_{ij}), \quad \exp(0) = 1$$

$$\text{mat-exp} : 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} \approx 2.708 \approx e$$

SOLO CUADRADAS

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \xrightarrow{\Sigma_j} \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$

SUMA POR EJE

Entrada a entrada, y a lo largo de un eje

Una expresión como $\exp(\mathbf{X})$ se entiende elemento a elemento:

$[\exp(\mathbf{X})]_{ij} = \exp(X_{ij})$, la exponencial tomada de cada entrada por separado, así que una entrada cero se convierte en 1. Compárelo con la verdadera exponencial matricial del álgebra lineal, $\text{mat-exp}(\mathbf{X}) = \sum_{k=0}^{\infty} \mathbf{X}^k / k!$, la suma de las potencias de la matriz, cada una dividida por k factorial. Para una matriz de uno por uno que contiene el número 1, los primeros términos son 1 más 1 más un medio más un sexto más un veinticuatroavo, unos 2,708, ya cerca de e , unos 2,718. La versión elemento a elemento funciona para tensores de cualquier forma, mientras que la exponencial matricial solo tiene sentido para matrices cuadradas; por eso bibliotecas como PyTorch guardan esas operaciones solo para matrices, también las inversas y los

determinantes, en un módulo aparte de álgebra lineal, como muestra el recuadro de código del libro con sus dos exponenciales.

Las reducciones como sumas y medias a lo largo de un eje se escriben sin límites cuando recorren todo el eje, $\sum_i X_i$, y en el código reciben un argumento de eje. Sumar las filas (1, 2) y (3, 4) a lo largo del segundo eje da un total por fila, 3 y 7.

¿Por qué se define la multiplicación de matrices de esta extraña manera fila por columna, en vez de entrada a entrada como Hadamard? La respuesta del libro: una función f de vectores es lineal si $f(\alpha \mathbf{x}_1 + \beta \mathbf{x}_2) = \alpha f(\mathbf{x}_1) + \beta f(\mathbf{x}_2)$, y toda función así puede escribirse como una matriz $\mathbf{A}$. Entonces el producto matriz-vector $\mathbf{A}\mathbf{x}$ es aplicar la función, y el producto de matrices $\mathbf{AB}$ es componer dos funciones: $(f \circ g)(\mathbf{x}) = f(g(\mathbf{x}))$ con $g(\mathbf{x}) = \mathbf{B}\mathbf{x}$. Compruébelo: sea $\mathbf{B}$ de filas (1, 1) y (0, 1), y $\mathbf{A}$ de filas (2, 0) y (0, 3); para $\mathbf{x}$ igual a (1, 1), $\mathbf{B}$ da (2, 1) y luego $\mathbf{A}$ da (4, 3). El producto $\mathbf{AB}$ tiene filas (2, 2) y (0, 3), y aplicado a (1, 1) da (4, 3) directamente: hacer una función tras otra es lo mismo que multiplicar primero sus matrices.

$$f(\alpha \mathbf{x}_1 + \beta \mathbf{x}_2) = \alpha f(\mathbf{x}_1) + \beta f(\mathbf{x}_2)$$

$$(f \circ g)(\mathbf{x}) = f(g(\mathbf{x}))$$

$$(1, 1) \xrightarrow{\mathbf{B}} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \rightarrow (2, 1) \xrightarrow{\mathbf{A}} \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \rightarrow (4, 3)$$

=

$$(1, 1) \xrightarrow{\quad \quad \quad} \mathbf{AB} = \begin{pmatrix} 2 & 2 \\ 0 & 3 \end{pmatrix} \xrightarrow{\quad \quad \quad} (4, 3)$$

Multiplicar matrices compone funciones

Pruébalo

En papel cuadriculado, escriba la matriz de filas (1, 2) y (3, 4) a la izquierda y la de filas (5, 6) y (7, 8) a la derecha. Recorra con un dedo la primera fila de la matriz izquierda y baje por la primera columna de la derecha, multiplicando y sumando: debería obtener 19. Haga las otras tres entradas y compruebe 22, 43 y 50; ha hecho a mano lo que el chip de un teléfono hace miles de millones de veces cuando etiqueta sus fotos.

Para recordar

La suma de las entradas de un vector es su producto escalar con el vector de unos, y la distancia al cuadrado entre dos vectores puede escribirse con productos escalares, $\|x - y\|^2 = \langle x, x \rangle + \langle y, y \rangle - 2\langle x, y \rangle$. Una matriz se ve mejor como un lote de vectores fila, y la entrada Z_{ij} de un producto XY es el producto escalar de la fila i de X con la columna j de Y , como 19, 22, 43, 50 en nuestro ejemplo. XW^T puntúa cada fila de un lote frente a cada patrón a la vez, y XX^T compara todos los pares de filas; el producto de Hadamard multiplica entrada a entrada y se usa para enmascarar y escalar. Funciones como $\exp$ actúan elemento a elemento sobre los tensores, a diferencia de la verdadera exponencial matricial de las matrices cuadradas; y la multiplicación de matrices se define como se define porque compone funciones lineales.

Preguntas

1. En un producto de matrices $Z = XY$, ¿qué es la entrada Z_{ij} ?
 - a. El producto de las entradas X_{ij} e Y_{ij}
 - b. El producto escalar de la fila i de X con la columna j de Y
 - c. La suma de la fila i de X más la columna j de Y
2. ¿Por qué la multiplicación de matrices se define fila por columna y no entrada a entrada?

- a. Porque así multiplicar matrices compone las funciones lineales que representan
 - b. Porque es más rápida de calcular que los productos entrada a entrada
 - c. Porque los productos entrada a entrada no están permitidos para matrices cuadradas
3. Si X es un lote de fotos, una por fila, ¿qué da XX^T ?
- a. La suma de todas las fotos
 - b. Cada foto multiplicada entrada a entrada consigo misma
 - c. Todos los productos escalares entre pares de fotos, como una matriz de n por n
4. ¿Por qué las bibliotecas guardan la exponencial matricial en un módulo aparte de álgebra lineal, mientras que $\exp$ funciona sobre cualquier tensor?
- a. Porque la exponencial matricial es solo una aproximación de $\exp$
 - b. Porque la $\exp$ elemento a elemento funciona para tensores de cualquier forma, mientras que la exponencial matricial solo está definida para matrices cuadradas
 - c. Porque $\exp$ no puede aplicarse a matrices en absoluto

Respuestas

1. **b)** Cada entrada es el producto escalar de la fila i de la primera matriz con la columna j de la segunda, y por eso los tamaños interiores deben coincidir; la multiplicación entrada a entrada es el producto de Hadamard.
2. **a)** Toda función lineal es una matriz; aplicarla es un producto matriz-vector, y el producto de matrices AB es la función que aplica primero $g = B$ y

luego $f = A$.

3. c) XX^T tiene forma (n, n) , y su entrada i, j es el producto escalar de las filas i y j : cada foto comparada con todas las demás en una sola operación.

4. b) $\exp(X)$ aplica la exponencial a cada entrada, válido para cualquier forma, mientras que la verdadera exponencial matricial suma potencias de la matriz, lo que solo tiene sentido para matrices cuadradas.

Ejemplos resueltos

Multiplicar dos matrices pequeñas Calcule XY para X con filas $(1, 2)$ y $(3, 4)$, e Y con filas $(5, 6)$ y $(7, 8)$.

Superior izquierda: fila 1 de X con columna 1 de Y , $1 \times 5 + 2 \times 7 = 19$.

Superior derecha: fila 1 con columna 2, $1 \times 6 + 2 \times 8 = 22$.

Inferior izquierda: fila 2 con columna 1, $3 \times 5 + 4 \times 7 = 43$.

Inferior derecha: fila 2 con columna 2, $3 \times 6 + 4 \times 8 = 50$.

Respuesta: XY tiene filas $(19, 22)$ y $(43, 50)$.

Distancia a partir de productos escalares Para $x = (1, 2)$ e $y = (4, 6)$, calcule $\|x - y\|^2$ directamente y con $\langle x, x \rangle + \langle y, y \rangle - 2\langle x, y \rangle$.

Directamente: $x - y = (-3, -4)$, y $9 + 16 = 25$.

$\langle x, x \rangle = 1 + 4 = 5$ y $\langle y, y \rangle = 16 + 36 = 52$.

$\langle x, y \rangle = 4 + 12 = 16$, así que $2\langle x, y \rangle = 32$.

$5 + 52 - 32 = 25$, lo mismo que el cálculo directo.

Respuesta: 25 de las dos maneras

Ejercicios

1. Calcule Wx para W con filas $(2, 1)$ y $(0, 3)$, y $x = (4, 5)$.

Pista: Cada entrada es el producto escalar de una fila de W con x .

Respuesta: Fila 1: $2 \times 4 + 1 \times 5 = 13$; fila 2: $0 \times 4 + 3 \times 5 = 15$; así que $Wx = (13, 15)$.

2. Enmascare la matriz de filas (5, 6) y (7, 8) usando el producto de Hadamard con la máscara cuyas filas son (0, 1) y (1, 0). ¿Qué queda?

Pista: Multiplique entrada a entrada.

Respuesta: Las filas (0, 6) y (7, 0): las entradas donde la máscara es 1 sobreviven, las demás se vuelven 0.

3. Sea $g(x) = Bx$ con B de filas (1, 0) y (1, 1), y $f(x) = Ax$ con A de filas (3, 0) y (0, 2). Calcule $f(g(x))$ para $x = (2, 1)$ de dos maneras: una función tras otra, y con la única matriz AB.

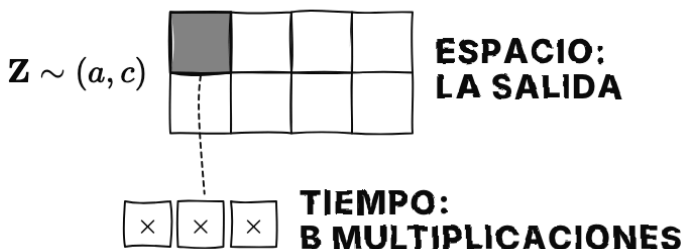
Pista: Primero Bx , luego A por el resultado; después calcule AB fila por columna.

Respuesta: $Bx = (2, 3)$, luego A da (6, 6). AB tiene filas (3, 0) y (2, 2), y AB por (2, 1) es (6, 6): lo mismo, porque el producto de matrices compone las funciones.

Cómo crece el coste

Cronometre cuánto tarda su teléfono en etiquetar un álbum pequeño, y luego imagine un álbum diez veces más ancho en cada dirección. Podría suponer que diez veces más tiempo, pero para la multiplicación de matrices la respuesta es unas mil veces más. Contar cómo crece el trabajo con el tamaño, antes de ejecutar nada, es el tema de esta parte.

10x MÁS ANCHO → 1000x TRABAJO



$$a = 2, b = 3, c = 4 : \quad 8 \times 3 = 24$$

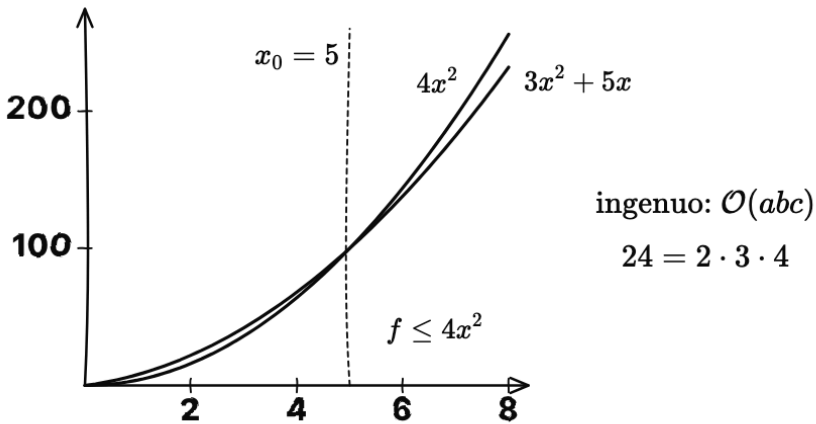
Contar el trabajo

El libro usa la multiplicación de matrices para introducir la complejidad de una operación. Para calcular $Z \sim (a, c)$ a partir de $X \sim$

(a, b) e $Y \sim (b, c)$ directamente desde la definición, necesitamos ac productos internos, uno por entrada de Z , cada uno de longitud b . Esa cuenta del trabajo se llama complejidad temporal. La memoria que necesita una implementación sencilla y secuencial es simplemente el tamaño de la matriz de salida, y eso se llama complejidad espacial. Con a igual a 2, b igual a 3 y c igual a 4, Z tiene 8 entradas, cada una un producto escalar de 3 multiplicaciones, así que 24 multiplicaciones en total, y 8 números que guardar.

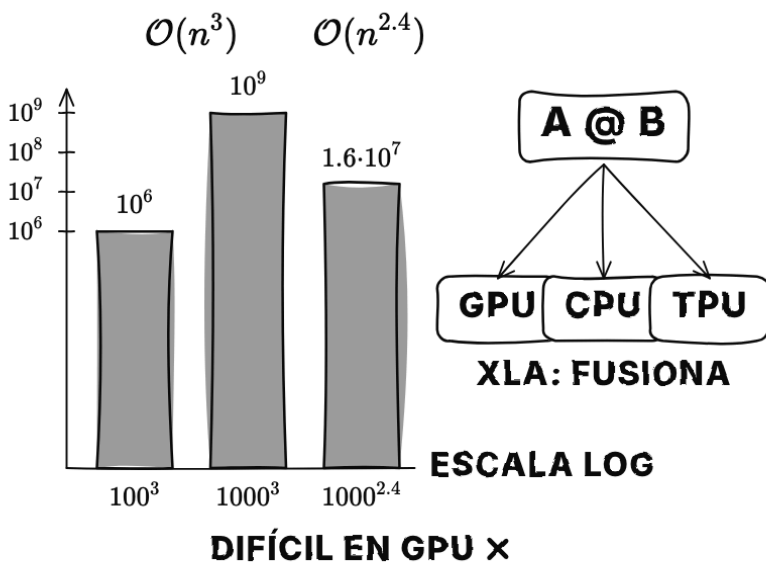
Para alejarse de los detalles de un hardware concreto, la informática usa la notación O grande, de la palabra alemana *Ordnung*, orden de aproximación. La definición: una función $f(x)$ es $\mathcal{O}(g(x))$, tomando entradas y salidas como no negativas, si existen una constante c y un valor inicial x_0 tales que $f(x) \leq cg(x)$ para cualquier $x \geq x_0$. En palabras: una vez que x es suficientemente grande, f nunca supera un múltiplo fijo de g , así que para entradas grandes todo salvo g puede ignorarse. Eso se llama análisis asintótico. Por ejemplo, $f(x) = 3x^2 + 5x$ es $\mathcal{O}(x^2)$: tome c igual a 4 y x cero igual a 5, porque tres equis al cuadrado más cinco equis es como mucho cuatro equis al cuadrado exactamente cuando cinco equis es como mucho equis al cuadrado, lo que se cumple a partir de 5.

$$3x^2 + 5x \leq 4x^2 \iff 5x \leq x^2$$



O grande: al final por debajo de c·g

Así que una multiplicación de matrices ingenua es $\mathcal{O}(abc)$: el trabajo crece en proporción a cada uno de los tres tamaños, como muestra nuestro 24 igual a 2 por 3 por 4. Para dos matrices cuadradas de tamaño (n, n) eso es $\mathcal{O}(n^3)$, cúbico en el tamaño. Con n igual a 100 es un millón de multiplicaciones; con n igual a 1000, mil millones: diez veces más ancho, mil veces el trabajo, el número del que partimos.



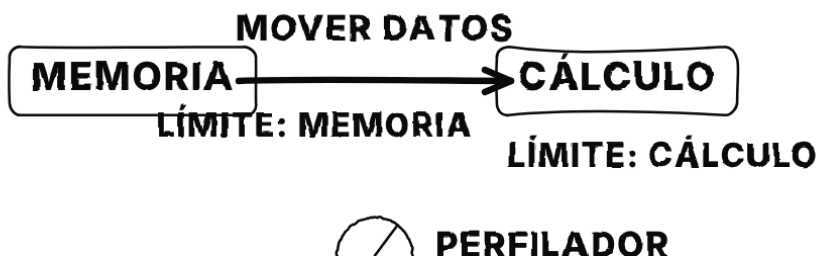
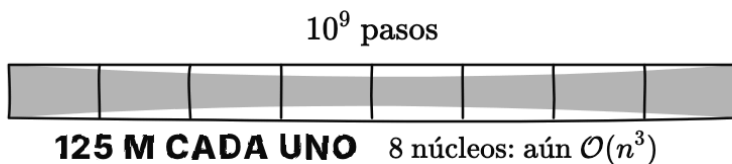
Coste cúbico, y quién lo ejecuta

El pensamiento asintótico es importante y elegante, dice el libro, pero elegir un algoritmo solo por su O grande no garantiza velocidad real, que depende del hardware, de cuánto trabajo en paralelo admite, y de mucho más. Cuando llama a la multiplicación de matrices en una biblioteca, escrita $A @ B$ en PyTorch, la rutina de bajo nivel que realmente se ejecuta, llamada kernel, se elige en tiempo de ejecución según su hardware, un proceso conocido como despacho; la misma línea de código puede ejecutarse en un kernel de GPU, de CPU o de TPU. Compiladores como XLA complican esto aún más, fusionando y remodelando operaciones para un chip de destino concreto.

El ejemplo del libro: los mejores algoritmos conocidos para multiplicar dos matrices de n por n escalan como $O(n^3)$ con una

constante c por debajo de 2,4, mucho mejor que el ingenuo $\mathcal{O}(n^3)$. Para n igual a 1000, n elevado a 2,4 es unos 16 millones, frente a mil millones para el cubo. Sin embargo, estos algoritmos ingeniosos son difíciles de ejecutar con eficiencia en paralelo en hardware como las GPU, así que en la práctica rara vez se usan.

Dos límites más del pensamiento asintótico. Primero, tener k procesadores no cambia en absoluto la O grande: como mucho divide el tiempo por k , un factor constante; ocho núcleos convierten mil millones de pasos en 125 millones cada uno, todavía cúbico en n . Segundo, la O grande ignora el tiempo dedicado a mover datos de un sitio a otro, que a menudo es el verdadero cuello de botella. Cuando eso ocurre, la implementación se llama limitada por memoria, en contraposición a limitada por cálculo, y la única manera práctica de saber cuál tiene es ejecutar un perfilador, una herramienta que mide adónde va realmente el tiempo. El libro promete que juzgar el coste de un algoritmo resultará lejos de trivial, porque la complejidad asintótica y la velocidad observada tiran en direcciones distintas.



Dos límites más

Pruébalo

Tome dos rejillas de números de 2 por 2 y multiplíquelas a mano, contando las multiplicaciones: contará 8. Ahora haga dos rejillas de 3 por 3: contará 27. Duplicar de 2 a 4 daría 64, ocho veces más: el tamaño subió dos veces, el trabajo dos por dos por dos veces, que es lo que significa cúbico.

Para recordar

Calcular $Z \sim (a, c)$ a partir de $X \sim (a, b)$ e $Y \sim (b, c)$ por definición requiere ac productos escalares de longitud b , su complejidad temporal, y necesita memoria para la salida, su complejidad espacial. La O grande dice que $f(x)$ es $O(g(x))$ cuando f se mantiene por debajo de c por g a partir de algún x_0 , así que solo importa la parte que crece más deprisa; la multiplicación de matrices ingenua es $O(abc)$, cúbica para matrices cuadradas. La velocidad real depende del hardware, los kernels y los compiladores: existen algoritmos por debajo de n elevado a 2,4, pero funcionan mal en las GPU, y k procesadores solo dividen el tiempo por una constante. Mover datos puede dominar, haciendo el código limitado por memoria en vez de por cálculo, y solo un perfilador dice cuál es el caso.

Preguntas

1. Dos matrices cuadradas crecen de tamaño 100 a tamaño 1000. ¿Aproximadamente cuánto más trabajo necesita la multiplicación ingenua?
 - a. Unas 10 veces más
 - b. Unas 100 veces más
 - c. Unas 1000 veces más
2. ¿Por qué los algoritmos que multiplican matrices en menos de n elevado a 2,4 pasos rara vez se usan en la práctica?

- a. Porque dan resultados ligeramente erróneos
 - b. Porque son difíciles de paralelizar con eficiencia en hardware como las GPU
 - c. Porque necesitan más memoria que la matriz de salida
3. ¿Por qué tener k procesadores no cambia la complejidad O grande de un algoritmo?
- a. Porque como mucho divide el tiempo por la constante k , y la O grande ignora las constantes
 - b. Porque los procesadores no pueden compartir memoria
 - c. Porque la O grande solo cuenta el procesador más lento
4. Un código está limitado por memoria. ¿Qué significa eso, y cómo lo averiguaría?
- a. Se queda sin memoria; lo averigua por un mensaje de error
 - b. Usa demasiadas multiplicaciones; el análisis O grande lo revela
 - c. Mover datos, no calcular, es el cuello de botella; un perfilador lo muestra

Respuestas

- 1. c)** La multiplicación ingenua de matrices de n por n es $O(n^3)$: diez veces el tamaño da diez por diez por diez, mil veces el trabajo, de un millón a mil millones de multiplicaciones.
- 2. b)** Su coste asintótico es mucho mejor, pero el rendimiento real depende del hardware y del paralelismo, y estos algoritmos funcionan mal en chips muy paralelos.
- 3. a)** El paralelismo puede dar como mucho una aceleración constante de $1/k$, y el análisis asintótico ignora los factores constantes.

4. c) La O grande ignora el movimiento de datos, que a menudo domina; ese código está limitado por memoria y no por cálculo, y solo ejecutar un perfilador muestra cuál es el caso.

Ejemplos resueltos

Contar el trabajo de un producto X tiene forma (2, 3) e Y tiene forma (3, 4). ¿Cuántos productos escalares, y cuántas multiplicaciones, necesita calcular $Z = XY$ por definición, y cuántos números hay que guardar?

Z tiene forma $(a, c) = (2, 4)$, así que tiene $ac = 8$ entradas.

Cada entrada es un producto escalar de longitud $b = 3$, así que 3 multiplicaciones cada una.

En total, $8 \times 3 = 24$ multiplicaciones, que es igual a $abc = 2 \times 3 \times 4$.

Una implementación secuencial guarda la salida, 8 números.

Respuesta: 8 productos escalares, 24 multiplicaciones, 8 números guardados

Comprobar una afirmación de O grande Demuestre que $f(x) = 3x^2 + 5x$ es $O(x^2)$ encontrando c y x_0 .

Pruebe $c = 4$: necesitamos $3x^2 + 5x \leq 4x^2$.

Reste $3x^2$: la condición se convierte en $5x \leq x^2$.

Para x positivo, divida por x : $5 \leq x$, así que se cumple para todo $x \geq 5$.

Respuesta: $c = 4$ y $x_0 = 5$ funcionan, así que f es $O(x^2)$.

Ejercicios

1. ¿Cuántas multiplicaciones necesita un producto ingenuo de dos matrices de 10 por 10, y de dos de 20 por 20?

Pista: Use n^3 para matrices cuadradas.

Respuesta: $10^3 = 1000$ y $20^3 = 8000$: duplicar el tamaño multiplica el trabajo por ocho.

2. Un producto de $X \sim (1000, 50)$ e $Y \sim (50, 10)$ se calcula por definición. ¿Cuál es su complejidad temporal en multiplicaciones, y cuántos números contiene la salida?

Pista: abc multiplicaciones; la salida tiene forma (a, c).

Respuesta: $1000 \times 50 \times 10 = 500\,000$ multiplicaciones; la salida tiene forma (1000, 10), 10 000 números.

3. Explique por qué elegir entre dos algoritmos solo por su O grande puede engañarle en un teléfono o una GPU reales.

Pista: Piense en el hardware paralelo, los kernels y el movimiento de datos.

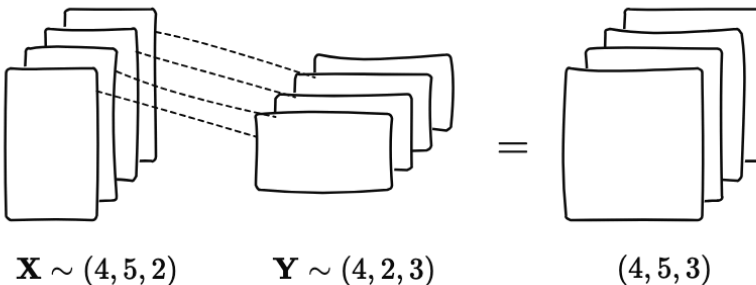
Respuesta: La O grande ignora los factores constantes, el paralelismo y el movimiento de datos. Un algoritmo teóricamente más rápido puede paralelizarse mal en una GPU, el kernel que realmente se ejecuta depende del hardware, y mover datos puede ser el verdadero cuello de botella, que solo un perfilador revela.

Muchos productos a la vez, y una abreviatura para todos ellos

Imagine una pila de cuatro hojas de papel cuadriculado, cada una con una pequeña tabla de números, y al lado una segunda pila de cuatro. Quiere multiplicar la hoja uno por la hoja uno, la hoja dos por la hoja dos, y así sucesivamente, cuatro productos de matrices separados hechos lado a lado. Dentro de cada pareja no pasa nada nuevo; lo único nuevo es la pila, y esta parte muestra cómo las bibliotecas manejan tales pilas, y una manera elegante de escribir cualquiera de estas operaciones como una corta cadena de letras.

$$[\text{BMM}(\mathbf{X}, \mathbf{Y})]_i = \mathbf{X}_i \mathbf{Y}_i$$

$$\mathbf{X} @ \mathbf{Y}$$



$$4 \times (5 \cdot 2 \cdot 3) = 4 \times 30 = 120$$

Un producto por hoja

Los vectores y las matrices son especiales, dice el libro, porque para ellos están definidas muchas operaciones que en más dimensiones no están definidas o son incómodas: la exponencial matricial, la multiplicación de matrices, los determinantes. En dimensiones superiores, la mayoría de las operaciones que necesitamos son o bien versiones por lotes de operaciones matriciales, o bien combinaciones de operaciones matriciales con reducciones como sumas.

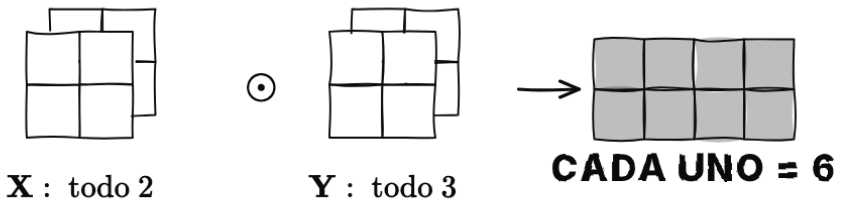
Primero el tipo por lotes. Tome dos tensores $X \sim (n, a, b)$ e $Y \sim (n, b, c)$, dos pilas de n matrices cada una. La multiplicación de matrices por lotes, BMM, es $[BMM(X, Y)]_i = X_i Y_i \sim (n, a, c)$. En palabras: para cada posición i de la pila, multiplique la matriz i de X por la matriz i de Y , y apile los n resultados. De dónde viene no es más que nuestras hojas de papel, un producto de matrices ordinario por hoja. El recuadro de código del libro usa pilas de 4: X de forma 4 por 5 por 2 e Y de forma 4 por 2 por 3 dan un resultado de forma 4 por 5 por 3. Contando el trabajo: cada producto es 5 por 2 por 3, que son 30 multiplicaciones, y hay 4, así que 120 en total.

Las bibliotecas manejan tales pilas de forma transparente: dan por supuesto que las dimensiones de lote van primero, las dimensiones iniciales, y en PyTorch la multiplicación por lotes se escribe exactamente igual que la ordinaria, con `matmul` o el signo `@`. Su aplicación de fotos hace esto constantemente: un lote entero de fotos pasa por la misma capa, la matriz de cada foto multiplicada por los mismos pesos, todo en una sola llamada.

Ahora el tipo con reducción. Para dos tensores de la misma forma, $X, Y \sim (a, b, c)$, un producto escalar generalizado, GDT, es $GDT(X, Y) = \sum_{i, j, k} [X \odot Y]_{i j k}$. En palabras: multiplique los dos tensores entrada a entrada, y luego súmelo todo. Es simplemente el producto escalar

ordinario de los dos tensores después de aplanar cada uno en un único vector largo. Para dos tensores de forma 2 por 2 por 2, con cada entrada de X igual a 2 y cada entrada de Y igual a 3, hay 8 productos de 6, así que el resultado es 48. Estos dos tipos, dice el libro, cubren la mayoría de las operaciones con tensores usadas en el resto del libro.

$$\text{GDT}(\mathbf{X}, \mathbf{Y}) = \sum_{i,j,k} [\mathbf{X} \odot \mathbf{Y}]_{ijk}$$



APLANAR, LUEGO PRODUCTO ESCALAR

$$8 \times 6 = 48$$

EINSUM →

Multiplicar todo, sumar todo

La siguiente sección, marcada como opcional, presenta einsum, una convención que puede expresar prácticamente cualquier operación con tensores, sumas, productos y reducciones, con una simple cadena de texto. Escribamos los dos ejemplos con cada índice visible.

Multiplicación por lotes: $M_{ijk} = \sum_z A_{ijz} B_{z_k}$. Producto escalar generalizado: $M = \sum_i \sum_j \sum_k X_{ijk} Y_{ijk}$.

$$M_{ijk} = A_{ijz} B_{izk}$$

$$M = X_{ijk} Y_{ijk}$$

FALTA A LA IZQUIERDA → SE SUMA

ijz, izk → ijk

A B SALIDA

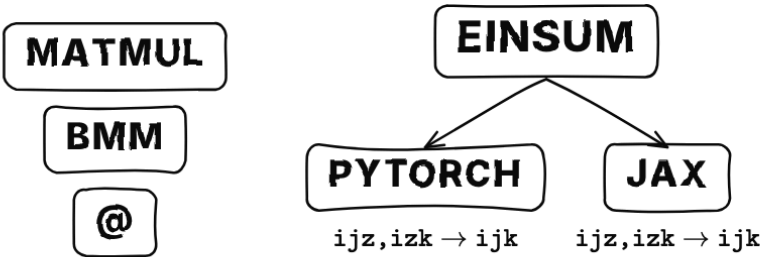
ijk, ijk →

De las sumas a una cadena

Siguiendo la notación de Einstein, en una forma simplificada que ignora la diferencia entre índices superiores e inferiores, quitamos los signos de suma, con una regla: cualquier índice que aparece a la derecha pero no a la izquierda se suma. Así, $M_{ijk} = A_{ijz} B_{izk}$ significa «sumar sobre z», porque z falta en M, y $M = X_{ijk} Y_{ijk}$ significa «sumar sobre los tres», porque M no tiene ningún índice.

Luego conservamos solo las letras de los índices, primero los operandos, y tras una flecha el resultado. La multiplicación de matrices por lotes se convierte en 'ijz,izk→ijk'. El producto escalar generalizado se convierte en 'ijk,ijk→'. Lea la primera en voz alta: el primer tensor tiene ejes i, j, z; el segundo i, z, k; la salida conserva i, j, k; z desaparece, así que se suma. Hay una correspondencia uno a uno

entre las fórmulas y las cadenas, y la mayoría de las bibliotecas lo implementan como una operación einsum.



REGLAS DISTINTAS

$$A \sim (n, a, b), B \sim (n, c, b)$$

EINOPS

$$ijz, ikz \rightarrow ijk$$

Una sintaxis en todas partes

La ventaja, dice el libro, es que ya no necesita recordar la interfaz de cada biblioteca. Solo PyTorch tiene varias funciones de multiplicación, como `matmul` y `bmm`, con distintas reglas de difusión y exigencias de forma, mientras que `einsum` da una sintaxis uniforme para todas ellas. Pasar de una biblioteca a otra también se vuelve transparente: la cadena `einsum` para la multiplicación por lotes es exactamente la misma en JAX que en PyTorch, como muestran los dos recuadros de código del libro.

Los ejes traspuestos también son fáciles. Para $A \sim (n, a, b)$ y $B \sim (n, c, b)$, multiplicar cada matriz de A por la traspuesta de la matriz correspondiente de B solo necesita intercambiar las letras de B: 'ijz,ikz→ijk'. La letra sumada z está ahora en último lugar en ambas entradas, que es exactamente lo que significa multiplicar por una traspuesta. Por estas razones einsum y sus parientes, como el popular paquete einops, se han extendido mucho.

Pruébalo

Escriba la cadena 'ijz,izk→ijk' en una tarjeta y, debajo, las formas 4 por 5 por 2 y 4 por 2 por 3. Empareje letras con tamaños: i es 4, j es 5, z es 2, k es 3. Lea la salida i, j, k: 4 por 5 por 3, la misma forma que imprime el código del libro, y z, la letra que desapareció, es el tamaño 2 que se sumó.

Para recordar

En más de dos dimensiones, la mayoría de las operaciones son operaciones matriciales por lotes u operaciones matriciales combinadas con reducciones. La multiplicación de matrices por lotes multiplica las matrices correspondientes de dos pilas, (n, a, b) con (n, b, c) da (n, a, c) , y las bibliotecas tratan las dimensiones iniciales como lotes automáticamente. El producto escalar generalizado multiplica dos tensores de la misma forma entrada a entrada y lo suma todo, un producto escalar de los tensores aplanados. Einsum escribe cualquier operación así como una cadena de letras de índice, sumando cada letra que falta en la salida, como en $'ijz,izk \rightarrow ijk'$ o $'ijk,ijk \rightarrow'$, con la misma sintaxis en todas las bibliotecas.

Preguntas

1. X tiene forma (n, a, b) e Y tiene forma (n, b, c) . ¿Qué produce la multiplicación de matrices por lotes?
 - a. Una matriz de forma (a, c) , la suma de los n productos
 - b. Una pila de n matrices de forma (a, c) , un producto por posición
 - c. Un solo número, el producto escalar de los tensores aplanados
2. En la cadena einsum $'ijz,izk \rightarrow ijk'$, ¿por qué se suma sobre el índice z ?

- a. Porque aparece en ambas entradas pero no en la salida
 - b. Porque es la última letra del alfabeto usada
 - c. Porque es la dimensión de lote
3. ¿Qué calcula la cadena 'ijk,ijk→'?
- a. Un tensor de la misma forma con las entradas multiplicadas
 - b. Un producto de matrices por lotes
 - c. El producto escalar generalizado: multiplicar entrada a entrada y sumarlo todo en un solo número
4. ¿Por qué el libro encuentra einsum cómodo entre bibliotecas?
- a. Porque se ejecuta más rápido que cualquier multiplicación incorporada
 - b. Porque una cadena uniforme sustituye a las distintas funciones de cada biblioteca y es la misma en PyTorch y JAX
 - c. Porque evita la necesidad de dimensiones de lote

Respuestas

1. **b)** Para cada posición i de la pila, la matriz i de X se multiplica por la matriz i de Y , y los n resultados se apilan en la forma (n, a, c) .
2. **a)** La convención es que cualquier índice que aparece a la derecha pero no en el resultado se suma; z falta en ijk , así que se suma.
3. **c)** La salida no tiene ningún índice, así que se suma sobre todos: los productos entrada a entrada se suman, el producto escalar de los tensores aplanados.
4. **b)** Solo PyTorch tiene varias funciones de multiplicación con reglas distintas, mientras que einsum da una sintaxis para todas ellas, y la misma cadena funciona en JAX.

Ejemplos resueltos

El producto por lotes del libro En el recuadro de código del libro, X tiene forma (4, 5, 2) e Y tiene forma (4, 2, 3). Halle la forma de $X @ Y$ y cuente las multiplicaciones hechas por definición.

La dimensión inicial 4 es el lote: hay 4 productos de matrices separados.

Cada producto multiplica una matriz de 5 por 2 por una de 2 por 3, dando 5 por 3.

Así que el resultado tiene forma (4, 5, 3).

Cada producto necesita $5 \times 2 \times 3 = 30$ multiplicaciones, y $4 \times 30 = 120$ en total.

Respuesta: Forma (4, 5, 3); 120 multiplicaciones

Ejercicios

1. X e Y tienen ambos forma (2, 2, 2); cada entrada de X es 1 y cada entrada de Y es 5. ¿Cuál es su producto escalar generalizado?

Pista: Cuente las entradas y multiplique cada pareja.

Respuesta: Hay $2 \times 2 \times 2 = 8$ entradas, cada producto es $1 \times 5 = 5$, así que la suma es 40.

2. A tiene forma (8, 10, 6) y B tiene forma (8, 6, 4). Usando 'ijz,izk→ijk', dé el tamaño de cada letra y la forma de la salida.

Pista: Empareje las letras de cada operando con sus tamaños.

Respuesta: $i = 8, j = 10, z = 6, k = 4$; la salida conserva i, j, k, así que tiene forma (8, 10, 4), y se suma sobre $z = 6$.

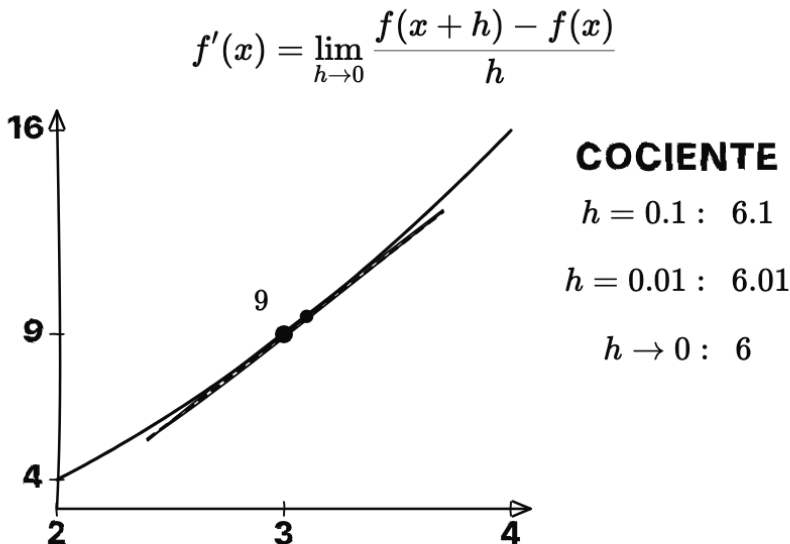
3. A tiene forma (n, a, b) y B tiene forma (n, c, b). ¿Qué cadena einsum multiplica cada matriz de A por la traspuesta de la matriz correspondiente de B, y cuál es la forma de la salida?

Pista: La letra sumada debe estar en el eje de tamaño b en ambas entradas.

Respuesta: 'ijz,ikz→ijk': z es el tamaño compartido b, sobre el que se suma, y la salida tiene forma (n, a, c).

La pendiente en un punto

Abra una calculadora, escriba 3 y elévelo al cuadrado: 9. Ahora escriba 3,1 y elévelo al cuadrado: 9,61, que creció 0,61 para un paso de 0,1, unas seis veces el paso. Pruebe 3,01: 9,0601, un crecimiento de 0,0601, aún más cerca de seis veces el paso. El cociente se va asentando en un solo número, y ese número es la derivada, el fundamento de todo lo que el libro llama diferenciable.



Cocientes que se asientan en 6

Los gradientes tienen un papel central en el libro, como sugiere la palabra diferenciable: permiten optimizar los modelos con métodos

semiautomáticos derivados del descenso de gradiente. Esta sección recuerda las definiciones básicas para derivar funciones, centrándose en lo que hará falta más adelante, a veces a costa del rigor matemático completo.

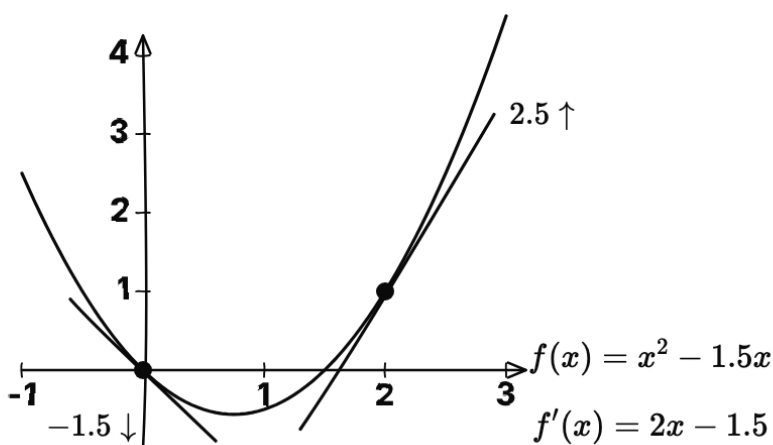
Empiece por el caso más sencillo, $y = f(x)$, un número que entra y un número que sale. Su derivada se define como $f'(x) = \lim_{h \rightarrow 0} (f(x + h) - f(x)) / h$. En palabras: dé un pasito, de tamaño h , mire cuánto cambió la salida, divida por el paso y deje que el paso se encoja hacia cero. Eso es exactamente lo que hizo nuestra calculadora: con f igual a equis al cuadrado en 3, pasos de 0,1 y 0,01 dieron cocientes de 6,1 y 6,01, acercándose a 6.

El libro usa varias notaciones. El símbolo ∂ representa derivadas y gradientes de cualquier dimensión; ∂_x , o ∂ sobre ∂x , nombra la variable respecto a la que derivamos cuando hace falta decirlo; y $f'(x)$, llamada notación de Lagrange, se reserva para funciones de un solo número. El libro simplemente da por supuesto que las derivadas que necesita existen, lo que no está garantizado ni siquiera para funciones continuas, y vuelve al asunto solo para funciones con esquinas, como el valor absoluto $f(x) = \text{abs}(x)$ en cero, en el capítulo 6.

Para funciones sencillas, aplicar la definición da las reglas conocidas. Para una potencia, $\partial x^p = p x^{p-1}$: el exponente baja delante y disminuye en uno; para equis al cubo en 2, eso es 3 por 2 al cuadrado, que es 12. Para el logaritmo, $\partial \log(x) = 1/x$: en 2 es un medio. Para el seno, $\partial \sin(x) = \cos(x)$: en cero es el coseno de cero, que es 1.

Geométricamente, la derivada es la pendiente de la recta tangente que pasa por un punto, o igualmente la mejor aproximación de la función por una recta en ese punto. Esa es la visión que importa: la pendiente dice cómo se comporta la función en las cercanías. Con pendiente

positiva la función sube hacia la derecha y baja hacia la izquierda, en un intervalo suficientemente pequeño; con pendiente negativa, al revés. La figura del libro dibuja $f(x) = x^2 - 1.5x$ con tangentes en dos puntos. Su derivada es $2x - 1.5$: en cero la pendiente es menos 1,5, así que la curva va bajando ahí, y en 2 es más 2,5, así que la curva va subiendo. Esta lectura de la pendiente se trasladará a las funciones de muchas variables.



Baja en cero, sube en dos

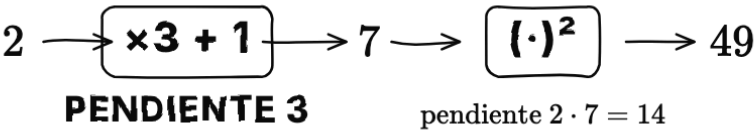
Tres propiedades se extienden a muchas dimensiones. Linealidad: la derivada de una suma es la suma de las derivadas, $\partial[f(x) + g(x)] = f'(x) + g'(x)$; para equis al cuadrado más seno de equis, la derivada es $2x$ más coseno de equis. La regla del producto: $\partial[f(x)g(x)] = f'(x)g(x) + f(x)g'(x)$; con f igual a x y g igual a equis al cuadrado, eso es 1 por

equis al cuadrado más equis por 2x, que es 3 equis al cuadrado, exactamente la derivada de equis al cubo.

$$\partial[f + g] = f' + g'$$

$$\partial[fg] = f'g + fg'$$

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



$$3 \times 14 = 42$$

UNA RED ES UNA LARGA CADENA

Las tasas se multiplican a lo largo de una cadena

Y la regla de la cadena, la que hace funcionar todo este libro:
 $\partial[f(g(x))] = f'(g(x)) g'(x)$. En palabras, cuando una función alimenta a otra, las tasas se multiplican. Tome g igual a 3x más 1 y f igual al cuadrado, en x igual a 2: g da 7 y tiene pendiente 3, el cuadrado tiene pendiente 2 por 7, que es 14, en 7, y el producto es 42. Esa es la cadena de la calculadora: dos, por tres, más uno, al cuadrado; un cambio diminuto en el 2 inicial mueve el 49 final cuarenta y dos veces más. Una red neuronal es una larga cadena de funciones así, y la regla de la cadena es como se encuentran sus pendientes.

Pruébalo

Con una calculadora, calcule 2 por 3 más 1, al cuadrado: 49. Ahora empiece desde 2,01: 2,01 por 3 más 1 es 7,03, y al cuadrado es 49,4209. La salida creció 0,4209 para un paso de 0,01, unas 42 veces el paso, justo como predijo la regla de la cadena al multiplicar 14 por 3.

Para recordar

La derivada $f'(x)$ es el límite del cambio en la salida dividido por un paso h que se encoge, 6 para equis al cuadrado en 3. Siguen las reglas conocidas: x^p da px^{p-1} , $\log x$ da 1 sobre x , el seno da el coseno; la notación ∂ cubre cualquier dimensión y f' es para funciones escalares. La derivada es la pendiente de la tangente, la mejor aproximación por una recta: positiva significa que sube, negativa que baja, como para $x^2 - 1.5x$ en 2 y en 0. Las derivadas son lineales, siguen la regla del producto y obedecen la regla de la cadena, $f'(g(x))$ por $g'(x)$, que multiplica las tasas a lo largo de una cadena, como 14 por 3 da 42.

Preguntas

1. La derivada de $f(x) = x^2 - 1.5x$ es $2x - 1.5$. ¿Qué dice sobre la función cerca de $x = 0$?
 - a. La función va bajando ahí, ya que la pendiente es $-1,5$

- b. La función va subiendo ahí, ya que la pendiente es 1,5
 - c. La función es plana ahí, ya que x es cero
2. ¿Por qué la regla de la cadena es la regla que «hace funcionar todo este libro»?
- a. Porque las redes neuronales no contienen sumas ni productos
 - b. Porque una red es una larga cadena de funciones, y la regla de la cadena multiplica sus tasas para hallar la pendiente del conjunto
 - c. Porque es la única regla válida para funciones con esquinas
3. En la definición de la derivada, ¿qué papel tiene h ?
- a. Es la altura de la función en x
 - b. Es un paso fijo de 0,1 usado en cada cálculo
 - c. Es un paso que se encoge hacia cero, y el cociente entre el cambio en la salida y h se acerca a la derivada

Respuestas

- 1. a)** En 0 la pendiente es $2 \times 0 - 1,5 = -1,5$; una pendiente negativa significa que la función decrece hacia la derecha y crece hacia la izquierda, en un intervalo suficientemente pequeño.
- 2. b)** Cuando una función alimenta a otra, la derivada es $f'(g(x))$ por $g'(x)$; una red es una larga composición, así que sus pendientes salen de multiplicar las tasas a lo largo de la cadena.
- 3. c)** La derivada es el límite de $(f(x + h) - f(x)) / h$ cuando h tiende a cero; para x^2 en 3 los cocientes 6,1 y 6,01 se acercan a 6.

Ejemplos resueltos

La derivada por pasos que se encogen Para $f(x) = x^2$ en $x = 3$, calcule el cociente $(f(x + h) - f(x)) / h$ para $h = 0,1$ y $h = 0,01$, y compárelo con la regla $\partial x^p = px^{p-1}$.

$h = 0,1$: $3,1^2 = 9,61$, así que $(9,61 - 9) / 0,1 = 6,1$.

$h = 0,01$: $3,01^2 = 9,0601$, así que $(9,0601 - 9) / 0,01 = 6,01$.

Los cocientes se acercan a 6 a medida que h se encoge.

La regla de la potencia da $2 \times 3^1 = 6$, el límite.

Respuesta: 6,1, luego 6,01, acercándose a la derivada 6

La regla de la cadena en la cadena de la calculadora Con $g(x) = 3x + 1$ y $f(u) = u^2$, halle la derivada de $f(g(x))$ en $x = 2$.

$g(2) = 7$, y $g'(x) = 3$.

$f'(u) = 2u$, así que $f'(g(2)) = 2 \times 7 = 14$.

Regla de la cadena: $f'(g(2)) \times g'(2) = 14 \times 3 = 42$.

Comprobación: desde 2,01, $(3 \times 2,01 + 1)^2 = 7,03^2 = 49,4209$, un crecimiento de 0,4209 para un paso de 0,01, unas 42 veces.

Respuesta: 42

Ejercicios

1. Use las reglas de la potencia y de la linealidad para hallar la derivada de $x^3 + 4x^2$ en $x = 1$.

Pista: Derive cada término y sume.

Respuesta: $3x^2 + 8x$, que en 1 es $3 + 8 = 11$.

2. Use la regla del producto en $x \cdot \log(x)$ y evalúe la derivada en $x = 1$.

Pista: $f = x$ tiene $f' = 1$; $g = \log x$ tiene $g' = 1/x$.

Respuesta: $1 \times \log(x) + x \times (1/x) = \log(x) + 1$, que en 1 es $0 + 1 = 1$.

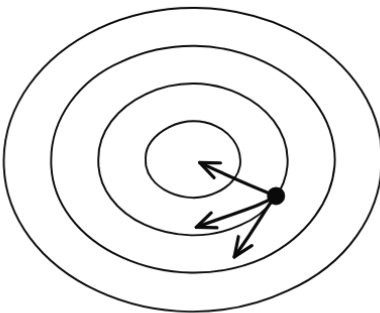
3. Halle la derivada de $\sin(2x)$ en $x = 0$ con la regla de la cadena.

Pista: La función exterior es el seno, la interior $2x$.

Respuesta: $\cos(2x) \times 2$, que en 0 es $\cos(0) \times 2 = 2$.

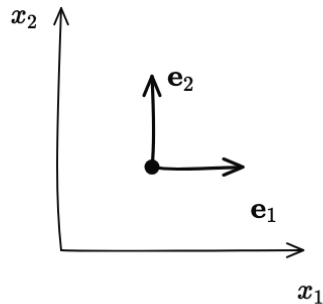
Pendientes en muchas direcciones

Póngase en la ladera de una colina y pregunte qué tan empinada es. La respuesta honesta es: depende de hacia dónde mire. Cuesta arriba en línea recta es empinada, a lo largo de la ladera es plana, y cada dirección intermedia tiene su propia inclinación. Con una función de un número solo había dos direcciones, izquierda y derecha; con una función de muchos números hay infinitas, y esta parte muestra que un solo vector, el gradiente, responde por todas ellas.



— EMPINADO
— PLANO

$$\partial_{x_i} f = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$$

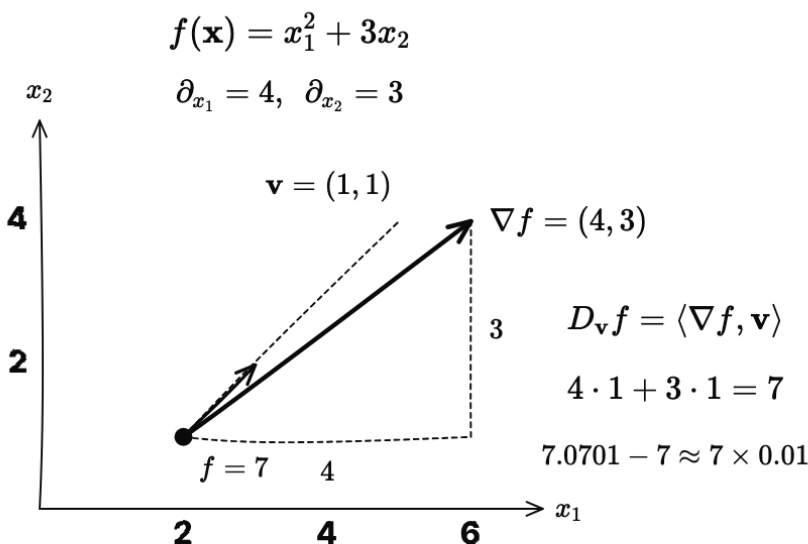


DERIVADA PARCIAL

La inclinación depende de la dirección

Tome una función $y = f(\mathbf{x})$ cuya entrada es un vector $\mathbf{x} \sim (d)$. Un empujoncito no significa nada hasta que decimos hacia dónde. La elección más sencilla es moverse a lo largo de un eje, el i -ésimo, manteniendo fijas todas las demás entradas: $\partial_{x_i} f(\mathbf{x}) = \partial y / \partial x_i = \lim_{h \rightarrow 0} (f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})) / h$. Aquí $\mathbf{e}_i$ es el i -ésimo vector de la base, la fila i de la matriz identidad: su entrada j es 1 cuando j es igual a i y 0 en caso contrario, así que sumar h veces ese vector solo mueve la entrada i . Esto se llama derivada parcial: la derivada ordinaria, tomada a lo largo de un eje.

Usemos una función pequeña todo el tiempo: $f(\mathbf{x}) = x_1^2 + 3x_2$, en el punto $\mathbf{x} = (2, 1)$, donde f es 4 más 3, que es 7. Moviendo solo x_1 , la pendiente es $2x_1$, que es 4; moviendo solo x_2 , la pendiente es 3.



El gradiente responde por cada dirección

Apilar todas las derivadas parciales en una columna da un vector de dimensión d , el gradiente: $\nabla f(\mathbf{x}) = \partial f(\mathbf{x}) = [\partial_{x_1} f(\mathbf{x}), \dots, \partial_{x_d} f(\mathbf{x})]^T$. Como es tan fundamental, tiene su propio símbolo, el triángulo invertido ∇ . Para nuestra función el gradiente en $(2, 1)$ es $(4, 3)$.

¿Y un empujón en una dirección general $\mathbf{v}$? Eso da la derivada direccional, $D_{\mathbf{v}} f(\mathbf{x}) = \lim_{h \rightarrow 0} (f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})) / h$. Como cualquier movimiento puede descomponerse en movimientos separados a lo largo de cada eje, la derivada direccional resulta ser un producto escalar: $D_{\mathbf{v}} f(\mathbf{x}) = \langle \nabla f(\mathbf{x}), \mathbf{v} \rangle = \sum_i \partial_{x_i} f(\mathbf{x}) v_i$. Cada término es la pendiente a lo largo de un eje por cuánto nos movemos a lo largo de ese eje. Para $\mathbf{v} = (1, 1)$: 4 por 1 más 3 por 1, que es 7. Compruébelo directamente: en $(2, 0.1, 1, 0.1)$, f es 4,0401 más 3,03, que es 7,0701, un cambio de 0,0701 para un paso de 0,01, unas 7 veces el paso. Así que conocer el gradiente basta para conocer la pendiente en cada dirección posible, que es la ladera respondida por un solo vector.

Ahora el caso general: una entrada vectorial $\mathbf{x} \sim (d)$ y una salida vectorial $\mathbf{y} \sim (o)$, que el libro dice que es el caso más general que necesitaremos. Con varias salidas tomamos un gradiente por salida, y apilarlos como filas da una matriz de o por d , el jacobiano: $\partial f(\mathbf{x}) \sim (o, d)$, cuya entrada en la fila j y la columna i es $\partial y_j / \partial x_i$. Tome $f(\mathbf{x}) = (x_1 x_2, x_1 + x_2)$ en $(2, 3)$: la primera salida tiene pendientes 3 y 2, la segunda 1 y 1, así que el jacobiano tiene filas $(3, 2)$ y $(1, 1)$. Con una salida, o igual a 1, el jacobiano es el gradiente; con una entrada y una salida es la derivada ordinaria.

$$\partial f(\mathbf{x}) \sim (o, d), \quad [\partial f]_{ji} = \frac{\partial y_j}{\partial x_i}$$

$$f(\mathbf{x}) = (x_1 x_2, x_1 + x_2) \text{ en } (2, 3) : \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix}$$

O = 1: EL GRADIENTE

$$\partial[f(g(\mathbf{x}))] = [\partial f] \partial g$$

$$g(3, 1) = (4, 2), \quad \partial f = (2, 4)$$

$$(2 \ 4) \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = (6, -2)$$

$$\nabla(x_1^2 - x_2^2) \text{ en } (3, 1) = (6, -2)$$

Los jacobianos se multiplican

Los jacobianos heredan las propiedades de las derivadas, y sobre todo la regla de la cadena se convierte en un producto de matrices:

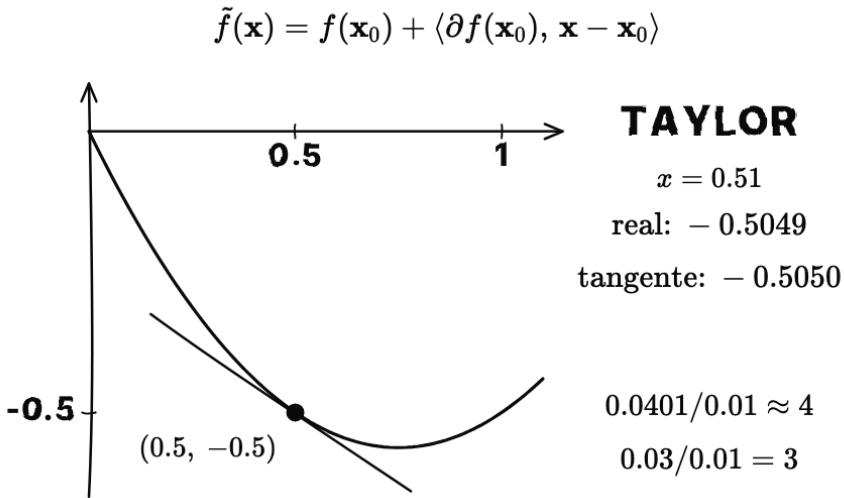
$\partial[f(g(\mathbf{x}))] = [\partial f(\bullet)] \partial g(\mathbf{x})$, con el primero evaluado en $g(\mathbf{x}) \sim (h)$.

Compruébelo con $g(\mathbf{x}) = (x_1 + x_2, x_1 - x_2)$ y $f(\mathbf{u}) = u_1 u_2$, en $\mathbf{x}$ igual a $(3, 1)$. g da $\mathbf{u}$ igual a $(4, 2)$, con jacobiano de filas $(1, 1)$ y $(1, -1)$; f tiene gradiente (u_2, u_1) , que ahí es $(2, 4)$. Multiplicando, 2 por 1 más 4 por 1 es 6, y 2 por 1 más 4 por menos 1 es menos 2: la fila $(6, -2)$.

Directamente, f de g es $x_1^2 - x_2^2$, cuyo gradiente en $(3, 1)$ es también $(6, -2)$.

Como en una dimensión, gradientes y jacobianos son funciones lineales tangentes en un punto. El gradiente da la mejor aproximación de primer orden cerca de $\mathbf{x}_0$: $\tilde{f}(\mathbf{x}) = f(\mathbf{x}_0) + \langle \partial f(\mathbf{x}_0), \mathbf{x} - \mathbf{x}_0 \rangle$, el valor en

el punto más la pendiente por el desplazamiento. Este es el teorema de Taylor. El recuadro de código del libro lo comprueba para $f(x) = x^2 - 1.5x$ en 0,5, donde f es menos 0,5 y la pendiente es menos 0,5: un paso de 0,01 da un valor verdadero de menos 0,5049 y una aproximación de menos 0,5050, casi idénticos.



La aproximación tangente

Pruébalo

En una calculadora, evalúe equis uno al cuadrado más 3 equis dos en $(2, 1)$, que da 7, y luego en $(2,01, 1)$, que da 7,0401, y en $(2, 1,01)$, que da 7,03. Los dos cambios, 0,0401 y 0,03, divididos por el paso 0,01, son aproximadamente 4 y 3, las derivadas parciales. Ahora dé el paso en ambas a la vez hasta $(2,01, 1,01)$: 7,0701, un cambio de unas 7 veces el paso, la suma que predijo el gradiente.

Para recordar

Para una entrada vectorial, una derivada parcial mueve una entrada a lo largo de su eje; apilarlas todas da el gradiente $\nabla f(\mathbf{x})$, $(4, 3)$ para $x_1^2 + 3x_2$ en $(2, 1)$. La derivada direccional a lo largo de cualquier $\mathbf{v}$ es el producto escalar del gradiente con $\mathbf{v}$, así que el gradiente da la pendiente en cada dirección. Para una salida vectorial, un gradiente por salida se apila en el jacobiano, una matriz de $\mathbf{o}$ por $\mathbf{d}$, y la regla de la cadena se convierte en un producto de jacobianos, como mostró $(6, -2)$ de las dos maneras. Cerca de un punto, f se aproxima por su valor más el gradiente multiplicado escalarmente por el desplazamiento, el teorema de Taylor, comprobado como $-0,5049$ frente a $-0,5050$.

Preguntas

1. ¿Por qué basta conocer el gradiente para hallar la pendiente de una función en cualquier dirección?
 - a. Porque la derivada direccional a lo largo de v es el producto escalar del gradiente con v
 - b. Porque una función de muchas entradas tiene una sola dirección que importa
 - c. Porque el gradiente enumera las pendientes en cada dirección posible una por una
2. Una función toma un vector de tamaño 5 y devuelve un vector de tamaño 3. ¿Cuál es la forma de su jacobiano?
 - a. (5, 3)
 - b. (3, 5)
 - c. (15)
3. ¿Cómo es la regla de la cadena para jacobianos?
 - a. Los jacobianos de las dos funciones se suman
 - b. Los jacobianos se multiplican entrada a entrada
 - c. El jacobiano de la función exterior, evaluado en $g(x)$, se multiplica como matriz por el jacobiano de g
4. ¿Qué dice el teorema de Taylor, en su forma de primer orden, sobre f cerca de un punto x_0 ?
 - a. Que f es igual a su valor en x_0 en todo el entorno
 - b. Que f se aproxima mejor por su valor en x_0 más el gradiente multiplicado escalarmente por el desplazamiento
 - c. Que f crece como el cuadrado de la distancia a x_0

Respuestas

1. a) Cualquier movimiento se descompone en movimientos a lo largo de cada eje, así que la derivada direccional es la suma de cada derivada parcial por el movimiento a lo largo de ese eje, el producto escalar $\langle \nabla f(x), v \rangle$.

2. b) El jacobiano apila un gradiente por salida como filas, así que tiene $o = 3$ filas y $d = 5$ columnas.

3. c) La derivada de una composición se convierte en un producto de matrices, ∂f evaluado en $g(x)$ por $\partial g(x)$, como mostró la fila $(6, -2)$ para $u_1 u_2$ tras $(x_1 + x_2, x_1 - x_2)$.

4. b) La mejor aproximación lineal es $f(x_0) + \langle \partial f(x_0), x - x_0 \rangle$, que en el ejemplo del libro da $-0,5050$ frente al verdadero $-0,5049$.

Ejemplos resueltos

La linealización del libro Para $f(x) = x^2 - 1.5x$, aproxime $f(0,51)$ con la tangente en $0,5$ y compárelo con el valor verdadero.

$$f(0,5) = 0,25 - 0,75 = -0,5.$$

$$f'(x) = 2x - 1,5, \text{ así que } f'(0,5) = -0,5.$$

$$\text{Aproximación: } -0,5 + (-0,5) \times 0,01 = -0,505.$$

$$\text{Valor verdadero: } 0,51^2 - 1,5 \times 0,51 = 0,2601 - 0,765 = -0,5049.$$

Respuesta: $-0,5050$ aproximado frente a $-0,5049$ verdadero

Un gradiente y una derivada direccional Para $f(x) = x_1^2 + 3x_2$ en $(2, 1)$, halle el gradiente y la derivada direccional a lo largo de $v = (1, 1)$.

$$\text{Parcial a lo largo de } x_1: 2x_1 = 4.$$

$$\text{Parcial a lo largo de } x_2: 3.$$

$$\text{Gradiente: } (4, 3).$$

A lo largo de v : $4 \times 1 + 3 \times 1 = 7$; comprobación: $f(2,01, 1,01) = 7,0701$, un cambio de $0,0701$ para un paso de $0,01$.

Respuesta: $\nabla f = (4, 3)$; derivada direccional 7

Ejercicios

1. Para $f(x) = x_1x_2 + x_2^2$ en $(1, 2)$, halle el gradiente y la derivada direccional a lo largo de $v = (1, -1)$.

Pista: Derive respecto a x_1 manteniendo x_2 fijo, y luego al revés.

Respuesta: $\partial/\partial x_1 = x_2 = 2$ y $\partial/\partial x_2 = x_1 + 2x_2 = 5$, así que $\nabla f = (2, 5)$; a lo largo de $(1, -1)$: $2 - 5 = -3$.

2. Halle el jacobiano de $f(x) = (3x_1, x_1x_2)$ en $(1, 4)$.

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

Respuesta: Fila 1: $(3, 0)$; fila 2: $(x_2, x_1) = (4, 1)$. El jacobiano tiene filas $(3, 0)$ y $(4, 1)$, forma $(2, 2)$.

3. Con $f(x) = x_1^2 + 3x_2$ en $x_0 = (2, 1)$, use la aproximación de primer orden para estimar $f(2,1, 0,9)$, y luego calcule el valor verdadero.

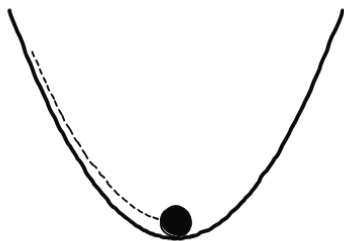
Pista: $f(x_0)$ más el gradiente $(4, 3)$ multiplicado escalarmente por el desplazamiento.

Respuesta: Desplazamiento $(0,1, -0,1)$: $7 + 4 \times 0,1 + 3 \times (-0,1) = 7,1$.

Verdadero: $2,1^2 + 3 \times 0,9 = 4,41 + 2,7 = 7,11$.

Tamaños ocultos, y el primer paso cuesta abajo

Ponga una canica en una ensaladera curva y suéltela. No sabe dónde está el fondo, y sin embargo llega: en cada instante rueda hacia donde se inclina el cuenco bajo ella. Entrenar un modelo es esa canica, y esta parte establece la regla que sigue, tras una advertencia breve pero importante sobre los tamaños de los jacobianos.



ENTRENAMIENTO

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

$$\partial_{\mathbf{x}}[\mathbf{W}\mathbf{x}] = \mathbf{W}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 2 & \boxed{0} \\ 0 & 1 & \boxed{4} \end{pmatrix}$$

**MOVER $x_3 \rightarrow$
SALIDAS 0 Y 4**

Una canica en un cuenco; $\mathbf{W}$ como tasa

El libro llama pedante a la advertencia, pero importa más adelante. Tome $\mathbf{y} = \mathbf{W}\mathbf{x}$, una matriz por un vector, el núcleo de cada capa de

una red. Vista como función de $\mathbf{x}$, su derivada es, como antes, una matriz de o por d , y es simplemente $\partial_{\mathbf{x}}[\mathbf{W}\mathbf{x}] = \mathbf{W}$. En palabras, la tasa a la que cambia la salida con la entrada es la propia matriz, igual que la derivada de $3x$ es 3 . Para $\mathbf{W}$ de filas $(1, 2, 0)$ y $(0, 1, 4)$, mover la tercera entrada un pasito mueve las salidas 0 y 4 veces ese paso: la tercera columna de $\mathbf{W}$.

Vista como función de $\mathbf{W}$, en cambio, la entrada es ella misma una matriz de o por d , y el «jacobiano» tiene entonces forma (o, o, d) , una tasa por cada salida y cada entrada de $\mathbf{W}$. Siempre podemos pasar a una función equivalente que toma la versión aplanada de $\mathbf{W}$, $\text{vect}(\mathbf{W}) \sim (od)$, un único vector largo, y entonces el jacobiano es una matriz ordinaria de forma (o, od) . Para nuestro ejemplo de 2 por 3 , la forma es 2 por 2 por 3 , que son 12 números, o 2 por 6 tras aplanar, los mismos 12 .

$$(o, o, d) = (2, 2, 3) \quad \text{o} \quad (o, od) = (2, 6)$$

$$y_i = \sum_j W_{ij} x_j$$

$$\partial y_i / \partial W_{ij} = x_j$$

1	2	3
---	---	---

0	0	0
---	---	---

SALIDA 1

0	0	0
---	---	---

1	2	3
---	---	---

SALIDA 2

$$1000 \times 1000 \times 1000 = 10^9$$

NO SE GUARDA: PRODUCTOS VECTOR-JACOBIANO

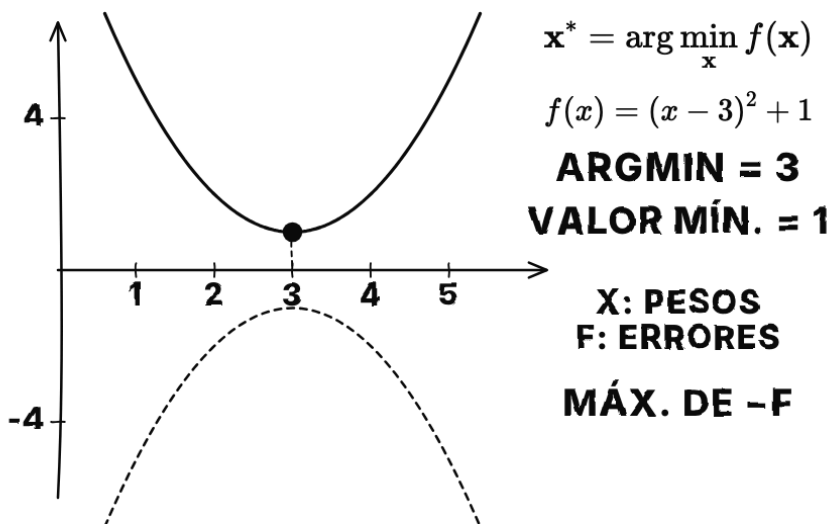
El jacobiano respecto a W

El recuadro del libro calcula cuáles son esos números. Escriba el producto entrada a entrada: $y_i = \sum_j W_{ij} x_j$. Entonces, de inmediato, $\partial y_i / \partial W_{ij} = x_j$: mover un peso de la fila i cambia la salida i en el valor de entrada de ese peso, y deja intactas todas las demás salidas. Con $x = (1, 2, 3)$, la salida 1 se mueve 1, 2 o 3 por cada unidad de empujón de los pesos de la fila 1, y no se mueve en absoluto con los pesos de la fila 2. Así que de nuestros 12 números, 6 son cero y los otros 6 son simplemente x copiado dos veces.

Ese es el punto: guardar el jacobiano explícitamente significaría conservar una gran cantidad de valores repetidos. Para una sola capa con 1000 entradas y 1000 salidas, W contiene un millón de pesos, y su jacobiano contendría 1000 por 1000 por 1000, mil millones de

números, en su mayoría ceros y copias. Como mostrará el capítulo 6, nunca necesitamos construirlo, porque en la práctica solo aplicamos el jacobiano a otro tensor, mediante los llamados productos vector-jacobiano, una visión que también puede formalizarse tratando los jacobianos como aplicaciones lineales abstractas. El libro mantiene la notación vectorial porque basta, pero advierte de que en el capítulo 6 los jacobianos matriciales esconderán sus verdaderos tamaños dentro de las formas de las entradas y las salidas, sobre todo los tamaños de lote.

Ahora la razón por la que los gradientes merecen todo esto: la sección 2.3, el descenso de gradiente. La tarea es minimizar una función $f(\mathbf{x})$ con $\mathbf{x} \sim (d)$: $\mathbf{x}^* = \operatorname{argmin} f(\mathbf{x})$. Como argmax , argmin significa «el valor de $\mathbf{x}$ que da la f más baja posible». Para $f(x) = (x - 3)^2 + 1$, el argmin es 3, donde f vale 1; el valor mínimo es 1, pero el argmin es el lugar, 3. El libro supone una sola salida, optimización con un solo objetivo, y ninguna restricción sobre adónde puede ir $\mathbf{x}$.



El argmin es un lugar

En el resto del libro, $\mathbf{x}$ contendrá los parámetros de un modelo y f medirá lo mal que lo hace el modelo con los datos, el planteamiento del aprendizaje supervisado del próximo capítulo. Para la aplicación de fotos de su teléfono, $\mathbf{x}$ son sus millones de pesos y f cuenta sus errores en fotos etiquetadas. Minimizar no pierde nada frente a maximizar, ya que minimizar f es maximizar menos f : dé la vuelta a una curva respecto al eje horizontal y sus valles se convierten en picos.

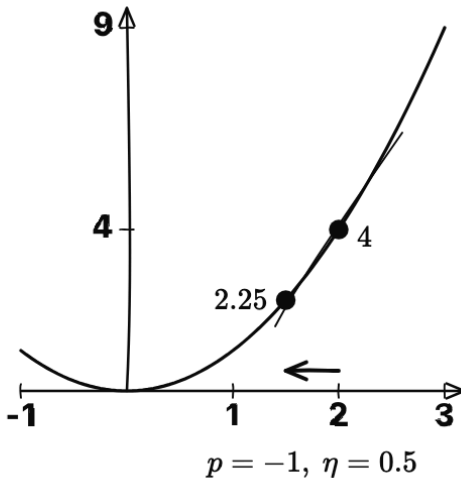
Rara vez el mínimo puede escribirse como una fórmula, en forma cerrada; el libro encontrará un caso así con los mínimos cuadrados en la sección 4.1.2. Normalmente debemos iterar. Partimos de una suposición al azar $\mathbf{x}_0$ y en cada iteración damos un paso hecho de una longitud η_t y una dirección $\mathbf{p}_t$: $\mathbf{x}_t = \mathbf{x}_{t-1} + \eta_t \mathbf{p}_t$. En palabras, la nueva

suposición es la vieja suposición más un desplazamiento. η_t se llama tamaño de paso o, en aprendizaje automático, tasa de aprendizaje.

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \eta_t \mathbf{p}_t$$

η : tasa de aprendizaje

$\mathbf{p}$: dirección



$$f(\mathbf{x}_t) \leq f(\mathbf{x}_{t-1})$$

$$D_{\mathbf{p}} f \leq 0 : -4$$

Un paso cuesta abajo

Una dirección es una dirección de descenso si alguna longitud de paso a lo largo de ella no hace mayor a f : $f(\mathbf{x}_t) \leq f(\mathbf{x}_{t-1})$. Si podemos elegir una dirección de descenso en cada iteración y escoger los tamaños de paso con cuidado, el procedimiento alcanza un mínimo, en un sentido que el libro precisa enseguida. Para funciones diferenciables la derivada direccional nombra exactamente cada dirección así: $\mathbf{p}_t$ de descenso $\Rightarrow D_{\mathbf{p}_t} f(\mathbf{x}_{t-1}) \leq 0$, una dirección en la que f no aumenta, al menos a primer orden. Pruebe $f(x) = x^2$ desde 2: la derivada es 4, así que la dirección menos 1 tiene derivada direccional menos 4, que es

negativa, y un paso de 0,5 a lo largo de ella cae en 1,5, donde f baja de 4 a 2,25.

Pruébalo

Dibuje la curva de equis al cuadrado en papel y ponga una moneda sobre ella en x igual a 2. Mueva la moneda media unidad a la izquierda, la dirección donde la pendiente dice que la curva baja, y lea la altura: 2,25 en vez de 4. Muévela en cambio a la derecha y la altura crece hasta 6,25; el signo de la pendiente le dijo de antemano hacia dónde estaba la bajada.

Para recordar

Para $y = Wx$, la derivada respecto a x es simplemente W , una matriz de o por d . Respecto a W el jacobiano tiene forma (o, o, d) , o (o, od) tras aplanar, con $\partial y_i / \partial W_{i,j} = x_j$: casi todo ceros y copias, así que el capítulo 6 evita guardarlo. La optimización busca $x^* = \operatorname{argmin} f(x)$; en el libro x contiene los parámetros de un modelo y f su error, y minimizar es lo mismo que maximizar $-f$. Los métodos iterativos actualizan $x_t = x_{t-1} + \eta_t p_t$ con una tasa de aprendizaje η_t , y una dirección de descenso es una con derivada direccional no positiva, como mostró el paso de 2 a 1,5 en x^2 .

Preguntas

1. Para $y = Wx$ con W de forma (o, d) , ¿por qué el jacobiano respecto a W nunca se guarda en la práctica?
 - a. Porque siempre es la matriz identidad
 - b. Porque tiene forma (o, o, d) , casi todo ceros y copias de x , y solo necesitamos aplicarlo a otro tensor
 - c. Porque no puede calcularse para entradas matriciales
2. ¿Cuál es la diferencia entre el mínimo de $f(x) = (x - 3)^2 + 1$ y su argmin?
 - a. El mínimo es 1, el valor más bajo; el argmin es 3, el lugar donde se alcanza
 - b. El mínimo es 3 y el argmin es 1
 - c. Son el mismo número
3. ¿Por qué el libro no pierde nada hablando siempre de minimizar?
 - a. Porque ninguna función útil tiene máximo
 - b. Porque el descenso de gradiente no puede encontrar máximos
 - c. Porque maximizar f es lo mismo que minimizar $-f$
4. ¿Cuándo es una dirección p una dirección de descenso para una f diferenciable en la suposición actual?
 - a. Cuando apunta en sentido contrario a la suposición inicial x_0
 - b. Cuando la derivada direccional de f a lo largo de p es como mucho cero
 - c. Cuando su longitud es igual a la tasa de aprendizaje

Respuestas

1. **b)** Como $\partial y_i / \partial W_{ij} = x_j$, el jacobiano repite x y por lo demás es cero; para una capa de 1000 por 1000 contendría mil millones de números, y el capítulo 6 lo evita con productos vector-jacobiano.

2. **a)** argmin nombra la entrada que da el valor más bajo de f ; aquí f es mínima, igual a 1, en $x = 3$.

3. **c)** Dar la vuelta a una función respecto al eje horizontal convierte sus picos en valles, así que cualquier maximización puede escribirse como una minimización.

4. **b)** Una dirección de descenso es una a lo largo de la cual f no aumenta; para funciones diferenciables esto significa $D_a f \leq 0$, como la dirección -1 para x^2 en 2, con derivada direccional -4 .

Ejemplos resueltos

El jacobiano de Wx respecto a W W tiene forma $(2, 3)$ y $x = (1, 2, 3)$. Usando $\partial y_i / \partial W_{ij} = x_j$, dé la forma del jacobiano de $y = Wx$ respecto a W y diga cuántas de sus entradas son cero.

$o = 2$ y $d = 3$, así que el jacobiano tiene forma $(o, o, d) = (2, 2, 3)$, 12 entradas.

La salida y_1 depende solo de la fila 1 de W : sus tasas son $x_1, x_2, x_3 = 1, 2, 3$, y 0 para los tres pesos de la fila 2.

Igualmente y_2 tiene tasas 1, 2, 3 para la fila 2 y 0 para la fila 1.

Así que 6 entradas son cero, y las otras 6 son x copiado dos veces.

Respuesta: Forma $(2, 2, 3)$, o $(2, 6)$ tras aplanar; 6 de las 12 entradas son cero.

Ejercicios

1. Para W de filas $(1, 2, 0)$ y $(0, 1, 4)$, ¿cuál es la derivada de Wx respecto a x , y cómo cambian las salidas si solo x_2 se mueve 0,1?

Pista: La derivada respecto a x es la propia W ; mire su segunda columna.

Respuesta: Es W . Mover x_2 en $0,1$ mueve las salidas $2 \times 0,1 = 0,2$ y $1 \times 0,1 = 0,1$, la segunda columna de W por el paso.

2. $f(x) = x^2$ y la suposición actual es 3 . Tome la dirección $p = -1$ con tamaño de paso $\eta = 0,5$. ¿Es una dirección de descenso, y cuáles son la nueva suposición y el nuevo valor de f ?

Pista: Compruebe el signo de $f'(3) \times p$, luego aplique $x_t = x_{t-1} + \eta p$.

Respuesta: $f'(3) = 6$, así que la derivada direccional es $6 \times (-1) = -6 \leq 0$: una dirección de descenso. Nueva suposición $3 + 0,5 \times (-1) = 2,5$, y f baja de 9 a $6,25$.

3. Una capa tiene 500 entradas y 200 salidas. ¿Cuántos números contendría su jacobiano respecto a W si se guardara explícitamente?

Pista: La forma es (o, o, d) .

Respuesta: $200 \times 200 \times 500 = 20\,000\,000$ números, aunque la propia W tiene solo $100\,000$ pesos.

El descenso más pronunciado, y dónde termina

Vuelva a la ladera y mire cuesta arriba, hacia donde la pendiente es más pronunciada. Ahora dese la vuelta: mira directamente hacia abajo, por el camino más empinado. Gire un cuarto de vuelta en cambio y camina en llano, ni arriba ni abajo; cualquier cosa entre los cuartos de vuelta del lado de bajada le lleva más abajo. Esta brújula sencilla, construida con el gradiente y el coseno, es todo el descenso de gradiente.

$$D_{\mathbf{p}}f = \|\nabla f\| \|\mathbf{p}\| \cos \alpha$$



$$\nabla f(3, 4) = (6, 8), \quad \|\nabla f\| = 10$$

$$(-0.6, -0.8) : -10$$

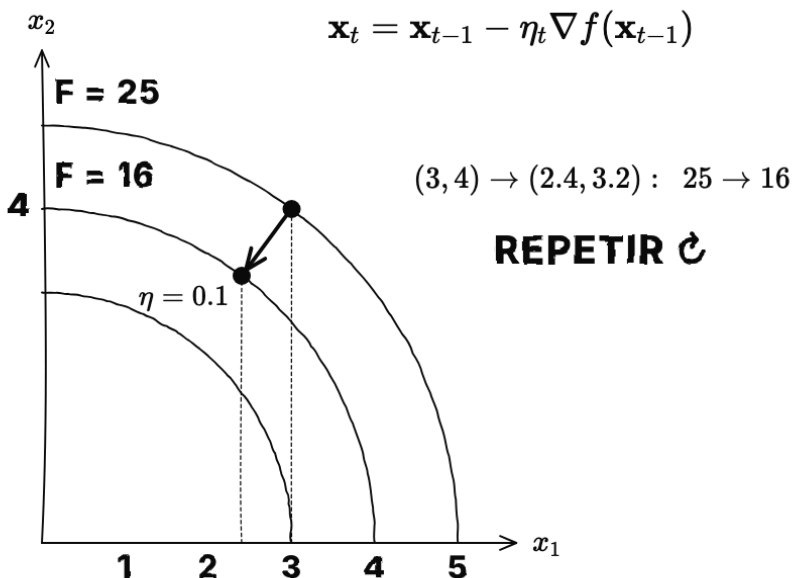
$$(-0.8, 0.6) : 0$$

Una brújula hecha con el gradiente

Combine la derivada direccional con la forma en coseno del producto escalar: $D_{\mathbf{p}_t} f(\mathbf{x}_{t-1}) = \langle \nabla f(\mathbf{x}_{t-1}), \mathbf{p}_t \rangle = \|\nabla f(\mathbf{x}_{t-1})\| \|\mathbf{p}_t\| \cos(\alpha)$, donde α es el ángulo entre la dirección del paso y el gradiente. La primera longitud no depende de la dirección que elijamos, y como la dirección solo dice hacia dónde ir, podemos fijar su longitud en uno. Así que todo depende del coseno.

El coseno es negativo para ángulos entre $\pi/2$ y $3\pi/2$, un cuarto de vuelta y tres cuartos de vuelta, así que cada dirección en ese rango es una dirección de descenso. Entre ellas, la de ángulo π , que apunta exactamente contra el gradiente, $\mathbf{p}_t = -\nabla f(\mathbf{x}_{t-1})$, da la derivada direccional más negativa, y se llama dirección de descenso más pronunciado. Tome $f(\mathbf{x}) = x_1^2 + x_2^2$ en $(3, 4)$: el gradiente es $(6, 8)$, de longitud 10. La dirección unitaria en su contra, $(-0,6, -0,8)$, da una derivada direccional de menos 10, la más pronunciada posible; la dirección lateral $(-0,8, 0,6)$ da menos 4,8 más 4,8, que es cero: caminar en llano.

Poner esto en el paso iterativo da un algoritmo capaz de minimizar cualquier función diferenciable, el descenso de gradiente más pronunciado. Dado un punto de partida $\mathbf{x}_0$ y una sucesión de tamaños de paso η_t , repita $\mathbf{x}_t = \mathbf{x}_{t-1} - \eta_t \nabla f(\mathbf{x}_{t-1})$. En palabras: desde donde está, dé un pasito contra el gradiente. Con nuestra función, un tamaño de paso de 0,1 desde $(3, 4)$ lleva a 3 menos 0,6 y 4 menos 0,8, es decir $(2,4, 3,2)$, y f baja de 25 a 5,76 más 10,24, que es 16.



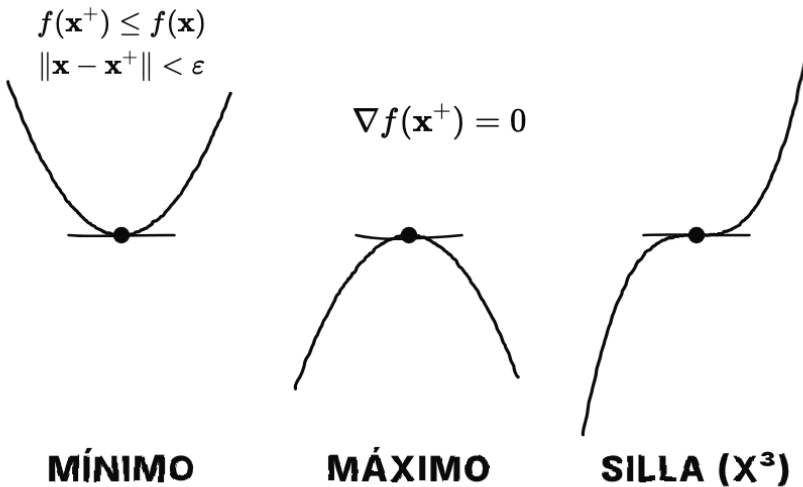
Un paso de descenso de gradiente

El libro deja de lado cómo elegir el tamaño de paso y simplemente supone que es lo bastante pequeño para que cada paso reduzca f . También señala que el descenso de gradiente es solo tan rápido como nuestra manera de calcular el gradiente, y por eso el capítulo 6 presenta un algoritmo general para ello. Cuando se entrenó su aplicación de fotos, este fue el bucle, repetido un número enorme de veces: calcular el gradiente del error respecto a millones de pesos, dar un pasito en su contra.

¿Dónde acaba el descenso de gradiente? Primero debemos decir qué es un mínimo. Un mínimo local es un punto $\mathbf{x}^+$ tal que, para algún radio pequeño $\varepsilon > 0$, $f(\mathbf{x}^+) \leq f(\mathbf{x})$ para todo $\mathbf{x}$ con $\|\mathbf{x} - \mathbf{x}^+\| < \varepsilon$. En palabras: dentro de una bola suficientemente pequeña a su alrededor,

nada está más bajo. Intuitivamente, en un punto así la tangente es plana y el gradiente a su alrededor apunta cuesta arriba, alejándose de él.

EL DG LLEGA A UN PUNTO ESTACIONARIO

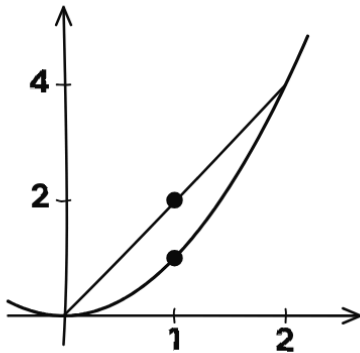


Puntos estacionarios

La idea de la tangente plana recibe su propio nombre: un punto estacionario es uno donde $\nabla f(\mathbf{x}^+) = 0$. Pero los puntos estacionarios no son solo mínimos: pueden ser máximos, que son mínimos de menos f , o puntos de silla, donde cambia la curvatura de la función. La figura del libro muestra una silla; en una dimensión, equis al cubo en cero es el caso clásico: su pendiente, 3 equis al cuadrado, es cero ahí, y sin embargo la curva sigue subiendo por ambos lados. Sin supuestos sobre f , solo puede demostrarse que el descenso de gradiente alcanza algún punto estacionario, cuál dependiendo de dónde empieza.

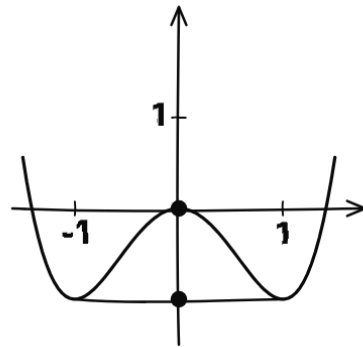
¿Podemos hacerlo mejor? Imagine una parábola. No tiene puntos de silla y tiene un único mínimo, y ese mínimo es especial: el valor más bajo en todo el dominio. Un mínimo global es un punto $\mathbf{x}^*$ con $f(\mathbf{x}^*) \leq f(\mathbf{x})$ para cualquier $\mathbf{x}$. En una parábola el descenso de gradiente lo alcanza desde cualquier punto de partida, porque todo gradiente apunta hacia él.

$$f(\alpha \mathbf{x}_1 + (1 - \alpha) \mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1 - \alpha) f(\mathbf{x}_2)$$



$$1 < 2 \checkmark$$

MÍNIMO GLOBAL EN 0



$$0 > -1 \times$$

DOS VALLES

Por debajo de toda cuerda

La convexidad generaliza la parábola. Una función es convexa si para dos puntos cualesquiera $\mathbf{x}_1$ y $\mathbf{x}_2$ y cualquier $\alpha \in [0, 1]$, $f(\alpha \mathbf{x}_1 + (1 - \alpha) \mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1 - \alpha) f(\mathbf{x}_2)$. En palabras: en el tramo entre dos puntos, la función nunca sube por encima de la cuerda recta que une sus valores. Para equis al cuadrado con los puntos 0 y 2 y alfa igual a un medio: el punto medio es 1, donde f vale 1, mientras que el punto

medio de la cuerda es la media de 0 y 4, que es 2; 1 está por debajo de 2, como exige la convexidad. Para $x^4 - 2x^2$, con los puntos menos 1 y 1, el punto medio 0 da 0, pero la media de la cuerda entre menos 1 y menos 1 es menos 1: la curva asoma por encima de la cuerda, así que no es convexa, y en efecto tiene dos valles separados.

Pruébalo

Dibuje una curva en forma de cuenco, equis al cuadrado, y marque dos puntos en ella, en 0 y en 2. Una sus alturas con una línea recta de regla: cada punto de la curva entre ellos queda bajo la regla. Ahora dibuje una curva en forma de W con dos hondonadas y ponga la regla sobre ambas: el bulto del medio sobresale por encima, que es exactamente lo que la convexidad prohíbe.

Para recordar

La derivada direccional es igual a la longitud del gradiente por la longitud del paso por $\cos \alpha$, así que toda dirección a más de un cuarto de vuelta del gradiente va cuesta abajo, y la más pronunciada es menos el gradiente. El descenso de gradiente repite $x_t = x_{t-1} - \eta_t \nabla f(x_{t-1})$: para $x_1^2 + x_2^2$ desde $(3, 4)$ con paso 0,1 llega a $(2,4, 3,2)$, y f baja de 25 a 16. Un mínimo local es el más bajo en una bola pequeña; un punto estacionario tiene gradiente cero y puede ser un mínimo, un máximo o una silla, y en general el descenso de gradiente solo alcanza algún punto estacionario. Un mínimo global es el más bajo en todas partes; las funciones convexas, que quedan por debajo de toda cuerda, generalizan la parábola, en la que el descenso de gradiente alcanza el mínimo global desde cualquier sitio.

Preguntas

1. ¿Por qué menos el gradiente se llama la dirección de descenso más pronunciado?
 - a. Porque es la única dirección en la que f decrece
 - b. Porque a un ángulo de π con el gradiente el coseno es -1 , lo que da la derivada direccional más negativa
 - c. Porque siempre lleva directamente al mínimo global
2. El descenso de gradiente se detiene en un punto donde el gradiente es cero. ¿Qué puede decir de ese punto en general?

- a. Es un punto estacionario, que podría ser un mínimo, un máximo o un punto de silla
 - b. Es siempre el mínimo global
 - c. Es siempre un mínimo local
3. ¿Qué garantiza la convexidad que hace que el descenso de gradiente se comporte como en una parábola?
- a. Que el tamaño de paso puede elegirse libremente
 - b. Que el gradiente nunca es cero
 - c. Que la función nunca sube por encima de la cuerda entre dos puntos, lo que descarta valles separados y sillas
4. Una dirección está exactamente en ángulo recto con el gradiente. ¿Qué le hace a f un pequeño paso a lo largo de ella, a primer orden?
- a. Disminuye f lo más rápido posible
 - b. Deja f sin cambios, ya que el coseno de un ángulo recto es cero
 - c. Aumenta f lo más rápido posible

Respuestas

- 1. b)** Con direcciones de longitud unidad, la derivada direccional es la longitud del gradiente por $\cos \alpha$; todo ángulo entre $\pi/2$ y $3\pi/2$ va cuesta abajo, pero $\cos \alpha$ es mínimo, -1 , en $\alpha = \pi$.
- 2. a)** Un gradiente cero define un punto estacionario, y sin supuestos sobre f solo puede demostrarse que el descenso de gradiente alcanza algún punto estacionario, según dónde empezó.
- 3. c)** Una función convexa queda por debajo de toda cuerda, generalizando la parábola, cuyo único mínimo es global y hacia el cual apuntan todos los

gradientes.

4. b) La derivada direccional es proporcional a $\cos \alpha$, que es 0 en un cuarto de vuelta, como $(-0,8, 0,6)$ contra el gradiente $(6, 8)$ dio $-4,8 + 4,8 = 0$.

Ejemplos resueltos

Un paso de descenso de gradiente Minimice $f(x) = x_1^2 + x_2^2$ partiendo de $(3, 4)$ con tamaño de paso 0,1. Halle el gradiente, el nuevo punto y el nuevo valor de f .

El gradiente es $(2x_1, 2x_2) = (6, 8)$ en $(3, 4)$.

Actualización: $(3, 4) - 0,1 \times (6, 8) = (3 - 0,6, 4 - 0,8) = (2,4, 3,2)$.

Valor anterior: $9 + 16 = 25$.

Valor nuevo: $2,4^2 + 3,2^2 = 5,76 + 10,24 = 16$.

Respuesta: Nuevo punto $(2,4, 3,2)$; f baja de 25 a 16

Probar la convexidad con una cuerda Compruebe la desigualdad de convexidad para $f(x) = x^4 - 2x^2$ con $x_1 = -1$, $x_2 = 1$ y $\alpha = 0,5$.

El punto mixto es $0,5 \times (-1) + 0,5 \times 1 = 0$, y $f(0) = 0$.

$f(-1) = 1 - 2 = -1$ y $f(1) = -1$, así que el valor de la cuerda es $0,5 \times (-1) + 0,5 \times (-1) = -1$.

La convexidad exigiría $0 \leq -1$, que es falso.

Respuesta: La desigualdad falla, así que $x^4 - 2x^2$ no es convexa; tiene dos valles separados.

Ejercicios

1. Para $f(x) = x^2$ partiendo de 5 con tamaño de paso 0,25, calcule dos pasos de descenso de gradiente.

Pista: El gradiente es $2x$; actualice $x \leftarrow x - 0,25 \times 2x$.

Respuesta: Paso 1: $5 - 0,25 \times 10 = 2,5$. Paso 2: $2,5 - 0,25 \times 5 = 1,25$.
Cada paso reduce x a la mitad, camino del mínimo en 0.

2. ¿Es $x = 0$ un punto estacionario de $f(x) = x^3$? ¿Es un mínimo?

Pista: Calcule la derivada en 0, luego mire f a cada lado.

Respuesta: $f'(x) = 3x^2$, que es 0 en 0, así que es estacionario. No es un mínimo: f es negativa justo a la izquierda y positiva a la derecha, así que es un punto donde la curva se aplana y sigue subiendo, como una silla en una dimensión.

3. Compruebe la desigualdad de convexidad para $f(x) = x^2$ con $x_1 = 1$, $x_2 = 3$ y $\alpha = 0,25$.

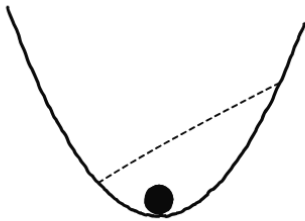
Pista: Compare f en $0,25 \times 1 + 0,75 \times 3$ con $0,25 f(1) + 0,75 f(3)$.

Respuesta: El punto es $0,25 + 2,25 = 2,5$, y $f(2,5) = 6,25$. La cuerda da $0,25 \times 1 + 0,75 \times 9 = 7$. Como $6,25 \leq 7$, la desigualdad se cumple, como exige la convexidad.

Cuencos, paisajes accidentados y un empujón de momento

Haga rodar una bola en una ensaladera lisa y siempre acabará en el fondo, la deje caer donde la deje caer. Hágala rodar por una huevera y se asentará en el hueco que quede más cerca. Y dele a la bola un poco de peso, para que conserve parte de su velocidad, y rodará más suave y más rápido hasta el fondo. Esas tres imágenes son esta parte: lo que garantiza la convexidad, lo que perdemos sin ella y cómo el momento acelera las cosas.

CONVEXA



**TODO INICIO → MÍN.
ESTRICTA → ÚNICO**

NO CONVEXA



**SOLO UN PUNTO
ESTACIONARIO**

La desigualdad de convexidad compara dos cosas. A la izquierda está el valor de f en cualquier punto del tramo de x_1 a x_2 ; a la derecha está el punto correspondiente de la recta que une $f(x_1)$ y $f(x_2)$. Si la función queda siempre por debajo de cada recta así, es convexa; una parábola que se abre hacia arriba es el ejemplo estándar, la ensaladera.

La convexidad nos dice lo fácil que es la optimización. Para una función general no convexa, el descenso de gradiente alcanza un punto estacionario y nada más puede decirse sin mirar las derivadas de orden superior, las derivadas de las derivadas. Para una función convexa, el descenso de gradiente alcanza un mínimo global sea cual sea el punto de partida. Y si la desigualdad se cumple estrictamente, con «menor que» en vez de «menor o igual», lo que se llama convexidad estricta, ese minimizador global es además único.

La diferencia es severa. Para estar seguro del mínimo global de un problema no convexo con descenso de gradiente, tendría que ejecutarlo desde cada punto de partida posible, infinitas veces, lo que hace la tarea NP-dura, entre las clases de problemas más difíciles de resolver. Esa es la huevera: cada salida encuentra un hueco, pero nada le dice cuál es el más bajo.

? MÍN. GLOBAL: NP-DURO



**REDES: NO
CONVEXAS**

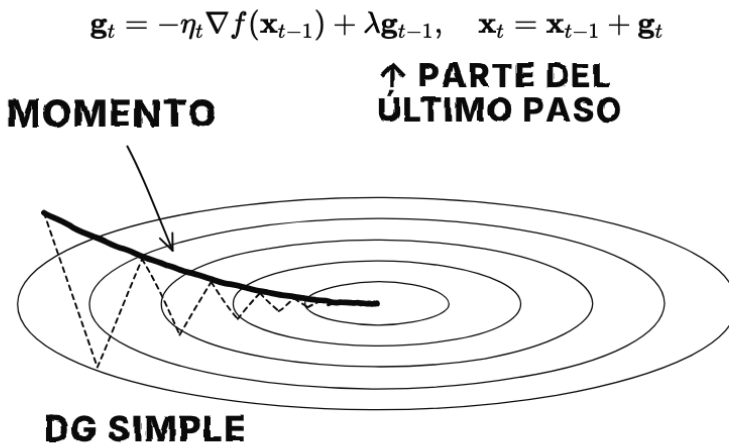
SVM: CONVEXA

✓ **BUENOS PUNTOS
EN LA PRÁCTICA** CHAT · FOTOS

¿Qué hueco es el más bajo?

Esto importa históricamente. Como mostraré el capítulo 5, toda red neuronal no trivial es no convexa, así que su optimización puede tener muchos puntos estacionarios. Otros métodos de aprendizaje, como las máquinas de vectores de soporte, mantienen un comportamiento no lineal y aun así tienen un problema de optimización convexo, lo que durante mucho tiempo se vio como una gran ventaja. Sin embargo, curiosamente, los modelos diferenciables complejos funcionan bien de todos modos: partiendo de una inicialización razonable, su optimización alcanza puntos que rinden bien en la práctica. El asistente de chat y la aplicación de fotos de su teléfono se entrenaron exactamente así, en hueveras de miles de millones de dimensiones, y funcionan.

Ahora la aceleración. El gradiente negativo es el camino más empinado hacia abajo solo en un entorno infinitamente pequeño del punto actual, y, como mostrará el capítulo 5 con la optimización estocástica, estas direcciones pueden ser muy ruidosas para modelos grandes. Muchas técnicas eligen mejores direcciones de descenso; por razones de cálculo el libro prefiere las que no necesitan derivadas de orden superior, como la hessiana, la matriz de segundas derivadas, ni llamadas extra a la función.

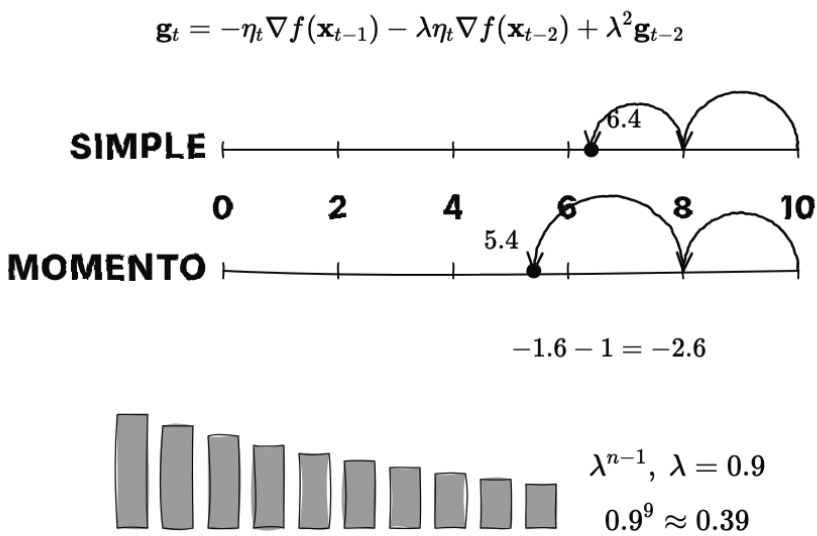


Zigzag frente a momento

La que describe es el momento. Imagine el descenso de gradiente como una bola que rueda cuesta abajo: su camino es a sacudidas, porque cada gradiente puede apuntar a un sitio bastante distinto; de hecho, con un paso perfectamente elegido en una función convexa,

dos gradientes sucesivos forman un ángulo recto. El momento suaviza esto conservando parte del paso anterior: $\mathbf{g}_t = -\eta_t \nabla f(\mathbf{x}_{t-1}) + \lambda \mathbf{g}_{t-1}$, y luego $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{g}_t$, empezando con $\mathbf{g}_0 = \mathbf{0}$. En palabras: el nuevo paso es el paso habitual de descenso más pronunciado más una fracción λ del paso anterior.

Ejecutémoslo. Tome $f(x) = x^2$, empiece en 10, con un tamaño de paso de 0,1 y $\lambda = 0,5$. Primer paso: el gradiente es 20, así que el paso es menos 2 y x pasa a ser 8, lo mismo que con el descenso de gradiente simple. Segundo paso: el gradiente es 16, la parte simple es menos 1,6, y el momento añade la mitad del menos 2 anterior, dando menos 2,6, así que x pasa a ser 5,4. El descenso de gradiente simple solo habría llegado a 6,4: el empujón recordado hizo que la bola rodara más lejos.



El momento rueda más lejos

Desplegar dos pasos muestra cómo se desvanece la memoria: $\mathbf{g}_t = -\eta_t \nabla f(\mathbf{x}_{t-1}) - \lambda \eta_t \nabla f(\mathbf{x}_{t-2}) + \lambda^2 \mathbf{g}_{t-2}$. En general, el gradiente de hace n pasos se amortigua por un factor λ^{n-1} . Con λ igual a 0,9, un gradiente de hace diez pasos todavía cuenta con 0,9 elevado a nueve, unos 0,39: el momento tiene una memoria larga. La figura del libro compara el descenso de gradiente simple y con momento en $x \sin(2x)$, empezando justo pasado 1 con λ igual a 0,3, y el momento muestra el camino más suave.

Otra idea común es adaptar el tamaño de paso para cada parámetro por separado, según lo grandes que hayan sido sus gradientes. El optimizador ampliamente usado llamado Adam combina varias de estas ideas. Una de sus ventajas es que es bastante robusto a la elección de sus hiperparámetros, los ajustes elegidos por el usuario en vez de aprendidos por descenso de gradiente, como el tamaño de paso y λ ; los valores por defecto de las bibliotecas son un buen comienzo en la mayoría de los casos. Diseñar un nuevo optimizador que pueda desbancar a Adam, o a sus variantes como AdamW, como el estándar en aprendizaje profundo sigue siendo un problema de investigación abierto, incluido el trabajo reciente que diseña optimizadores para redes neuronales desde primeros principios.

Pruébalo

En una calculadora, ejecute el descenso de gradiente simple en equis al cuadrado desde 10 con paso 0,1: cada paso multiplica x por 0,8, dando 8, 6,4, 5,12. Ahora ejecute el momento con λ igual a 0,5: 8, luego 5,4, luego calcule usted el tercer paso: el gradiente 10,8 da menos 1,08, más la mitad de menos 2,6, total menos 2,38, así que x es 3,02. Compare 3,02 con el 5,12 del descenso simple tras los mismos tres pasos.

Para recordar

En una función convexa el descenso de gradiente alcanza un mínimo global desde cualquier salida, y con convexidad estricta ese minimizador es único; en una no convexa solo alcanza un punto estacionario, y hallar el mínimo global se vuelve NP-duro. Las redes neuronales son no convexas, a diferencia de las máquinas de vectores de soporte, y sin embargo desde una salida razonable alcanzan puntos que funcionan bien en la práctica. El momento añade una fracción λ del paso anterior, $g_t = -\eta_t \nabla f(x_{t-1}) + \lambda g_{t-1}$, así que los gradientes viejos se desvanecen como λ^{n-1} ; en x^2 desde 10 llegó a 5,4 en vez de 6,4 en dos pasos. Adam combina el momento con tamaños de paso por parámetro, es robusto a sus hiperparámetros y sigue siendo el estándar que los nuevos optimizadores intentan superar.

Preguntas

1. ¿Qué garantiza el descenso de gradiente en una función convexa que no puede garantizar en una no convexa?
 - a. Alcanzar un mínimo global desde cualquier punto de partida
 - b. Dar menos pasos que con momento
 - c. No encontrarse nunca con un punto de gradiente cero
2. ¿Por qué hallar el mínimo global de un problema no convexo con descenso de gradiente se llama NP-duro?
 - a. Porque cada paso de gradiente es demasiado caro
 - b. Porque habría que ejecutarlo desde cada punto de partida posible para estar seguro
 - c. Porque las funciones no convexas no tienen mínimo
3. En el momento, $g_t = -\eta_t \nabla f(x_{t-1}) + \lambda g_{t-1}$. ¿Qué controla λ ?
 - a. Lo grande que es el paso de gradiente actual
 - b. Cuántos gradientes se calculan por paso
 - c. Cuánto del paso anterior se conserva, de modo que los gradientes viejos se desvanecen por potencias de λ
4. ¿Por qué Adam es a menudo una elección cómoda?
 - a. Porque hace convexas las redes neuronales
 - b. Porque es bastante robusto a sus hiperparámetros, y los valores por defecto funcionan en la mayoría de los casos
 - c. Porque necesita la hessiana, lo que lo hace exacto

Respuestas

1. **a)** En una función convexa el descenso de gradiente alcanza un mínimo global sin importar la inicialización; en una no convexa solo alcanza algún punto estacionario.
2. **b)** Cada ejecución encuentra algún punto estacionario, como una bola que se asienta en un hueco de una huevera, y solo probar cada salida garantizaría el más bajo.
3. **c)** λ es la fracción del paso anterior que se arrastra; al desplegar se ve el gradiente de hace n pasos ponderado por λ^{n-1} .
4. **b)** Adam combina el momento con tamaños de paso por parámetro y resulta relativamente robusto a la elección de sus hiperparámetros, así que los valores por defecto de las bibliotecas son un buen comienzo.

Ejemplos resueltos

Dos pasos con momento Minimice $f(x) = x^2$ desde $x = 10$ con tamaño de paso $0,1$ y $\lambda = 0,5$, empezando con $g_0 = 0$. Compare con el descenso de gradiente simple tras dos pasos.

Paso 1: gradiente $2 \times 10 = 20$, así que $g_1 = -0,1 \times 20 + 0,5 \times 0 = -2$ y $x = 8$.

Paso 2: gradiente 16 , así que $g_2 = -1,6 + 0,5 \times (-2) = -2,6$ y $x = 8 - 2,6 = 5,4$.

Descenso simple: $10 \rightarrow 8 \rightarrow 8 - 1,6 = 6,4$.

El momento está más cerca del mínimo en 0 tras los mismos dos pasos.

Respuesta: El momento llega a $5,4$, el descenso simple a $6,4$

Ejercicios

1. Continúe el ejemplo de momento con un tercer paso desde $x = 5,4$ con $g_2 = -2,6$.

Pista: El gradiente es $2 \times 5,4$; añada la mitad de g_2 .

Respuesta: Gradiente 10,8, parte simple $-1,08$, más $0,5 \times (-2,6) = -1,3$, así que $g_3 = -2,38$ y $x = 5,4 - 2,38 = 3,02$, frente a 5,12 para el descenso simple.

2. Con $\lambda = 0,9$, ¿por qué factor se pondera el gradiente de hace 4 pasos en el paso de momento actual?

Pista: El gradiente de hace n pasos se amortigua por λ^{n-1} .

Respuesta: $0,9^3 = 0,729$.

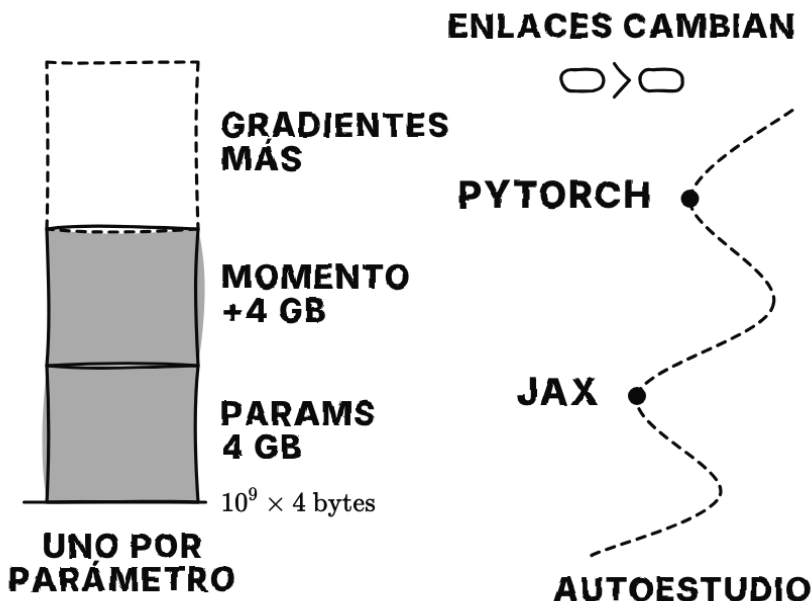
3. Explique por qué las redes neuronales se entrenan con éxito aunque su optimización es no convexa.

Pista: Piense en lo que dice el libro sobre los puntos de partida y el rendimiento empírico.

Respuesta: La no convexidad significa muchos puntos estacionarios y ninguna garantía del mínimo global. Pero, como observa el libro, cuando se parte de una inicialización razonable, la optimización alcanza puntos con buen rendimiento en la práctica, así que la garantía global no es necesaria para que los modelos funcionen bien.

El precio de la memoria, y dónde practicar

Una bola con momento recuerda su último empujón, y recordar cuesta espacio. El momento guarda en memoria el paso anterior, un número por cada parámetro, lo que duplica el espacio que necesita el propio optimizador. Póngale números: un modelo con mil millones de parámetros, cada uno guardado en 4 bytes, ocupa 4 gigabytes; la memoria del momento sobre el último paso añade otros 4. El libro añade que, en la mayoría de los casos, la memoria necesaria para calcular el gradiente importa aún más, un punto al que vuelve en la sección 6.3.

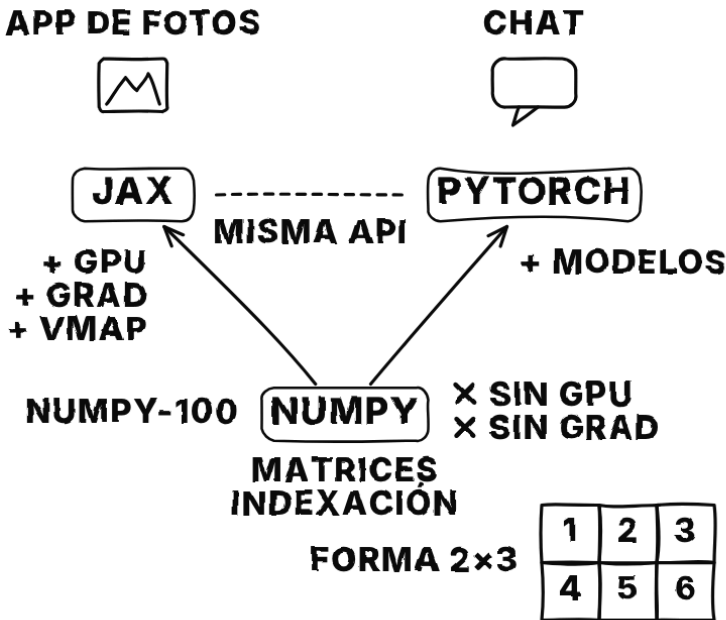


El momento cuesta memoria

Luego el capítulo pasa de la teoría a la práctica. El libro no tiene ejercicios clásicos de fin de capítulo, ya que muchos libros de texto ya los ofrecen. En su lugar propone un camino de autoaprendizaje a través de dos bibliotecas, JAX y PyTorch, que crece a medida que avanza el libro, con soluciones que se publicarán en la página web del libro. Advierte de que estas secciones están llenas de enlaces a material en línea, que puede haberse movido o caducado cuando usted lo consulte.

El punto de partida para cualquiera que diseñe modelos diferenciables, dice el libro, es un estudio cuidadoso de NumPy. NumPy ofrece un conjunto general de funciones para manejar matrices multidimensionales, lo que el libro llama tensores, junto con

funciones para indexar y transformar su contenido. Debería sentirse cómodo manejando matrices en NumPy, sobre todo indexándolas, las rebanadas y los dos puntos que vimos antes. La guía de inicio rápido de la biblioteca es el lugar para empezar, y una colección de cien ejercicios de NumPy, el repositorio [rougier/numpy-100](https://github.com/rougier/numpy-100), ofrece una manera pausada de ponerse a prueba.



NumPy, luego JAX y PyTorch

Con toda su influencia, NumPy tiene límites. Sin bibliotecas adicionales tiene poco soporte para hardware paralelo como las GPU, y no tiene diferenciación automática, la maquinaria del capítulo 6. JAX copia la interfaz de NumPy y añade un soporte de hardware más amplio, cálculo automático de gradientes y más transformaciones, como el mapa vectorizado, `jax.vmap`, que aplica una función a todo un

lote de una vez. PyTorch también tiene un núcleo al estilo de NumPy, con pequeños cambios de nombres y comportamiento, y añade herramientas de alto nivel para construir modelos diferenciables.

El libro sugiere hojear la documentación de `jax.numpy.array` y `torch.tensor` para ver cuánto comparten ambos con NumPy. Los módulos de alto nivel como `torch.nn` pueden ignorarse por ahora; el libro explica cómo están diseñadas estas bibliotecas en el capítulo 6, una vez presentada su maquinaria de gradientes. Los modelos detrás de su aplicación de fotos y su asistente de chat casi seguro se escribieron en una de estas dos bibliotecas, así que este camino lleva directamente a las herramientas de los sistemas reales.

Pruébalo

Si tiene Python a mano, cree con NumPy una pequeña matriz con los números del 1 al 6 en 2 filas de 3, y pregunte su forma: debería responder 2 por 3. Haga lo mismo con `jax.numpy` o `torch`, y fíjese en que las órdenes son casi idénticas. Esa casi identidad es lo que el libro quiere decir cuando afirma que estas bibliotecas replican la interfaz de NumPy.

Para recordar

Los optimizadores acelerados cuestan memoria: el momento guarda el paso anterior, duplicando el espacio del optimizador, 4 gigabytes más para mil millones de parámetros, aunque calcular gradientes suele necesitar aún más. En vez de ejercicios clásicos, el libro ofrece un camino de autoaprendizaje a través de JAX y PyTorch, con soluciones en línea. Empiece con NumPy, la biblioteca estándar para matrices, y practique la indexación, por ejemplo con los ejercicios `numpy-100`. NumPy carece de soporte sencillo para GPU y de diferenciación automática; JAX copia su interfaz y añade ambas cosas, más transformaciones como `vmap`, mientras que PyTorch ofrece un núcleo al estilo de NumPy con herramientas para construir modelos.

Preguntas

1. ¿Por qué el momento aumenta la memoria que necesita un optimizador?
 - a. Porque guarda el paso anterior, un número por parámetro, duplicando el espacio del optimizador
 - b. Porque calcula segundas derivadas
 - c. Porque guarda una copia de todo el conjunto de datos
2. ¿Qué añade JAX a la interfaz de NumPy?
 - a. Una manera distinta de indexar matrices
 - b. Soporte para cadenas y enteros en las matrices

- c. Soporte de hardware más amplio, gradientes automáticos y transformaciones como vmap
3. ¿Por qué el libro empieza su camino práctico con NumPy en vez de con una biblioteca completa?
- a. Porque NumPy es más rápido que JAX y PyTorch en GPU
 - b. Porque manejar e indexar matrices en NumPy es la base que JAX y PyTorch copian
 - c. Porque NumPy calcula gradientes automáticamente

Respuestas

1. **a)** El momento tiene que recordar el paso de gradiente anterior para cada parámetro, duplicando el espacio del algoritmo de optimización, aunque calcular el gradiente suele necesitar aún más memoria.
2. **c)** NumPy carece de soporte sencillo para GPU y de diferenciación automática; JAX replica su interfaz añadiendo ambas cosas, más transformaciones como el mapa vectorizado.
3. **b)** NumPy es el estándar para manipular matrices multidimensionales, y tanto JAX como PyTorch se construyen sobre una interfaz al estilo de NumPy, así que la comodidad con NumPy se traslada.

Ejemplos resueltos

Coste de memoria del momento Un modelo tiene mil millones de parámetros, cada uno guardado en 4 bytes. ¿Cuánta memoria ocupan los parámetros, y cuánta añade el momento?

Parámetros: $1\,000\,000\,000 \times 4 \text{ bytes} = 4\,000\,000\,000 \text{ bytes}$, unos 4 gigabytes.

El momento guarda el paso anterior, un número por parámetro, también de 4 bytes cada uno.

Así que añade otros 4 gigabytes, duplicando el espacio del optimizador.

Respuesta: 4 GB para los parámetros, más 4 GB para el momento

Ejercicios

1. Un modelo tiene 7 mil millones de parámetros guardados en 2 bytes cada uno. ¿Cuánta memoria ocupa el paso guardado del momento?

Pista: Un número por parámetro, del mismo tamaño que los parámetros.

Respuesta: $7\,000\,000\,000 \times 2 \text{ bytes} = 14\,000\,000\,000 \text{ bytes}$, unos 14 gigabytes.

2. Explique con sus propias palabras por qué el libro sugiere ignorar por ahora módulos como `torch.nn`.

Pista: Piense en lo que el libro aún no ha presentado.

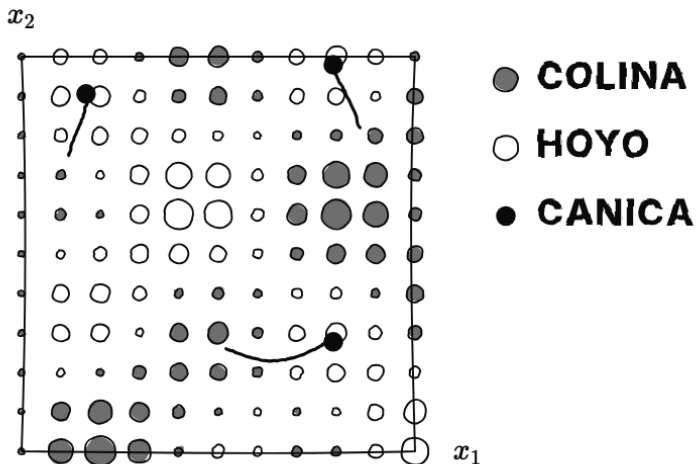
Respuesta: Son herramientas de alto nivel para construir modelos, y el libro explica cómo están diseñadas las bibliotecas solo en el capítulo 6, tras presentar cómo se calculan los gradientes; por ahora el objetivo es ver cuánto comparten `jax.numpy.array` y `torch.tensor` con NumPy.

Un paisaje para practicar, y qué es un conjunto de datos

Imagine una lámina de huevera extendida sobre una mesa, de diez por diez unidades, con bultos y hondonadas repartidos por ella. Deje caer unas canicas en sitios distintos y mire dónde acaba cada una. Ese es el ejercicio que el libro plantea antes de seguir adelante, y justo después viene la pregunta de la que depende todo modelo entrenado: ¿de dónde vienen los ejemplos, y podemos confiar en ellos?

$$f(\mathbf{x}) = \sin x_1 \cos x_2 + \sin 0.5x_1 \cos 0.5x_2$$

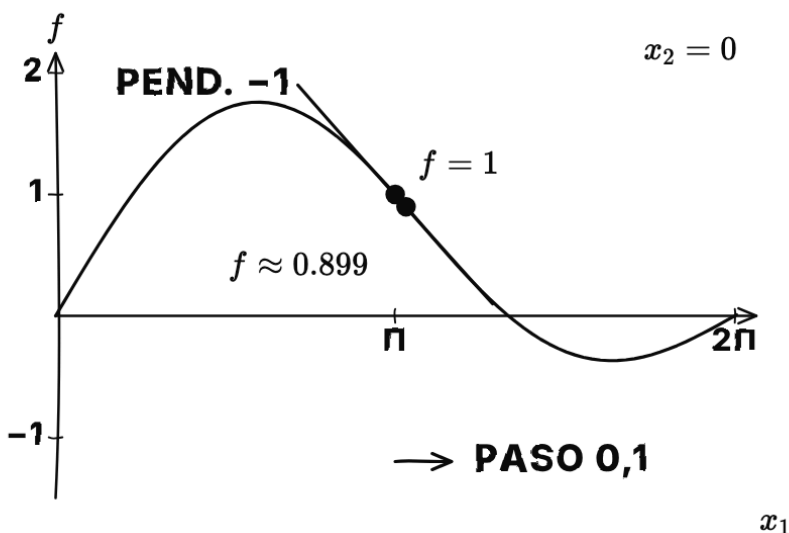
ONDA RÁPIDA ONDA LENTA



Un paisaje de bultos y hondonadas

El libro pide al lector repetir un ejercicio tres veces, en NumPy, en JAX y en PyTorch, cada uno de solo unos minutos una vez que la sintaxis es familiar. El paisaje es una función de dos entradas, $\mathbf{x} \sim (2)$, sobre el cuadrado de 0 a 10 en ambas direcciones: $f(\mathbf{x}) = \sin(x_1)\cos(x_2) + \sin(0.5x_1)\cos(0.5x_2)$. En palabras, dos ondas multiplicadas entre sí, más una copia más lenta del mismo patrón, lo que da un terreno con varios picos y valles. El libro añade una nota honesta: esta función se generó pidiendo a ChatGPT una con varios mínimos y máximos, y nada más en el libro fue producido por un modelo de lenguaje.

Las tareas vienen en cuatro pasos. Primero, implementar la función en forma vectorizada: dada una matriz $\mathbf{X} \sim (n, 2)$ de n puntos de entrada, devolver un vector de n valores, $[f(\mathbf{X})]_i = f(\mathbf{X}_i)$, todos a la vez y no en un bucle. Segundo, escribir a mano una función para el gradiente, ya que la diferenciación automática aún no se ha presentado. Tercero, escribir un descenso de gradiente básico y dibujar los caminos desde varios puntos de partida, las canicas en la huevera; en JAX o PyTorch este es un buen sitio para probar vmap. Cuarto, añadir momento y representar la longitud del gradiente, que debería encogerse hacia cero a medida que el método se acerca a un punto estacionario.



Un paso cuesta abajo

Hagamos el gradiente a mano en un punto, para ver que el ejercicio está al alcance. La derivada parcial a lo largo de x_1 es $\cos(x_1)\cos(x_2) + 0.5\cos(0.5x_1)\cos(0.5x_2)$, donde la regla de la cadena da el factor un medio para la onda lenta. En el punto $(\pi, 0)$, f es 0 más seno de $\pi/2$, que es 1; la primera parcial es menos 1 más un medio por 0, que es menos 1, y la segunda parcial es 0. Un paso de descenso de gradiente de 0,1 sube por tanto x_1 hasta unos 3,24, y f baja hasta unos 0,899, una caída de aproximadamente 0,1, justo como predijo la pendiente de menos 1.

Ahora el capítulo 3, que formaliza el aprendizaje supervisado: conjuntos de datos, pérdidas, minimización del riesgo empírico, los supuestos básicos y, por último, una formulación probabilística

construida sobre la máxima verosimilitud. El libro llama a este breve capítulo la columna vertebral de todo lo que sigue.

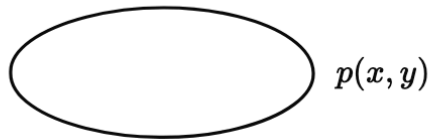
SUPERVISADO

<i>x</i>	<i>y</i>	
	PERRO	} N PARES
	GATO	
	PERRO	
	GATO	
<i>n</i> = 10,000		

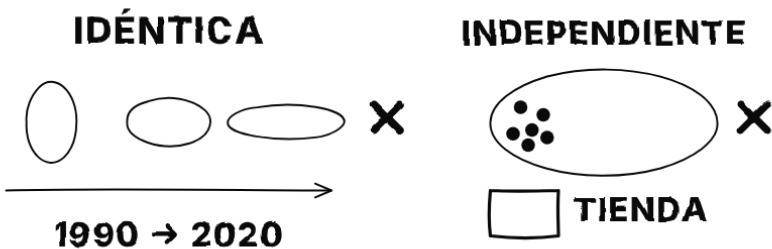
Un conjunto de datos es una lista de pares

La situación de partida: escribir la función a mano es imposible, reconocer objetos en fotos reales, por ejemplo, pero reunir ejemplos del comportamiento deseado es bastante fácil. El reconocimiento de voz y la navegación de robots son más ejemplos. La definición: un conjunto de datos supervisado $\mathcal{S}_n$ de tamaño n es un conjunto de n pares (x_i, y_i) , cada uno un ejemplo de la relación entrada-salida que queremos modelar. La aplicación de fotos de su teléfono podría partir de 10 000 pares así: una foto como x , y la palabra «perro» o «gato» como y .

Se añade un supuesto más, y es crucial: cada ejemplo es una extracción independiente e idénticamente distribuida, i.i.d., de alguna distribución de probabilidad $p(x, y)$ desconocida, y de hecho incognoscible. Suena técnico, pero es lo que da sentido a la tarea. Idénticamente distribuida significa que estamos aproximando algo estable que no cambia con el tiempo. El ejemplo del libro es reconocer modelos de coche en fotos: recogidas en un periodo corto, el supuesto se cumple; recogidas a lo largo de varias décadas, falla, porque los modelos cambiaron, y un modelo entrenado con esos datos no reconocerá los coches nuevos o rendirá mal.



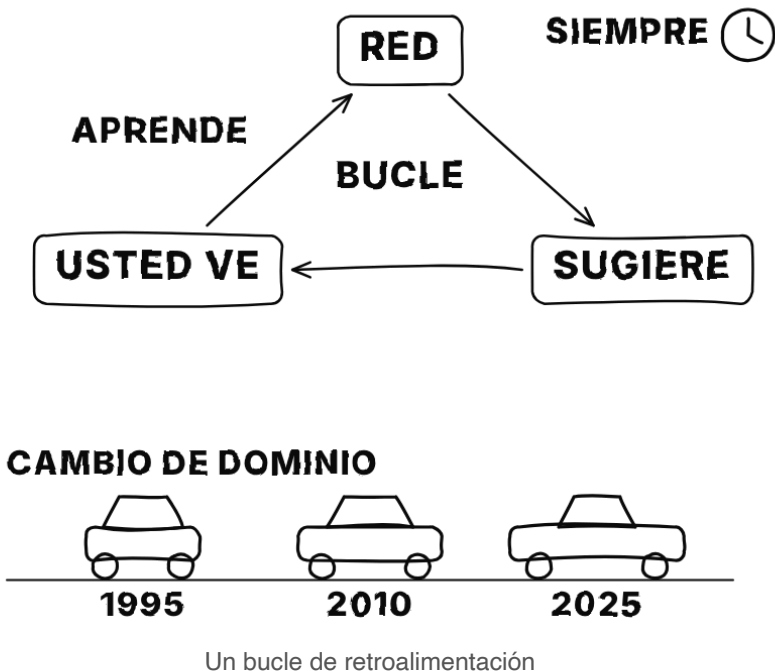
COMPROBAR  **I.I.D.**



Misma fuente, extraída con honestidad

Independientemente distribuida significa que la recogida no tiene sesgo y representa toda la distribución. Reúna fotos de coches junto a

un concesionario de Tesla y tendrá demasiados de un tipo y muy pocos de todas las demás marcas. La validez también depende de dónde se use el modelo: un conjunto de datos de coches recogido en Italia puede servir en Roma o Milán, pero no en Tokio o Taiwán. Así que el supuesto i.i.d. debe comprobarse siempre. Y para los modelos de lenguaje modernos, entrenados con distribuciones enormes de datos, incluso distinguir qué tareas están dentro de la distribución y cuáles fuera, y hasta dónde generalizan los modelos, se vuelve borroso.



El recuadro del libro añade que la comprobación nunca se hace de una vez por todas; debe continuar durante toda la vida de un modelo. Sin vigilancia, los cambios lentos en qué coches circulan degradarán

un clasificador de coches, un ejemplo de cambio de dominio. Un sistema de recomendación es aún más delicado: cambia cómo la gente usa una aplicación, porque los usuarios empiezan a reaccionar a sus propias sugerencias, lo que crea bucles de retroalimentación que exigen reevaluar constantemente el sistema y la aplicación. Los vídeos que le sugiere su aplicación de streaming son un caso en vivo: lo que usted ve moldea lo que ella aprende, y lo que ella aprende moldea lo que usted ve.

Pruébalo

En una calculadora en radianes, evalúe $\sin(3.14159)$ más $\sin(1.5708)$: obtiene unos 0 más 1, que es 1. Ahora evalúe $\sin(3.24159)$ más $\sin(1.62080)$: unos menos 0,0998 más 0,9988, que es 0,899. El paisaje bajó unos 0,1 cuando dio un paso de 0,1 contra el gradiente, exactamente lo que promete un paso de descenso de gradiente.

Para recordar

El ejercicio propuesto implementa $f(x) = \sin(x_1)\cos(x_2) + \sin(0.5x_1)\cos(0.5x_2)$ en NumPy, JAX y PyTorch: vectorizado sobre un lote, con un gradiente escrito a mano, caminos de descenso de gradiente desde varias salidas y momento con una norma del gradiente que se encoge. Un conjunto de datos supervisado es un conjunto de n pares entrada-salida, usado cuando escribir la función a mano es imposible pero los ejemplos son fáciles de reunir. Los pares se suponen i.i.d. de una distribución desconocida: idénticamente distribuidos significa estables en el tiempo, independientemente distribuidos significa recogidos sin sesgo, como muestran las décadas de fotos de coches y el concesionario de Tesla. El supuesto debe vigilarse durante toda la vida del modelo: el cambio de dominio degrada los modelos con el tiempo, y los sistemas de recomendación crean bucles de retroalimentación al cambiar el mismo comportamiento del que aprenden.

Preguntas

1. Un clasificador de coches se entrena con fotos recogidas durante los últimos cuarenta años. ¿Qué parte del supuesto i.i.d. se rompe, y por qué?
 - a. Idénticamente distribuido, porque los modelos de coche cambiaron con el tiempo, así que la relación no es estable

- b. Independientemente distribuido, porque las fotos se tomaron en una sola ciudad
 - c. Ninguna, porque más años dan más datos
2. ¿Por qué recoger fotos de coches junto al concesionario de una marca rompería el supuesto i.i.d.?
- a. Porque las fotos serían demasiado parecidas en iluminación
 - b. Porque la recogida estaría sesgada, con demasiados coches de un tipo y muy pocos de los demás
 - c. Porque los concesionarios solo venden coches nuevos
3. ¿Por qué un sistema de recomendación necesita una reevaluación constante?
- a. Porque sus gradientes se vuelven ruidosos con el tiempo
 - b. Porque su conjunto de datos siempre es demasiado pequeño
 - c. Porque los usuarios reaccionan a sus propias sugerencias, creando bucles de retroalimentación que cambian los datos de los que aprende
4. En el ejercicio práctico, ¿qué debería pasar con la longitud del gradiente a medida que avanza el descenso de gradiente?
- a. Debería encogerse hacia cero a medida que el método se acerca a un punto estacionario
 - b. Debería crecer, porque el momento se suma a cada paso
 - c. Debería mantenerse constante si el tamaño de paso es fijo

Respuestas

1. a) Idénticamente distribuido significa que lo que se aproxima es estable en el tiempo; a lo largo de décadas los modelos de coche cambian, y el

modelo entrenado no reconocerá los nuevos.

2. b) Independientemente distribuido significa que la recogida no tiene sesgo y representa toda la distribución; un concesionario sobrerrepresenta a un fabricante.

3. c) Un recomendador cambia cómo los usuarios interactúan con una aplicación, así que los datos se desplazan en respuesta al propio sistema, lo que exige una reevaluación continua.

4. a) En un punto estacionario el gradiente es cero, así que a medida que el optimizador se acerca a uno, la norma del gradiente converge a cero.

Ejemplos resueltos

Un paso en el paisaje de práctica Para $f(x) = \sin(x_1)\cos(x_2) + \sin(0.5x_1)\cos(0.5x_2)$ en $(\pi, 0)$, calcule f y el gradiente, luego dé un paso de descenso de gradiente de tamaño 0,1.

$$f(\pi, 0) = \sin(\pi) \cdot 1 + \sin(\pi/2) \cdot 1 = 0 + 1 = 1.$$

$$\partial f / \partial x_1 = \cos(x_1)\cos(x_2) + 0.5\cos(0.5x_1)\cos(0.5x_2) = -1 + 0,5 \times 0 = -1.$$

$$\partial f / \partial x_2 = -\sin(x_1)\sin(x_2) - 0.5\sin(0.5x_1)\sin(0.5x_2) = 0 \text{ en } x_2 = 0.$$

Paso: $x_1 = \pi - 0,1 \times (-1) \approx 3,2416$, x_2 se queda en 0; $f \approx -0,0998 + 0,9988 \approx 0,899$.

Respuesta: f baja de 1 a unos 0,899 tras un paso

Ejercicios

1. Escriba la derivada parcial de $f(x) = \sin(x_1)\cos(x_2) + \sin(0.5x_1)\cos(0.5x_2)$ respecto a x_2 , y evalúela en $(\pi/2, \pi/2)$.

Pista: La derivada de $\cos$ es menos $\sin$, y la regla de la cadena trae un factor 0,5 para la onda lenta.

Respuesta: $\partial f / \partial x_2 = -\sin(x_1)\sin(x_2) - 0.5\sin(0.5x_1)\sin(0.5x_2)$. En $(\pi/2, \pi/2)$: $-1 \times 1 - 0,5 \times \sin(\pi/4)^2 = -1 - 0,5 \times 0,5 = -1,25$.

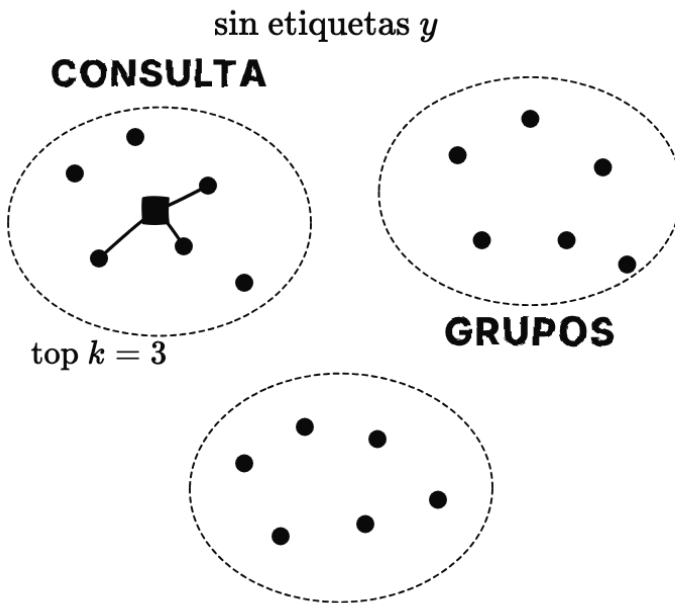
2. Un equipo entrena un modelo de desbloqueo facial con fotos de un solo país y lo distribuye por todo el mundo. Explique, con los términos del libro, qué podría salir mal.

Pista: Piense en el conjunto de datos de coches recogido en Italia y usado en Tokio.

Respuesta: Los datos pueden ser válidos donde se recogieron pero no en otros sitios: la distribución de uso difiere, así que el supuesto i.i.d. falla y el rendimiento puede caer para usuarios distintos de los del conjunto de entrenamiento; hay que comprobarlo, y volver a comprobarlo con el tiempo.

Aprender sin etiquetas, y reutilizar lo aprendido

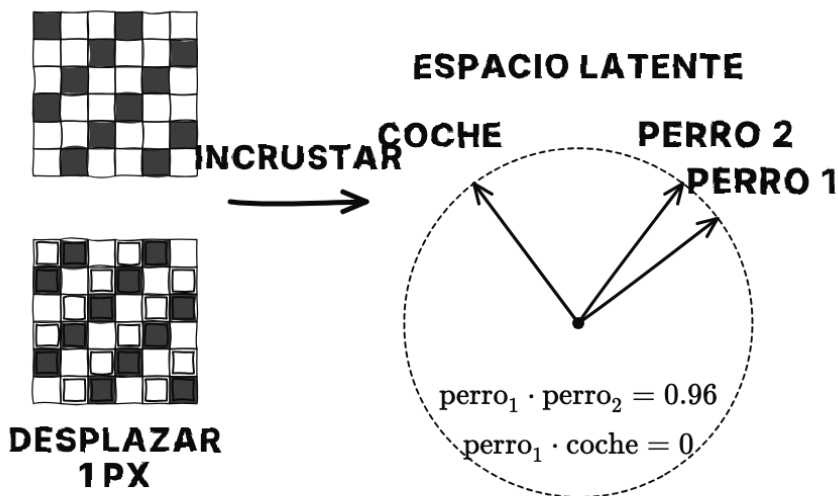
Abra su aplicación de fotos, elija una imagen de su perro y busque un botón que diga algo como «mostrar similares». Le devuelve otras fotos de perros, incluidas algunas que nadie etiquetó nunca, y aleja las fotos de su coche. Nadie le dijo a la aplicación qué se parece a qué; algo dentro de un modelo entrenado lo hizo. Esta parte explica ese algo, y las muchas maneras en que un modelo entrenado puede reutilizarse.



Agrupar y encontrar similares

La mayoría de las aplicaciones exitosas usan aprendizaje supervisado de alguna forma, pero hay muchas variaciones. Cuando el conjunto de datos no tiene objetivos y_i en absoluto, hablamos de aprendizaje no supervisado. Un uso clásico es el agrupamiento: reunir las entradas de modo que los puntos del mismo grupo sean parecidos y los de grupos distintos no lo sean. Otro es la recuperación: buscar en una gran base de datos los k elementos más parecidos a la consulta de un usuario.

Con datos complejos como imágenes esto es difícil, porque las distancias calculadas sobre píxeles crudos significan poco: incluso un cambio diminuto puede alterar millones de píxeles. Una foto de teléfono de 12 millones de píxeles, desplazada un solo píxel hacia un lado, nos parece idéntica, y sin embargo casi todos sus 36 millones de números de color cambian.



De los píxeles a un espacio latente

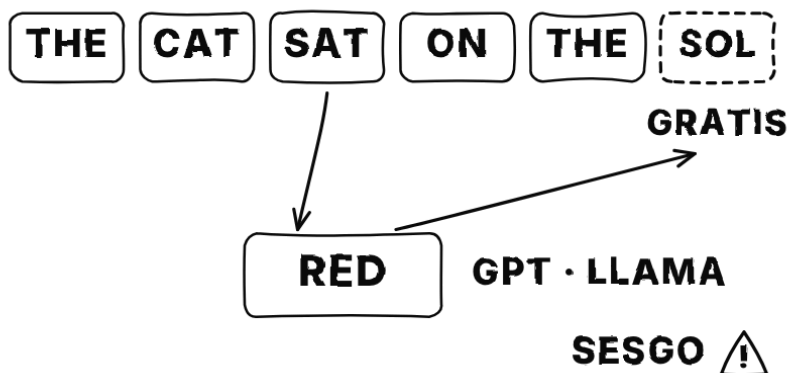
Ahora suponga que ya tenemos un modelo diferenciable optimizado para otra tarea bastante general, como la clasificación de imágenes; esto se llama un modelo preentrenado. Como mostrará el libro, los estados internos de un modelo así pueden leerse como vectores en un espacio de alta dimensión, y a menudo tienen una geometría útil: las cosas parecidas en significado se envían, se incrustan, a puntos cercanos entre sí. La figura del libro llama a esto hacer un embedding de los datos: moverlos de un espacio sin una distancia con sentido, las imágenes crudas, a un espacio con una, el espacio latente.

Aquí vuelve el producto escalar de las partes anteriores. Suponga que, en un espacio latente diminuto de dos dimensiones, una foto de perro cae en $(0,8, 0,6)$, otro perro en $(0,6, 0,8)$, y un coche en $(-0,6, 0,8)$,

todos de longitud uno. La similitud coseno de los dos perros es 0,48 más 0,48, que es 0,96, casi la misma dirección; el primer perro y el coche dan menos 0,48 más 0,48, que es 0, sin relación. Así funciona «mostrar similares», y los mismos vectores latentes pueden darse a métodos de agrupamiento estándar como los modelos de mezcla de gaussianas.

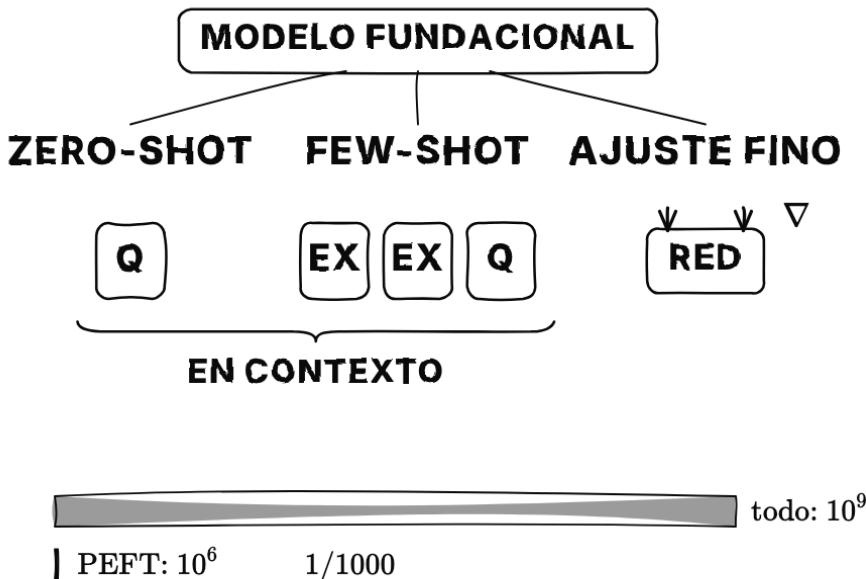
¿Y si no hay disponible ningún modelo preentrenado? Una variante común del aprendizaje no supervisado es el aprendizaje autosupervisado, SSL. Su idea es encontrar una tarea supervisada escondida dentro de los datos sin etiquetar, y usarla para entrenar un modelo que sirva a muchas tareas posteriores. Con un gran cuerpo de texto, por ejemplo, siempre podemos entrenar un programa para predecir cómo continúa un trozo corto de texto: las palabras siguientes son etiquetas gratis. El descubrimiento de que las redes neuronales preentrenadas de este modo también producen embeddings eficientes del texto tuvo un impacto profundo en el campo. El libro añade una advertencia: los grandes conjuntos de datos de la web están llenos de sesgos, groserías y contenido vulgar, los modelos entrenados con ellos absorben esos sesgos, y esta es una crítica importante a los modelos fundacionales de código cerrado.

AUTOSUPERVISADO



Etiquetas escondidas en el texto

Los grandes modelos de lenguaje, como mostrarán los capítulos 8 y 10, son versiones modernas de esta idea: modelos como GPT o Llama empiezan siempre con un entrenamiento autosupervisado para predecir el siguiente token, un pequeño trozo de texto. Tales modelos se llaman a veces modelos fundacionales, y hay tres maneras de usarlos, que la figura del libro compara. Zero-shot: darle al modelo una tarea nueva directamente, como una pregunta, y usar su respuesta. Prompting few-shot: poner primero unos pocos ejemplos de la nueva tarea en el prompt; ambos funcionan solo si el modelo generaliza muy bien. Ajuste fino: tomar el modelo preentrenado y optimizar sus parámetros por descenso de gradiente sobre un conjunto de datos pequeño para la nueva tarea, casi como si se entrenara desde cero.



Tres maneras de usar un modelo

Cuando le pide a un asistente de chat que traduzca una frase, eso es zero-shot; cuando primero le muestra dos ejemplos traducidos en el estilo que quiere, eso es prompting few-shot; y una empresa que lo entrena más con sus propios documentos lo está ajustando finamente. Cuando los ejemplos del prompt bastan y no hace falta ajuste fino, el libro dice que el modelo está haciendo aprendizaje en contexto; el aprendizaje few-shot también puede hacerse con ajuste fino. El propio libro se centra en construir modelos desde cero, señalando que el ajuste fino funciona por medios parecidos.

El ajuste fino es especialmente fácil hoy gracias a los grandes repositorios de modelos de código abierto. Puede actualizar todos los parámetros del modelo de partida, o solo un pequeño subconjunto, o

un pequeño número de parámetros añadidos; las últimas opciones se llaman ajuste fino eficiente en parámetros, PEFT. Para hacerse una idea del ahorro: entrenar un millón de números añadidos sobre un modelo de mil millones significa ajustar una milésima parte de los valores que ajustaría el ajuste fino completo. El libro deja las técnicas PEFT para su siguiente volumen.

Pruébalo

Dibuje tres flechas de longitud uno desde el centro de una hoja: una hacia arriba a la derecha para «perro uno», otra algo más empinada para «perro dos» y otra hacia arriba a la izquierda para «coche». Mida los ángulos con un transportador: las dos flechas de perro están cerca, a unos 16 grados, mientras que la flecha del coche está en ángulo recto con el primer perro. Un ángulo pequeño significa un coseno grande, es decir, «similar», que es todo lo que necesita un sistema de recuperación para ordenar sus resultados.

Para recordar

Sin objetivos tenemos aprendizaje no supervisado, como el agrupamiento y la recuperación, pero las distancias entre píxeles crudos no tienen sentido para las imágenes. Los vectores internos de un modelo preentrenado incrustan los datos en un espacio latente donde las cosas parecidas quedan cerca, así que los productos escalares y el agrupamiento funcionan ahí, como mostraron el 0,96 para dos perros y el 0 para un perro y un coche. El aprendizaje autosupervisado inventa etiquetas a partir de datos sin etiquetar, como predecir las palabras siguientes de un texto; así empiezan los modelos de lenguaje como GPT y Llama, y también importa los sesgos de los datos de la web. Los modelos fundacionales pueden usarse zero-shot, con prompts few-shot, o ajustarse finamente por descenso de gradiente, ya sea por completo o con ajuste fino eficiente en parámetros de un conjunto pequeño de parámetros.

Preguntas

1. ¿Por qué es difícil encontrar imágenes parecidas midiendo distancias directamente sobre sus píxeles?
 - a. Porque los píxeles no pueden guardarse como números
 - b. Porque incluso un cambio diminuto, como un desplazamiento de un píxel, altera millones de valores de píxel
 - c. Porque las imágenes no tienen etiquetas

2. ¿Qué hace útiles los vectores internos de un modelo preentrenado para la recuperación y el agrupamiento?
 - a. Las cosas parecidas en significado se incrustan cerca unas de otras en el espacio latente
 - b. Siempre son bidimensionales
 - c. Contienen los píxeles originales en forma comprimida
3. ¿Cómo obtiene etiquetas el aprendizaje autosupervisado de un gran cuerpo de texto sin etiquetar?
 - a. Pidiendo a personas que etiqueten una pequeña parte
 - b. Agrupando primero el texto
 - c. Entrenando el modelo para predecir cómo continúa un trozo de texto, de modo que las palabras siguientes sirven de etiquetas
4. Le muestra a un asistente de chat dos ejemplos de su estilo de resumen preferido y luego le pide un tercer resumen, sin cambiar sus parámetros. ¿Cómo se llama esto?
 - a. Ajuste fino
 - b. Prompting few-shot, un ejemplo de aprendizaje en contexto
 - c. Uso zero-shot

Respuestas

1. **b)** Las distancias sobre píxeles crudos están mal definidas: pequeñas perturbaciones modifican millones de píxeles mientras la imagen nos parece la misma.
2. **a)** Los estados internos del modelo a menudo tienen una geometría útil, enviando objetos semánticamente parecidos a puntos cercanos, así que las distancias y los productos escalares cobran sentido ahí.

- 3. c)** El SSL encuentra un objetivo supervisado dentro de datos sin etiquetar; con texto siempre puede entrenarse un modelo para predecir la continuación, que es como empiezan modelos como GPT y Llama.
- 4. b)** Dar unos pocos ejemplos en el prompt es prompting few-shot; cuando no hace falta ajuste fino, el modelo está haciendo aprendizaje en contexto.

Ejemplos resueltos

Similitud en un espacio latente En un espacio latente de dos dimensiones, el perro uno está en $(0,8, 0,6)$, el perro dos en $(0,6, 0,8)$ y un coche en $(-0,6, 0,8)$, todos de longitud uno. Calcule la similitud coseno del perro uno con el perro dos y con el coche.

Para vectores unitarios la similitud coseno es el producto escalar.

Perro uno con perro dos: $0,8 \times 0,6 + 0,6 \times 0,8 = 0,48 + 0,48 = 0,96$.

Perro uno con el coche: $0,8 \times (-0,6) + 0,6 \times 0,8 = -0,48 + 0,48 = 0$.

Así que los dos perros apuntan casi en la misma dirección, y el coche está en ángulo recto con el perro uno.

Respuesta: 0,96 para los dos perros, 0 para perro y coche

Ejercicios

1. Una foto tiene 4000 por 3000 píxeles con 3 valores de color cada uno. ¿Cuántos números la describen, y por qué hace esto que las distancias entre píxeles sean una mala medida de similitud?

Pista: Multiplique los tres tamaños; piense en un desplazamiento de un píxel.

Respuesta: $4000 \times 3000 \times 3 = 36\,000\,000$ números. Desplazar la foto un píxel cambia casi todos, así que dos fotos que parecen iguales pueden estar muy lejos en distancia de píxeles.

2. Un modelo base tiene mil millones de parámetros. El ajuste fino completo los actualiza todos, mientras que un método eficiente en parámetros entrena 2 millones de parámetros añadidos. ¿Qué fracción del número de parámetros entrena el método eficiente en parámetros?

Pista: Divida los dos números.

Respuesta: $2\,000\,000 / 1\,000\,000\,000 = 0,002$, es decir, dos milésimas del tamaño del modelo base.

3. Clasifique cada uso de un modelo de lenguaje como zero-shot, prompting few-shot o ajuste fino: (a) pedirle directamente que traduzca una frase; (b) entrenarlo más con los tickets de soporte de una empresa; (c) mostrarle tres acertijos resueltos antes de preguntarle un cuarto.

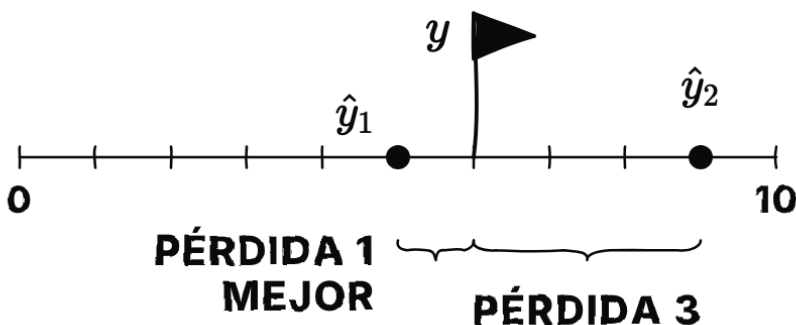
Pista: Pregúntese si hay ejemplos en el prompt y si los parámetros cambian.

Respuesta: (a) zero-shot; (b) ajuste fino, ya que sus parámetros se optimizan por descenso de gradiente; (c) prompting few-shot, ya que los ejemplos están en el prompt y los parámetros no cambian.

Puntuar los errores: la pérdida

Juegue a las adivinanzas con un amigo: él piensa un número y usted lo adivina. Tras cada intento le dice cuánto se alejó, y usted intenta hacer pequeño ese número. Sin ese único número, «cuánto se alejó», no tendría manera de mejorar; con él, mejorar se convierte en cuestión de aritmética. Una función de pérdida es ese número, y entrenar un modelo es el juego jugado millones de veces.

$$l(y, \hat{y})$$



$$l(y, \hat{y}_1) < l(y, \hat{y}_2)$$

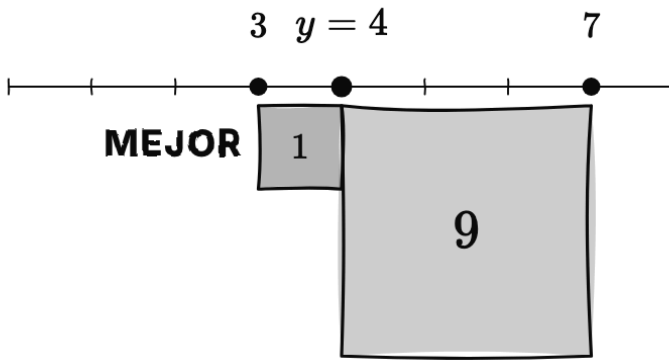
Cuánto se aleja la suposición

Primero, el libro completa su lista de escenarios de aprendizaje con unas pocas indicaciones. Cuando solo una parte del conjunto de datos

está etiquetada, el escenario es semisupervisado, con ejemplos que llegan en el capítulo 12. Cuando hay varios conjuntos de datos de distribuciones parecidas, o una distribución que cambia con el tiempo, surgen incontables problemas según el orden en que llegan las tareas o los datos: adaptación de dominio, metaaprendizaje, aprendizaje continuo, aprendizaje de métricas, desaprendizaje y más, algunos de los cuales trata el siguiente volumen.

La sección 3.2 pasa a las funciones de pérdida, que hacen precisa la idea de «aproximar» un comportamiento. La definición: dado un objetivo y y la predicción del modelo $\hat{y} = f(x)$, una pérdida $l(y, \hat{y})$ es un único número real, diferenciable, cuyo valor sigue lo bien que lo hace el modelo. Concretamente, $l(y, \hat{y}_1) < l(y, \hat{y}_2)$ significa que la predicción uno es mejor que la predicción dos para ese objetivo.

Un ejemplo común es el error cuadrático, la diferencia al cuadrado. Si el precio verdadero de una casa es 4 y dos modelos predicen 3 y 7, las pérdidas son 1 al cuadrado, que es 1, y 3 al cuadrado, que es 9; la pérdida menor, 1, señala la mejor predicción. Una pérdida convierte nuestra comprensión de la tarea, y nuestras preferencias entre soluciones, en un único número real con el que un optimizador puede trabajar. Y como es diferenciable, convierte el aprendizaje en un problema de optimización matemática que el descenso de gradiente puede resolver, minimizando la pérdida media sobre el conjunto de datos.



$$f^* = \arg \min_f \frac{1}{n} \sum_{i=1}^n l(y_i, f(x_i)) \quad \text{ERM}$$

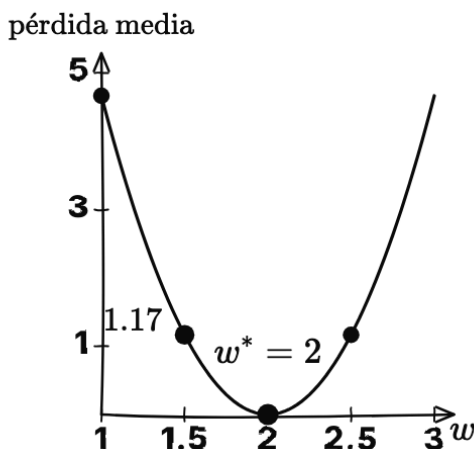
Error cuadrático, luego la media

Así, dado un conjunto de datos S_n de pares y una pérdida l , un objetivo sensato es el modelo con la menor pérdida media: $f^* = \arg \min_f (1/n) \sum_{i=1}^n l(y_i, f(x_i))$. En palabras: pruebe el modelo con cada ejemplo, puntúe cada predicción, promedie las puntuaciones y elija el modelo con la media más baja. Por razones históricas esto se llama minimización del riesgo empírico, ERM, donde «riesgo» es simplemente otra palabra para pérdida.

Calculemos un riesgo empírico. Tres ejemplos tienen objetivos 2, 4 y 6, y un modelo predice 3, 4 y 4. Las pérdidas cuadráticas son 1, 0 y 4; su media es 5 dividido por 3, unos 1,67.

y	$\hat{y}$	pérdida
2	3	1
4	4	0
6	4	4

media = $5/3 \approx 1.67$



$$w^* = \arg \min_w \frac{1}{n} \sum_{i=1}^n l(y_i, f(x_i, w))$$

La pérdida media y su valle

Esa fórmula busca entre todas las funciones posibles de x . En la práctica, como mostrará el libro, un modelo se describe por un conjunto de tensores w , sus parámetros, y escribimos $f(x, w)$. El entrenamiento busca entonces los mejores parámetros por optimización numérica, minimizando el riesgo empírico con descenso de gradiente: $w^* = \operatorname{argmin}_w (1/n) \sum_{i=1}^n l(y_i, f(x_i, w))$. El único cambio es que la búsqueda es ahora sobre el tensor de parámetros w .

Hagámoslo concreto con el modelo más sencillo, $f(x, w) = wx$, sobre las entradas 1, 2, 3 con objetivos 2, 4, 6. Con w igual a 1,5, las predicciones son 1,5, 3 y 4,5, las pérdidas cuadráticas 0,25, 1 y 2,25, y la media unos 1,17. Con w igual a 2, cada predicción es exacta y la pérdida media es 0, así que aquí w asterisco es 2. La aplicación de

fotos de su teléfono se entrenó de la misma manera, solo que con millones de fotos, millones de parámetros y una pérdida adecuada para etiquetas en vez de precios.

Pruébalo

Tome los tres pares (1, 2), (2, 4), (3, 6) y pruebe unos pocos valores de w en el modelo w por x : 1, 1,5, 2, 2,5. Para cada uno, calcule los tres errores cuadráticos y su media. Debería ver cómo la media baja hacia 0 en w igual a 2 y vuelve a subir más allá, un pequeño valle cuyo fondo es el parámetro entrenado.

Para recordar

Más allá del aprendizaje supervisado simple están el aprendizaje semisupervisado y muchos escenarios con varios conjuntos de datos, como la adaptación de dominio, el aprendizaje continuo y el desaprendizaje. Una pérdida $l(y, \hat{y})$ es un número real diferenciable que es menor para mejores predicciones, como un error cuadrático de 1 frente a 9. La minimización del riesgo empírico elige el modelo con la menor pérdida media sobre el conjunto de datos, unos 1,67 para las predicciones 3, 4, 4 frente a los objetivos 2, 4, 6. Con un modelo parametrizado $f(x, w)$, el entrenamiento busca $w^* = \operatorname{argmin}_w$ sobre w de la pérdida media, usando descenso de gradiente; para wx sobre nuestros tres pares, w^* es 2.

Preguntas

1. ¿Por qué una función de pérdida debe ser diferenciable?
 - a. Para que pueda imprimirse como un número real
 - b. Para que el aprendizaje se convierta en un problema de optimización que el descenso de gradiente pueda resolver
 - c. Para que sea siempre positiva
2. ¿Qué elige la minimización del riesgo empírico?
 - a. El modelo con la menor pérdida en un ejemplo elegido
 - b. El modelo con el mayor número de parámetros
 - c. El modelo con la menor pérdida media sobre todo el conjunto de datos
3. Para un objetivo de 4, la predicción A es 3 y la predicción B es 7. Con el error cuadrático, ¿cuál es mejor y por qué?
 - a. A, porque su pérdida es 1 frente al 9 de B
 - b. B, porque es mayor que el objetivo
 - c. Son igual de buenas, ya que ambas están equivocadas
4. ¿Qué cambia cuando pasamos de minimizar sobre todas las funciones f a minimizar sobre los parámetros w ?
 - a. La función de pérdida ya no hace falta
 - b. La búsqueda se convierte en una optimización numérica sobre los tensores de parámetros de una familia de modelos fija
 - c. El conjunto de datos se sustituye por un solo ejemplo

Respuestas

1. **b)** Al ser diferenciable, la pérdida convierte el aprendizaje en un problema de optimización matemática resuelto por descenso de gradiente, minimizando la pérdida media en el conjunto de datos.
2. **c)** La ERM puntúa cada predicción del conjunto de datos, promedia las pérdidas y elige el modelo, o los parámetros, con la media más baja; riesgo es sinónimo de pérdida.
3. **a)** Una pérdida menor significa una mejor predicción: $(4 - 3)^2 = 1$ es menor que $(4 - 7)^2 = 9$.
4. **b)** Los modelos se describen por tensores w , así que el entrenamiento busca $w^* = \operatorname{argmin}_w$ sobre w de la pérdida media mediante optimización numérica como el descenso de gradiente.

Ejemplos resueltos

Riesgo empírico de un modelo sencillo El modelo $f(x, w) = wx$ se prueba con los pares $(1, 2)$, $(2, 4)$, $(3, 6)$ con la pérdida cuadrática. Calcule el riesgo empírico para $w = 1,5$ y para $w = 2$.

$w = 1,5$: predicciones 1,5, 3, 4,5; errores cuadráticos 0,25, 1, 2,25.

Media: $(0,25 + 1 + 2,25) / 3 = 3,5 / 3 \approx 1,17$.

$w = 2$: predicciones 2, 4, 6, todas exactas; errores cuadráticos 0, 0, 0; media 0.

Así que $w = 2$ minimiza el riesgo empírico en este conjunto de datos.

Respuesta: Unos 1,17 para $w = 1,5$, y 0 para $w = 2$, que es w^*

Ejercicios

1. Los objetivos son 1, 3 y 5; un modelo predice 2, 3 y 3. Calcule el riesgo empírico con la pérdida cuadrática.

Pista: Eleve cada diferencia al cuadrado, sume, divida por 3.

Respuesta: Errores cuadráticos 1, 0 y 4; suma 5; riesgo empírico $5/3 \approx 1,67$.

2. Para $f(x, w) = wx$ con los pares (1, 2), (2, 4), (3, 6), calcule el riesgo empírico para $w = 2,5$ y compárelo con $w = 1,5$.

Pista: Predicciones 2,5, 5, 7,5.

Respuesta: Errores cuadráticos 0,25, 1 y 2,25, media $\approx 1,17$, la misma que con $w = 1,5$: el riesgo sube simétricamente a ambos lados del mínimo en $w = 2$.

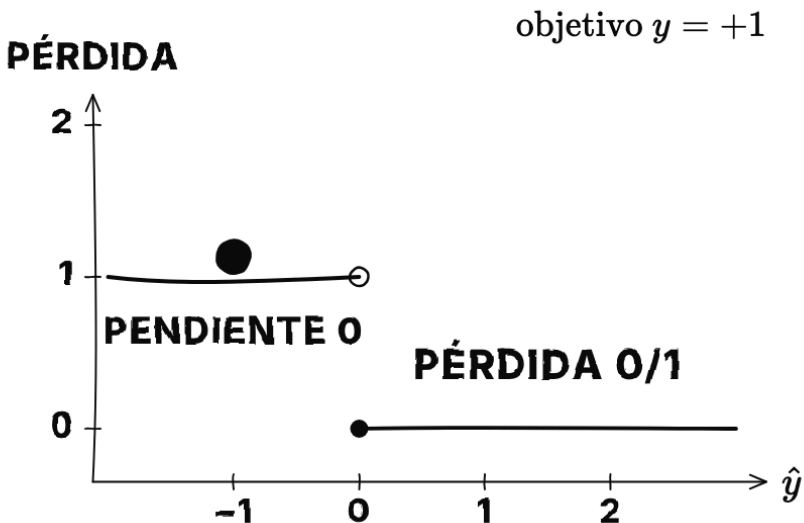
3. Un filtro de spam se entrena con un conjunto de datos donde solo el 5% de los correos están etiquetados. ¿Qué escenario de aprendizaje es este, y qué dice el libro sobre los escenarios en que los datos llegan con el tiempo desde distribuciones cambiantes?

Pista: Piense en las etiquetas parciales, y en la lista de problemas con varios conjuntos de datos.

Respuesta: Los datos parcialmente etiquetados son aprendizaje semisupervisado. Los datos de distribuciones parecidas o cambiantes que llegan con el tiempo dan lugar a problemas como la adaptación de dominio y el aprendizaje continuo, algunos tratados en el siguiente volumen.

Pérdidas que pueden seguirse, y modelos que solo recuerdan

Juegue otra vez a las adivinanzas, pero ahora su amigo solo dice «bien» o «mal». Dice 30 y oye «mal»; dice 31 y oye «mal» otra vez. No aprende nada sobre hacia dónde moverse, porque la respuesta no cambia cuando mueve un poco su suposición. Una pérdida con ese defecto no puede guiar el descenso de gradiente, y esta parte muestra cómo evitarlo, antes de pasar a una trampa más profunda: un modelo que puntúa perfectamente simplemente recordando.

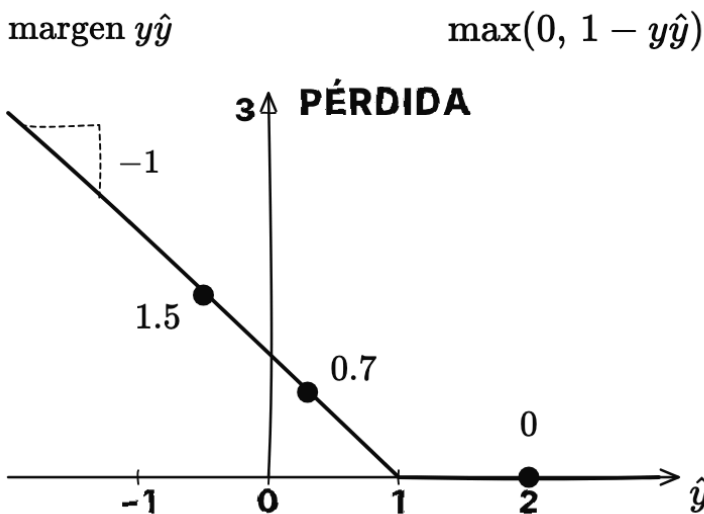


La pérdida bien-o-mal es plana

Primera observación: la diferenciabilidad de la pérdida es esencial.

Tome una tarea de clasificación binaria, donde el objetivo $y = \pm 1$ toma solo dos valores, y un modelo $f(x)$ que produce cualquier número real. Podemos leer la decisión del modelo como el signo de su salida, $\text{sign}(f(x))$, y definir la pérdida 0/1: $l(y, \hat{y}) = 0$ si $\text{sign}(\hat{y}) = y$, y 1 en caso contrario. Coincide con nuestra intuición de acertar, pero es inútil para entrenar. Su gradiente es cero casi en todas partes, salvo exactamente donde cambia el signo, así que el descenso de gradiente nunca se mueve de su punto de partida: bien o mal, sin nada en medio.

Una cantidad menos intuitiva pero más útil es el margen, $y\hat{y}$. Es positivo cuando el signo del modelo coincide con el objetivo y negativo cuando no, y, a diferencia de la pérdida 0/1, cambia suavemente. Una pérdida construida sobre él es la pérdida de bisagra, $l(y, \hat{y}) = \max(0, 1 - y\hat{y})$, usada para entrenar máquinas de vectores de soporte.



La pérdida de bisagra tiene pendiente

Calcúlela para un objetivo de más 1. Una predicción de 0,3 es correcta en signo, así que la pérdida 0/1 es 0, pero la pérdida de bisagra es 1 menos 0,3, que es 0,7: correcta, pero sin confianza. Una predicción de 2 da un margen de 2 y una pérdida de bisagra de 0. Una predicción de menos 0,5 da un margen de menos 0,5 y una pérdida de bisagra de 1,5, mayor porque es errónea. Y por debajo de un margen de 1, la pérdida de bisagra baja una unidad por cada unidad que sube la predicción: una pendiente que el descenso de gradiente puede seguir. El libro extrae la lección: hay una tensión incorporada entre las pérdidas que expresan nuestra idea de buen rendimiento y las pérdidas que funcionan bien para la optimización numérica.

Un recuadro anota la historia: la minimización del riesgo empírico y del riesgo esperado está ligada al trabajo del informático ruso Vladimir Vapnik, que fundó la teoría del aprendizaje estadístico. Esa teoría estudia lo bien que el riesgo empírico aproxima al riesgo esperado a partir de un número finito de muestras, para familias restringidas de funciones y medidas de su complejidad. Las redes neuronales modernas, que siguen generalizando bien mucho después de cuando se habría esperado el sobreajuste, han abierto muchas preguntas de investigación nuevas en esa teoría.

Segunda observación: el riesgo empírico siempre es trivial de minimizar. Defina un modelo que busca su entrada en el conjunto de entrenamiento: si el par está ahí, devuelve su y ; si no, devuelve un valor fijo por defecto como 0. Esta tabla de consulta hace la pérdida de entrenamiento tan baja como puede ser, y sin embargo es inútil, porque no sabe nada de ninguna entrada que no haya visto. Esa es la diferencia entre memorizar y aprender. Optimizamos la pérdida media sobre los datos de entrenamiento, pero nuestro objetivo real es una pérdida baja en entradas futuras aún no vistas; el conjunto de entrenamiento es solo un sustituto.

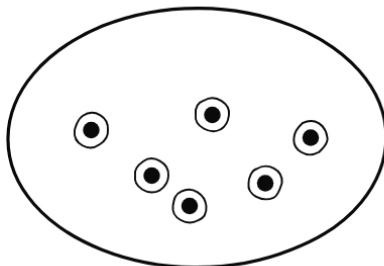
CONSULTA

1 → 2
2 → 4
3 → 6

PÉRDIDA ENTREN. 0

5 → 0 ✕

$$ER[f] = \mathbb{E}_{p(x,y)}[l(y, f(x))]$$



TODOS LOS PARES

MEMORIZAR

EMPÍRICO $\approx$

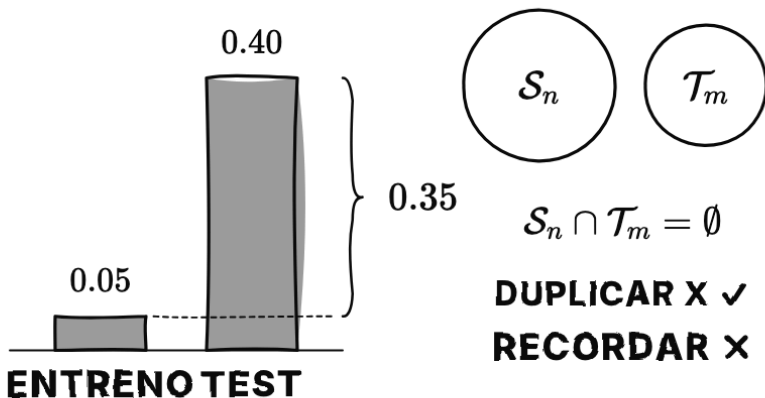
Recordar frente al riesgo esperado

El libro formaliza esto como el riesgo esperado: $ER[f] = \mathbb{E}_{p(x,y)}[l(y, f(x))]$. En palabras: la pérdida media sobre cada par entrada-salida que el modelo podría encontrar jamás, ponderado por lo probable que es cada uno, por ejemplo cada correo que un filtro de spam podría recibir. Un modelo con bajo riesgo esperado tendría garantizado funcionar, pero calcularlo es imposible, ya que nunca podemos enumerar y etiquetar cada entrada posible. El riesgo empírico lo estima a partir del conjunto de datos que tenemos, como una aproximación de Monte Carlo: una media sobre muestras aleatorias que sustituye a una media sobre todo.

La diferencia entre el riesgo esperado y el empírico es la brecha de generalización. La tabla de consulta tiene una brecha enorme:

sobreajusta sus datos de entrenamiento. En la práctica probamos la generalización con un conjunto de prueba aparte $\mathcal{T}_m$ de m puntos nunca usados en el entrenamiento, sin nada en común con el conjunto de entrenamiento: $\mathcal{S}_n \cap \mathcal{T}_m = \emptyset$. Si un clasificador de fotos alcanza una pérdida media de 0,05 en sus fotos de entrenamiento pero 0,40 en fotos de prueba apartadas, la brecha de 0,35 es una medida de cuánto sobreajusta.

BRECHA



La brecha de generalización

Pruébalo

Escriba cinco pares, como (1, 2), (2, 4), (3, 6), (4, 8), (5, 10), y esconda dos de ellos. Construya un «modelo» que simplemente recuerde los tres que conservó, y otro que diga «duplica la entrada». Compruebe ambos con los dos pares escondidos: el modelo de recuerdo no tiene respuesta y recurre al 0, el modelo de duplicar acierta ambos; eso es memorización frente a aprendizaje, medido en un conjunto de prueba.

Para recordar

La pérdida debe ser diferenciable: la pérdida 0/1 coincide con acertar pero tiene gradiente cero casi en todas partes, así que el descenso de gradiente no puede moverse. El margen $y\hat{y}$ varía suavemente, y la pérdida de bisagra $\max(0, 1 - y\hat{y})$ da 0,7 para un 0,3 correcto pero tímido, 0 para un 2 confiado y 1,5 para un -0,5 erróneo. El riesgo empírico puede llevarse a su mínimo con una tabla de consulta que solo memoriza; lo que realmente queremos es un riesgo esperado bajo, la pérdida media sobre todas las entradas posibles, que solo puede estimarse. La brecha entre el riesgo esperado y el empírico mide el sobreajuste, y un conjunto de prueba aparte nunca usado en el entrenamiento lo estima en la práctica.

Preguntas

1. ¿Por qué la pérdida 0/1 es inútil para entrenar con descenso de gradiente?
 - a. Porque da la respuesta equivocada para las predicciones correctas
 - b. Porque su gradiente es cero casi en todas partes, así que el descenso de gradiente nunca se mueve
 - c. Porque solo puede usarse con máquinas de vectores de soporte
2. Para un objetivo de +1, ¿qué predicción recibe una pérdida de bisagra de 0,7?
 - a. 0,3
 - b. 2
 - c. -0,5
3. Una tabla de consulta que devuelve la respuesta guardada para cada entrada de entrenamiento alcanza el menor riesgo empírico. ¿Por qué sigue siendo inútil?
 - a. Porque consultar es demasiado lento
 - b. Porque su pérdida no es diferenciable
 - c. Porque solo memoriza, y no tiene nada útil que decir sobre entradas que no ha visto
4. ¿Cómo se mide el sobreajuste en la práctica?
 - a. Comparando la pérdida en el conjunto de entrenamiento con la pérdida en un conjunto de prueba aparte nunca usado en el entrenamiento
 - b. Calculando el riesgo esperado sobre todas las entradas posibles

c. Contando los parámetros del modelo

Respuestas

1. **b)** La pérdida 0/1 solo cambia donde cambia el signo de la predicción; en todo lo demás mover la predicción no cambia nada, así que el gradiente es cero y el optimizador se queda en su salida.
2. **a)** La pérdida de bisagra es $\max(0, 1 - y\hat{y})$: para 0,3 es $1 - 0,3 = 0,7$, para 2 es 0, y para $-0,5$ es 1,5.
3. **c)** Nuestro verdadero objetivo es una pérdida baja en entradas futuras no vistas, el riesgo esperado; la tabla de consulta recurre a un valor por defecto y sobreajusta, con una brecha de generalización enorme.
4. **a)** El riesgo esperado no puede calcularse, así que un conjunto de prueba apartado lo estima; la diferencia entre la pérdida de entrenamiento y la de prueba aproxima la brecha de generalización.

Ejemplos resueltos

Pérdida de bisagra frente a pérdida 0/1 Para un objetivo $y = +1$, calcule el margen, la pérdida 0/1 y la pérdida de bisagra $\max(0, 1 - y\hat{y})$ para las predicciones $\hat{y} = 0,3, 2$ y $-0,5$.

$\hat{y} = 0,3$: margen 0,3; el signo coincide, así que pérdida 0/1 de 0; bisagra $1 - 0,3 = 0,7$.

$\hat{y} = 2$: margen 2; pérdida 0/1 de 0; bisagra $\max(0, -1) = 0$.

$\hat{y} = -0,5$: margen $-0,5$; el signo no coincide, así que pérdida 0/1 de 1; bisagra $1 + 0,5 = 1,5$.

La pérdida 0/1 no distingue 0,3 de 2, mientras que la pérdida de bisagra sigue empujando 0,3 hacia un margen confiado.

Respuesta: Pérdidas de bisagra 0,7, 0 y 1,5; pérdidas 0/1 de 0, 0 y 1

Ejercicios

1. Para un objetivo $y = -1$, calcule la pérdida de bisagra para las predicciones $\hat{y} = -3, -0,2$ y $0,4$.

Pista: Primero calcule el margen $y\hat{y}$.

Respuesta: Márgenes 3, 0,2 y $-0,4$; pérdidas de bisagra $\max(0, -2) = 0$, 0,8 y 1,4.

2. Un modelo tiene pérdida media 0,12 en su conjunto de entrenamiento y 0,15 en un conjunto de prueba; otro tiene 0,01 en entrenamiento y 0,45 en prueba. ¿Cuál sobreajusta más, y cuál es la estimación de la brecha de generalización de cada uno?

Pista: Reste la pérdida de entrenamiento a la de prueba.

Respuesta: Brechas: 0,03 y 0,44. El segundo modelo sobreajusta mucho más: casi memoriza sus datos de entrenamiento pero lo hace mal con datos no vistos.

3. Explique con sus propias palabras por qué el riesgo empírico se llama una aproximación de Monte Carlo del riesgo esperado.

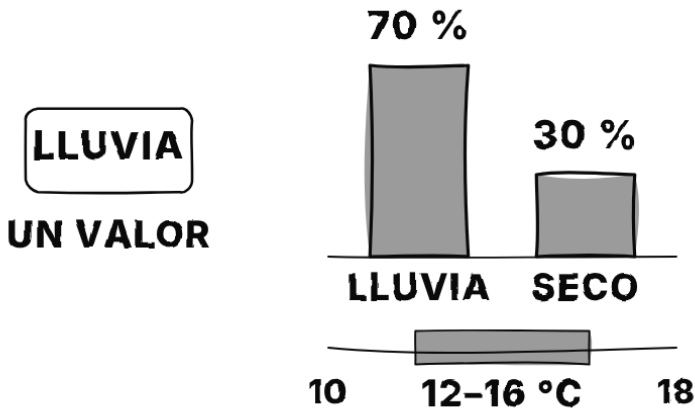
Pista: Piense sobre qué promedia el riesgo esperado y qué aporta el conjunto de datos.

Respuesta: El riesgo esperado promedia la pérdida sobre cada par entrada-salida posible, ponderado por la distribución, lo que no puede calcularse. El conjunto de datos es una muestra aleatoria de esa distribución, y promediar la pérdida sobre él estima la media completa, igual que los métodos de Monte Carlo estiman medias a partir de muestras aleatorias.

Predecir una distribución en vez de un número

Pregunte a una aplicación del tiempo por mañana y rara vez dirá simplemente «lluvia». Dirá «70 por ciento de probabilidad de lluvia», o «entre 12 y 16 grados». Esa es una clase de respuesta más rica: no un valor, sino lo probable que es cada valor. El libro muestra ahora que un modelo puede construirse para responder así, y que hacerlo nos dice qué pérdida usar.

LO PROBABLE QUE ES CADA UNO



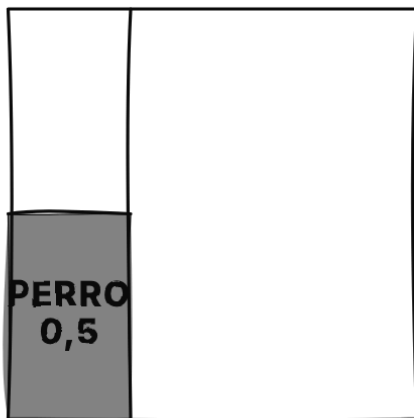
PROBABILÍSTICO

Un valor o lo probable que es cada uno

El libro sugiere que este es un buen momento para estudiar u hojear su apéndice sobre probabilidad, en especial las distribuciones, los estadísticos suficientes y la estimación por máxima verosimilitud. La pérdida, dice, codifica nuestro conocimiento previo sobre la tarea y tiene un gran efecto en el rendimiento. A veces basta un razonamiento sencillo sobre el problema para diseñar una, como con la pérdida de bisagra. Pero hay una vía más fundamentada: reformular todo el entrenamiento en términos puramente probabilísticos, un punto de vista que muchos libros prefieren y que puede ser más intuitivo o más útil en algunos contextos.

La observación clave parte del supuesto de que los ejemplos vienen de una distribución $p(x, y)$. Por la regla del producto de la probabilidad, $p(x, y) = p(x) p(y | x)$. En palabras: la probabilidad de ver juntos una entrada y una salida concretas es la probabilidad de la entrada, por la probabilidad de esa salida una vez conocida la entrada. Si el 30 por ciento de las fotos se toman al aire libre, y la mitad de las fotos al aire libre muestran un perro, la probabilidad de una foto al aire libre con un perro es 0,3 por 0,5, que es 0,15.

TODAS LAS FOTOS



AIRE LIBRE 0,3

$$p(x, y) = p(x) p(y | x) \\ = p(x | y) p(y)$$

$$0.3 \times 0.5 = 0.15$$

$p(y | x)$: discriminativo

$p(x)$: generativo

La regla del producto como áreas

El libro añade que el producto también puede separarse al revés, $p(x, y) = p(x | y) p(y)$. Los métodos que estiman $p(x)$ o $p(x | y)$ se llaman generativos, y los que estiman $p(y | x)$ se llaman discriminativos; aparte del modelado del lenguaje, este libro se centra en los discriminativos y deja el modelado generativo para su siguiente volumen.

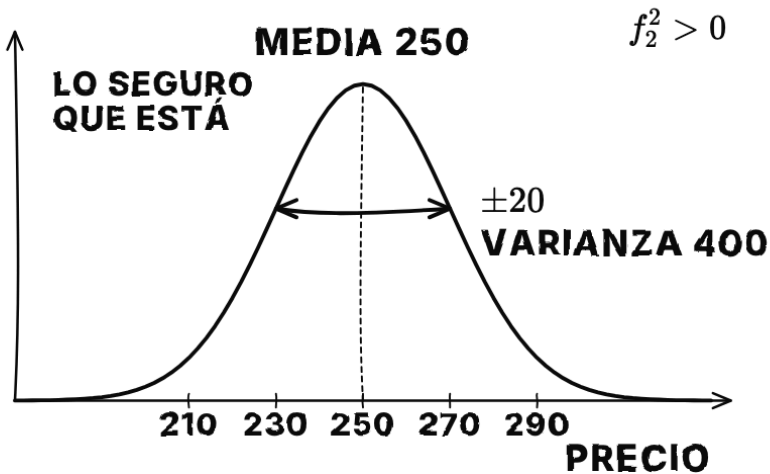
Aproximar $p(y | x)$ con una sola función $f(x)$ que produce una predicción tiene sentido solo si, para cada entrada, casi toda la probabilidad se concentra en un valor de y . Una distribución así se parece a lo que los matemáticos llaman una delta de Dirac, un pico en un solo punto, y suponerla simplifica enormemente el problema.

La relajación: que las salidas del modelo no sean la predicción en sí, sino los estadísticos suficientes, los parámetros, de una distribución condicional $p(y \mid f(x))$ sobre las salidas posibles. Para un problema de clasificación con tres clases, numeradas 1, 2 y 3, el modelo puede tener tres salidas que definen una distribución categórica: $p(y \mid f(x)) = \prod_{i=1}^3 f_i(x)^{y_i}$. Aquí $y \sim \text{Binary}(3)$ es la codificación one-hot de la clase: todo ceros salvo un 1 en la posición de la clase, y $f(x) \sim \Delta(3)$ lista la probabilidad predicha de cada clase.

Leamos la fórmula con números. Suponga que la aplicación de fotos predice (0,7, 0,2, 0,1) para perro, gato y coche, y la clase verdadera es perro, one-hot (1, 0, 0). El producto es 0,7 elevado a 1, por 0,2 elevado a 0, por 0,1 elevado a 0, y como cualquier cosa elevada a 0 es 1, el resultado es simplemente 0,7: la probabilidad que el modelo dio a la clase correcta. Si la foto fuera en realidad un coche, one-hot (0, 0, 1), la misma fórmula escogería 0,1. Los exponentes funcionan como un interruptor que selecciona la probabilidad de la clase verdadera.

Para la regresión, predecir un número real $y \in \mathbb{R}$, el modelo puede producir dos valores, $f(x) \sim (2)$, la media y la dispersión de una gaussiana, una curva de campana: $p(y \mid f(x)) = \mathcal{N}(y \mid f_1(x), f_2^2(x))$. La segunda salida se eleva al cuadrado para que la varianza sea siempre positiva. Si un modelo de precios de casas produce 250 y menos 20, la previsión es una curva de campana centrada en 250 con varianza 400, una dispersión típica de 20; el signo menos desaparece al elevar al cuadrado.

$$p(y | f(x)) = \mathcal{N}(y | f_1(x), f_2^2(x))$$



Una curva de campana para un precio

Este planteamiento es muy general e incluye todo lo dicho hasta ahora. También da al diseñador más libertad, porque elegir una forma para $p(y | x)$ puede ser más fácil que inventar una pérdida $l(y, \hat{y})$ directamente. Y modela la incertidumbre de forma natural: la varianza de la gaussiana dice lo seguro que está el modelo, igual que el porcentaje de la aplicación del tiempo.

Pruébalo

Escriba tres probabilidades que sumen uno, digamos 0,6, 0,3 y 0,1, para tres respuestas posibles a una pregunta de un cuestionario. Elija la respuesta verdadera y escriba su vector one-hot, como (0, 1, 0). Eleve cada probabilidad al 0 o al 1 correspondiente y multiplique: obtiene 0,3, la probabilidad que asignó a la verdad, que es el número que las próximas partes intentarán hacer grande.

Para recordar

La pérdida lleva nuestro conocimiento previo de la tarea; en vez de diseñarla a mano, podemos reformular el entrenamiento en términos probabilísticos. Por la regla del producto, $p(x, y) = p(x)p(y | x)$; estimar $p(y | x)$ es discriminativo, estimar $p(x)$ o $p(x | y)$ es generativo, y el libro se centra en lo discriminativo. Un modelo puede producir los parámetros de una distribución: tres probabilidades de clase para una categórica, donde los exponentes one-hot escogen la probabilidad de la clase verdadera, 0,7 para un perro predicho con 0,7. Para la regresión puede producir la media de una gaussiana y una dispersión al cuadrado, lo que mantiene positiva la varianza y expresa la incertidumbre del modelo.

Preguntas

1. Un clasificador produce las probabilidades (0,7, 0,2, 0,1) y la clase verdadera es la tercera. ¿Qué da la fórmula categórica $\Pi_i f_i(x)^{y_i}$?
 - a. 0,7
 - b. 0,1
 - c. 1,0
2. En el modelo de regresión gaussiano, ¿por qué se eleva al cuadrado la segunda salida?
 - a. Para hacer la media más grande
 - b. Para asegurar que la varianza predicha sea siempre positiva
 - c. Para convertir la regresión en clasificación
3. ¿Cuándo tiene sentido predecir un solo valor $f(x)$ en vez de toda una distribución sobre y ?
 - a. Cuando, para cada entrada, casi toda la probabilidad se concentra en un valor de y , como una delta de Dirac
 - b. Cuando el conjunto de datos es muy grande
 - c. Cuando la tarea tiene más de dos clases
4. ¿Qué clase de método estima $p(y | x)$?
 - a. Generativo
 - b. No supervisado
 - c. Discriminativo

Respuestas

1. **b)** El vector one-hot (0, 0, 1) pone exponente 1 solo en la tercera probabilidad, y cualquier cosa elevada a 0 es 1, así que el producto escoge 0,1, la probabilidad dada a la clase verdadera.

- 2. b)** Una varianza no puede ser negativa; elevar al cuadrado la segunda salida, $f_2^2(x)$, la mantiene positiva sea lo que sea lo que produzca el modelo, así que -20 da 400 .
- 3. a)** Una sola predicción es un buen resumen solo si $p(y \mid x)$ está concentrada alrededor de un punto; si no, modelar la distribución captura la incertidumbre.
- 4. c)** Los métodos que estiman $p(x)$ o $p(x \mid y)$ son generativos, los que estiman $p(y \mid x)$ discriminativos; aparte del modelado del lenguaje, el libro se centra en estos últimos.

Ejemplos resueltos

La fórmula categórica como interruptor Un modelo predice $f(x) = (0,7, 0,2, 0,1)$ para perro, gato, coche. Calcule $p(y \mid f(x)) = \prod_i f_i(x)^{y_i}$ cuando la clase verdadera es perro, y cuando es coche.

Perro tiene one-hot $(1, 0, 0)$: $0,7^1 \times 0,2^0 \times 0,1^0$.

Cualquier número elevado a 0 es 1, así que el producto es 0,7.

Coche tiene one-hot $(0, 0, 1)$: $0,7^0 \times 0,2^0 \times 0,1^1 = 0,1$.

Respuesta: 0,7 para perro, 0,1 para coche: la probabilidad que el modelo dio a la clase verdadera

La regla del producto con números El 30% de las fotos son al aire libre, y la mitad de las fotos al aire libre muestran un perro. ¿Cuál es la probabilidad de una foto al aire libre que muestre un perro?

$p(x) = 0,3$ para una foto al aire libre.

$p(y \mid x) = 0,5$ para un perro dada una foto al aire libre.

Regla del producto: $p(x, y) = p(x) p(y \mid x) = 0,3 \times 0,5$.

Respuesta: 0,15

Ejercicios

1. Un modelo con cuatro clases predice (0,1, 0,4, 0,4, 0,1); la clase verdadera es la segunda. Escriba su vector one-hot y calcule $\Pi_i f_i(x)^{\wedge} y_i$.

Pista: Solo el exponente de la clase verdadera es 1.

Respuesta: One-hot (0, 1, 0, 0); el producto es 0,4.

2. Un modelo de regresión para temperaturas produce $f(x) = (14, -3)$. Describa la distribución predicha.

Pista: La primera salida es la media; la segunda se eleva al cuadrado para dar la varianza.

Respuesta: Una gaussiana centrada en 14 con varianza $(-3)^2 = 9$, es decir, una dispersión típica de 3 grados.

3. Explique por qué predecir una distribución puede ser más útil que predecir un número, con un ejemplo de la vida cotidiana.

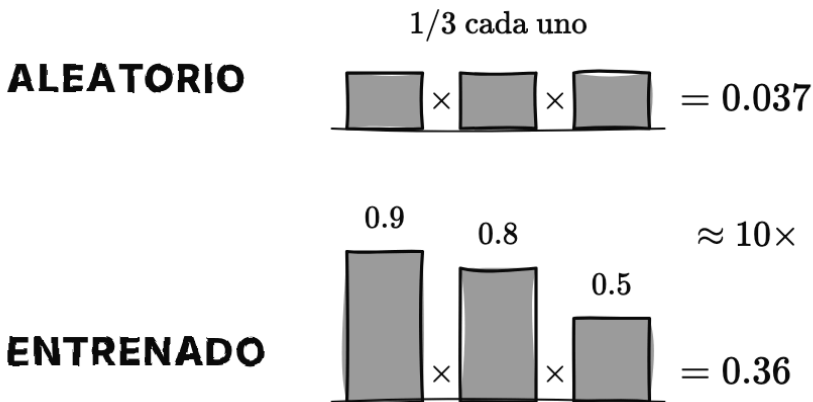
Pista: Piense en lo que una previsión del tiempo le dice más allá del resultado más probable.

Respuesta: Una distribución también dice lo segura que es la predicción: «70% de probabilidad de lluvia» o un rango de temperaturas le permiten decidir si llevar paraguas, mientras que un simple «lluvia» esconde la incertidumbre. La varianza de la gaussiana cumple ese papel en el modelo de regresión del libro.

Hacer probables los datos, y conservar todo modelo bueno

Imagine dos meteorólogos que hicieron ambos la previsión de la semana pasada. Para cada día busca la probabilidad que cada uno dio al tiempo que realmente hizo, y multiplica esas probabilidades entre sí. El meteorólogo con el producto mayor hizo que la semana pasada pareciera lo menos sorprendente, y en ese confiaría más. Ese producto es la verosimilitud, y elegir el modelo que lo hace máximo es como se entrenan los modelos probabilísticos.

$$p(\mathcal{S}_n | f) = \prod_{i=1}^n p(y_i | f(x_i))$$



Verosimilitud: multiplicar las probabilidades

Recuerde el supuesto de que el conjunto de datos $\mathcal{S}_n$ es un conjunto de muestras independientes e idénticamente distribuidas. Como las muestras son independientes, la probabilidad que un modelo dado f asigna a todo el conjunto de datos es el producto sobre los ejemplos: $p(\mathcal{S}_n | f) = \prod_{i=1}^n p(y_i | f(x_i))$. Esta cantidad se llama la verosimilitud del conjunto de datos. Un modelo elegido al azar reparte su probabilidad más o menos uniformemente entre todas las salidas posibles, así que da a nuestros datos concretos una verosimilitud pequeña.

Tome la aplicación de fotos con tres clases y tres fotos etiquetadas. Un modelo aleatorio da a cada clase verdadera una probabilidad de un tercio, así que la verosimilitud es un tercio al cubo, unos 0,037. Un modelo entrenado da a las clases verdaderas 0,9, 0,8 y 0,5, así que la verosimilitud es 0,9 por 0,8 por 0,5, que es 0,36, unas diez veces mayor.

La estrategia sensata es por tanto elegir el modelo que hace el conjunto de datos lo más probable posible, que es el principio de máxima verosimilitud. La definición: dado un conjunto de datos y una familia de distribuciones $p(y | f(x))$ parametrizada por f , la solución de máxima verosimilitud es $f^* = \operatorname{argmax}_f \prod_{i=1}^n p(y_i | f(x_i))$. Es otra vez un problema de optimización, pero ahora, una vez elegidas las distribuciones, se sigue de las leyes de la probabilidad, mientras que antes la pérdida era algo que teníamos que diseñar.

MÁXIMA VEROSIMILITUD

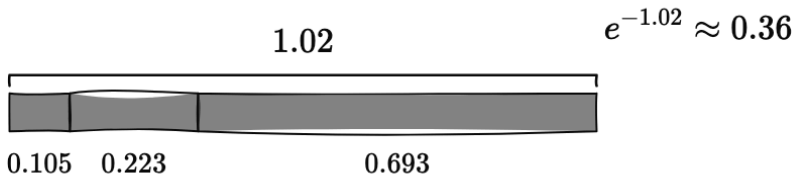
$$f^* = \arg \max_f \prod_{i=1}^n p(y_i | f(x_i))$$

log
↓

$$\arg \min_f \sum_{i=1}^n -\log p(y_i | f(x_i))$$

PSEUDOPÉRDIDA

$$(1/2)^{1000} \approx 10^{-301} \quad \mathbf{X} \quad 1000 \times 0.693 = 693$$

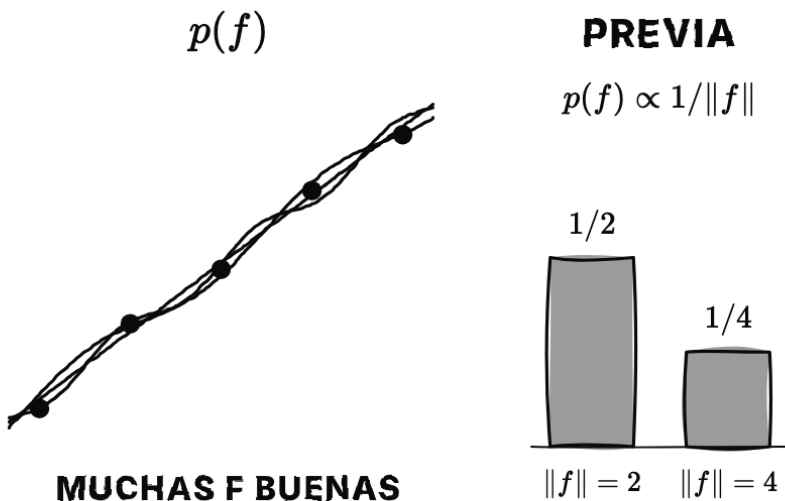


De un producto a una suma de logaritmos

Las dos visiones están estrechamente conectadas. Tome logaritmos, que convierten productos en sumas, y pase de maximizar a minimizar cambiando el signo: $\arg \max_f \log \prod_i p(y_i | f(x_i)) = \arg \min_f \sum_i -\log p(y_i | f(x_i))$. El logaritmo conserva el orden de los números, así que el mejor modelo es el mismo, y las sumas son mucho más amables con los computadores que los productos largos: mil factores de un medio se multiplican hasta un número con unos trescientos ceros tras la coma, mientras que sus menos logaritmos simplemente suman unos 693. Para nuestro modelo entrenado, menos logaritmo de 0,9, 0,8 y 0,5 son unos 0,105, 0,223 y 0,693, que suman unos 1,02, y e elevado a menos 1,02 es en efecto unos 0,36. Así que las dos formulaciones coinciden si tratamos $-\log p(y | f(x))$ como una pseudopérdida, y el

libro dice que toda pérdida usada en la práctica surge así, de una elección concreta de ese término.

La sección 3.3 va un paso más allá, hacia el aprendizaje bayesiano y las redes neuronales bayesianas, de las que el libro da solo la idea general. Una probabilidad sobre las salidas ya permite a un modelo expresar varias predicciones plausibles. Pero el entrenamiento sigue devolviendo una sola función f , mientras que muchos ajustes distintos de parámetros pueden ser todos buenos, y tenerlos todos podría dar predicciones más fieles.



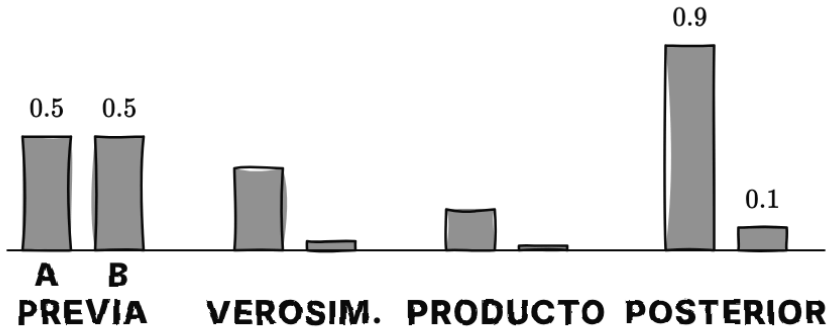
Muchos modelos buenos, y una previa

De nuevo la solución es otra distribución de probabilidad, esta vez sobre funciones. Partimos de una previa $p(f)$, nuestra preferencia

antes de ver ningún dato; como f se describe por un conjunto finito de parámetros, en la práctica esto es una previa sobre los pesos. Por ejemplo, las funciones con menor norma suelen preferirse porque son más estables, y entonces podría tomarse $p(f) \propto 1/\|f\|$: una función de norma 2 sería a priori el doble de probable que una de norma 4.

Tras observar los datos, la probabilidad sobre las funciones se desplaza según la previa y la verosimilitud, por el teorema de Bayes: $p(f | \mathcal{S}_n) = p(\mathcal{S}_n | f) p(f) / p(\mathcal{S}_n)$. El lado izquierdo es la posterior, la creencia después de los datos; el denominador, la evidencia, solo hace que la posterior sume uno. Suponga que hay solo dos modelos candidatos, A y B, igual de probables a priori, con verosimilitudes 0,36 y 0,04. Previa por verosimilitud da 0,18 y 0,02; la evidencia es su suma, 0,2; así que la posterior es 0,9 para A y 0,1 para B.

$$p(f \mid \mathcal{S}_n) = \frac{p(\mathcal{S}_n \mid f) p(f)}{p(\mathcal{S}_n)}$$



$$0.9 \times 0.8 + 0.1 \times 0.3 = 0.75$$

Bayes con dos modelos

A diferencia del único mejor modelo, la posterior puede seguir prefiriendo varias funciones a la vez, lo que puede predecir mejor. Para una entrada nueva promediamos sobre los modelos, cada uno ponderado por su posterior: $p(y \mid x) = \int p(y \mid f(x)) p(f \mid \mathcal{S}_n) df$, que en la práctica se aproxima promediando sobre una muestra finita de k modelos. Con nuestros dos modelos, si A dice «perro» con probabilidad 0,8 y B con 0,3, la predicción promediada es 0,9 por 0,8 más 0,1 por 0,3, que es 0,75: sobre todo la visión de A, suavizada un poco por la duda de B.

Pruébalo

Escriba tres días de tiempo y las probabilidades de dos meteorólogos para lo que realmente ocurrió, por ejemplo 0,9, 0,8 y 0,5 para el primero y 0,4, 0,5 y 0,2 para el segundo.

Multiplique los tres números de cada meteorólogo: 0,36 frente a 0,04. Luego sume en cambio los menos logaritmos en una calculadora: unos 1,02 frente a 3,22; la suma menor pertenece al mismo meteorólogo que el producto mayor, y por eso maximizar la verosimilitud y minimizar la pérdida logarítmica son la misma elección.

Para recordar

Para datos i.i.d., la verosimilitud de un conjunto de datos bajo un modelo es el producto de las probabilidades que da a cada salida verdadera, 0,36 para un modelo entrenado frente a 0,037 para uno aleatorio. La máxima verosimilitud elige el modelo que hace los datos más probables; tomar logaritmos convierte esto en minimizar la suma de $-\log p(y_i | f(x_i))$, una pseudopérdida de la que derivan las pérdidas prácticas. El aprendizaje bayesiano conserva una distribución sobre los modelos: una previa $p(f)$, actualizada por el teorema de Bayes en una posterior, con la evidencia como normalizador, 0,9 y 0,1 para nuestros dos modelos. Las predicciones promedian entonces todos los modelos ponderados por la posterior, aproximada por una muestra finita de modelos, como 0,75 a partir de 0,8 y 0,3.

Preguntas

1. ¿Por qué la verosimilitud de un conjunto de datos puede escribirse como un producto sobre sus ejemplos?
 - a. Porque los ejemplos se suponen independientes, así que sus probabilidades se multiplican
 - b. Porque el modelo tiene un parámetro por ejemplo
 - c. Porque la pérdida es siempre un producto
2. ¿Por qué maximizar la verosimilitud es lo mismo que minimizar la suma de $-\log p(y_i | f(x_i))$?
 - a. Porque el logaritmo invierte el orden de los números
 - b. Porque el logaritmo conserva el orden de los números y convierte el producto en una suma, y el signo menos convierte maximizar en minimizar
 - c. Porque ambos son siempre iguales a cero en el mejor modelo
3. ¿Qué conserva el aprendizaje bayesiano que la máxima verosimilitud desecha?
 - a. Los propios datos de entrenamiento
 - b. La función de pérdida
 - c. Toda una distribución sobre los modelos, de modo que varias funciones buenas pueden contribuir a las predicciones
4. En el teorema de Bayes, ¿cuál es el papel de la evidencia $p(S_n)$?
 - a. Normaliza la posterior para que sume uno
 - b. Es la preferencia previa por normas pequeñas
 - c. Es la verosimilitud del mejor modelo

Respuestas

1. **a)** Las muestras son i.i.d., así que la probabilidad que el modelo asigna a todo el conjunto de datos es el producto de las probabilidades que asigna a cada salida verdadera.
2. **b)** El logaritmo es creciente, así que el mejor modelo no cambia; convierte los productos en sumas, y cambiar el signo da una minimización de una pseudopérdida.
3. **c)** La máxima verosimilitud devuelve una función, mientras que la posterior puede preferir varias funciones a la vez, y las predicciones las promedian por su peso posterior.
4. **a)** La posterior es la verosimilitud por la previa dividido por la evidencia, lo que asegura que la posterior sea una distribución propia, como 0,18 y 0,02 divididos por 0,2 dieron 0,9 y 0,1.

Ejemplos resueltos

Verosimilitud y pérdida logarítmica Un clasificador da a las clases verdaderas de tres fotos probabilidades 0,9, 0,8 y 0,5. Calcule la verosimilitud y la log-verosimilitud negativa, y compárelas con un modelo aleatorio que da $1/3$ a cada una.

Verosimilitud: $0,9 \times 0,8 \times 0,5 = 0,36$.

Log-verosimilitud negativa: $-\ln 0,9 - \ln 0,8 - \ln 0,5 \approx 0,105 + 0,223 + 0,693 \approx 1,02$.

Comprobación: $e^{(-1,02)} \approx 0,36$.

Modelo aleatorio: $(1/3)^3 \approx 0,037$, unas diez veces menor.

Respuesta: Verosimilitud 0,36, log-verosimilitud negativa de unos 1,02; el modelo aleatorio obtiene unos 0,037

Una posterior sobre dos modelos Los modelos A y B tienen previas iguales 0,5 y verosimilitudes 0,36 y 0,04. Halle sus

posteriores, y la probabilidad promediada de «perro» si A predice 0,8 y B predice 0,3.

Prevía por verosimilitud: $0,5 \times 0,36 = 0,18$ y $0,5 \times 0,04 = 0,02$.

Evidencia: $0,18 + 0,02 = 0,2$.

Posteriores: $0,18 / 0,2 = 0,9$ para A y $0,02 / 0,2 = 0,1$ para B.

Predicción promediada: $0,9 \times 0,8 + 0,1 \times 0,3 = 0,72 + 0,03 = 0,75$.

Respuesta: Posteriores 0,9 y 0,1; probabilidad promediada de perro 0,75

Ejercicios

1. Un modelo da a las salidas verdaderas de cuatro ejemplos probabilidades 0,5, 0,5, 0,8 y 0,25. Calcule la verosimilitud.

Pista: Multiplique los cuatro números.

Respuesta: $0,5 \times 0,5 \times 0,8 \times 0,25 = 0,05$.

2. Los modelos A y B tienen previas 0,7 y 0,3, y verosimilitudes 0,1 y 0,4. Calcule sus posteriores.

Pista: Multiplique la previa por la verosimilitud para cada uno, luego divida por la suma.

Respuesta: 0,07 y 0,12; evidencia 0,19; posteriores $\approx 0,37$ para A y $\approx 0,63$ para B. Los datos desplazaron la creencia de A hacia B.

3. Con la previa $p(f) \propto 1/\|f\|$, ¿cuánto más probable a priori es una función de norma 1 que una de norma 5?

Pista: Compare $1/1$ con $1/5$.

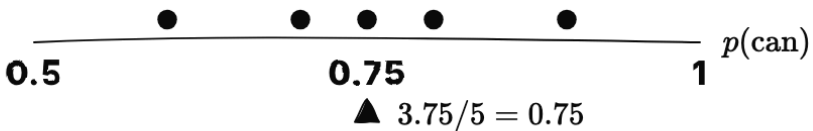
Respuesta: Cinco veces más probable, ya que $1/1$ es cinco veces $1/5$; las funciones de menor norma se prefieren por ser más estables.

Aproximaciones, previas como penalizaciones y aprendizaje a plazos

Pida a cinco amigos que estimen cuánto durará un paseo: 20, 25, 30, 22 y 28 minutos. Su media, 25, suele ser una mejor suposición que cualquiera de ellas por sí sola. La predicción bayesiana es esa idea hecha precisa: preguntar a muchos modelos plausibles y promediarlos, y esta parte muestra cómo se hace en la práctica, y qué métodos más sencillos se esconden dentro.



K MODELOS DE LA POSTERIOR



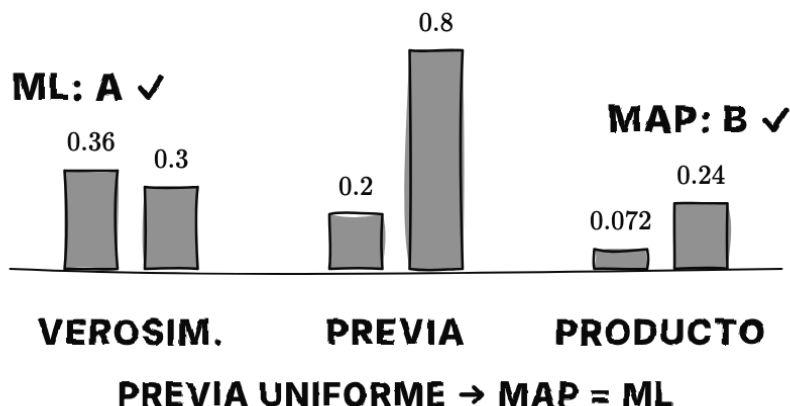
POSTERIOR: RARA VEZ EXACTA MCMC · VARIACIONAL

Promediar muchas suposiciones

La predicción promediada sobre todos los modelos es una integral, y en la práctica se sustituye por una media de Monte Carlo sobre k modelos extraídos al azar de la posterior, $f_k \sim p(f \mid \mathcal{S}_n)$. Si cinco modelos muestreados dan a una foto la probabilidad de ser un perro de 0,8, 0,7, 0,9, 0,6 y 0,75, la predicción promediada es su suma, 3,75, dividida por 5, que es 0,75. La belleza de este planteamiento, dice el libro, la estropea un hecho: la posterior casi nunca puede calcularse exactamente, salvo para elecciones muy particulares de previa y verosimilitud. Hay que aproximarla, bien con métodos de Monte Carlo con cadenas de Markov, bien con inferencia variacional, y la sección 9.3.1 mostrará un tratamiento bayesiano de los parámetros de un modelo, el dropout de Monte Carlo.

El libro cierra el capítulo con dos observaciones. Primera, suponga que solo queremos la única función con la mayor densidad posterior. Entonces la evidencia puede ignorarse, y el problema se separa en dos términos: $f^* = \operatorname{argmax}_f p(\mathcal{S}_n \mid f) p(f) = \operatorname{argmax}_f [\log p(\mathcal{S}_n \mid f) + \log p(f)]$. El primer término es la verosimilitud, el segundo un término de regularización, y la respuesta se llama solución de máximo a posteriori, o MAP.

$$f^* = \arg \max_f [\log p(\mathcal{S}_n | f) + \log p(f)]$$



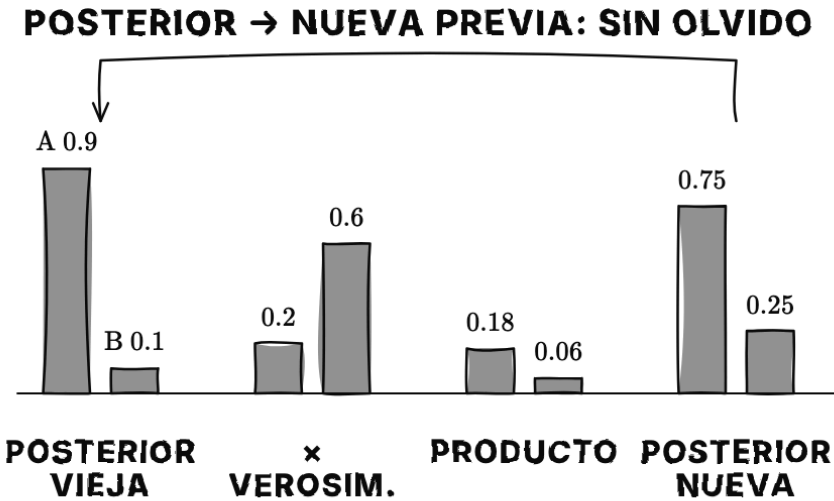
Verosimilitud o verosimilitud por previa

Compare las dos con números. El modelo A explica los datos con verosimilitud 0,36 pero tiene previa 0,2; el modelo B tiene verosimilitud 0,30 y previa 0,8. La máxima verosimilitud elige A, ya que 0,36 supera a 0,30. MAP multiplica por la previa: 0,36 por 0,2 es 0,072, mientras que 0,30 por 0,8 es 0,24, así que MAP elige B, el modelo que ajusta un poco peor pero es mucho más plausible de antemano.

Si toda función tiene el mismo peso previo, una previa uniforme, el segundo término es constante y MAP se reduce a la máxima verosimilitud. En otro caso MAP penaliza las funciones que se alejan demasiado de la previa, lo que ayuda contra el sobreajuste y nos permite imponer restricciones a f . Por eso $\log p(f)$ se llama un

regularizador: tira de la solución hacia la región que la previa favorece. Una nota al pie enlaza esto con las dos interpretaciones de la probabilidad: a grandes rasgos, la máxima verosimilitud trata los parámetros como incógnitas fijas y los datos como aleatorios, la visión frecuentista, mientras que la visión bayesiana trata los datos como fijos y los parámetros como aleatorios.

Segunda observación: el tratamiento bayesiano completo absorbe nuevos datos con facilidad. Dado un nuevo conjunto de datos S'_n de la misma distribución, usamos la posterior calculada con los primeros datos como nueva previa, y la actualizamos con la nueva verosimilitud. El libro sugiere ver la previa original como la posterior después de no observar nada en absoluto.



La posterior de ayer, la previa de hoy

Tome los modelos A y B con posteriores 0,9 y 0,1 tras el primer lote. Un nuevo lote tiene verosimilitud 0,2 bajo A y 0,6 bajo B: multiplicando da 0,18 y 0,06, cuya suma es 0,24, así que las nuevas posteriores son 0,75 y 0,25. La creencia se movió hacia B, pero el primer lote no se olvidó: A sigue por delante. Por eso la visión bayesiana puede aliviar un problema bien conocido de los modelos entrenados en línea, con datos que llegan con el tiempo: el olvido catastrófico, en el que aprender de los datos nuevos borra lo aprendido antes. Un asistente de chat actualizado solo con las conversaciones de este mes, sin que nada se conserve, estaría expuesto exactamente a ese riesgo.

Pruébelo

Sostenga dos monedas y decida de antemano que cree 9 a 1 que la primera es la justa. Láncelas unas cuantas veces y, para cada moneda, multiplique su creencia por lo probables que parecen sus resultados, luego reescale los dos números para que sumen uno. Hágalo otra vez tras más lanzamientos, partiendo de las creencias que acaba de calcular y no de 9 a 1: eso es la posterior convirtiéndose en la siguiente previa.

Para recordar

Las predicciones bayesianas promedian sobre modelos extraídos de la posterior, una aproximación de Monte Carlo como 0,75 a partir de cinco muestras, pero la propia posterior suele necesitar aproximación por Monte Carlo con cadenas de Markov o inferencia variacional. Conservar solo la función más probable da la solución MAP, el $\arg \max$ del logaritmo de la verosimilitud más el logaritmo de la previa, donde el logaritmo de la previa actúa como regularizador; con una previa uniforme es igual a la máxima verosimilitud. MAP puede preferir un modelo que ajusta un poco peor pero es más plausible a priori, como B sobre A con 0,24 frente a 0,072. Los datos nuevos se absorben usando la posterior antigua como nueva previa, lo que ayuda contra el olvido catastrófico en el entrenamiento en línea.

Preguntas

1. ¿Cuándo se reduce la solución MAP a la solución de máxima verosimilitud?
 - a. Cuando la previa es uniforme, dando a toda función el mismo peso
 - b. Cuando el conjunto de datos es muy pequeño
 - c. Cuando la posterior se calcula exactamente
2. ¿Por qué $\log p(f)$ se llama un regularizador?
 - a. Porque hace mayor la verosimilitud

- b. Porque tira de la solución hacia las funciones que la previa favorece, penalizando las que se alejan mucho de ella
 - c. Porque quita la evidencia del teorema de Bayes
3. ¿Cómo absorbe el tratamiento bayesiano un nuevo lote de datos?
- a. Reentrenando desde la previa original solo con el nuevo lote
 - b. Promediando los conjuntos de datos antiguo y nuevo
 - c. Usando la posterior de los datos antiguos como previa para los datos nuevos
4. ¿Por qué las predicciones bayesianas suelen tener que aproximarse?
- a. Porque la posterior rara vez puede calcularse en forma cerrada
 - b. Porque la previa es siempre desconocida
 - c. Porque las medias de Monte Carlo son exactas

Respuestas

- 1. a)** MAP maximiza el logaritmo de la verosimilitud más el logaritmo de la previa; con una previa uniforme el segundo término es una constante, y queda la máxima verosimilitud.
- 2. b)** El logaritmo de la previa penaliza las funciones que se desvían de la previa, lo que ayuda contra el sobreajuste e impone restricciones a f.
- 3. c)** La posterior anterior se convierte en la creencia de partida, actualizada con la nueva verosimilitud, así que la información antigua se conserva, lo que ayuda contra el olvido catastrófico.
- 4. a)** Salvo para previas y verosimilitudes muy concretas, la posterior no tiene forma cerrada, así que se usa Monte Carlo con cadenas de Markov o

inferencia variacional, y se promedia sobre modelos muestreados.

Ejemplos resueltos

Máxima verosimilitud frente a MAP El modelo A tiene verosimilitud 0,36 y previa 0,2; el modelo B tiene verosimilitud 0,30 y previa 0,8. ¿Cuál elige la máxima verosimilitud, y cuál elige MAP?

La máxima verosimilitud compara verosimilitudes: $0,36 > 0,30$, así que elige A.

MAP compara verosimilitud por previa: A da $0,36 \times 0,2 = 0,072$.

B da $0,30 \times 0,8 = 0,24$.

$0,24 > 0,072$, así que MAP elige B.

Respuesta: La máxima verosimilitud elige A; MAP elige B

Actualizar a plazos Tras un primer lote, los modelos A y B tienen posteriores 0,9 y 0,1. Un nuevo lote tiene verosimilitud 0,2 bajo A y 0,6 bajo B. Calcule las posteriores actualizadas.

Use la posterior antigua como nueva previa.

Previa por verosimilitud: $0,9 \times 0,2 = 0,18$ y $0,1 \times 0,6 = 0,06$.

Evidencia: $0,18 + 0,06 = 0,24$.

Posteriores: $0,18 / 0,24 = 0,75$ y $0,06 / 0,24 = 0,25$.

Respuesta: 0,75 para A y 0,25 para B

Ejercicios

1. Cuatro modelos muestreados de la posterior predicen probabilidades 0,6, 0,9, 0,7 y 0,8 de que un correo sea spam. ¿Cuál es la predicción de Monte Carlo?

Pista: Promedie las cuatro predicciones.

Respuesta: $(0,6 + 0,9 + 0,7 + 0,8) / 4 = 3,0 / 4 = 0,75$.

2. Con una previa uniforme sobre dos modelos con verosimilitudes 0,4 y 0,1, ¿qué modelo elige MAP, y coincide con la máxima verosimilitud?

Pista: Una previa uniforme multiplica ambos por el mismo número.

Respuesta: Ambos se multiplican por la misma previa, así que MAP elige el modelo con verosimilitud 0,4, exactamente como hace la máxima verosimilitud.

3. Explique el olvido catastrófico y cómo usar la posterior como siguiente previa puede ayudar.

Pista: Piense en un modelo entrenado con datos que llegan con el tiempo.

Respuesta: Cuando un modelo entrenado en línea aprende solo de los datos nuevos, puede perder lo que aprendió antes. Usar la posterior de los datos anteriores como previa para los nuevos arrastra la información antigua, así que la actualización desplaza la creencia sin borrarla.