capitulo 19

Aprendiendo de ejemplos

En el que describimos agentes que pueden mejorar su comportamiento a través del estudio diligente del pasado.

Experiencias y predicciones sobre el futuro.

un agente es**aprendizaje**si mejora su desempeño después de hacer observaciones sobre el mundo. El aprendizaje puede variar desde lo trivial, como anotar una lista de compras, hasta lo profundo, como cuando Albert Einstein infirió una nueva teoría del universo. Cuando el agente es una computadora, lo llamamos**aprendizaje automático**: una computadora observa algunos datos, construye un**modelo** basado en los datos, y usa el modelo como un**hipótesis**sobre el mundo y un pedazo de software que puede resolver problemas.

Aprendizaje automático

¿Por qué querríamos que una máquina aprenda? ¿Por qué no simplemente programarlo de la manera correcta para comenzar? ¿con? Hay dos razones principales. Primero, los diseñadores no pueden anticipar todo el futuro posible. situaciones Por ejemplo, un robot diseñado para navegar por laberintos debe aprender el diseño de cada uno. nuevo laberinto que encuentra; un programa para predecir los precios del mercado de valores debe aprender a adaptarse cuando las condiciones cambian de auge a caída. En segundo lugar, a veces los diseñadores no tienen idea cómo programar una solución ellos mismos. La mayoría de las personas son buenas para reconocer las caras de miembros de la familia, pero lo hacen inconscientemente, por lo que incluso los mejores programadores no saben cómo programar una computadora para realizar esa tarea, excepto mediante el uso de aprendizaje automático algoritmos

En este capítulo, intercalamos una discusión de varias clases de modelos: árboles de decisión (Sección 19.3-), modelos lineales (Sección 19.6-), modelos no paramétricos como vecinos más cercanos (Sección 19.7-), modelos de conjuntos como bosques aleatorios (Sección 19.8-)—con prácticas

consejos sobre la construcción de sistemas de aprendizaje automático (Sección 19.9) y discusión de la teoría de aprendizaje automático (Secciones 19.1 a 19.5).

19.1 Formas de aprendizaje

Cualquier componente de un programa de agente se puede mejorar mediante el aprendizaje automático. los
Las mejoras y las técnicas utilizadas para realizarlas dependen de estos factores:

- Cual*componente*es para mejorar.
- Qué*conocimiento previo*que tiene el agente, que influye en la*modelo*construye
- Qué*datos y retroalimentación*en que los datos están disponibles.

Capitulo 2describió varios diseños de agentes. los**componentes**de estos agentes incluyen:

1.Un mapeo directo de las condiciones del estado actual a las acciones.

2.Un medio para inferir propiedades relevantes del mundo a partir de la secuencia de percepción.

3.Información sobre la forma en que evoluciona el mundo y sobre los resultados de posibles acciones que el agente puede tomar.

4.Información de utilidad que indica la conveniencia de los estados mundiales.

5.Información de valor de acción que indica la conveniencia de las acciones.

6.Metas que describen los estados más deseables.

7.Un elemento generador de problemas, crítico y de aprendizaje que permite al sistema mejorar.

Cada uno de estos componentes se puede aprender. Considere un agente de autos sin conductor que aprende observar a un conductor humano. Cada vez que el conductor frena, el agente puede aprender una condición: regla de acción para cuándo frenar (componente 1). Al ver muchas imágenes de cámara que se dice contienen autobuses, puede aprender a reconocerlos (componente 2). Intentando acciones y
Al observar los resultados, por ejemplo, frenar con fuerza en una carretera mojada, puede aprender los efectos de su acciones (componente 3). Luego, cuando recibe quejas de los pasajeros que han sido completamente sacudido durante el viaje, puede aprender un componente útil de su utilidad general función (componente 4).

La tecnología de aprendizaje automático se ha convertido en una parte estándar de la ingeniería de software.
Siempre que esté construyendo un sistema de software, incluso si no lo considera un agente de IA,
Los componentes del sistema se pueden mejorar potencialmente con el aprendizaje automático. Por ejemplo, software para analizar imágenes de galaxias bajo lentes gravitacionales fue acelerado por un

factor de 10 millones con un modelo de aprendizaje automático (Hezaveh*et al.*, 2017), y uso de energía para la refrigeración de los centros de datos se redujo en un 40 % con otro modelo de aprendizaje automático (Gao, 2014). El ganador del Premio Turing, David Patterson, y el jefe de Google AI, Jeff Dean, declararon el amanecer de un "Edad de oro" para la arquitectura informática debido al aprendizaje automático (Decano*et al.*, 2018).

Hemos visto varios ejemplos de modelos para componentes de agentes: atómico, factorizado y modelos relacionales basados en lógica o probabilidad, etc. Los algoritmos de aprendizaje han sido pensado para todos ellos.

Este capítulo asume poco**conocimiento previo**por parte del agente: se parte de cero y aprende de los datos. EnSección 21.7.2consideramos**transferir el aprendizaje**, en el cual el conocimiento de un dominio se transfiere a un nuevo dominio, de modo que el aprendizaje puede continuar más rápido con menos datos. Suponemos, sin embargo, que el diseñador del sistema elige un marco modelo que puede conducir a un aprendizaje efectivo.

Conocimiento previo

Pasar de un conjunto específico de observaciones a una regla general se llama**inducción**; desde el observaciones de que el sol salió todos los días en el pasado, inducimos que el sol saldrá mañana. Esto difiere de la**deducción**estudiamos enCapítulo 7porque la inductiva conclusiones pueden ser incorrectas, mientras que las conclusiones deductivas están garantizadas para ser correctas si las premisas son correctas.

Este capítulo se concentra en problemas donde la entrada es un**representación factorizada**a vector de valores de atributo. También es posible que la entrada sea cualquier tipo de estructura de datos, incluyendo atómica y relacional.

Cuando la salida es uno de un conjunto finito de valores (como*soleado/nublado/lluvioso*o*verdadero Falso*), el problema de aprendizaje se llama**clasificación**. Cuando es un número (como mañana temperatura, medida ya sea como un número entero o un número real), el problema de aprendizaje tiene la (ciertamente oscuro1) nombre**regresión**.

1Un mejor nombre hubiera sido*aproximación de funciones*o*predicción numérica*. Pero en 1886 Francis Galton escribió un influyente artículo sobre el concepto de*regresión a la media*(por ejemplo, es probable que los hijos de padres altos sean más altos que el promedio, pero no tan altos como el padres). Galton mostró gráficos con lo que llamó "líneas de regresión", y los lectores llegaron a asociar la palabra "regresión" con la técnica estadística de aproximación de funciones más que con el tema de la regresión a la media.

Clasificación

Regresión

Hay tres tipos de**retroalimentación**que pueden acompañar a las entradas, y que determinan la tres tipos principales de aprendizaje:

- En**aprendizaje supervisado**el agente observa pares de entrada-salida y aprende una función que mapas de entrada a salida. Por ejemplo, las entradas podrían ser imágenes de cámara, cada una acompañado de una salida que dice "autobús" o "peatón", etc. Una salida como esta se llama a**etiqueta**. El agente aprende una función que, cuando se le da una nueva imagen, predice la etiqueta apropiada. En el caso de acciones de frenado (componente 1 anterior), una entrada es el estado actual (velocidad y dirección del coche, estado de la carretera), y una salida es el distancia que tardó en detenerse. En este caso, el agente puede obtener un conjunto de valores de salida. de sus propias percepciones (después del hecho); el ambiente es el maestro, y el agente aprende una función que asigna estados a la distancia de frenado.

Aprendizaje supervisado

Etiqueta

- En**aprendizaje sin supervisión**el agente aprende patrones en la entrada sin ningún explícito retroalimentación. La tarea de aprendizaje no supervisado más común es**agrupamiento**: detectar grupos potencialmente útiles de ejemplos de entrada. Por ejemplo, cuando se muestran millones de imágenes tomadas de Internet, un sistema de visión por computadora puede identificar un gran grupo de imágenes similares que un hablante de inglés llamaría "gatos".

Aprendizaje sin supervisión

- En**aprendizaje reforzado**el agente aprende de una serie de refuerzos: recompensas y castigos Por ejemplo, al final de una partida de ajedrez se le dice al agente que ha ganado (una recompensa) o perdido (un castigo). Corresponde al agente decidir cuál de las acciones antes del refuerzo eran los principales responsables de ello, y alterar sus acciones para apuntar hacia más recompensas en el futuro.

Aprendizaje reforzado

Retroalimentación

19.2 Aprendizaje supervisado

Más formalmente, la tarea del aprendizaje supervisado es la siguiente:

> Dado un**conjunto de entrenamiento**de$n$pares de entrada-salida de ejemplo
>
> $$(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n),$$
>
> *Conjunto de entrenamiento*
>
> donde cada par fue generado por una función desconocida$y = f(x)$,descubrir una función$h$que se aproxima a la función verdadera$f$.

La función$h$se llama un**hipótesis**Acerca del mundo. Se extrae de un**espacio de hipótesis** Hde funciones posibles. Por ejemplo, el espacio de hipótesis podría ser el conjunto de polinomios de grado 3; o el conjunto de funciones de Javascript; o el conjunto de fórmulas lógicas booleanas 3-SAT.

Espacio de hipótesis

Con vocabulario alternativo, podemos decir que$h$es un**modelo**de los datos, extraídos de un**modelo clase**H,o podemos decir un**función**extraído de un**clase de función**. Llamamos a la salida$y$ila **verdad básica**—la respuesta verdadera que le estamos pidiendo a nuestro modelo que prediga.

clase de modelo

verdad básica

¿Cómo elegimos un espacio de hipótesis? Es posible que tengamos algún conocimiento previo sobre el proceso que generó los datos. Si no, podemos realizar**análisis exploratorio de datos**: examinar los datos con pruebas estadísticas y visualizaciones: histogramas, diagramas de dispersión, cuadros gráficos: para tener una idea de los datos y una idea de lo que podría ser el espacio de hipótesis adecuado. O simplemente podemos probar múltiples espacios de hipótesis y evaluar cuál funciona. mejor.

Análisis exploratorio de datos

Hipótesis consistente

¿Cómo elegimos una buena hipótesis dentro del espacio de hipótesis? Podríamos esperar a**hipótesis consistente**: $h$tal que cada uno$x_i$en el conjunto de entrenamiento tiene$h(x_i) = y_i$.Con salidas de valor continuo, no podemos esperar una coincidencia exacta con la realidad básica; en cambio nosotros busca un**función de mejor ajuste**por lo que cada$h(x_i)$esta cerca dey_i(de una manera que lo haremos formalizar enSección 19.4.2-).

La verdadera medida de una hipótesis no es cómo funciona en el conjunto de entrenamiento, sino qué tan bien maneja entradas que aún no ha visto. Podemos evaluar eso con una segunda muestra de (x_i, y_i) pares llamados**equipo de prueba**. Nosotros decimos eso$h$**generaliza**bien si predice con precisión las salidas de el conjunto de prueba.

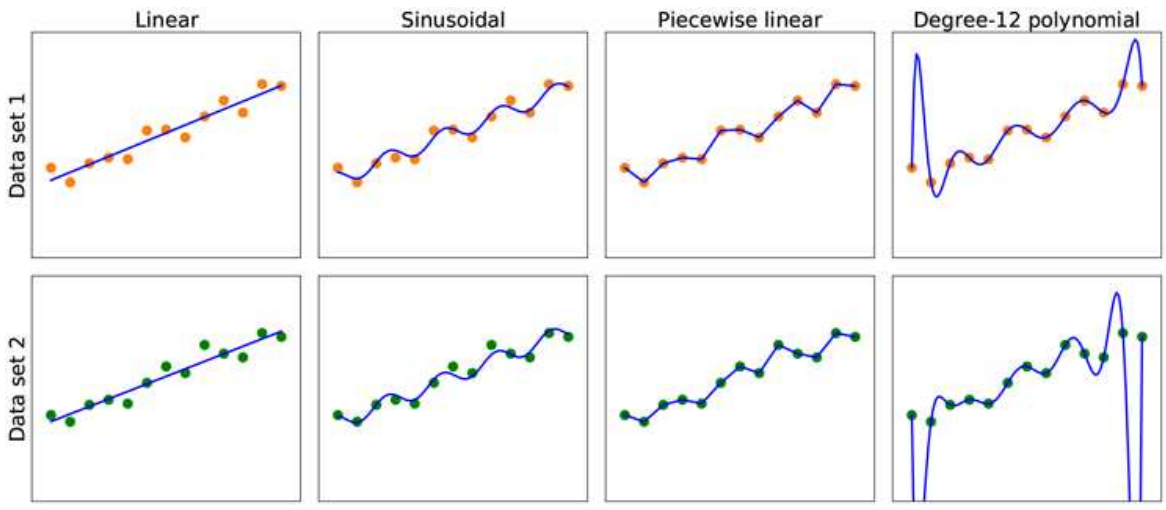
Equipo de prueba

Generalización

Figura 19.1muestra que la funciónhque descubre un algoritmo de aprendizaje depende de la espacio de hipótesisHconsidera y sobre el conjunto de entrenamiento se le da. Cada una de las cuatro parcelas en la fila superior tiene el mismo conjunto de entrenamiento de 13 puntos de datos en el (x, y)plano. Las cuatro parcelas en la fila inferior tiene un segundo conjunto de 13 puntos de datos; ambos conjuntos son representativos del mismo función desconocidaf(x).Cada columna muestra la hipótesis de mejor ajusteh de un diferente espacio de hipótesis:

- **COLUMNA 1:**Lineas rectas; funciones de la formah(x) = $w_1 x + w_0$.no hay linea que sería una hipótesis consistente para los puntos de datos.
- **COLUMNA 2:**Funciones sinusoidales de la formah(x) = $w_1 x + \text{sen}(w_0 x)$.Esta elección no es bastante consistente, pero se ajusta muy bien a ambos conjuntos de datos.
- **COLUMNA 3:**Funciones lineales por partes donde cada segmento de línea conecta los puntos de un punto de datos al siguiente. Estas funciones son siempre consistentes.
- **COLUMNA 4:**Polinomios de grado 12,h(x) = $\sum_{yo=0}^{12} w_i x_i$.Estos son consistentes: podemos siempre obtenga un polinomio de grado 12 para que se ajuste perfectamente a 13 puntos distintos. Pero solo porque el hipótesis es consistente no significa que sea una buena conjetura.

Figura 19.1



Encontrar hipótesis para ajustar los datos.**Fila superior**: cuatro gráficos de funciones de mejor ajuste de cuatro espacios de hipótesis diferentes entrenados en el conjunto de datos 1.**Fila inferior**: las mismas cuatro funciones, pero entrenadas en un conjunto de datos ligeramente diferente (muestreado del mismof(x)función).

Una forma de analizar los espacios de hipótesis es por el sesgo que imponen (independientemente del entrenamiento).
conjunto de datos) y la varianza que producen (de un conjunto de entrenamiento a otro).

Parcialidad

Por**parcialidad**queremos decir (vagamente) la tendencia de una hipótesis predictiva a desviarse de la
valor esperado cuando se promedia sobre diferentes conjuntos de entrenamiento. El sesgo a menudo resulta de
restricciones impuestas por el espacio de hipótesis. Por ejemplo, el espacio de hipótesis de lineal
funciones induce un fuerte sesgo: solo permite funciones que consisten en líneas rectas. Sí hay
hay algún patrón en los datos que no sea la pendiente general de una línea, una función lineal no
ser capaz de representar esos patrones. Decimos que una hipótesis es**desajustar**cuando falla
encontrar un patrón en los datos. Por otro lado, la función lineal por partes tiene un sesgo bajo; la
la forma de la función es impulsada por los datos.

---

subequipamiento

Por**diferencia**nos referimos a la cantidad de cambio en la hipótesis debido a la fluctuación en el
datos de entrenamiento. Las dos filas deFigura 19.1-representan conjuntos de datos que fueron muestreados
del mismo$f(x)$función. Los conjuntos de datos resultaron ser ligeramente diferentes. Por el primero
tres columnas, la pequeña diferencia en el conjunto de datos se traduce en una pequeña diferencia en el
hipótesis. A eso lo llamamos varianza baja. Pero los polinomios de grado 12 en la cuarta columna
tienen una varianza alta: mira cuán diferentes son las dos funciones en ambos extremos de la$X$-eje.
Claramente, al menos uno de estos polinomios debe ser una mala aproximación a la verdadera$f(x)$.Nosotros
decir que una función es**sobreajuste**los datos cuando se presta demasiada atención a los datos particulares
conjunto en el que está entrenado, lo que hace que funcione mal en datos no vistos.

Diferencia

sobreajuste

A menudo hay un**compensación sesgo-varianza**: una elección entre hipótesis más complejas y de bajo sesgo que se ajustan bien a los datos de entrenamiento e hipótesis más simples y de baja varianza que pueden generalizarse mejor. Albert Einstein dijo en 1933, “el objetivo supremo de toda teoría es hacer que los elementos básicos irreductibles sean lo más simples y mínimos posible sin tener que renunciar a la representación adecuada de un solo dato de experiencia”. En otras palabras, Einstein
recomienda elegir la hipótesis más simple que coincida con los datos. Este principio se remonta más atrás al filósofo inglés del siglo XIV William of Ockham.[2]Su
El principio de que “la pluralidad [de entidades] no debe postularse sin necesidad” se llama **la navaja de Ockham**porque se usa para “afeitar” explicaciones dudosas.

[2]El nombre a menudo se escribe mal como "Occam".

Compensación de sesgo-varianza

Definir la simplicidad no es fácil. Parece claro que un polinomio con solo dos parámetros es más simple que uno con trece parámetros. Precisaremos esta intuición en Sección 19.3.4-. Sin embargo, encapitulo 21-veremos que los modelos de redes neuronales profundas pueden a menudo generalizan bastante bien, a pesar de que son muy complejos, algunos de ellos tienen miles de millones de parámetros Entonces, el número de parámetros por sí solo no es una buena medida de la capacidad de un modelo. aptitud física. Tal vez deberíamos apuntar a la "adecuación", no a la "simplicidad" en una clase modelo. Consideraremos este tema enSección 19.4.1-.

¿Qué hipótesis es mejor enFigura 19.1-? No podemos estar seguros. Si supiéramos los datos representaba, por ejemplo, el número de visitas a un sitio web que crece día a día, pero también ciclos dependiendo de la hora del día, entonces podríamos favorecer la función sinusoidal. Si nosotros Sabía que los datos definitivamente no eran cíclicos pero tenían mucho ruido, eso favorecería el lineal función.

En algunos casos, un analista está dispuesto a decir no solo que una hipótesis es posible o imposible, sino lo probable que es. El aprendizaje supervisado se puede hacer eligiendo el hipótesish_*eso es lo más probable dados los datos:

$$h_* = \underset{h \in H}{\text{argmax}}\, P(h \mid \text{datos}) .$$

Por la regla de Bayes esto es equivalente a

$$h_* = \underset{h \in H}{\text{argmax}}\, P(\text{datos} \mid h)\, P(h).$$

Entonces podemos decir que la probabilidad previa$P(h)$es alto para un grado suave-1 o -2 polinomio y menor para un polinomio de grado 12 con picos grandes y afilados. Permitimos funciones de apariencia inusual cuando los datos dicen que realmente las necesitamos, pero las desaconsejamos dándoles una probabilidad previa baja.

¿Por qué no dejar$H$ser la clase de todos los programas de computadora, o todas las máquinas de Turing? El problema es eso*hay una compensación entre la expresividad de un espacio de hipótesis y el computacional complejidad de encontrar una buena hipótesis dentro de ese espacio.*Por ejemplo, ajustando una línea recta a los datos son un cálculo fácil; ajustar polinomios de alto grado es algo más difícil; y ajuste Las máquinas de Turing son indecidibles. Una segunda razón para preferir espacios de hipótesis simples es que presumiblemente querremos usar$h$después de que lo hayamos aprendido, y computando$h(x)$cuando$h$es un Se garantiza que la función lineal sea rápida, mientras se calcula una máquina de Turing arbitraria Ni siquiera se garantiza que el programa finalice.

Por estas razones, la mayor parte del trabajo sobre el aprendizaje se ha centrado en representaciones simples. En los últimos años ha habido un gran interés en el aprendizaje profundo (capitulo 21.), donde las representaciones no son simples pero donde el$h(x)$el cálculo aún toma solo un*número acotado de pasos*a computar con el hardware adecuado.

Veremos que la compensación expresividad-complejidad no es simple: a menudo es el caso, como vimos con lógica de primer orden enCapítulo 8., que un lenguaje expresivo hace posible para*simple*hipótesis para ajustar los datos, mientras que restringir la expresividad del lenguaje significa que cualquier hipótesis consistente debe ser compleja.

19.2.1 Problema de ejemplo: espera en un restaurante

Describiremos en detalle un ejemplo de problema de aprendizaje supervisado: el problema de decidir si esperar por una mesa en un restaurante. Este problema se utilizará a lo largo del capítulo. para demostrar diferentes clases de modelos. Para este problema la salida, y, es una variable booleana que llamaremos *Esperará*; es cierto para los ejemplos en los que esperamos una mesa. la entrada, X, es un vector de diez valores de atributo, cada uno de los cuales tiene valores discretos:

1. **ALTERNO:** si hay un restaurante alternativo adecuado cerca.
2. **BARRA:** si el restaurante tiene una cómoda área de bar para esperar.
3. **VIE/SÁBADO:** cierto los viernes y sábados.
4. **HAMBRE:** si tenemos hambre en este momento.
5. **PATROCINADORES:** cuántas personas hay en el restaurante (los valores son *Ninguna*, *Alguno*, y *Completo*).
6. **PRECIO:** el rango de precios del restaurante ($, $$, $$$).
7. **LLUVIA:** si está lloviendo afuera.
8. **RESERVA:** si hicimos una reserva.
9. **TIPO:** el tipo de restaurante (francés, italiano, tailandés o hamburguesería).
10. **TIEMPO DE ESPERA:** estimación de espera del anfitrión: $0 - 10$, $10 - 30$, $30 - 60$ o >60 minutos.

Un conjunto de 12 ejemplos, tomados de la experiencia de uno de nosotros (SR), se muestra en Figura 19.2-. Tenga en cuenta lo escasos que son estos datos: hay $2_6 \times 3_2 \times 4_2 = 9,216$ posibles combinaciones de valores para los atributos de entrada, pero se nos da la salida correcta para solo 12 de ellos; cada de los otros 9.204 podrían ser verdaderos o falsos; no sabemos Esta es la esencia de inducción: necesitamos hacer nuestra mejor conjetura en estos 9,204 valores de salida que faltan, dados solo la evidencia de los 12 ejemplos.

Figura 19.2

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Ejemplos para el dominio restaurante.

19.5 La teoría del aprendizaje

¿Cómo podemos estar seguros de que nuestra hipótesis aprendida predirá bien para antes no visto entradas? Es decir, ¿cómo sabemos que la hipótesishestá cerca de la función objetivoFsi nosotros no se queF¿es? Estas preguntas han sido ponderadas durante siglos por Ockham, Hume y otros. En las últimas décadas han surgido otras preguntas: ¿cuántos ejemplos ¿Necesitamos conseguir un buenh?¿Qué espacio de hipótesis debemos usar? Si el espacio de hipótesis es muy complejo, ¿podemos encontrar el mejorh,¿O tenemos que conformarnos con un máximo local? ¿Qué tan complejo debeh¿ser? ¿Cómo evitamos el sobreajuste? Esta sección examina estos preguntas.

Comenzaremos con la pregunta de cuántos ejemplos se necesitan para aprender. vimos desde la curva de aprendizaje para el aprendizaje del árbol de decisión en el problema del restaurante (Figura 19.7 en página 661) que la precisión mejora con más datos de entrenamiento. Las curvas de aprendizaje son útiles, pero son específicos de un algoritmo de aprendizaje particular sobre un problema particular. ¿Hay algunos principios más generales que rigen el número de ejemplos necesarios?

Preguntas como esta son abordadas por**teoría del aprendizaje computacional**, que se encuentra en el intersección de IA, estadística e informática teórica. El principio subyacente es que cualquier hipótesis que esté seriamente equivocada casi con seguridad será "descubierta" con gran probabilidad después de un pequeño número de ejemplos, porque hará una predicción incorrecta. Por lo tanto, cualquier hipótesis que sea consistente con un conjunto suficientemente grande de ejemplos de entrenamiento es improbable que esté seriamente equivocado: es decir, debe ser**probablemente aproximadamente correcto (PAC)**.

Teoría del aprendizaje computacional

Probablemente aproximadamente correcto (PAC)

Cualquier algoritmo de aprendizaje que devuelve hipótesis que probablemente sean aproximadamente correctas es llamado a**aprendizaje PAC**algoritmo; podemos usar este enfoque para proporcionar límites en el rendimiento de varios algoritmos de aprendizaje.

---

aprendizaje PAC

Los teoremas de aprendizaje PAC, como todos los teoremas, son consecuencias lógicas de los axiomas. Cuando una teorema (a diferencia de, digamos, un experto político) establece algo sobre el futuro basado en pasado, los axiomas tienen que proporcionar el "jugo" para hacer esa conexión. Para el aprendizaje de PAC, el jugo es proporcionado por el supuesto de estacionariedad presentado en la página665-, que dice que los ejemplos futuros se van a extraer de la misma distribución fija $P(E) = P(X, Y)$como ejemplos pasados. (Tenga en cuenta que no tenemos que saber qué distribución es, solo que no cambia.) Además, para mantener las cosas simples, supondremos que el función verdaderaFes determinista y es miembro del espacio de hipótesisHeso es ser consideró.

Los teoremas PAC más simples tratan con funciones booleanas, para las cuales la pérdida 0/1 es adecuado. los**Tasa de error**de una hipótesish,definido informalmente antes, se define formalmente aquí como el error de generalización esperado para ejemplos extraídos de la estacionaria distribución:

$$error(h) = GenLoss_{L_{0/1}}(h) = \sum_{x,y} L_{0/1}(y, h(x))\, P(x, y).$$

En otras palabras,error(h)es la probabilidad de quehclasifica incorrectamente un nuevo ejemplo. Este es el misma cantidad medida experimentalmente por las curvas de aprendizaje mostradas anteriormente.

Una hipótesishse llama**aproximadamente correcto**sierror(h) ≤ ϵ,donde ϵ es una pequeña constante. Demostraremos que podemos encontrar unnortetal que, después del entrenamiento ennorteejemplos, con alto probabilidad, todas las hipótesis consistentes serán aproximadamente correctas. Uno puede pensar en un hipótesis aproximadamente correcta como "cercana" a la función verdadera en el espacio de hipótesis: se encuentra dentro de lo que se llama ϵ-pelotaalrededor de la verdadera funciónF.El espacio de hipótesis fuera esta pelota se llamaH_{malo}.

ϵ - bola

Podemos derivar un límite en la probabilidad de que una hipótesis "gravemente errónea"$h_b \in H_{malo}$es consistente con la primera$norte$ejemplos de la siguiente manera. Lo sabemos$error(h_b) > \epsilon$.Por lo tanto, la probabilidad de que esté de acuerdo con un ejemplo dado es como máximo$1 - \epsilon$.Dado que los ejemplos son independiente, el límite para$norte$ejemplos es:

$$pag_b\text{concuerda con N ejemplos)} \leq (1 - \epsilon)^{norte}.$$

La probabilidad de queH_{malo}contiene al menos una hipótesis consistente está limitada por la suma de las probabilidades individuales:

$$P(H_{malo}\text{contiene una hipótesis consistente)} \leq \|H_{malo}\|(1 - \epsilon)^{norte} \leq \|H\|(1 - \epsilon)^{norte},$$

donde hemos utilizado el hecho de queH_{malo}es un subconjunto de$H$y así$|H_{malo}| \leq |H|$.Lo haríamos quisiera reducir la probabilidad de este evento por debajo de un pequeño número δ:

$$P(H_{malo}\text{contiene una hipótesis consistente)} \leq |H|(1 - \epsilon)^{norte} \leq d.$$

Dado que$1 - \epsilon \leq mi^{-\epsilon}$,podemos lograr esto si permitimos que el algoritmo vea

(19.1)

$$norte \geq \frac{1}{\epsilon}(en\ \frac{1}{d} + en\ |H|)$$

ejemplos Por lo tanto, con probabilidad al menos$1 - d$,después de ver tantos ejemplos, el aprendizaje algoritmo devolverá una hipótesis que tiene un error como máximo ϵ. En otras palabras, es probable que aproximadamente correcto. El número de ejemplos requeridos, en función de ϵ y δ, se llama la**complejidad de la muestra**del algoritmo de aprendizaje.

Como vimos antes, si$H$es el conjunto de todas las funciones booleanas en $n$atributos, entonces $\|H\| = 2^{2^n}$. Así, la complejidad muestral del espacio crece a medida que2^n.Porque el número de posibles ejemplos es también2^n,esto sugiere que el aprendizaje PAC en la clase de todas las funciones booleanas requiere ver todos, o casi todos, los ejemplos posibles. Un momento de reflexión revela la motivo de esto:Hcontiene suficientes hipótesis para clasificar cualquier conjunto dado de ejemplos en todos caminos posibles. En particular, para cualquier conjunto de$n$ejemplos, el conjunto de hipótesis consistentes con esos ejemplos contienen igual número de hipótesis que predicenX_{n+1}ser positivo y hipótesis que predicenX_{n+1}ser negativo.

Entonces, para obtener una generalización real a ejemplos no vistos, parece que necesitamos restringir el espacio de hipótesis$H$de alguna manera; pero por supuesto, si restringimos el espacio, podríamos eliminar la verdadera función por completo. Hay tres formas de escapar de este dilema.

El primero es aplicar conocimientos previos al problema.

El segundo, que presentamos enSección 19.4.3-, es insistir en que el algoritmo regrese no cualquier hipótesis consistente, sino preferiblemente una simple (como se hace en el árbol de decisión aprendizaje). En los casos en que es factible encontrar hipótesis consistentes simples, la muestra los resultados de complejidad son generalmente mejores que los de los análisis basados únicamente en la consistencia.

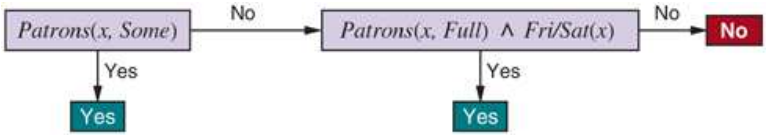
El tercero, que seguiremos a continuación, es centrarnos en subconjuntos aprendibles de toda la hipótesis. espacio de funciones booleanas. Este enfoque se basa en la suposición de que la restricción el espacio de hipótesis contiene una hipótesis$h$que está lo suficientemente cerca de la verdadera funciónF;la Los beneficios son que el espacio de hipótesis restringido permite una generalización efectiva y es típicamente más fácil de buscar. Ahora examinamos uno de estos espacios de hipótesis restringidos en más detalle.

19.5.1 Ejemplo de aprendizaje de PAC: listas de decisiones de aprendizaje

Ahora mostramos cómo aplicar el aprendizaje de PAC a un nuevo espacio de hipótesis:**listas de decisiones**. A lista de decisión consta de una serie de pruebas, cada una de las cuales es una conjunción de literales. si una prueba tiene éxito cuando se aplica a una descripción de ejemplo, la lista de decisiones especifica el valor a ser devuelto Si la prueba falla, el procesamiento continúa con la siguiente prueba de la lista. Listas de decisiones se parecen a los árboles de decisión, pero su estructura general es más simple: se ramifican solo en una dirección. En cambio, las pruebas individuales son más complejas.Figura 19.10-muestra una decisión lista que representa la siguiente hipótesis:

WillWait ⇔ (Patronos = Algunos) ∨ (Patronos = Completo ∧ Vie/Sab).

Figura 19.10



Una lista de decisiones para el problema del restaurante.

Listas de decisiones

Si permitimos pruebas de tamaño arbitrario, las listas de decisiones pueden representar cualquier función booleana (Ejercicio19DLEX). Por otro lado, si restringimos el tamaño de cada prueba a como máximokliterales, entonces es posible que el algoritmo de aprendizaje generalice con éxito a partir de un pequeño número de ejemplos Usamos la notaciónk-DLpara una lista de decisiones con hastakconjunciones. los ejemplo enFigura 19.10está en 2-DL. Es fácil de mostrar (Ejercicio19DLEX) quek-DLincluye como un subconjuntok–DT,el conjunto de todos los árboles de decisión de profundidad como máximok.Usaremos la notación k-DL(norte)para denotar unk-DLusandonorteAtributos booleanos.

k-DT

La primera tarea es demostrar quek-DLes aprendible, es decir, que cualquier función enk-DLpuede ser aproximado con precisión después del entrenamiento en un número razonable de ejemplos. Para hacer esto, nosotros necesidad de calcular el número de hipótesis posibles. Sea el conjunto de conjunciones de a lo sumok literales usandonorteatributos serConj(n, k).Debido a que una lista de decisiones se construye a partir de pruebas, y porque cada prueba se puede adjuntar a un*Sí*o un*No*resultado o puede estar ausente del lista de decisiones, hay como máximo3|Conj(n,k)|distintos conjuntos de pruebas de componentes. Cada uno de estos conjuntos de las pruebas puede estar en cualquier orden, por lo que

$$|k\text{-DL}(n)| \leq 3^c c! \text{ donde } c = |\text{Conj}(n, k)|.$$

El número de conjunciones de como máximokliterales denortelos atributos vienen dados por

$$|\text{Conj}(n, k)| = \sum_{i=0}^{k} \binom{2n}{i} = O(n^k).$$

Por lo tanto, después de un poco de trabajo, obtenemos

$$|k\text{-DL}(n)| = 2^{En^k \text{Iniciar sesión}_2(n^k))}.$$

Podemos enchufar esto enEcuación (19.1)para mostrar que el número de ejemplos necesarios para PAC-aprendizaje ak-DL(norte)la función es polinomial ennorte:

$$n \geq \frac{1}{\epsilon}\left(\ln \frac{1}{\delta} + O(n^k \text{Iniciar sesión}_2(n^k))\right).$$

Por lo tanto, cualquier algoritmo que devuelva una lista de decisiones consistente PAC-aprenderá unk-DL funcionar en un número razonable de ejemplos, para pequeñosk.

La siguiente tarea es encontrar un algoritmo eficiente que devuelva una lista de decisiones consistente. Lo haremos usa un algoritmo codicioso llamado Descisión-List-Lganadorque encuentra repetidamente una prueba que está de acuerdo exactamente con algún subconjunto del conjunto de entrenamiento. Una vez que encuentra tal prueba, la agrega a la lista de decisión en construcción y elimina los ejemplos correspondientes. Luego construye el resto de la lista de decisiones, usando solo los ejemplos restantes. Esto se repite hasta no quedan ejemplos. El algoritmo se muestra enFigura 19.11-.

Figura 19.11

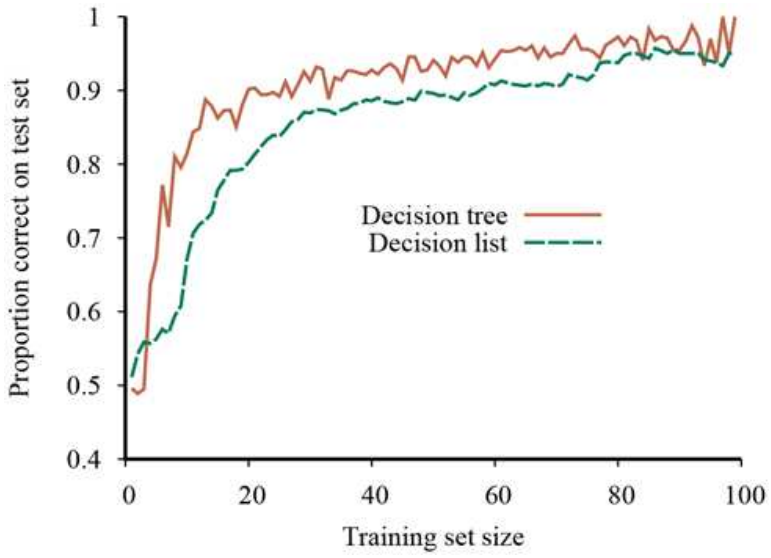
function DECISION-LIST-LEARNING(*examples*) **returns** a decision list, or *failure*

if *examples* is empty **then return** the trivial decision list *No*
$t \leftarrow$ a test that matches a nonempty subset $examples_t$ of *examples*
such that the members of $examples_t$ are all positive or all negative
if there is no such t **then return** *failure*
if the examples in $examples_t$ are positive **then** $o \leftarrow Yes$ **else** $o \leftarrow No$
return a decision list with initial test t and outcome o and remaining tests given by
DECISION-LIST-LEARNING(*examples* $-$ $examples_t$)

Un algoritmo para el aprendizaje de listas de decisiones.

Este algoritmo no especifica el método para seleccionar la siguiente prueba para agregar a la decisión lista. Aunque los resultados formales dados anteriormente no dependen del método de selección, sí Parecería razonable preferir pruebas pequeñas que coincidan con grandes conjuntos de clasificaciones uniformes. ejemplos, para que la lista de decisiones general sea lo más compacta posible. Lo más simple La estrategia es encontrar la prueba más pequeña.tque coincida con cualquier subconjunto uniformemente clasificado, independientemente del tamaño del subconjunto. Incluso este enfoque funciona bastante bien, ya queFigura 19.12sugiere. Para este problema, el árbol de decisiones aprende un poco más rápido que la lista de decisiones, pero tiene más variación. Ambos métodos tienen una precisión de más del 90 % después de 100 intentos.

Figura 19.12



Curva de aprendizaje para Descisión-List-Lganadoralgoritmo en los datos del restaurante. La curva para Lganar-Descisión- Treese muestra para comparar; los árboles de decisión funcionan un poco mejor en este problema en particular.

19.6 Regresión lineal y clasificación

Ahora es el momento de pasar de los árboles de decisión y las listas a un espacio de hipótesis diferente, uno que se ha utilizado durante cientos de años: la clase de **funciones lineales** de valor continuo entradas. Comenzaremos con el caso más simple: regresión con una función lineal univariante, también conocido como "ajuste de una línea recta". Sección 19.6.3 cubre el caso multivariable. Secciones 19.6.4 y 19.6.5 mostrar cómo convertir funciones lineales en clasificadores aplicando Umbrales duros y suaves.

Función lineal

19.6.1 Regresión lineal univariante

Una función lineal univariada (una línea recta) con entrada x y salida y tiene la forma $y = w_1x + w_0$, dónde w_0 y w_1 son coeficientes con valores reales que se deben aprender. Usamos la letra w porque pensamos en los coeficientes como **pesos**; El valor de y se cambia cambiando el peso relativo de un término u otro. definiremos w ser el vector $\langle w_0, w_1 \rangle$, y definir la función lineal con esos pesos como

$$h_w(x) = w_1x + w_0.$$

Peso

Figura 19.13(a) muestra un ejemplo de un conjunto de entrenamiento de $norte$ puntos en el x, y plano, cada punto que representa el tamaño en pies cuadrados y el precio de una casa que se ofrece a la venta. La tarea de encontrando el h_w que mejor se ajusta a estos datos se llama **regresión lineal**. Para ajustar una línea a los datos, todos lo que tenemos que hacer es encontrar los valores de los pesos $\langle w_0, w_1 \rangle$ que minimizan la pérdida empírica. Está

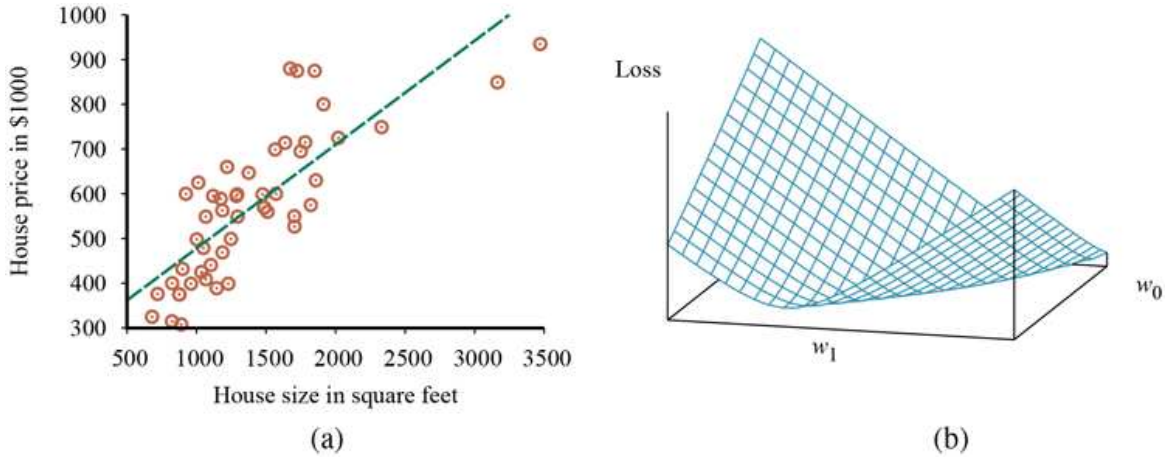
tradicional (volviendo a Gauss[6]) para usar la función de pérdida de error al cuadrado, L_2, resumido todos los ejemplos de entrenamiento:

[6] Gauss demostró que si el y_j valores tienen ruido normalmente distribuido, entonces los valores más probables de w_1 y w_0 se obtienen usando L_2 pérdida, minimizando la suma de los cuadrados de los errores. (Si los valores tienen ruido que sigue a un Laplace (doble exponencial) distribución, entonces L_1 la pérdida es apropiada.)

$$\text{Pérdida}(h_w) = \sum_{j=1}^{n} L_2(y_j, h_w(X_j)) = \sum_{j=1}^{n} (y_j - h_w(X_j))^2 = \sum_{j=1}^{n} (y_j - (w_1 X_j + w_0))^2.$$

regresión lineal

Figura 19.13



(a) Puntos de datos de precio versus espacio de piso de casas en venta en Berkeley, CA, en julio de 2009, junto con la hipótesis de la función lineal que minimiza la pérdida de error cuadrático: $y = 0.232x + 246$. (b) Gráfico de la función de pérdida $\sum_j (y_j - w_1 X_j + w_0)^2$ para varios valores de w_0, w_1. Tenga en cuenta que la función de pérdida es convexa, con una mínimo global único.

nos gustaría encontrar $w^* = \text{argmin}_w \text{Pérdida}(h_w)$. La suma $\sum_{j=1}^{n} (y_j - (w_1 X_j + w_0))^2$ es minimiza cuando sus derivadas parciales con respecto a w_0 y w_1 son cero:

(19.2)

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{n} (y_j - (w_1 X_j + w_0))^2 = 0 \text{ y} \qquad \frac{\partial}{\partial w_1} \sum_{j=1}^{n} (y_j - (w_1 X_j + w_0))^2 = 0.$$

Estas ecuaciones tienen una única solución:

(19.3)

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \qquad w_0 = (\sum y_j - w_1(\sum x_j))/N.$$

Para el ejemplo enFigura 19.13(a)-, la solucion es$w_1 = 0.232$, $w_0 = 246$,y la línea con esos pesos se muestran como una línea discontinua en la figura.

Muchas formas de aprendizaje implican el ajuste de pesos para minimizar una pérdida, por lo que ayuda tener un imagen mental de lo que está pasando en**espacio de peso**—el espacio definido por todos los escenarios posibles de los pesos Para la regresión lineal univariante, el espacio de peso definido porw_0y$w_1$es bidimensional, por lo que podemos graficar la pérdida como una función dew_0y$w_1$en un gráfico 3D (ver Figura 19.13(b)-). Vemos que la función de pérdida es**convexo**, como se define en la página122; esto es cierto para*cada*problema de regresión lineal con unL_2función de pérdida, e implica que hay sin mínimos locales. En cierto sentido, ese es el final de la historia de los modelos lineales; si tenemos que encajar líneas a datos, aplicamosEcuación (19.3)-.[7]

[7]Con algunas salvedades: elL_2La función de pérdida es apropiada cuando hay un ruido normalmente distribuido que es independiente de$X$;todos los resultados confiar en el supuesto de estacionariedad; etc.

espacio de peso

19.6.2 Descenso de pendiente

El modelo lineal univariante tiene la buena propiedad de que es fácil encontrar una solución óptima donde las derivadas parciales son cero. Pero ese no siempre será el caso, así que presentamos aquí un método para minimizar la pérdida que no depende de resolver para encontrar ceros de la derivadas, y se puede aplicar a cualquier función de pérdida, sin importar cuán compleja sea.

Descenso de gradiente

Como se discutió enSección 4.2-(página119) podemos buscar a través de un espacio de peso continuo mediante la modificación incremental de los parámetros. Allí llamamos al algoritmo**Montañismo**, pero aquí estamos minimizando la pérdida, no maximizando la ganancia, por lo que usaremos el término**degradado descendencia**. Elegimos cualquier punto de partida en el espacio de pesos—aquí, un punto en el (w_0, w_1) plano- y luego calcule una estimación del gradiente y mueva una pequeña cantidad en la pendiente más pronunciada dirección cuesta abajo, repitiendo hasta converger en un punto en el espacio de peso con (local) pérdida mínima. El algoritmo es como sigue:

(19.4)

w ← cualquier punto en el espacio de
parámetros mientras no converja do
para cada uno$_i$en lo que hacemos

$$w_i \leftarrow w_i - a\frac{\partial \text{Pérdida}}{\partial w_i}(w)$$

Descenso de gradiente

El parámetro α, al que llamamos**Numero de pie**en*Sección 4.2-*, suele llamarse el **tasa de aprendizaje**cuando estamos tratando de minimizar la pérdida en un problema de aprendizaje. puede ser fijo constante, o puede decaer con el tiempo a medida que avanza el proceso de aprendizaje.

Tasa de aprendizaje

Para la regresión univariante, la pérdida es cuadrática, por lo que la derivada parcial será lineal. (Los El único cálculo que necesitas saber es el**cadena de reglas**:$\partial g(f(x))/\partial x = g'(f(x))\ \partial f(x)/\partial x$,más el hechos que$\frac{\partial}{\partial x}x^2 = 2x$y $\frac{\partial}{\partial x}x = 1$.)Primero resolvamos las derivadas parciales—las pendientes—en el caso simplificado de un solo ejemplo de entrenamiento, (x, y):

(19.5)

$$\frac{\partial}{\partial w_i}\text{Pérdida}(w) = \frac{\partial}{\partial w_i}(y - h_w(X))^2 = 2(y - h_w(x)) \times \frac{\partial}{\partial w_i}(y - h_w(X))$$
$$= 2(y - h_w(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)).$$

Cadena de reglas

Aplicando esto a ambosw_0y$w_1$obtenemos:

$$\frac{\partial}{\partial w_0}\text{Pérdida}(w) = -2\,(y - h_w(X)) ; \qquad \frac{\partial}{\partial w_1}\text{Pérdida}(w) = -2\,(y - h_w(x)) \times x.$$

Enchufando esto enEcuación (19.4)., y plegando el 2 en la tasa de aprendizaje no especificada α, obtenemos la siguiente regla de aprendizaje para los pesos:

$$w_0 \leftarrow w_0 + \alpha\,(y - h_w(X)) ; \; w_1 \leftarrow w_1 + \alpha\,(y - h_w(x)) \times x.$$

Estas actualizaciones tienen un sentido intuitivo: si$h_w(x) > y$(es decir, la salida es demasiado grande), reduzcaw_0a poco, y reducir$w_1$si$X$fue un aporte positivo pero aumentó$w_1$si$X$fue una entrada negativa.

Las ecuaciones anteriores cubren un ejemplo de entrenamiento. Para$n$ejemplos de entrenamiento, queremos minimizar la suma de las pérdidas individuales para cada ejemplo. La derivada de una suma es el suma de las derivadas, entonces tenemos:

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_w(X_j)) ; \qquad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_w(X_j)) \times x_j.$$

Estas actualizaciones constituyen la**descenso de gradiente por lotes**regla de aprendizaje para lineal univariado regresión (también llamada**descenso de gradiente determinista**). La superficie de pérdida es convexa, lo que significa que no hay mínimos locales en los que atascarse, y la convergencia al global el mínimo está garantizado (siempre y cuando no elijamos un α que sea tan grande que se sobrepase), pero puede ser muy lento: tenemos que sumar sobre todo$n$ejemplos de entrenamiento para cada paso, y hay pueden ser muchos pasos. El problema se complica si$n$es más grande que la memoria del procesador Talla. Un paso que cubre todos los ejemplos de entrenamiento se llama**época**.

Descenso de gradiente por lotes

Una variante más rápida se llama**descenso de gradiente estocásticoo****USD**: selecciona aleatoriamente un pequeño número de ejemplos de entrenamiento en cada paso, y actualizaciones de acuerdo conEcuación (19.5)-. los versión original de SGD seleccionó solo un ejemplo de entrenamiento para cada paso, pero ahora es más común para seleccionar un**minilote**demetrofuera denorteejemplos Supongamos que tenemosn = 10, 000 ejemplos y elija un minilote de tamañometro = 100Entonces en cada paso hemos reducido el cantidad de cálculo por un factor de 100; pero debido a que el error estándar de la estimación gradiente medio es proporcional a la raíz cuadrada del número de ejemplos, el estándar error aumenta sólo por un factor de 10. Así que incluso si tenemos que dar 10 veces más pasos antes convergencia, el minibatch SGD sigue siendo 10 veces más rápido que el lote completo SGD en este caso.

Época

Descenso de gradiente estocástico

USD

minilote

Con algunas arquitecturas de CPU o GPU, podemos elegirmetropara aprovechar el vector paralelo operaciones, dando un paso conmetroejemplos casi tan rápido como un paso con un solo ejemplo. Dentro de estas limitaciones, trataríamosmetrocomo un hiperparámetro que debe ser afinado para cada problema de aprendizaje.

La convergencia de SGD de minilotes no está estrictamente garantizada; puede oscilar alrededor del mínimo sin asentarse. vamos a ver en la pagina684cómo un programa de disminución de la la tasa de aprendizaje, α, (como en el recocido simulado) garantiza la convergencia.

SGD puede ser útil en un entorno en línea, donde los nuevos datos ingresan uno a la vez, y la suposición de estacionariedad puede no ser válida. (De hecho, SGD también se conoce como**gradiente en línea descendencia**.) Con una buena elección para α, un modelo evolucionará lentamente, recordando algo de lo que aprendido en el pasado, sino también adaptándose a los cambios que representan los nuevos datos.

Descenso de gradiente en línea

SGD se aplica ampliamente a modelos distintos de la regresión lineal, en particular a las redes neuronales. Incluso cuando la superficie de pérdida no es convexa, el enfoque ha demostrado ser eficaz para encontrar buenas mínimos locales cercanos al mínimo global.

19.6.3 Regresión lineal multivariable

Podemos extendernos fácilmente a**regresión lineal multivariable**problemas, en los que cada ejemplo $\mathbf{x}_j$es unnorte-elemento vectorial[8]Nuestro espacio de hipótesis es el conjunto de funciones de la forma

[8]El lector puede desear consultarApéndice**A**-para un breve resumen de álgebra lineal. Además, tenga en cuenta que usamos el término "multivariable regresión" para significar que la entrada es un vector de múltiples valores, pero la salida es una sola variable. Usaremos el término "multivariado regresión" para el caso en que la salida sea también un vector de múltiples variables. Sin embargo, otros autores utilizan los dos términos indistintamente.

$$h_{\mathbf{w}}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_{norte} x_{jn} = w_0 + \sum_i w_i x_{ji}.$$

Regresión lineal multivariable

los$w_0$término, el intercepto, se destaca como diferente de los demás. Podemos arreglar eso por inventar un atributo de entrada ficticio,$x_{j,0}$,que se define como siempre igual a 1. Entonceshes

simplemente el producto punto de los pesos y el vector de entrada (o de manera equivalente, la matriz producto de la transpuesta de los pesos y el vector de entrada):

$$h_w(X_j) = w \cdot x_j = w^T X_j = \sum_i w_i X_{ji}.$$

El mejor vector de pesos,w^*,minimiza la pérdida de error cuadrático sobre los ejemplos:

$$w^* = \text{argmín}_w \sum_j L_2(y_j, w \cdot x_j).$$

La regresión lineal multivariable en realidad no es mucho más complicada que la univariante. caso que acabamos de cubrir. El descenso del gradiente alcanzará el mínimo (único) de la pérdida función; la ecuación de actualización para cada peso$w_i$es

(19.6)

$$w_i \leftarrow w_i + \alpha \sum_j (y_j - h_w(X_j)) \times x_{ji}.$$

Con las herramientas del álgebra lineal y el cálculo vectorial, también es posible resolver analíticamente Para el$w$que minimiza la pérdida. Dejar$y$ser el vector de salidas para los ejemplos de entrenamiento, y Xser el**matriz de datos**—es decir, la matriz de entradas con una$n$orte-ejemplo dimensional por fila. Entonces el vector de salidas pronosticadas es $\hat{y} = Xw$y la pérdida de error cuadrático sobre todos los los datos de entrenamiento son

$$L(w) = \| \hat{y} - y \|_2^2 = \| Xw - y \|^2.$$

Matriz de datos

Ponemos el gradiente a cero:

$$\nabla_w L(\text{ancho}) = 2X^T(Xw - y) = 0.$$

Reorganizando, encontramos que el vector de peso de pérdida mínima está dado por

(19.7)

$$w^* = (X^\top X)^{-1} X^\top y.$$

Llamamos a la expresión $(X^\top X)^{-1} X^\top$ la **pseudoinverso** de la matriz de datos, y Ecuación (19.7) se llama el **ecuación normal**.

---

pseudoinverso

---

ecuación normal

Con la regresión lineal univariante no tuvimos que preocuparnos por el sobreajuste. Pero con regresión lineal multivariable en espacios de alta dimensión es posible que algunos La dimensión que en realidad es irrelevante parece ser útil por casualidad, lo que resulta en un sobreajuste.

Por lo tanto, es común usar **regularización** en funciones lineales multivariables para evitar sobreajuste Recuerde que con la regularización minimizamos el costo total de una hipótesis, contando tanto la pérdida empírica como la complejidad de la hipótesis:

$$\text{Costo}(h) = \text{EmpLoss}(h) + \lambda\, \text{Complejidad}(h).$$

Para funciones lineales, la complejidad se puede especificar como una función de los pesos. Podemos Considere una familia de funciones de regularización:

$$\text{Complejidad}(h_w) = L_q(a) = \sum_i |a_i|_q.$$

Al igual que con las funciones de pérdida, con $q = 1$ tenemos L_1 regularización 9, que minimiza la suma de los valores absolutos; con $q = 2$, L_2 la regularización minimiza la suma de cuadrados. ¿Qué función de regularización debería elegir? Eso depende del problema específico, pero L_1 regularización tiene una ventaja importante: tiende a producir una **modelo escaso**. Es decir, a menudo establece muchos pesos en cero, declarando efectivamente que los atributos correspondientes son

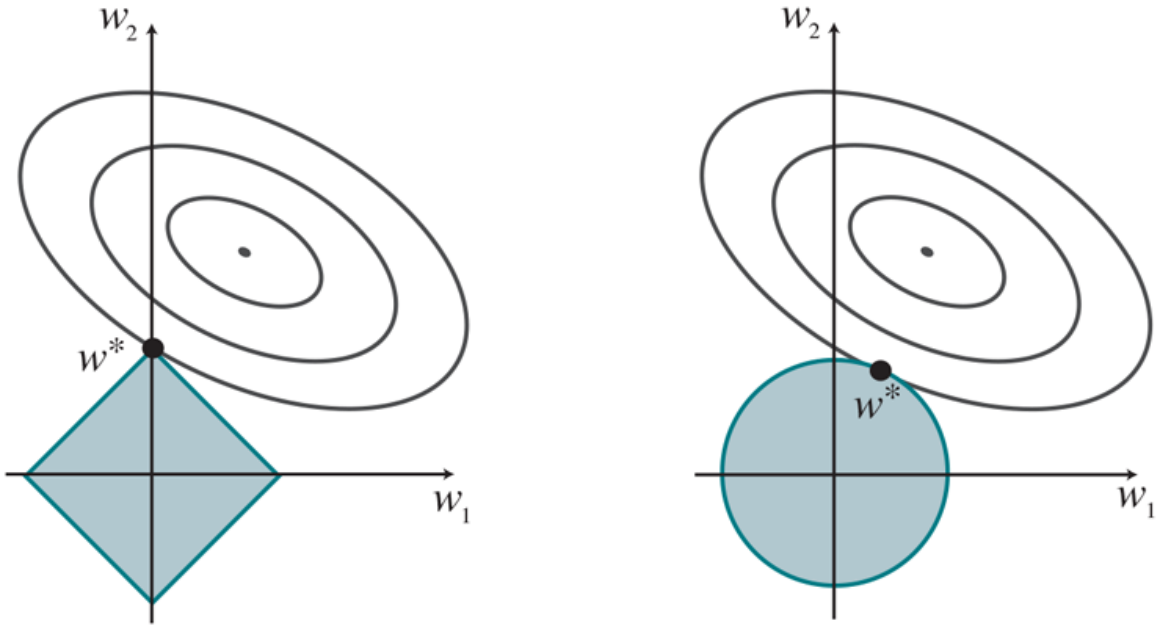
completamente irrelevante, al igual que LGANAR-DESCISIÓN-TREElo hace (aunque por un mecanismo diferente). Las hipótesis que descartan atributos pueden ser más fáciles de entender para un ser humano y pueden ser menos probable que se sobreajuste.

9Tal vez sea confuso que la notaciónL1yL2se utiliza tanto para funciones de pérdida como para funciones de regularización. no necesitan ser usado en pares: podrías usarL2pérdida conL1regularización, o viceversa.

modelo disperso

Figura 19.14da una explicación intuitiva de por quéL1la regularización conduce a ponderaciones de cero, mientrasL2la regularización no. Tenga en cuenta que minimizarPérdida(w) + λComplejidad(w)es equivalente a minimizarPérdida (w)sujeto a la restricción de queComplejidad(w) ≤ c,por alguna constanteCque está relacionado con λ. Ahora enFigura 19.14(a)la caja en forma de diamante representa el conjunto de puntoswen un espacio de peso bidimensional que tieneL1menor complejidad queC;nuestra solución tendrá que estar en algún lugar dentro de esta caja. Los óvalos concéntricos representan contornos de la función de pérdida, con la pérdida mínima en el centro. Queremos encuentre el punto en el cuadro que está más cerca del mínimo; se puede ver en el diagrama que, para una posición arbitraria del mínimo y sus contornos, será común para la esquina de la caja para encontrar su camino más cercano al mínimo, solo porque las esquinas son puntiagudas. Y por supuesto las esquinas son los puntos que tienen un valor de cero en alguna dimensión.

Figura 19.14



Por quéL_1la regularización tiende a producir un modelo disperso. Irse conL_1regularización (caja), la pérdida mínima alcanzable (contornos concéntricos) a menudo ocurre en un eje, lo que significa un peso de cero. Derecha: ConL_2 regularización (círculo), es probable que la pérdida mínima ocurra en cualquier parte del círculo, sin dar preferencia a los pesos cero.

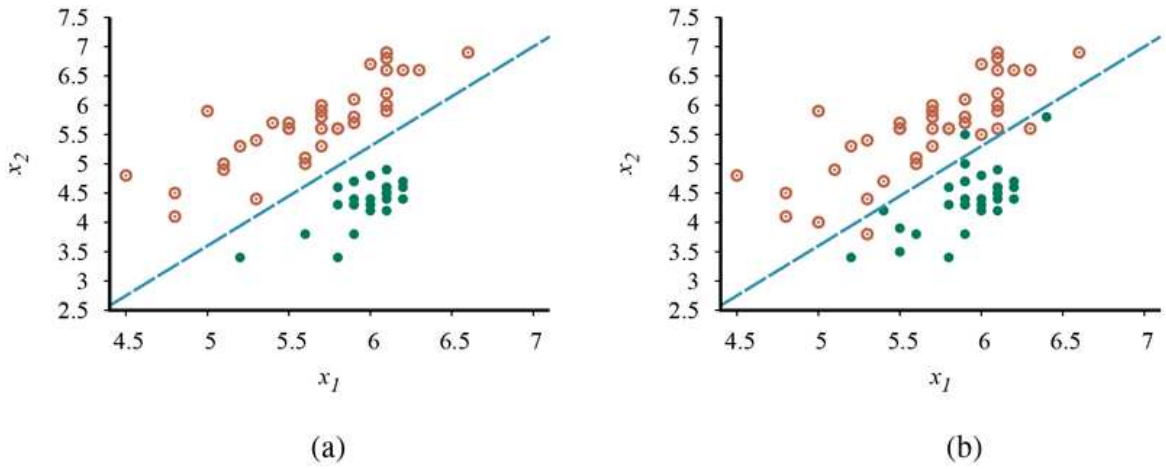
En Figura 19.14(b)-, hemos hecho lo mismo para elL_2medida de complejidad, que representa un círculo en lugar de un diamante. Aquí se puede ver que, en general, no hay razón para la intersección para aparecer en uno de los ejes; de este modoL_2la regularización no tiende a producir pesos cero. El resultado es que el número de ejemplos necesarios para encontrar una buenahes lineal en el número de características irrelevantes paraL_2regularización, pero sólo logarítmica conL_1 regularización La evidencia empírica sobre muchos problemas apoya este análisis.

Otra forma de verlo es queL_1la regularización toma en serio los ejes dimensionales, tiempoL_2los trata como arbitrarios. losL_2la función es esférica, lo que la hace rotacionalmente invariante: Imagine un conjunto de puntos en un plano, medido por su$x$y$y$coordenadas Ahora imagina rotar los ejes por45^o.Obtendrías un conjunto diferente de (x', y')valores que representan la mismos puntos. si aplicasL_2regularización antes y después de la rotación, se obtiene exactamente el mismo punto que la respuesta (aunque el punto se describiría con el nuevo (x', y') coordenadas). Eso es apropiado cuando la elección de los ejes es realmente arbitraria, cuando no importa si sus dos dimensiones son distancias norte y este; o distancias al noreste y sureste. ConL_1regularización obtendría una respuesta diferente, porque elL_1función no es rotacionalmente invariante. Eso es apropiado cuando los ejes no son intercambiables; eso no tiene sentido rotar "número de baños"45^ohacia el "tamaño del lote".

19.6.4 Clasificadores lineales con umbral duro

Las funciones lineales se pueden usar para hacer clasificación y regresión. Por ejemplo,Figura 19.15(a)muestra puntos de datos de dos clases: terremotos (que son de interés para sismólogos) y explosiones subterráneas (que son de interés para los expertos en control de armas). Cada punto está definido por dos valores de entrada,X_1yX_2,que se refieren a onda de cuerpo y de superficie magnitudes calculadas a partir de la señal sísmica. Dados estos datos de entrenamiento, la tarea de clasificación es aprender una hipótesishque tomará nuevo (X_1, X_2)puntos y devuelve 0 para terremotos o 1 para explosiones.

Figura 19.15



(a) Gráfica de dos parámetros de datos sísmicos, magnitud de onda corporalX_1y magnitud de onda superficialX_2,para terremotos (círculos naranjas abiertos) y explosiones nucleares (círculos verdes) que ocurrieron entre 1982 y 1990 en Asia y el Medio Oriente (Kebfácil*et al.*, 1998). También se muestra un límite de decisión entre las clases. (b) El mismo dominio con más puntos de datos. Los terremotos y las explosiones ya no son linealmente separables.

A**límite de decisión**es una línea (o una superficie, en dimensiones superiores) que separa los dos clases EnFigura 19.15(a), el límite de decisión es una línea recta. Una decisión lineal límite se llama**separador lineal**y los datos que admiten tal separador se llaman **separables linealmente**. El separador lineal en este caso está definido por

$$X_2 = 1{,}7x_1 - 4{,}9 \text{ o } -4{,}9 + 1{,}7x_1 - x_2 = 0.$$

Límite de decisión

separador lineal

separabilidad lineal

Las explosiones, que queremos clasificar con valor 1, están debajo ya la derecha de este línea; son puntos para los cuales $-4{,}9 + 1{,}7x_1 - x_2 > 0$,mientras que los terremotos tienen $-4{,}9 + 1{,}7x_1 - x_2 < 0$..Podemos hacer que la ecuación sea más fácil de manejar cambiándola a la forma de producto escalar vectorial—con$x_0 = 1$tenemos

$$-4{,}9x_0 + 1.7x_1 - x_2 = 0 ,$$

y podemos definir el vector de pesos,

$$w = \langle -4.9, 1.7, -1 \rangle,$$

y escribe la hipótesis de clasificación

$$h_w(x) = 1 \text{ si } w \cdot x \geq 0 \text{ y } 0 \text{ en caso contrario.}$$

Alternativamente, podemos pensar en$h$como resultado de pasar la función lineal$w \cdot x$a través de un **función de umbral**:

$$h_w(x) = \text{Umbral}\,(w \cdot x) \text{ donde Umbral } (z) = 1 \text{ si } z \geq 0 \text{ y } 0 \text{ en caso contrario.}$$

Función de umbral

La función de umbral se muestra enFigura 19.17(a)-.

Ahora que la hipótesis $h_w(X)$ tiene una forma matemática bien definida, podemos pensar en eligiendo los pesos w para minimizar la pérdida. En Secciones 19.6.1 y 19.6.3, hicimos esto tanto en forma cerrada (estableciendo el gradiente en cero y resolviendo los pesos) y por descenso de gradiente en el espacio de peso. Aquí no podemos hacer ninguna de las dos cosas porque el el gradiente es cero en casi todas partes en el espacio de peso, excepto en aquellos puntos donde $w \cdot x = 0$, y en esos puntos el gradiente es indefinido.

Sin embargo, existe una regla de actualización de peso simple que converge a una solución, es decir, a un separador lineal que clasifica los datos perfectamente, siempre que los datos sean linealmente separables. Para un solo ejemplo (x, y), tenemos

(19.8)

$$w_i \leftarrow w_i + \alpha\,(y - h_w(x)) \times x_i$$

que es esencialmente idéntico a Ecuación (19.6), la regla de actualización para la regresión lineal. Este regla se llama la **regla de aprendizaje del perceptrón**, por razones que se aclararán en Capítulo 21. Sin embargo, debido a que estamos considerando un problema de clasificación 0/1, el comportamiento es algo diferente Tanto el valor real y y la salida de la hipótesis $h_w(X)$ son 0 o 1, entonces hay tres posibilidades:

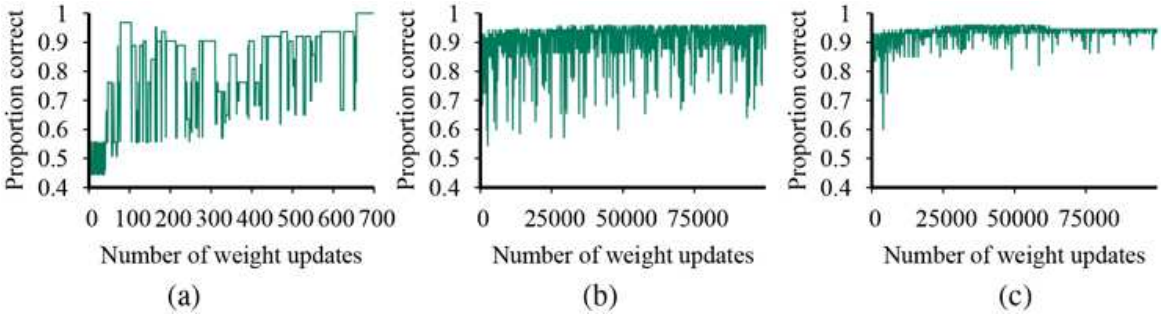
- Si la salida es correcta (es decir, $y = h_w(X)$) entonces los pesos no se modifican.
- Si y es 1 pero $h_w(X)$ es 0, entonces w_i es *aumentó* cuando la entrada correspondiente x_i es positivo y *disminuido* cuando x_i es negativo Esto tiene sentido, porque queremos hacer $w \cdot x$ más grande para que $h_w(X)$ emite un 1.
- Si y es 0 pero $h_w(X)$ es 1, entonces w_i es *disminuido* cuando la entrada correspondiente x_i es positivo y *aumentó* cuando x_i es negativo Esto tiene sentido, porque queremos hacer $w \cdot x$ más pequeño para que $h_w(X)$ emite un 0.

Regla de aprendizaje del perceptrón

Por lo general, la regla de aprendizaje se aplica un ejemplo a la vez, eligiendo ejemplos al azar (como en el descenso de gradiente estocástico). Figura 19.16(a) muestra una **curva de entrenamiento** por este aprendizaje

regla aplicada a los datos de terremotos/explosiones mostrados enFigura 19.15(a). Una curva de entrenamiento mide el rendimiento del clasificador en un conjunto de entrenamiento fijo a medida que avanza el proceso de aprendizaje una actualización a la vez en ese conjunto de entrenamiento. La curva muestra la regla de actualización que converge a una separador lineal de error cero. El proceso de "convergencia" no es exactamente agradable, pero siempre obras. Esta ejecución en particular toma 657 pasos para converger, para un conjunto de datos con 63 ejemplos, por lo que cada ejemplo se presenta aproximadamente 10 veces en promedio. Por lo general, la variación entre ejecuciones es largo.

Figura 19.16



(a) Gráfica de la precisión total del conjunto de entrenamiento frente al número de iteraciones a través del conjunto de entrenamiento para la regla de aprendizaje del perceptrón, dados los datos de terremoto/explosión enFigura 19.15(a). (b) La misma gráfica para los datos no separables con ruido enFigura 19.15(b); nótese el cambio de escala de laX-eje. (c) La misma gráfica que en (b), con un programa de tasa de aprendizaje $\alpha(t) = 1000/(1000 + t)$.

Curva de entrenamiento

Hemos dicho que la regla de aprendizaje del perceptrón converge a un separador lineal perfecto cuando los puntos de datos son linealmente separables; pero ¿y si no lo son? Esta situación es demasiado común en el mundo real. Por ejemplo,Figura 19.15(b)vuelve a agregar en los puntos de datos que quedan fuera porKebfácil*et al.*, (1998)cuando graficaron los datos que se muestran enFigura 19.15(a). En Figura 19.16(b), mostramos que la regla de aprendizaje del perceptrón no converge incluso después de 10.000 pasos: a pesar de que llega a la solución de error mínimo (tres errores) muchas veces, el algoritmo sigue cambiando los pesos. En general, la regla del perceptrón puede no converger a un solución estable para tasa de aprendizaje fija α, pero si α decae como$O(1/t)$dóndetes la iteración número, entonces se puede demostrar que la regla converge a una solución de error mínimo cuando los ejemplos se presentan en una secuencia aleatoria.[10]También se puede demostrar que encontrar el

solución de mínimo error es NP-difícil, por lo que uno espera que muchas presentaciones de los ejemplos serán necesarios para lograr la convergencia.Figura 19.16(c)-muestra el entrenamiento proceso con un programa de tasa de aprendizaje $\alpha(t) = 1000/(1000 + t)$:la convergencia no es perfecta después de 100.000 iteraciones, pero es mucho mejor que el caso de α fijo.

10Técnicamente, requerimos que$\sum_{t=1}^{\infty}\alpha(t) = \infty$y$\sum_{t=1}^{\infty}\alpha^2(t) < \infty$.La tasa de aprendizaje $\alpha(t) = O(1/t)$satisface estas condiciones. Con frecuencia usamosc/(c + t)para alguna constante bastante grandeC.

19.6.5 Clasificación lineal con regresión logística

Hemos visto que pasar la salida de una función lineal a través de la función de umbral crea un clasificador lineal; sin embargo, la naturaleza dura del umbral causa algunos problemas: el hipótesis$h_{\mathbf{w}}(\mathbf{X})$no es diferenciable y de hecho es una función discontinua de sus entradas y sus pesos Esto hace que aprender con la regla del perceptrón sea una aventura muy impredecible. Además, el clasificador lineal siempre anuncia una predicción completamente segura de 1 o 0, incluso para ejemplos que están muy cerca del límite; sería mejor si pudiera clasifique algunos ejemplos como un claro 0 o 1, y otros como casos límite poco claros.

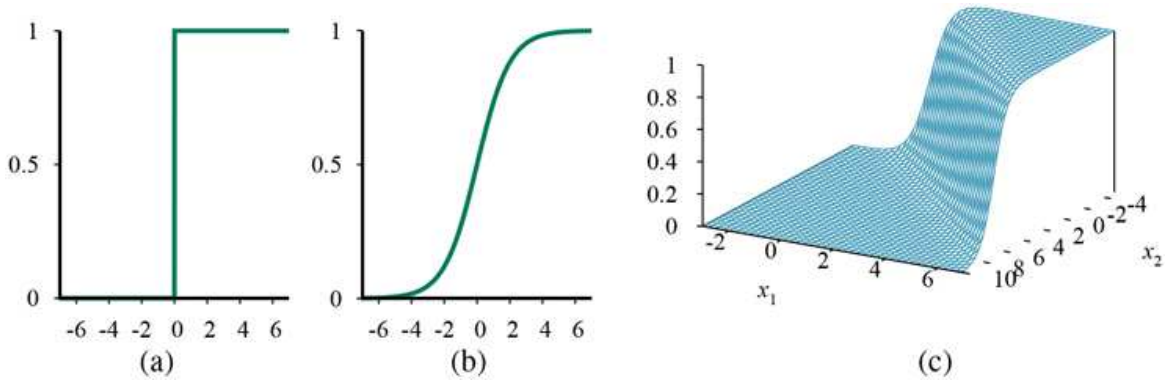
Todos estos problemas se pueden resolver en gran medida suavizando la función de umbral: aproximando el umbral duro con una función continua y diferenciable. EnCapítulo 13-(página424), vimos dos funciones que parecen umbrales suaves: la integral del estándar distribución normal (usada para el modelo probit) y la función logística (usada para el modelo logit modelo). Aunque las dos funciones son muy similares en forma, la función logística

$$\text{Logística}(z) = \frac{1}{1 + mi^{-z}}$$

tiene propiedades matemáticas más convenientes. La función se muestra enFigura 19.17(b)-. Con la función logística reemplazando la función de umbral, ahora tenemos

$$h_{\mathbf{w}}(\mathbf{x}) = \text{Logística}(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + mi^{-\mathbf{w}\cdot\mathbf{x}}}.$$

Figura 19.17



(a) La función de umbral duroUmbral (z)con salida 0/1. Nótese que la función no es derivable az = 0. (b) La función logística,Logística(z) = $\frac{1}{1+e^{-z}}$,también conocida como función sigmoidea. (c) Parcela de un hipótesis de regresión logísticah$_w$(x) = Logística(w · x)por los datos mostrados enFigura 19.15(b)-.

Se muestra un ejemplo de tal hipótesis para el problema de terremoto/explosión de dos entradas enFigura 19.17(c)-. Observe que la salida, siendo un número entre 0 y 1, puede ser interpretado como un*probabilidad*de pertenecer a la clase etiquetada como 1. La hipótesis forma una suave límite en el espacio de entrada y da una probabilidad de 0.5 para cualquier entrada en el centro del región límite, y se aproxima a 0 o 1 a medida que nos alejamos del límite.

El proceso de ajustar los pesos de este modelo para minimizar la pérdida en un conjunto de datos se denomina **Regresión logística**. No existe una solución fácil de forma cerrada para encontrar el valor óptimo dew con este modelo, pero el cálculo del descenso del gradiente es sencillo. Porque nuestro las hipótesis ya no generan solo 0 o 1, usaremos elL_2función de pérdida; también, para mantener la fórmulas legibles, usaremosgramorepresentar la función logística, congramo′su derivado.

Regresión logística

Para un solo ejemplo (x, y),la derivación del gradiente es la misma que para la regresión lineal (Ecuación (19.5)-) hasta el punto en que la forma real dehse inserta. (Para esto derivación, nuevamente necesitamos la regla de la cadena.) Tenemos

$$\frac{\partial}{\partial w_i}\text{Pérdida}(w) = \frac{\partial}{\partial w_i}(y - h_w(X))^2$$
$$= 2(y - h_w(x)) \times \frac{\partial}{\partial w_i}(y - h_w(X))$$
$$= -2(y - h_w(x)) \times \text{gramo}'(w \cdot x) \times \frac{\partial}{\partial w_i} w \cdot x$$
$$= -2(y - h_w(x)) \times \text{gramo}'(w \cdot x) \times x_i.$$

La derivadagramo′de la función logística satisfacegramo′(z) = g(z)(1 − g(z)),entonces tenemos

$$\text{gramo}'(w \cdot x) = g(w \cdot x)(1 - g(w \cdot x)) = h_w(x)(1 - h_w(X))$$

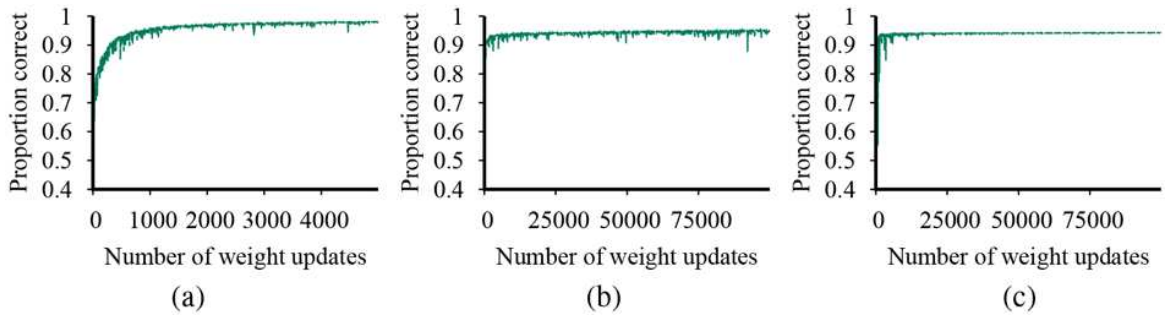
por lo que la actualización de peso para minimizar la pérdida da un paso en la dirección de la diferencia entre entrada y predicción, $(y - h_w(X))$,y la longitud de ese paso depende de la constante α ygramo′:

(19.9)

$$w_i \leftarrow w_i + \alpha\,(y - h_w(x)) \times h_w(x)(1 - h_w(x)) \times x_i.$$

Repitiendo los experimentos deFigura 19.16-con regresión logística en lugar de la lineal clasificador de umbral, obtenemos los resultados que se muestran enFigura 19.18-. En (a), el linealmente caso separable, la regresión logística es algo más lenta para converger, pero se comporta mucho más predeciblemente En (b) y (c), donde los datos son ruidosos y no separables, la regresión logística converge de forma mucho más rápida y fiable. Estas ventajas tienden a trasladarse al mundo real. aplicaciones, y la regresión logística se ha convertido en una de las clasificaciones más populares técnicas para problemas en medicina, marketing, análisis de encuestas, calificación crediticia, público salud y otras aplicaciones.

Figura 19.18



Repetición de los experimentos enFigura 19.16-mediante regresión logística. La gráfica en (a) cubre 5000 iteraciones en lugar de 700, mientras que las gráficas en (b) y (c) usan la misma escala que antes.

19.9 Desarrollo de sistemas de aprendizaje automático

En este capítulo nos hemos concentrado en explicar la*teoría*de aprendizaje automático. los
*práctica*de usar el aprendizaje automático para resolver problemas prácticos es una disciplina separada. Sobre
En los últimos 50 años, la industria del software ha desarrollado una metodología de desarrollo de software
eso hace que sea más probable que un proyecto de software (tradicional) sea un éxito. Pero somos
aún en las primeras etapas de definición de una metodología para proyectos de aprendizaje automático; las herramientas
y las técnicas no están tan bien desarrolladas. Aquí hay un desglose de los pasos típicos en el
proceso.

19.9.1 Formulación del problema

El primer paso es averiguar qué problema quieres resolver. Hay dos partes en esto.
Primero pregúntese, "¿qué problema quiero resolver para mis usuarios?" Una respuesta como “hacerlo más fácil
para que los usuarios organicen y accedan a sus fotos” es demasiado vago; “ayudar a un usuario a encontrar todas las fotos que
coincidir con un término específico, como*París*" es mejor. Luego pregunte, “¿qué parte(s) del problema se puede
¿Resuelto por aprendizaje automático? tal vez decidirse por “aprender una función que asigna una foto a un
conjunto de etiquetas; luego, cuando se le dé una etiqueta como consulta, recupere todas las fotos con esa etiqueta”.

Para concretar esto, debe especificar una función de pérdida para su aprendizaje automático
componente, tal vez midiendo la precisión del sistema para predecir una etiqueta correcta. Este
El objetivo debe estar correlacionado con sus verdaderas metas, pero por lo general será distinto: la verdadera
El objetivo podría ser maximizar la cantidad de usuarios que gana y mantiene en su sistema, y la
ingresos que producen. Esas son métricas que debe rastrear, pero no necesariamente las que
puede construir directamente un modelo de aprendizaje automático para.

Cuando haya descompuesto su problema en partes, puede encontrar que hay múltiples
componentes que pueden ser manejados por ingeniería de software anticuada, no por máquina
aprendizaje. Por ejemplo, para un usuario que solicita "mejores fotos", podría implementar un simple
procedimiento que ordena las fotos por el número de "me gusta" y vistas. Una vez que haya desarrollado
su sistema general hasta el punto en que sea viable, luego puede volver atrás y optimizar,
reemplazando los componentes simples con modelos de aprendizaje automático más sofisticados.

Parte de la formulación del problema es decidir si se trata de un problema supervisado o aprendizaje no supervisado o por refuerzo. Las distinciones no siempre son tan claras. En **aprendizaje semisupervisado**se nos dan algunos ejemplos etiquetados y los usamos para extraer más información de una gran colección de ejemplos no etiquetados. Esto se ha vuelto común emergente, con empresas emergentes cuya misión es etiquetar rápidamente algunos ejemplos, en para ayudar a los sistemas de aprendizaje automático a hacer un mejor uso del resto sin etiquetar ejemplos

---

Aprendizaje semisupervisado

A veces tienes la opción de elegir qué enfoque utilizar. Considere un sistema para recomendar canciones o películas a los clientes. Podríamos abordar esto como un problema de aprendizaje supervisado, donde las entradas incluyen una representación del cliente y la salida etiquetada es si o no les gustó la recomendación, o podríamos abordarlo como un aprendizaje de refuerzo problema, donde el sistema hace una serie de acciones de recomendación, y ocasionalmente obtiene una recompensa del cliente por hacer una buena sugerencia.

Las etiquetas en sí pueden no ser las verdades oraculares que esperamos. Imagina que eres tratando de construir un sistema para adivinar la edad de una persona a partir de una foto. Reúnes algunos etiquetados ejemplos haciendo que las personas carguen fotos y digan su edad. Eso es aprendizaje supervisado. Pero en realidad algunas de las personas mintieron sobre su edad. No es solo que haya ruido aleatorio en los datos; más bien, las inexactitudes son sistemáticas, y descubrirlas es un trabajo no supervisado. problema de aprendizaje que involucra imágenes, edades autoinformadas y edades verdaderas (desconocidas). Así, ambos el ruido y la falta de etiquetas crean un continuo entre el aprendizaje supervisado y no supervisado. El campo de**aprendizaje poco supervisado**se centra en el uso de etiquetas ruidosas, imprecisas o proporcionada por no expertos.

Aprendizaje débilmente supervisado

19.9.2 Recopilación, evaluación y gestión de datos

Cada proyecto de aprendizaje automático necesita datos; en el caso de nuestro proyecto de identificación con foto hay conjuntos de datos de imágenes disponibles gratuitamente, como**ImageNet**, que tiene más de 14 millones fotos con unas 20.000 etiquetas diferentes. A veces puede que tengamos que fabricar los nuestros datos, que pueden ser realizados por nuestro propio trabajo, o por**colaboración colectiva**a los trabajadores remunerados o no remunerados voluntarios que operan a través de un servicio de Internet. A veces los datos provienen de sus usuarios. Para Por ejemplo, el servicio de navegación Waze anima a los usuarios a cargar datos sobre atascos de tráfico, y lo utiliza para proporcionar instrucciones de navegación actualizadas para todos los usuarios. Transferir el aprendizaje (ver Sección 21.7.2-) se puede usar cuando no tiene suficientes datos propios: comience con un Conjunto de datos de propósito general disponible públicamente (o un modelo que ha sido entrenado previamente en este data), y luego agregue datos específicos de sus usuarios y vuelva a entrenar.

ImageNet

Si implementa un sistema para los usuarios, sus usuarios proporcionarán comentarios, tal vez haciendo clic en un elemento e ignorando los demás. Necesitará una estrategia para manejar estos datos. Que implica una revisión con expertos en privacidad (verSección 27.3.2-) para asegurarse de que obtiene la permiso adecuado para los datos que recopila, y que tiene procesos para asegurar la integridad de los datos del usuario, y que ellos entiendan lo que harás con ellos. También necesitas para asegurarse de que sus procesos sean justos e imparciales (verSección 27.3.3-). si hay datos que cree que es demasiado sensible para recopilar pero que sería útil para un aprendizaje automático modelo, considere un enfoque de aprendizaje federado donde los datos permanecen en el dispositivo del usuario, pero los parámetros del modelo se comparten de una manera que no revela datos privados.

Es una buena práctica mantener**procedencia de los datos**para todos sus datos. Para cada columna de su conjunto de datos, debe conocer la definición exacta, de dónde provienen los datos, cuáles son los posibles son los valores y quién ha trabajado en ello. ¿Hubo períodos de tiempo en los que una fuente de datos fue interrumpido? ¿La definición de alguna fuente de datos evolucionó con el tiempo? tendrás que saber esto si desea comparar resultados entre períodos de tiempo.

Procedencia de los datos

Esto es especialmente cierto si se basa en datos producidos por otra persona: sus sus necesidades y las suyas pueden diferir, y pueden terminar cambiando la forma en que se procesan los datos. producido, o podría dejar de actualizarlo todo junto. Necesita monitorear sus fuentes de datos para atrapa esto. Tener una tubería de manejo de datos confiable, flexible y segura es más crítico para éxito que los detalles exactos del algoritmo de aprendizaje automático. La procedencia también es importante por razones legales, como el cumplimiento de la ley de privacidad.

Para cualquier tarea habrá preguntas sobre los datos: ¿Son estos los datos correctos para mi tarea? Lo hace capturar suficientes entradas correctas para darnos la oportunidad de aprender un modelo? ¿Contiene las salidas que quiero predecir? Si no, ¿puedo construir un modelo sin supervisión? ¿O puedo etiquetar un parte de los datos y luego hacer un aprendizaje semisupervisado? ¿Son datos relevantes? es genial tienes 14 millones de fotos, pero si todos tus usuarios son especialistas interesados en un tema específico, entonces una base de datos general no ayudará; deberá recopilar fotos sobre el tema específico. Cómo ¿Cuántos datos de entrenamiento son suficientes? (¿Necesito recopilar más datos? ¿Puedo descartar algunos datos para hacer el cálculo más rápido?) La mejor manera de responder esto es razonar por analogía con un proyecto con un tamaño de conjunto de entrenamiento conocido.

Una vez que comience, puede dibujar una curva de aprendizaje (verFigura 19.7-) a ver si hay más datos ayudará, o si el aprendizaje ya se ha estancado. Hay un sinfín de reglas ad hoc, injustificadas de pulgar para ver la cantidad de ejemplos de capacitación que necesitará: millones para problemas difíciles; miles para problemas promedio; cientos o miles para cada clase en una clasificación problema; 10 veces más ejemplos que parámetros del modelo; 10 veces más ejemplos que características de entrada;O(d registro d)ejemplos paradfunciones de entrada; más ejemplos de no lineal modelos que para los modelos lineales; más ejemplos si se requiere mayor precisión; menos ejemplos si usa la regularización; suficientes ejemplos para lograr el poder estadístico necesario rechazar la hipótesis nula en la clasificación. Todas estas reglas vienen con advertencias, como hace la regla sensata que sugiere probar lo que ha funcionado en el pasado para problemas similares.

Debe pensar a la defensiva sobre sus datos. ¿Puede haber errores en la entrada de datos? Qué puede hacerse con los campos de datos que faltan? Si recopila datos de sus clientes (u otras personas) ¿Podrían algunas de las personas ser adversarios para jugar con el sistema? ¿Hay errores ortográficos o

terminología inconsistente en los datos de texto? (Por ejemplo, escriba "Apple", "AAPL" y "Apple Inc". ¿Todos se refieren a la misma empresa?) Necesitará un proceso para detectar y corregir todos estos posibles fuentes de error de datos.

Cuando los datos son limitados,**aumento de datos**poder ayudar. Por ejemplo, con un conjunto de datos de imágenes, puede crear múltiples versiones de cada imagen girando, traduciendo, recortando o escalando cada imagen, cambiando el brillo o el balance de color o agregando ruido. Mientras estos son pequeños cambios, la etiqueta de la imagen debe permanecer igual, y un modelo entrenado en tal los datos aumentados serán más robustos.

Aumento de datos

A veces los datos son abundantes, pero se clasifican en**clases desequilibradas**. por ejemplo, un conjunto de entrenamiento de transacciones de tarjeta de crédito puede consistir en 10,000,000 transacciones válidas y 1.000 fraudulentos. Un clasificador que dice "válido" independientemente de la entrada logrará 99,99 % de precisión en este conjunto de datos. Para ir más allá, un clasificador tendrá que pagar más atención a los ejemplos fraudulentos. Para ayudarlo a hacer eso, puede**submuestreo**la mayoría clase (es decir, ignorar algunos de los ejemplos de clase "válidos") o**exceso de muestra**la clase minoritaria (es decir, duplicar algunos de los ejemplos de clases "fraudulentas"). Puede utilizar una función de pérdida ponderada eso da una penalización mayor por perder un caso fraudulento.

Clases desequilibradas

---

submuestreo

Sobremuestra

Impulsar también puede ayudarlo a concentrarse en la clase minoritaria. Si está utilizando un conjunto método, puede cambiar las reglas por las que el conjunto vota y dar "fraudulento" como el respuesta incluso si solo una minoría del conjunto vota por "fraudulento". Puedes ayudar equilibrar clases desequilibradas generando datos sintéticos con técnicas como SMOTE (Chawla*et al.*, 2002) o unaDASYN(Él*et al.*, 2008).

Debes considerar cuidadosamente**valores atípicos**en tus datos Un valor atípico es un punto de datos que está lejos de otros puntos Por ejemplo, en el problema del restaurante, si el precio fuera un valor numérico en lugar de que uno categórico, y si un ejemplo tuviera un precio de $316mientras que todos los demás eran $ 30 o menos, ese ejemplo sería un caso atípico. Métodos como la regresión lineal son susceptibles a valores atípicos porque deben formar un solo modelo lineal global que tome todas las entradas en cuenta: no pueden tratar el valor atípico de manera diferente a otros puntos de ejemplo y, por lo tanto, un solo El valor atípico puede tener un gran efecto en todos los parámetros del modelo.

Parte aislada

Con atributos como el precio que son números positivos, podemos disminuir el efecto de los valores atípicos por transformando los datos, tomando el logaritmo de cada valor, entonces $20, $25,y $316convertirse en 1.3, 1.4 y 2.5. Esto tiene sentido desde un punto de vista práctico porque el alto valor ahora tiene menos influencia sobre el modelo, y desde un punto de vista teórico porque, como vimos en Sección 16.3.2-, la utilidad del dinero es logarítmica.

Los métodos como los árboles de decisión que se construyen a partir de múltiples modelos locales pueden tratar los valores atípicos individualmente: no importa si el mayor valor es $300o $31;de cualquier manera se puede tratar en su propio nodo local después de una prueba de la formacosto ≤ 30.Eso hace que los árboles de decisión (y por lo tanto bosques aleatorios y aumento de gradiente) más robustos a los valores atípicos.

Ingeniería de características

Después de corregir los errores manifiestos, es posible que también desee preprocesar sus datos para que sea más fácil digerir. Ya hemos visto el proceso de cuantización: forzar una entrada valorada continua, como el tiempo de espera, en contenedores fijos (0 − 10 minutos, 10 − 30, 30 − 60 o >60) .Dominio el conocimiento puede decirle qué umbrales son importantes, como compararedad ≥ 18cuando

estudiar los patrones de votación. También vimos (página688.) que los algoritmos del vecino más cercano funcionan mejor cuando los datos se normalizan para tener una desviación estándar de 1. Con categórica atributos como soleado/nublado/lluvioso, a menudo es útil transformar los datos en tres atributos booleanos separados, exactamente uno de los cuales es verdadero (a esto lo llamamos**codificación one-hot**). Esto es particularmente útil cuando el modelo de aprendizaje automático es una red neuronal.

Codificación one-hot

También puede introducir nuevos atributos basados en su conocimiento del dominio. Por ejemplo, dado un conjunto de datos de compras de clientes donde cada entrada tiene un atributo de fecha, podría desea aumentar los datos con nuevos atributos que indiquen si la fecha es un fin de semana o fiesta.

Como otro ejemplo, considere la tarea de estimar el valor real de las casas que son para rebaja. EnFigura 19.13.mostramos una versión de juguete de este problema, haciendo una regresión lineal de tamaño de la casa al precio de venta. Pero realmente queremos estimar el precio de venta de una casa, no el precio de venta. Para resolver esta tarea necesitaremos datos sobre las ventas reales. Pero eso no significa que nosotros debería desechar los datos sobre el precio solicitado; podemos usarlo como una de las características de entrada. Además del tamaño de la casa, necesitaremos más información: el número de habitaciones, dormitorios y baños; si la cocina y los baños han sido recientemente remodelado; la edad de la casa y tal vez su estado de conservación; si tiene central calefacción y aire acondicionado; el tamaño del patio y el estado del paisaje.

También necesitaremos información sobre el lote y el vecindario. Pero, ¿cómo definimos ¿vecindario? ¿Por código postal? ¿Qué pasa si un código postal está a caballo entre lo deseable y lo indeseable? ¿vecindario? ¿Qué pasa con el distrito escolar? Si el*nombre*del distrito escolar ser un característica, o la*puntajes promedio de las pruebas*? La capacidad de hacer un buen trabajo de ingeniería de características es fundamental para el éxito. ComoPedro Domingos (2012)dice: "Al final del día, alguna máquina los proyectos de aprendizaje tienen éxito y algunos fracasan. ¿Qué hace la diferencia? Fácilmente el más factor importante son las características utilizadas."

Análisis y visualización de datos exploratorios

John Tukey (1977)acuñar el termino**análisis exploratorio de datos**(EDA) para el proceso de explorar datos para comprenderlos, no para hacer predicciones o probar hipótesis Esto se hace principalmente con visualizaciones, pero también con estadísticas de resumen. Mirar algunos histogramas o diagramas de dispersión a menudo puede ayudar a determinar si faltan datos o si erróneo; si sus datos se distribuyen normalmente o son de cola pesada; y que aprendizaje modelo puede ser apropiado.

Puede ser útil agrupar sus datos y luego visualizar un prototipo de punto de datos en el centro de cada racimo. Por ejemplo, en el conjunto de datos de imágenes, puedo identificar que aquí hay un grupo de caras de gato; cerca hay un grupo de gatos dormidos; otros grupos representan otros objetos. Esperar iterar varias veces entre la visualización y el modelado: para crear clústeres necesita un función de distancia para decirle qué elementos están cerca uno del otro, pero para elegir una buena distancia función que necesita cierta sensación de los datos.

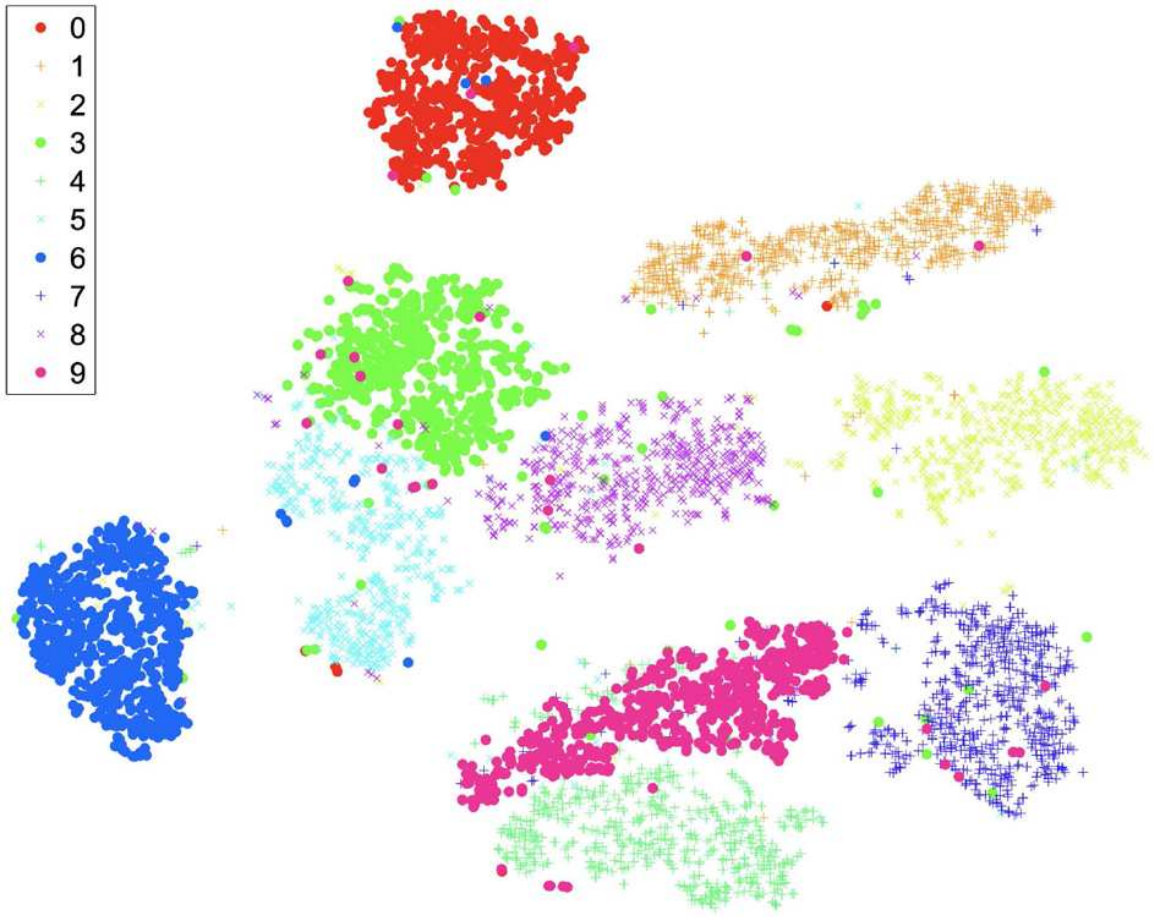
También es útil para detectar valores atípicos que están lejos de los prototipos; estos pueden ser considerados **críticos**del modelo prototipo, y puede darle una idea de qué tipo de errores su sistema Podría hacer. Un ejemplo sería un gato disfrazado de león.

Los dispositivos de visualización de nuestra computadora (pantallas o papel) son bidimensionales, lo que significa que es fácil visualizar datos bidimensionales. Y nuestros ojos son experimentados en entender datos tridimensionales que han sido proyectados en dos dimensiones. Pero muchos datos los conjuntos tienen docenas o incluso millones de dimensiones. Para visualizarlos podemos hacer reducción de la dimensionalidad, proyectando los datos a un**mapa**en dos dimensiones (o a veces a tres dimensiones, que luego se pueden explorar de forma interactiva).17

17Geoffrey Hinton brinda el útil consejo "Para manejar un espacio de 14 dimensiones, visualice un espacio en 3D y diga 'catorce' a en voz muy alta.

El mapa no puede mantener todas las relaciones entre puntos de datos, pero debe tener la propiedad que puntos similares en el conjunto de datos original están muy juntos en el mapa. Una técnica llamada **incrustación de vecinos estocásticos distribuidos en t (t-SNE)**hace justo eso.Figura 19.27-muestra una Mapa t-SNE del conjunto de datos de reconocimiento de dígitos MNIST. Paquetes de análisis y visualización de datos como Pandas, Bokeh y Tableau pueden facilitar el trabajo con sus datos.

Figura 19.27



Un mapa t-SNE bidimensional del conjunto de datos MNIST, una colección de 60.000 imágenes de dígitos escritos a mano, cada uno28×28píxeles y por lo tanto 784 dimensiones. Puede ver claramente los grupos de diez dígitos, con algunas confusiones en cada grupo; por ejemplo, el grupo superior es para el dígito 0, pero dentro de los límites del grupo hay algunos puntos de datos que representan los dígitos 3 y 6. El algoritmo t-SNE encuentra una representación que acentúa las diferencias entre los grupos.

Incrustación de vecinos estocásticos distribuidos en T (t-SNE)

19.9.3 Selección y entrenamiento del modelo

Con datos limpios en la mano y una sensación intuitiva para ellos, es hora de construir un modelo. Eso significa elegir una clase modelo (¿bosques aleatorios? ¿redes neuronales profundas? ¿un conjunto?), entrenamiento su modelo con los datos de entrenamiento, ajustando cualquier hiperparámetro de la clase (número de ¿árboles? número de capas?) con los datos de validación, depurando el proceso, y finalmente evaluar el modelo en los datos de prueba.

No existe una forma garantizada de elegir la mejor clase de modelo, pero hay algunas pautas. Los bosques aleatorios son buenos cuando hay muchas características categóricas y Creo que muchos de ellos pueden ser irrelevantes. Los métodos no paramétricos son buenos cuando tiene muchos datos y ningún conocimiento previo, y cuando no quiere preocuparse demasiado sobre elegir las características correctas (siempre y cuando haya menos de 20). Sin embargo, los métodos no paramétricos generalmente te dan una función $h$que es más caro de ejecutar.

La regresión logística funciona bien cuando los datos son linealmente separables, o se pueden convertir para ser así con la ingeniería de características inteligente. Las máquinas de vectores de soporte son un buen método para probar cuando el conjunto de datos no es demasiado grande; funcionan de manera similar a la regresión logística en datos separables y puede ser mejor para datos de alta dimensión. Problemas relacionados con el reconocimiento de patrones, como el procesamiento de imágenes o del habla, se abordan con mayor frecuencia con redes neuronales profundas (ver capitulo 21-).

La elección de hiperparámetros se puede hacer con una combinación de experiencia: haga lo que funcionó bien en problemas anteriores similares, y busque: realice experimentos con múltiples valores posibles para hiperparámetros. A medida que realice más experimentos obtendrá ideas para diferentes modelos para probar. Sin embargo, si mide el rendimiento en los datos de validación, obtenga una nueva idea y ejecute más experimentos, entonces corre el riesgo de sobreajustar los datos de validación. Si usted tiene suficientes datos, es posible que desee tener varios conjuntos de datos de validación separados para evitar esto problema. Esto es especialmente cierto si inspecciona los datos de validación a ojo, en lugar de simplemente ejecutar valoraciones al respecto.

Suponga que está creando un clasificador, por ejemplo, un sistema para clasificar el correo electrónico no deseado. Etiquetado un correo legítimo como spam se denomina **falso positivo**. Habrá un intercambio entre falsos positivos y falsos negativos (etiquetar una pieza de spam como legítima); si quieres mantenga más correo legítimo fuera de la carpeta de spam, necesariamente terminará enviando más spam a la bandeja de entrada. Pero, ¿cuál es la mejor manera de hacer el intercambio? Puedes probar diferentes valores de hiperparámetros y obtener tasas diferentes para los dos tipos de errores: diferentes puntos en esta compensación. Un gráfico llamado **Curva característica operativa del receptor (ROC)** traza falsos positivos versus verdaderos positivos para cada valor del hiperparámetro, ayudándole visualizar valores que serían buenas opciones para la compensación. Una métrica llamada "área bajo la curva ROC" o **ABC** proporciona un resumen de un solo número de la curva ROC, que es útil si desea implementar un sistema y permitir que cada usuario elija su punto de equilibrio.

Falso positivo

Curva característica operativa del receptor (ROC)

ABC

Otra herramienta de visualización útil para problemas de clasificación es un**matriz de confusión**: un dos-
tabla dimensional de recuentos de la frecuencia con la que cada categoría se clasifica o se clasifica erróneamente como cada
otra categoría.

Matriz de confusión

Puede haber compensaciones en factores distintos de la función de pérdida. Si puedes entrenar un mercado de valores
modelo de predicción que te hace ganar $10 en cada operación, eso es genial, pero no si te cuesta $20
en el costo de cómputo para cada predicción. Un programa de traducción automática que se ejecuta en su
teléfono y le permite leer las señales en una ciudad extranjera es útil, pero no si se agota
batería después de una hora de uso. Lleve un registro de todos los factores que conducen a la aceptación o el rechazo
de su sistema, y diseñe un proceso donde pueda iterar rápidamente el proceso de obtener un
nueva idea, realizar un experimento y evaluar los resultados del experimento para ver si
han hecho progresos. Hacer que este proceso de iteración sea rápido es uno de los factores más importantes
para el éxito en el aprendizaje automático.

19.9.4 Confianza, interpretabilidad y explicabilidad

Hemos descrito una metodología de aprendizaje automático en la que desarrolla su modelo con
datos de entrenamiento, elija hiperparámetros con datos de validación y obtenga una métrica final con prueba
datos. Hacerlo bien en esa métrica es una condición necesaria pero no suficiente para que**confianza**

tu modelo Y no es solo usted, otras partes interesadas, incluidos los reguladores, los legisladores, el prensa, y sus usuarios también están interesados en la confiabilidad de su sistema (así como en atributos relacionados tales como confiabilidad, responsabilidad y seguridad).

Un sistema de aprendizaje automático sigue siendo una pieza de software, y puede generar confianza con todos los herramientas típicas para verificar y validar cualquier sistema de software:

- **FUENTE DE CONTROL:**Sistemas para control de versiones, compilación y seguimiento de errores/problemas.
- **PRUEBAS:**Pruebas unitarias para todos los componentes que cubren casos canónicos simples así como casos complicados contradictorios, pruebas de fuzz (donde se generan entradas aleatorias), regresión pruebas, pruebas de carga y pruebas de integración del sistema: todos estos son importantes para cualquier software sistema. Para el aprendizaje automático, también tenemos pruebas en el entrenamiento, validación y prueba. conjuntos de datos
- **REVISIÓN:**Revisiones y revisiones del código, revisiones de privacidad, revisiones de imparcialidad (ver Sección 27.3.3.), y otras revisiones de cumplimiento legal.
- **VIGILANCIA:**Tableros y alertas para asegurarse de que el sistema esté en funcionamiento y sigue funcionando a un alto nivel de precisión.
- **RESPONSABILIDAD:**¿Qué pasa cuando el sistema está mal? ¿Cuál es el proceso para quejarse o apelar la decisión del sistema? ¿Cómo podemos rastrear quién fue responsable del error? La sociedad espera (pero no siempre obtiene) responsabilidad por decisiones importantes tomadas por los bancos, los políticos y la ley, y deben esperar responsabilidad de los sistemas de software, incluidos los sistemas de aprendizaje automático.

Además, hay algunos factores que son especialmente importantes para el aprendizaje automático sistemas, como detallaremos a continuación.

INTERPRETACIÓN: Decimos que un modelo de aprendizaje automático es**interpretable**si puede inspeccionar el modelo real y comprender por qué obtuvo una respuesta particular para una entrada determinada, y cómo cambiaría la respuesta cuando cambia la entrada.[18]Los modelos de árboles de decisión son considerado altamente interpretable; podemos entender que siguiendo el camino Mecenas = CompletoyEstimación de espera = 0–10en un árbol de decisión conduce a una decisión deEspere.Un árbol de decisión es interpretable por dos razones. En primer lugar, los humanos tenemos experiencia en comprensión de las reglas SI/ENTONCES. (Por el contrario, es muy difícil para los humanos obtener una comprensión intuitiva del resultado de una matriz multiplicada seguida de una función de activación, como se hace en algunos modelos de redes neuronales). En segundo lugar, el árbol de decisión se construyó en cierto sentido.

para ser interpretable: la raíz del árbol fue elegida para ser el atributo con la mayor ganancia de información.

[18] Esta terminología no es universalmente aceptada; algunos autores usan "interpretable" y "explicable" como sinónimos, ambos referidos a llegar a algún tipo de comprensión de un modelo.

Interpretabilidad

Los modelos de regresión lineal también se consideran interpretables; podemos examinar un modelo para predecir el alquiler de un apartamento y ver que por cada dormitorio añadido, el alquiler aumenta en $500, según el modelo. Esta idea de "Si cambioX,como sera la salida ¿cambio?" está en el centro de la interpretabilidad. Por supuesto, la correlación no es causalidad, por lo que modelos interpretables están respondiendo*qué*es el caso, pero no necesariamente*por qué*es el caso

---

explicabilidad

EXPLICABILIDAD: Un modelo explicable es aquel que puede ayudarte a entender "*por qué*estaba esta salida producida para esta entrada? En nuestra terminología, la interpretabilidad se deriva de inspeccionando el modelo real, mientras que la explicabilidad puede ser proporcionada por un proceso separado. Es decir, el modelo en sí puede ser una caja negra difícil de entender, pero un módulo de explicación puede resumir lo que hace el modelo. Para un sistema de reconocimiento de imagen de red neuronal que clasifica una imagen como*perro*, si tratáramos de interpretar el modelo directamente, lo mejor que podríamos llegar fuera sería algo así como "después de procesar las capas convolucionales, la activación Para el*perro*la producción en la capa softmax fue más alta que cualquier otra clase". Eso no es muy argumento de peso. Pero un módulo de explicación separado podría ser capaz de examinar el modelo de red neuronal y proponga la explicación "tiene cuatro patas, pelaje, una cola, orejas y un hocico largo; es más pequeño que un lobo, y está acostado en la cama de un perro, así que creo que es un perro." Las explicaciones son una forma de generar confianza, y algunas normativas como la europea GDPR (Reglamento General de Protección de Datos) requiere que los sistemas proporcionen explicaciones.

Como ejemplo de un módulo de explicación separado, el modelo local interpretable-agnóstico El sistema de explicaciones (LIME) funciona así: no importa qué clase de modelo use, LIME construye un modelo interpretable, a menudo un árbol de decisiones o un modelo lineal, que es un aproximación de su modelo, y luego interpreta el modelo lineal para crear explicaciones que dicen cuán importante es cada característica. LIME logra esto tratando la máquina-modelo aprendido como una caja negra, y probarlo con diferentes valores de entrada aleatorios para crear un conjunto de datos a partir del cual se puede construir el modelo interpretable. Este enfoque es apropiado para datos estructurados, pero no para cosas como imágenes, donde cada píxel es una función y nadie El píxel es "importante" por sí mismo.

A veces elegimos una clase modelo debido a su explicabilidad; podemos elegir la decisión árboles sobre redes neuronales no porque tengan mayor precisión, sino porque la la explicabilidad nos da más confianza en ellos.

Sin embargo, una simple explicación puede dar lugar a una falsa sensación de seguridad. Después de todo, normalmente elegir usar un modelo de aprendizaje automático (en lugar de un programa tradicional escrito a mano) porque el problema que estamos tratando de resolver es intrínsecamente complejo, y no sabemos cómo para escribir un programa tradicional. En ese caso, no debemos esperar que haya necesariamente ser una explicación simple para cada predicción.

Si está creando un modelo de aprendizaje automático principalmente con el fin de comprender el dominio, entonces la interpretabilidad y la explicabilidad lo ayudarán a llegar a esa comprensión. Pero si solo desea el software con el mejor rendimiento, las pruebas pueden brindarle más seguridad y confianza que explicaciones. ¿En qué confiaría usted: un avión experimental? que nunca ha volado antes pero tiene una explicación detallada de por qué es seguro, o un avión que completó con seguridad 100 vuelos anteriores y se ha mantenido cuidadosamente, pero viene sin explicación garantizada?

19.9.5 Operación, monitoreo y mantenimiento

Una vez que esté satisfecho con el rendimiento de su modelo, puede implementarlo para sus usuarios. lo harás enfrentar desafíos adicionales. En primer lugar, está el problema de la**cola larga**de entradas del usuario. Tú puede haber probado su sistema en un conjunto de prueba grande, pero si su sistema es popular, pronto ver entradas que nunca se probaron antes. Necesita saber si su modelo generaliza bien por ellos, lo que significa que usted necesita**monitor**su desempeño en datos en vivo—seguimiento estadísticas, mostrar un tablero y enviar alertas cuando las métricas clave caen por debajo de un

límite. Además de actualizar automáticamente las estadísticas sobre las interacciones de los usuarios, puede necesita contratar y capacitar evaluadores humanos para observar su sistema y calificar qué tan bien lo está haciendo.

Cola larga

Vigilancia

En segundo lugar, está el problema de**no estacionariedad**—el mundo cambia con el tiempo. Suponer su sistema clasifica el correo electrónico como spam o no spam. Tan pronto como clasifique con éxito un lote de mensajes de spam, los spammers verán lo que ha hecho y cambiarán su tácticas, enviando un nuevo tipo de mensaje que no ha visto antes. El no-spam también evoluciona, como los usuarios cambian la combinación de correo electrónico versus mensajería o escritorio versus servicios móviles que usar.

---

no estacionariedad

Continuamente se enfrentará a la pregunta de qué es mejor: un modelo que ha sido bien probado pero se creó a partir de datos más antiguos, frente a un modelo que se creó a partir de los datos más recientes pero que no ha sido probado en uso real. Diferentes sistemas tienen diferentes requisitos de frescura: algunos los problemas se benefician de un nuevo modelo cada día, o incluso cada hora, mientras que otros problemas puede mantener el mismo modelo durante meses. Si está implementando un nuevo modelo cada hora, será No sería práctico ejecutar un conjunto de pruebas pesado y un proceso de revisión manual para cada actualización. Tú necesitará automatizar el proceso de prueba y lanzamiento para que los pequeños cambios puedan ser aprobado automáticamente, pero los cambios más grandes desencadenan una revisión adecuada. Puedes considerar el compensación entre un modelo en línea donde los nuevos datos modifican incrementalmente los existentes modelo, frente a un modelo sin conexión en el que cada nueva versión requiere la creación de un nuevo modelo a partir de rascar.

No es solo que los datos cambiarán, por ejemplo, se usarán nuevas palabras en el correo no deseado.

mensajes de correo electrónico También es que todo el esquema de datos puede cambiar: puede comenzar

clasificación de correo electrónico no deseado y necesidad de adaptarse para clasificar mensajes de texto no deseados, mensajes de voz no deseados

mensajes, videos spam, etc.Figura 19.28da una rúbrica general para guiar al practicante en elegir el nivel apropiado de prueba y monitoreo.

Figura 19.28

Tests for Features and Data

(1) Feature expectations are captured in a schema. (2) All features are beneficial. (3) No feature's cost is too much. (4) Features adhere to meta-level requirements. (5) The data pipeline has appropriate privacy controls. (6) New features can be added quickly. (7) All input feature code is tested.

Tests for Model Development

(1) Every model specification undergoes a code review. (2) Every model is checked in to a repository. (3) Offline proxy metrics correlate with actual metrics (4) All hyperparameters have been tuned. (5) The impact of model staleness is known. (6) A simpler model is not better. (7) Model quality is sufficient on all important data slices. The model has been tested for considerations of inclusion.

Tests for Machine Learning Infrastructure

(1) Training is reproducible. (2) Model specification code is unit tested. (3) The full ML pipeline is integration tested. (4) Model quality is validated before attempting to serve it. (5) The model allows debugging by observing the step-by-step computation of training or inference on a single example. (6) Models are tested via a canary process before they enter production serving environments. (7) Models can be quickly and safely rolled back to a previous serving version.

Monitoring Tests for Machine Learning

(1) Dependency changes result in notification. (2) Data invariants hold in training and serving inputs. (3) Training and serving features compute the same values. (4) Models are not too stale. (5) The model is numerically stable. (6) The model has not experienced regressions in training speed, serving latency, throughput, or RAM usage. (7) The model has not experienced a regression in prediction quality on served data.

Un conjunto de criterios para ver qué tan bien está implementando su modelo de aprendizaje automático con suficientes pruebas. abreviado deBreck*et al.*(2016), que también proporcionan una métrica de puntuación.

Resumen

Este capítulo introdujo el aprendizaje automático y se centró en el aprendizaje supervisado de ejemplos Los puntos principales fueron:

- El aprendizaje toma muchas formas, dependiendo de la naturaleza del agente, el componente a ser mejorado y los comentarios disponibles.
- Si la retroalimentación disponible proporciona la respuesta correcta para entradas de ejemplo, entonces el problema de aprendizaje se llama**aprendizaje supervisado**. La tarea es aprender una función.y = h(x). Aprender una función cuya salida es un valor continuo u ordenado (como*peso*) se llama **regresión**; aprender una función con un pequeño número de posibles categorías de salida es llamó**clasificación**;
- Queremos aprender una función que no solo concuerde con los datos sino que también sea probable que concuerde con datos futuros. Necesitamos equilibrar el acuerdo con los datos contra la simplicidad del hipótesis.
- **Árboles de decisión**puede representar todas las funciones booleanas. los**ganancia de información**heurístico proporciona un método eficiente para encontrar un árbol de decisión simple y consistente.
- El rendimiento de un algoritmo de aprendizaje se puede visualizar mediante un**curva de aprendizaje**, cual muestra la precisión de la predicción en el**equipo de prueba**en función de la**conjunto de entrenamiento**Talla.
- Cuando hay varios modelos para elegir,**selección de modelo**puede elegir buenos valores de hiperparámetros, como lo confirma**validación cruzada**sobre datos de validación. Una vez el se eligen los valores de los hiperparámetros, construimos nuestro mejor modelo usando todos los datos de entrenamiento.
- A veces no todos los errores son iguales. A**función de pérdida**nos dice qué tan malo es cada error; la el objetivo es entonces minimizar la pérdida sobre un conjunto de validación.
- **Teoría del aprendizaje computacional**analiza la complejidad de la muestra y computacional complejidad del aprendizaje inductivo. Existe un equilibrio entre la expresividad de la el espacio de hipótesis y la facilidad de aprendizaje.
- **regresión lineal**es un modelo muy utilizado. Los parámetros óptimos de una regresión lineal El modelo se puede calcular exactamente, o se puede encontrar mediante la búsqueda de descenso de gradiente, que es un técnica que se puede aplicar a modelos que no tienen una solución de forma cerrada.
- Un clasificador lineal con un umbral estricto, también conocido como**perceptrón**—puede ser entrenado por una regla de actualización de peso simple para ajustar los datos que son**separables linealmente**. En otros casos, el la regla no converge.

- **Regresión logística**reemplaza el umbral duro del perceptrón con un umbral suave definida por una función logística. El descenso de gradiente funciona bien incluso para datos ruidosos que son no linealmente separables.
- **Modelos no paramétricos**usar todos los datos para hacer cada predicción, en lugar de intentar resumir los datos con algunos parámetros. Ejemplos incluyen**vecinos más cercanos**y **regresión ponderada localmente**.
- **Máquinas de vectores de soporte**encontrar separadores lineales con**margen máximo**para mejorar el rendimiento de generalización del clasificador.**Métodos del núcleo**transformar implícitamente el datos de entrada en un espacio de alta dimensión donde puede existir un separador lineal, incluso si el los datos originales no son separables.
- Métodos de conjunto como**harpillera**y**impulsar**a menudo funcionan mejor que los individuos métodos. En**aprender en línea**podemos agregar las opiniones de los expertos por venir arbitrariamente cerca del desempeño del mejor experto, incluso cuando la distribución de los datos están cambiando constantemente.
- Construir un buen modelo de aprendizaje automático requiere experiencia en el completo proceso de desarrollo, desde la gestión de datos hasta la selección y optimización de modelos, hasta mantenimiento continuado.

capitulo 21

Aprendizaje profundo

En el que el descenso de gradiente aprende programas de varios pasos, con implicaciones significativas para los principales subcampos de la inteligencia artificial.

Aprendizaje profundoes una amplia familia de técnicas para el aprendizaje automático en el que las hipótesis toman la forma de circuitos algebraicos complejos con puntos fuertes de conexión ajustables. La palabra "profundo" se refiere al hecho de que los circuitos normalmente están organizados en muchos**capas**, cual significa que las rutas de cálculo desde las entradas hasta las salidas tienen muchos pasos. El aprendizaje profundo es actualmente el enfoque más utilizado para aplicaciones tales como el reconocimiento visual de objetos, traducción automática, reconocimiento de voz, síntesis de voz y síntesis de imágenes; también juega un papel importante en las aplicaciones de aprendizaje por refuerzo (vercapitulo 22-).

Aprendizaje profundo

Capa

El aprendizaje profundo tiene su origen en los primeros trabajos que intentaron modelar redes de neuronas en el cerebro (McCulloch y Pitts, 1943) con circuitos computacionales. Por eso, las redes entrenados por métodos de aprendizaje profundo a menudo se denominan**Redes neuronales**, a pesar de el parecido con las células y estructuras neurales reales es superficial.

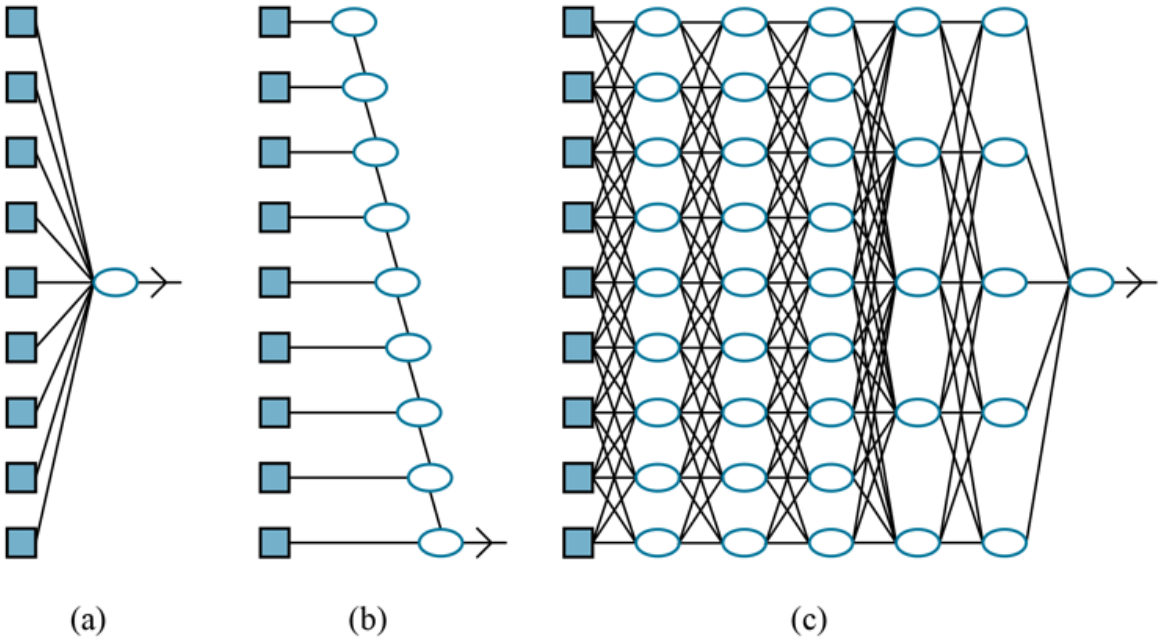
---

red neuronal

Si bien las verdaderas razones del éxito del aprendizaje profundo aún no se han dilucidado por completo, se ha ventajas evidentes sobre algunos de los métodos cubiertos encapitulo 19—particularmente para datos de alta dimensión como imágenes. Por ejemplo, aunque métodos como el lineal y regresión logística puede manejar una gran cantidad de variables de entrada, la ruta de cálculo de cada entrada a la salida es muy corta: la multiplicación por un solo peso, luego se suma a la producción agregada. Además, las diferentes variables de entrada contribuyen de forma independiente a la salida, sin interactuar entre sí (Figura 21.1(a)). Esto limita significativamente la fuerza expresiva de tales modelos. Solo pueden representar funciones lineales y límites.

en el espacio de entrada, mientras que la mayoría de los conceptos del mundo real son mucho más complejos.

Figura 21.1



(a) Un modelo poco profundo, como la regresión lineal, tiene rutas de cálculo cortas entre las entradas y la salida. (b) Una red de listas de decisiones (página674) tiene algunas rutas largas para algunos valores de entrada posibles, pero la mayoría de las rutas son cortas. (c) Una red de aprendizaje profundo tiene rutas de cálculo más largas, lo que permite que cada variable interactúe con todas las demás.

Las listas de decisión y los árboles de decisión, por otro lado, permiten largas rutas de cálculo que puede depender de muchas variables de entrada, pero solo para una fracción relativamente pequeña de las posibles vectores de entrada (Figura 21.1(b)). Si un árbol de decisión tiene largas rutas de cálculo para un fracción significativa de las entradas posibles, debe ser exponencialmente grande en el número de variables de entrada. La idea básica del aprendizaje profundo es entrenar circuitos de tal manera que el cálculo las rutas son largas, lo que permite que todas las variables de entrada interactúen de manera compleja (Figura

21.1(c)). Estos modelos de circuitos resultan lo suficientemente expresivos para captar la complejidad de los datos del mundo real para muchos tipos importantes de problemas de aprendizaje.

Sección 21.1describe redes simples de feedforward, sus componentes y los elementos esenciales de aprendizaje en dichas redes.Sección 21.2entra en más detalles sobre qué tan profundas son las redes se juntan ySección 21.3cubre una clase de redes llamadas neuronales convolucionales redes que son especialmente importantes en aplicaciones de visión.Secciones 21.4y21.5Vamos en más detalles sobre algoritmos para entrenar redes a partir de datos y métodos para mejorar generalización.Sección 21.6cubre redes con estructura recurrente, que están bien adecuado para datos secuenciales.Sección 21.7describe formas de usar el aprendizaje profundo para tareas que no son que el aprendizaje supervisado. Finalmente,Sección 21.8examina la gama de aplicaciones de la profunda aprendizaje.

21.1 Redes realimentadas simples

A**red de realimentación**, como sugiere el nombre, tiene conexiones solo en una dirección: que es decir, forma un gráfico acíclico dirigido con nodos de entrada y salida designados. cada nodo calcula una función de sus entradas y pasa el resultado a sus sucesores en la red.
La información fluye a través de la red desde los nodos de entrada a los nodos de salida, y allí no hay bucles. A**red recurrente**, en cambio, alimenta sus productos intermedios o finales volver a sus propias entradas. Esto significa que los valores de la señal dentro de la red forman un sistema dinámico que tiene estado interno o memoria. Consideraremos las redes recurrentes en Sección 21.6-.

Red de realimentación

---

red recurrente

Los circuitos booleanos, que implementan funciones booleanas, son un ejemplo de feedforward. redes En un circuito booleano, las entradas están limitadas a 0 y 1, y cada nodo implementa una función booleana simple de sus entradas, produciendo un 0 o un 1. En las redes neuronales, la entrada los valores son típicamente continuos, y los nodos toman entradas continuas y producen continuos salidas. Algunas de las entradas a los nodos son**parámetros**de la red; la red aprende por ajustando los valores de estos parámetros para que la red en su conjunto se ajuste al entrenamiento datos.

21.1.1 Redes como funciones complejas

Unidad

Cada nodo dentro de una red se llama**unidad**. Tradicionalmente, siguiendo el diseño propuesto por McCulloch y Pitts, una unidad calcula la suma ponderada de las entradas del predecesor nodos y luego aplica una función no lineal para producir su salida. Dejara$_j$denota la salida de unidadjy dejaw$_{i,j}$sea el peso adjunto al enlace de la unidadia la unidadj;entonces tenemos

$$a_j = \text{gramo}_j(\sum_i w_{i,j} a_i) \equiv \text{gramo}_j(en_j),$$

dóndegramo$_j$es un no lineal**función de activación**asociado con la unidadjyen$_j$es el ponderado suma de las entradas a la unidadj.

Función de activación

Como enSección 19.6.3(página679), estipulamos que cada unidad tiene una entrada extra de un unidad ficticia 0 que se fija en +1y un pesow$_{0,j}$para esa entrada. Esto permite el total entrada ponderadaen$_j$a la unidadjser distinto de cero incluso cuando las salidas de la capa anterior son todo cero Con esta convención, podemos escribir la ecuación anterior en forma vectorial:

(21.1)

$$a_j = \text{gramo}_j(w^T x)$$

dóndewes el vector de pesos que conduce a la unidadj (incluidow$_{0,j}$) yXes el vector de entradas a la unidadj (incluyendo el +1).

El hecho de que la función de activación sea no lineal es importante porque si no lo fuera, cualquier la composición de las unidades aún representaría una función lineal. La no linealidad es lo que permite redes suficientemente grandes de unidades para representar funciones arbitrarias. los**universal aproximación**El teorema establece que una red con solo dos capas de unidades computacionales, el primero no lineal y el segundo lineal, puede aproximar cualquier función continua a un grado arbitrario de precisión. La prueba funciona al mostrar que un exponencialmente grande La red puede representar exponencialmente muchos "baches" de diferentes alturas en diferentes ubicaciones en el espacio de entrada, aproximándose así a la función deseada. En otras palabras,

las redes suficientemente grandes pueden implementar una tabla de búsqueda para funciones continuas, al igual que

Los árboles de decisión suficientemente grandes implementan una tabla de búsqueda para funciones booleanas.

Se utiliza una variedad de diferentes funciones de activación. Los más comunes son los siguientes:

- La logística o**sigmoideo**función, que también se utiliza en la regresión logística (consulte la página685):

$$\sigma(x) = 1/(1 + e^{-x}).$$

Sigmoideo

- los**ReLU**función, cuyo nombre es una abreviatura de**unidad lineal rectificada**:

$$\text{ReLU}(x) = \max(0,x).$$

ReLU

- los**softplus**función, una versión fluida de la función ReLU:

$$\text{softplus}(x) = \log(1 + e^{x}).$$

---

softplus

  La derivada de la función softplus es la función sigmoidea.

- los**bronceado**función:

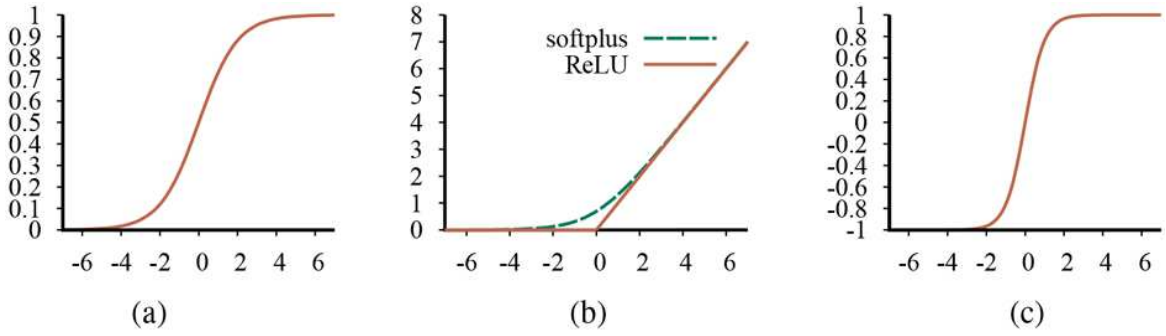
$$\tanh(x) = \frac{mi^{2x} - 1}{mi^{2x}+1}$$

Tanh

Tenga en cuenta que el rango de tanh es (−1, + 1).Tanh es una versión escalada y desplazada del sigmoide, comotanh(x) = 2σ(2x) − 1.

Estas funciones se muestran enFigura 21.2-. Tenga en cuenta que todos ellos son monótonamente no decreciente, lo que significa que sus derivadasgramo′son no negativos. tendremos más para decir acerca de la elección de la función de activación en secciones posteriores.

Figura 21.2



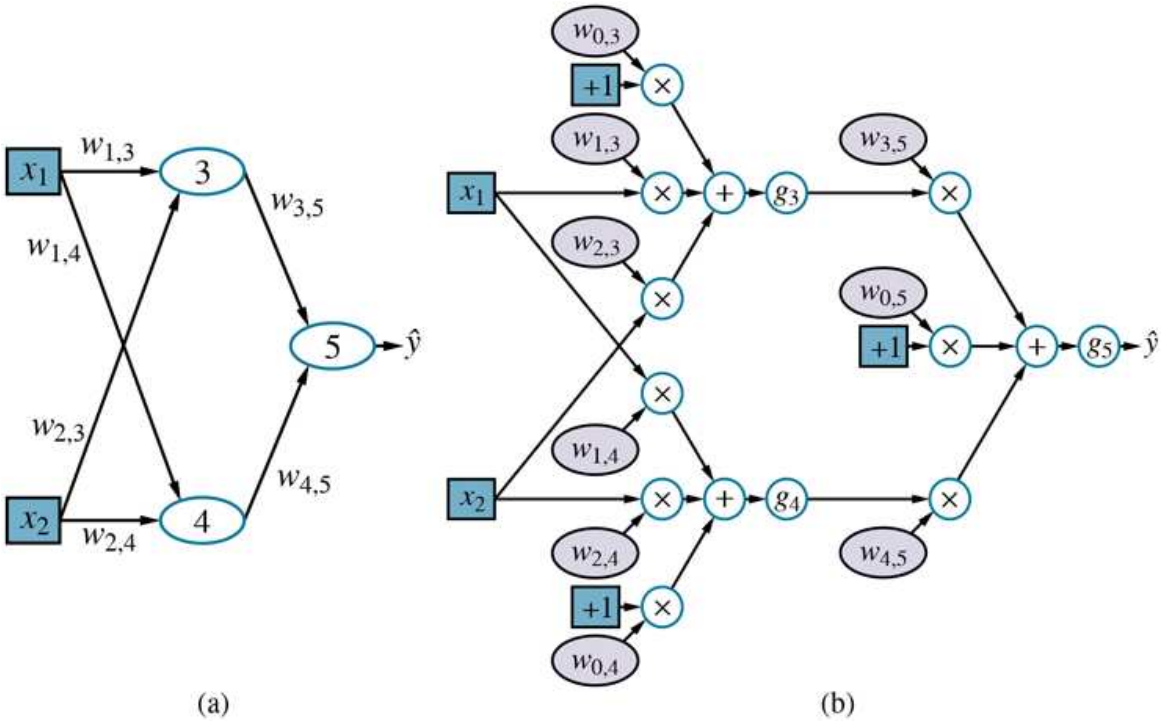
Funciones de activación comúnmente utilizadas en sistemas de aprendizaje profundo: (a) la función logística o sigmoidea; (b) la función ReLU y la función softplus; (c) la función tanh.

El acoplamiento de varias unidades en una red crea una función compleja que es un composición de las expresiones algebraicas representadas por las unidades individuales. Por ejemplo, la red que se muestra enFigura 21.3(a)-representa una funciónhw(X),parametrizado por pesosw,que mapea un vector de entrada de dos elementosXa un valor de salida escalar ŷ. El interno La estructura de la función refleja la estructura de la red. Por ejemplo, podemos escribir un expresión para la salida ŷ como sigue:

(21.2)

$$\begin{aligned}\hat{y} &= g_5(en_5) = gramo_5(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4)\\ &= gramo_5(w_{0,5} + w_{3,5}gramo_3(en_3) + w_{4,5}gramo_4(en_4))\\ &= gramo_5(w_{0,5} + w_{3,5}gramo_3(w_{0,3} + w_{1,3}X_1 + w_{2,3}X_2)\\ &\qquad + w_{4,5}gramo_4(w_{0,4} + w_{1,4}X_1 + w_{2,4}X_2)).\end{aligned}$$

Figura 21.3



(a) Una red neuronal con dos entradas, una capa oculta de dos unidades y una unidad de salida. No se muestran las entradas ficticias y sus pesos asociados. (b) La red en (a) desempaquetada en su gráfico de cálculo completo.

Así, tenemos la salida ŷ expresada como una función$h_w(X)$de las entradas y los pesos.

Figura 21.3(a)muestra la forma tradicional en que se podría representar una red en un libro sobre redes neuronales. redes Una forma más general de pensar en la red es como un**gráfico de cálculo**o **gráfico de flujo de datos**—esencialmente un circuito en el que cada nodo representa un elemental cálculo.Figura 21.3(b)muestra el gráfico de cálculo correspondiente a la red enFigura 21.3(a); el gráfico hace explícito cada elemento del cálculo general. También distingue entre las entradas (en azul) y los pesos (en malva claro): los pesos pueden ajustarse para que la salida ŷ coincida más con el valor realyen el entrenamiento datos. Cada peso es como una perilla de control de volumen que determina cuánto el siguiente nodo en el gráfico escucha de ese predecesor en particular en el gráfico.

Gráfico de cálculo

Gráfico de flujo de datos

Tal comoEcuación (21.1)describió el funcionamiento de una unidad en forma vectorial, podemos hacer algo similar para la red en su conjunto. Generalmente usaremos$W$para denotar un peso matriz; por esta red,$W^{(1)}$denota los pesos en la primera capa ($w_{1,3}$,$w_{1,4}$, etc) y$W^{(2)}$ denota los pesos en la segunda capa ($w_{3,5}$etc.). Finalmente, deja$\text{gramo}^{(1)}$y$\text{gramo}^{(2)}$denota el funciones de activación en la primera y segunda capas. Entonces toda la red se puede escribir como sigue:

(21.3)

$$h_w(x) = \text{gramo}^{(2)}(W^{(2)}\text{gramo}^{(1)}(W^{(1)}X)).$$

Me gustaEcuación (21.2), esta expresión corresponde a un gráfico de cálculo, aunque mucho más simple que el gráfico enFigura 21.3(b): aquí, el gráfico es simplemente una cadena con peso matrices que alimentan cada capa.

El gráfico de cálculo enFigura 21.3(b)es relativamente pequeño y poco profundo, pero la misma idea se aplica a todas las formas de aprendizaje profundo: construimos gráficos de cálculo y ajustamos su pesos para ajustar los datos. El gráfico enFigura 21.3(b)es también**completamente conectado**, significa que cada nodo en cada capa está conectado a cada nodo en la siguiente capa. Esto es en cierto sentido por defecto, pero lo veremos enSección 21.3que elegir la conectividad de la red es también es importante para lograr un aprendizaje efectivo.

Totalmente conectado

21.1.2 Gradientes y aprendizaje

EnSección 19.6, introdujimos un enfoque para el aprendizaje supervisado basado en**degradado descendencia**: calcule el gradiente de la función de pérdida con respecto a los pesos y ajuste los pesos a lo largo de la dirección del gradiente para reducir la pérdida. (Si aún no has leído

Sección 19.6-, le recomendamos encarecidamente que lo haga antes de continuar). Podemos aplicar exactamente el mismo enfoque para aprender los pesos en gráficos de cálculo. para los pesos conduciendo a unidades en el**capa de salida**—los que producen la salida de la red, los el cálculo del gradiente es esencialmente idéntico al proceso enSección 19.6-. para pesas conduciendo a unidades en el**capas ocultas**, que no están conectados directamente a las salidas, el El proceso es sólo un poco más complicado.

Capa de salida

capa oculta

Por ahora, usaremos la función de pérdida al cuadrado,L_2, y calcularemos el gradiente para el red enFigura 21.3-con respecto a un solo ejemplo de entrenamiento (x, y). (para múltiples ejemplos, el gradiente es simplemente la suma de los gradientes de los ejemplos individuales). la red genera una predicción $\hat{y} = h_w(X)$y el verdadero valor esy,entonces tenemos

$$\text{Pérdida}\,(h_w) = L_2(y, h_w(x)) = \| y - h_w(x) \|_2 = (y - \hat{y})_2.$$

Para calcular el gradiente de la pérdida con respecto a los pesos, necesitamos las mismas herramientas de cálculo que usamos encapitulo 19-—principalmente el**cadena de reglas**, $\partial g(f(x))/\partial x = g'(f(x))\partial f(x)/\partial x$.Empezaremos con el caso fácil: un peso como$w_{3,5}$eso es conectado a la unidad de salida. Operamos directamente en las expresiones que definen la red de Ecuación (21.2)-:

(21.4)

$$\frac{\partial}{\partial w_{3,5}}\text{Pérdida}(h_w) = \frac{\partial}{\partial w_{3,5}}(y-\hat{y})^2 = -2(y-\hat{y})\frac{\partial \hat{y}}{\partial w_{3,5}}$$

$$= -2(y-\hat{y})\frac{\partial}{\partial w_{3,5}}\text{gramo}_5(\text{en}_5) = -2(y-\hat{y})g'_5(\text{en}_5)\frac{\partial}{\partial w_{3,5}}\text{en}_5$$

$$= -2(y-\hat{y})g'_5(\text{en}_5)\frac{\partial}{\partial w_{3,5}}(w_{0,5}+w_{3,5}a_3+w_{4,5}a_4)$$

$$= -2(y-\hat{y})g'_5(\text{en}_5)a_3.$$

La simplificación en la última línea sigue porque$w_{0,5}$y$w_{4,5}a_4$no dependas de$w_{3,5}$, ni el coeficiente de$w_{3,5}$,a_3.

El caso un poco más difícil implica un peso como$w_{1,3}$que no está conectado directamente a la unidad de salida. Aquí, tenemos que aplicar la regla de la cadena una vez más. Los primeros pasos son idénticos, por lo que los omitimos:

(21.5)

$$\frac{\partial}{\partial w_{1,3}}\text{Pérdida}(h_w) = -2(y-\hat{y})g'_5(\text{en}_5)\frac{\partial}{\partial w_{1,3}}(w_{0,5}+w_{3,5}a_3+w_{4,5}a_4)$$

$$= -2(y-\hat{y})g'_5(\text{en}_5)w_{3,5}\frac{\partial}{\partial w_{1,3}}a_3$$

$$= -2(y-\hat{y})g'_5(\text{en}_5)w_{3,5}\frac{\partial}{\partial w_{1,3}}\text{gramo}_3(\text{en}_3)$$

$$= -2(y-\hat{y})g'_5(\text{en}_5)w_{3,5}\text{gramo}'_3(\text{en}_3)\frac{\partial}{\partial w_{1,3}}\text{en}_3$$

$$= -2(y-\hat{y})g'_5(\text{en}_5)w_{3,5}\text{gramo}'_3(\text{en}_3)\frac{\partial}{\partial w_{1,3}}(w_{0,3}+w_{1,3}X_1+w_{2,3}X_2)$$

$$= -2(y-\hat{y})g'_5(\text{en}_5)w_{3,5}\text{gramo}'_3(\text{en}_3)X_1.$$

Entonces, tenemos expresiones bastante simples para el gradiente de pérdida con respecto a los pesos $w_{3,5}$y$w_{1,3}$.

Si definimos $\Delta_5 = 2(\hat{y}-y)g'_5(\text{en}_5)$como una especie de "error percibido" en el punto donde la unidad 5 recibe su entrada, entonces el gradiente con respecto a$w_{3,5}$es solo $\Delta_5 a_3$. esto hace perfecto sentido: si $\Delta_5$es positivo, eso significa que ŷ es demasiado grande (recuerde quegramo′siempre es no negativo); si$a_3$es también positivo, luego creciente$w_{3,5}$sólo empeorará las cosas, mientras que si$a_3$es negativo, luego aumentando$w_{3,5}$reducirá el error. La magnitud de$a_3$también importa: si$a_3$es pequeño para este ejemplo de entrenamiento, entonces$w_{3,5}$no desempeñó un papel importante en la producción del error y no hay que cambiar mucho.

Si también definimos $\Delta_3 = \Delta_5 w_{3,5} g'(in_3)$, entonces el gradiente para $w_{1,3}$ se convierte simplemente en $\Delta_3 x_1$. Por lo tanto, la error percibido en la entrada a la unidad 3 es el error percibido en la entrada a la unidad 5, multiplicado por información a lo largo del camino de 5 a 3. Este fenómeno es completamente general, y da lugar al término **retropropagación** por la forma en que el error en la salida es pasado de vuelta a través de la red.

---

retropropagación

Otra característica importante de estas expresiones de gradiente es que tienen como factores los derivados locales $g'_j(in_j)$. Como se señaló anteriormente, estas derivadas siempre son no negativas, pero pueden estar muy cerca de cero (en el caso de las funciones sigmoide, softplus y tanh) o exactamente cero (en el caso de ReLU), si las entradas del ejemplo de entrenamiento en cuestión pasar a poner unidad j en la región de operación plana. Si la derivada g'_j es pequeño o cero, que significa que cambiar los pesos que conducen a la unidad j tendrá un efecto insignificante en su producción. Como resultado, las redes profundas con muchas capas pueden sufrir una **gradiente de desaparición**—las señales de error se extinguen por completo a medida que se propagan de regreso a través del la red. Sección 21.3.3 proporciona una solución a este problema.

Gradiente de fuga

Hemos demostrado que los gradientes en nuestra pequeña red de ejemplo son expresiones simples que pueden calcularse pasando información de vuelta a través de la red desde las unidades de salida. Eso resulta que esta propiedad se mantiene de manera más general. De hecho, como mostramos en Sección 21.4.1, la cálculos de gradiente para *ningún* gráfico de cálculo feedforward tienen la misma estructura que el gráfico de cálculo subyacente. Esta propiedad se sigue directamente de las reglas de diferenciación.

Hemos mostrado los detalles sangrientos de un cálculo de gradiente, pero no se preocupe: no hay necesidad de rehacer las derivaciones en Ecuaciones (21.4) y (21.5) para cada nueva estructura de red! Todos

tales gradientes se pueden calcular por el método de**diferenciación automática**, que se aplica las reglas del cálculo de forma sistemática para calcular gradientes para cualquier programa numérico.[1]En De hecho, el método de propagación hacia atrás en el aprendizaje profundo es simplemente una aplicación de **modo inverso**diferenciación, que aplica la regla de la cadena "de afuera hacia adentro" y obtiene las ventajas de eficiencia de la programación dinámica cuando la red en cuestión tiene muchos insumos y relativamente pocos productos.

[1]Los métodos de diferenciación automática se desarrollaron originalmente en las décadas de 1960 y 1970 para optimizar los parámetros de los sistemas. definida por programas Fortran grandes y complejos.

Diferenciación automática

---

modo inverso

Todos los paquetes principales para el aprendizaje profundo brindan una diferenciación automática, de modo que los usuarios puede experimentar libremente con diferentes estructuras de red, funciones de activación, funciones de pérdida, y formas de composición sin tener que hacer mucho cálculo para derivar un nuevo aprendizaje algoritmo para cada experimento. Esto ha fomentado un enfoque llamado**de extremo a extremo aprendizaje**, en el que un sistema computacional complejo para una tarea como la traducción automática puede estar compuesto por varios subsistemas entrenables; todo el sistema es entonces entrenado en un Moda de extremo a extremo de pares de entrada/salida. Con este enfoque, el diseñador necesita tener sólo una vaga idea sobre cómo debe estructurarse el sistema general; no hay necesidad de saber de antemano exactamente qué debe hacer cada subsistema o cómo etiquetar sus entradas y salidas.

Aprendizaje de extremo a extremo