Aprendizaje automático 19/02/2024

El aprendizaje profundo pertenece históricamente al campo del aprendizaje automático estadístico, ya que se refiere fundamentalmente a métodos capaces de aprender representaciones a partir de datos. Las técnicas implicadas proceden originalmente de las redes neuronales artificiales, y el calificativo "profundo" subraya que los modelos son largas composiciones de mapeos, ahora conocidas para lograr un mayor rendimiento.

La modularidad, versatilidad y escalabilidad de los modelos profundos han dado lugar a una plétora de métodos matemáticos específicos y herramientas de desarrollo de software, estableciendo el aprendizaje profundo como un campo técnico distinto.

1.1 Aprender de los datos

El caso de uso más sencillo para un modelo entrenado a partir de datos es cuando se tiene acceso a una señal x, por ejemplo, la imagen de una matrícula, a partir de la cual se quiere predecir una cantidad y, como la cadena de caracteres escritos en la matrícula.

En muchas situaciones del mundo real, en las que x es una señal de alta dimensionalidad captada en un entorno no controlado, es demasiado complicado desarrollar una receta analítica o fórmula matemática que relacione x e y de manera precisa.

Lo que sí se puede hacer es recopilar un gran conjunto de entrenamiento 𝒟 de pares (xn,yn), e idear un modelo paramétrico f. Se trata de un fragmento de código informático que incorpora parámetros entrenables w que modulan su comportamiento y que, con los valores adecuados w∗ sea un buen predictor. "Bueno" significa aquí que si se da una x a este trozo de código, el valor yˆ= f(x;w∗) que calcula es una buena estimación del valor y que habría estado asociado con x en el conjunto de entrenamiento si hubiera estado allí.

Esta noción de bondad suele formalizarse con una pérdida ℒ(w) que es pequeña cuando f(.;w) es buena en 𝒟. Entonces, el entrenamiento del modelo consiste en calcular un valor w∗ que minimice ℒ(w∗).

La mayor parte del contenido de este libro trata sobre la definición de f, que, en escenarios realistas, es una combinación compleja de submódulos predefinidos.

Los parámetros entrenables que componen w suelen denominarse pesos, por analogía con los pesos sinápticos de las redes neuronales biológicas. Además de estos parámetros, los modelos suelen depender de los meta-parámetros, que se establecen de acuerdo con el conocimiento previo del dominio previo del dominio, las mejores prácticas o las limitaciones de recursos. También pueden optimizarsepero con técnicas distintas de las utilizadas para optimizar w.

1.2 Regresión de la función de base

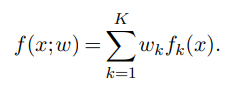
Podemos ilustrar el entrenamiento de un modelo en un caso sencillo en el que xn e yn son dos números reales, la pérdida es el error cuadrático medio:

Imagen que contiene objeto, reloj

Descripción generada automáticamente

*N es el número total de muestras.*

f(-;w) es una combinación lineal de una base predefinida de funciones f1,...,fK, con w = (w1,...,wK):



Como f(xn;w) es lineal con respecto a wks y ℒ(w) es cuadrática con respecto a f(xn;w), la pérdida ℒ(w) es cuadrática con respecto a las wks, y encontrar w∗ que la minimice se reduce a resolver un sistema lineal.

1.3 Under y overfitting

Un elemento clave es la interacción entre la capacidad del modelo, es decir, su flexibilidad y habilidad para ajustarse a datos diversos, y la cantidad y calidad de los datos de entrenamiento. Cuando la capacidad es insuficiente, el modelo no puede ajustarse a los datos, lo que se traduce en un alto error durante el entrenamiento. Esto se denomina underfitting.

Por el contrario, cuando la cantidad de datos es insuficiente, el modelo suele aprender características específicas de los ejemplos de entrenamiento, lo que se traduce en un excelente rendimiento durante el entrenamiento, a costa de un peor ajuste a la estructura global de los datos y un rendimiento deficiente a las nuevas entradas. Esto se denomina overfitting.

Por lo tanto, una gran parte del arte del aprendizaje automático consiste en diseñar modelos que no sean demasiado flexibles pero capaces de ajustarse a los datos. Esto se consigue creando el sesgo inductivo correcto en un modelo, lo que significa que su estructura se corresponde a la estructura subyacente de los datos en cuestión.

Aunque esta perspectiva clásica es relevante para los modelos profundos de tamaño razonable, las cosas se vuelven confusas con aquellos que son muy grandes y tienen un gran número de parámetros entrenables y una capacidad extrema, pero aun así rinden bien en la predicción.

1.4 Categorías de modelos

Podemos organizar el uso de modelos de aprendizaje en tres grandes categorías:

- Regresión: consiste en predecir un vector de valor continuo y ∈ R^K, por ejemplo, la posición geométrica de un objeto, dada una señal de entrada X. Se trata de una generalización multidimensional de la configuración que vimos en § 1.2. El conjunto de entrenamiento se compone de pares de una señal de entrada y un valor de verdad fundamental.

- Clasificación: El objetivo es predecir un valor de un conjunto finito {1,...,C}, por ejemplo, la etiqueta Y de una imagen X . Al igual que en la regresión, el conjunto de entrenamiento se compone de pares de señales de entrada y la cantidad fundamental de verdad, en este caso una etiqueta de ese conjunto.

- Modelado de densidad: tiene como objetivo modelar la función de densidad de probabilidad de los datos µX, por ejemplo, imágenes. En ese caso, el conjunto de entrenamiento se compone de valores xn sin cantidades asociadas para predecir, y el modelo entrenado debería permitir la evaluación de la función de densidad de probabilidad, o el muestreo de la distribución, o ambos.

Tanto la regresión como la clasificación se refieren generalmente al aprendizaje supervisado, ya que el valor a predecir, que se requiere como objetivo durante el entrenamiento, debe ser proporcionado, por ejemplo, por expertos humanos.

Por el contrario, la modelización de densidad generalmente se considera aprendizaje no supervisado, ya que basta con tomar datos existentes sin necesidad de producir una verdad de referencia asociada.

Estas tres categorías no son independientes; por ejemplo, la clasificación puede considerarse como una regresión de puntuaciones de clase, o la modelización de densidad de secuencia discreta como clasificación iterada.

Además, no cubren todos los casos. Uno puede querer predecir cantidades compuestas, o múltiples clases, o modelar una densidad condicional en función de una señal.

Regresión Lineal 20/02/2024

Los problemas de regresión surgen siempre que queremos predecir un valor numérico. Ejemplos comunes incluyen la predicción de precios (de viviendas, acciones, etc.), la predicción de la duración de la estancia (para los pacientes en el hospital), la previsión de la demanda (para las ventas al por menor), entre muchos otros. No todos los problemas de predicción son de regresión clásica. Más adelante introduciremos los problemas de clasificación, en los que el objetivo es predecir la pertenencia a un conjunto de categorías.

Como ejemplo práctico, supongamos que deseamos estimar los precios de las casas (en dólares) en función de su superficie (en pies cuadrados) y su antigüedad (en años). Para desarrollar un modelo de predicción del precio de la vivienda, necesitamos disponer de datos que incluyan el precio de venta, la superficie y la antigüedad de cada casa. En la terminología del aprendizaje automático, el conjunto de datos se denomina conjunto de datos de entrenamiento o conjunto de entrenamiento, y cada fila (que contiene los datos correspondientes a una venta) se denomina ejemplo (o punto de datos, instancia, muestra). Lo que intentamos predecir (el precio) se denomina etiqueta (u objetivo). Las variables (edad y zona) en las que se basan las predicciones se denominan características (o covariables).

3.1.1. Conceptos básicos

La regresión lineal es a la vez la más sencilla y la más popular de las herramientas estándar para abordar los problemas de regresión. La regresión lineal, que se remonta a los albores del siglo XIX (Gauss, 1809, Legendre, 1805), parte de unos cuantos supuestos sencillos. En primer lugar, suponemos que la relación entre las características x y el objetivo y es aproximadamente lineal, es decir, que la media condicional E[Y|X=x] puede expresarse como una suma ponderada de las características x. Esta configuración permite que el valor objetivo pueda desviarse de su valor esperado debido al ruido de observación. A continuación, podemos imponer el supuesto de que dicho ruido se comporta bien, siguiendo una distribución gaussiana. Normalmente, utilizaremos n para indicar el número de ejemplos del conjunto de datos. Utilizamos superíndices para enumerar las muestras y los objetivos, y subíndices para indexar las coordenadas. Más concretamente, x^(i) indica la muestra i^(th) y xj^(i) indica la muestra j^(th).

3.1.1.1. Modelo

En el núcleo de toda solución hay un modelo que describe cómo pueden transformarse las características en una estimación del objetivo. El supuesto de linealidad significa que el valor esperado del objetivo (precio) puede expresarse como una suma ponderada de las características (superficie y edad):

Precio = warea \* área + wage \* age + b

(3.1.1)

Aquí warea y wage se denominan pesos (o ponderaciones), y b se denomina sesgo (o desplazamiento o intercepto). Las ponderaciones determinan la influencia de cada característica en nuestra predicción. El sesgo determina el valor de la estimación cuando todas las características son cero. Aunque nunca veamos viviendas de nueva construcción con una superficie exactamente igual a cero, necesitamos el sesgo porque nos permite expresar todas las funciones lineales de nuestras características (en lugar de restringirnos a las líneas que pasan por el origen). Estrictamente hablando, (3.1.1) es una transformación afín de las características de entrada, que se caracteriza por una transformación lineal de las características mediante una suma ponderada, combinada con una traslación mediante el sesgo añadido. Dado un conjunto de datos, nuestro objetivo es elegir las ponderaciones w y el sesgo b que, en promedio, hagan que las predicciones de nuestro modelo se ajusten lo más posible a los precios reales observados en los datos.

En las disciplinas en las que es habitual centrarse en conjuntos de datos con pocas características, es habitual expresar los modelos de forma explícita y detallada, como en (3.1.1). En el aprendizaje automático, solemos trabajar con conjuntos de datos de alta dimensión, en los que es más conveniente emplear una notación compacta de álgebra lineal. Cuando nuestras entradas constan de d características, podemos asignar a cada una un índice (entre 1 y d) y expresar nuestra predicción^y (en general, el símbolo "sombrero" indica una estimación) como:

^y = w1x1 + … + wd+xd +b

(3.1.2)

Recogiendo todas las características en un vector x E R^d y todos los pesos en un vector w E R^d, podemos expresar nuestro modelo de forma compacta mediante el producto entre w y x :

^y = w^T \* x + b

(3.1.3)

En (3.1.3), el vector x corresponde a las características de un único ejemplo. A menudo nos resultará conveniente referirnos a las características de todo nuestro conjunto de datos de n ejemplos mediante la matriz de diseño X E R^(nxd). En este caso, X contiene una fila para cada ejemplo y una columna para cada característica. Para una colección de características X, las predicciones ^y E R^n pueden expresarse mediante el producto matriz-vector:

^y = X \* w + b

(3.1.4)

Dadas las características de un conjunto de datos de entrenamiento X y las correspondientes etiquetas (conocidas) y, el objetivo de la regresión lineal es encontrar el vector de pesos w y el término de sesgo b de forma que, dadas las características de un nuevo ejemplo de datos muestreado a partir de la misma distribución que X la etiqueta del nuevo ejemplo se prediga (en principio) con el menor error.

Aunque creamos que el mejor modelo para predecir y a partir de x es lineal, no podemos esperar encontrar un conjunto de datos reales de n ejemplos en los que y^(i) sea exactamente igual a w^Tx^(i)+ b para todos1 <= i <= n. Por ejemplo, independientemente de los instrumentos que utilicemos para observar las características X y las etiquetas y, puede haber un pequeño error de medición. Por lo tanto, aunque estemos seguros de que la relación subyacente es lineal, incorporaremos un término de ruido para tener en cuenta dichos errores.

Antes de buscar los mejores parámetros (o parámetros del modelo) w y b necesitaremos dos cosas más: (i) una medida de la calidad de un modelo determinado; y (ii) un procedimiento para actualizar el modelo con el fin de mejorar su calidad.

3.1.1.2. Función de pérdida

Naturalmente, ajustar nuestro modelo a los datos requiere que nos pongamos de acuerdo en alguna medida de adecuación/ajuste (o, equivalentemente, de inadecuación). Las funciones de pérdida cuantifican la distancia entre los valores reales y predichos del objetivo. La pérdida suele ser un número no negativo en el que los valores más pequeños son mejores y las predicciones perfectas incurren en una pérdida de 0. Para los problemas de regresión, la función de pérdida más común es el error al cuadrático. Cuando nuestra predicción para un ejemplo i es ^y^(i) y la etiqueta verdadera correspondiente es y^(i) el error al cuadrado viene dado por:



(3.1.5)

La constante 1/2 no supone ninguna diferencia real, pero resulta conveniente desde el punto de vista de la notación, ya que se anula cuando tomamos la derivada de la pérdida. Dado que el conjunto de datos de entrenamiento nos viene dado, y por tanto está fuera de nuestro control, el error empírico es sólo una función de los parámetros del modelo. En la Fig. 3.1.1, visualizamos el ajuste de un modelo de regresión lineal en un problema con entradas unidimensionales.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Obsérvese que las grandes diferencias entre las estimaciones ^y^(i) creadas por el modelo y los objetivos/valores reales y(i) dan lugar a contribuciones aún mayores a la pérdida, debido a su forma cuadrática. Esto dignifica que el modelo es penalizado de manera más significativa cuando comete errores grandes (esta cuadraticidad puede ser un arma de doble filo; aunque anima al modelo a evitar grandes errores, también puede dar lugar a una sensibilidad excesiva a los datos anómalos). Para medir la calidad de un modelo en todo el conjunto de datos de n ejemplos, basta con promediar (o, lo que es lo mismo, sumar) las pérdidas en el conjunto de entrenamiento:



Al entrenar el modelo, buscamos parámetros (w\*, b\* ) que minimicen la pérdida total en todos los ejemplos de entrenamiento:

Texto

Descripción generada automáticamente­

3.1.1.3. Solución analítica

A diferencia de la mayoría de los modelos que trataremos, la regresión lineal nos presenta un problema de optimización sorprendentemente sencillo. En particular, podemos encontrar los parámetros óptimos (evaluados en los datos de entrenamiento) analíticamente aplicando una sencilla fórmula como la siguiente. En primer lugar, podemos subsumir el sesgo b en el parámetro w añadiendo una columna a la matriz de diseño formada por todos los 1s. Entonces nuestro problema de predicción es minimizar ||y - Xw||^2. Mientras la matriz de diseño X tenga rango completo (ninguna característica depende linealmente de las demás), sólo habrá un punto crítico en la superficie de pérdida, que corresponde al mínimo de la pérdida en todo el dominio. Si se toma la derivada de la pérdida con respecto a w y haciéndola igual a cero se obtiene:



Resolviendo nos proporciona la solución óptima para el problema de optimización. Nótese que esta solución w

w\* = (X^TX)^(-1)X^Ty

sólo será única cuando la matriz sea invertible, es decir, cuando las columnas de la matriz de diseño sean linealmente independientes X^TX.

Aunque problemas sencillos como la regresión lineal pueden admitir soluciones analíticas, no hay que acostumbrarse a tan buena fortuna. Aunque las soluciones analíticas permiten un bonito análisis matemático, el requisito de una solución analítica es tan restrictivo que excluiría casi todos los aspectos apasionantes del aprendizaje profundo.

3.1.1.4. Descenso Gradiente Estocástico por Mini-Lotes

Afortunadamente, incluso en los casos en los que no podemos resolver los modelos analíticamente, a menudo podemos entrenar modelos de forma eficaz en la práctica. Es más, para muchas tareas, esos modelos difíciles de optimizar resultan ser mucho mejores que los modelos más simples, por lo que averiguar cómo entrenarlos acaba mereciendo la pena.

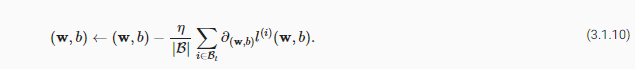
La técnica clave para optimizar casi todos los modelos de aprendizaje profundo, y a la que recurriremos a lo largo de este libro, consiste en reducir iterativamente el error mediante la actualización de los parámetros en la dirección que disminuye incrementalmente la función de pérdida. Este algoritmo se denomina descenso gradiente.

La aplicación más básica del descenso de gradiente consiste en calcular la derivada de la función de pérdida, que es una media de las pérdidas calculadas en cada ejemplo individual del conjunto de datos. En la práctica, esto puede ser extremadamente lento: hay que pasar por todo el conjunto de datos antes de hacer una sola actualización, aunque los pasos de actualización puedan ser muy potentes (Liu y Nocedal, 1989). Peor aún, si hay mucha redundancia en los datos de entrenamiento, el beneficio de una actualización completa es limitado.

El otro extremo es considerar un único ejemplo a la vez y dar pasos de actualización basados en una observación a la vez. El algoritmo resultante, el descenso de gradiente estocástico (SGD), puede ser una estrategia eficaz (Bottou, 2010), incluso para grandes conjuntos de datos. Por desgracia, el SGD presenta inconvenientes, tanto computacionales como estadísticos. Un problema surge del hecho de que los procesadores son mucho más rápidos multiplicando y sumando números que trasladando datos de la memoria principal a la caché del procesador. Es hasta un orden de magnitud más eficiente realizar una multiplicación matriz-vector que un número correspondiente de operaciones vector-vector. Esto significa que puede llevar mucho más tiempo procesar una muestra a la vez en comparación con un lote completo. Un segundo problema es que algunas de las capas, como la normalización por lotes (que se describirá en la sección 8.5), sólo funcionan bien cuando tenemos acceso a más de una observación a la vez.

La solución a ambos problemas es elegir una estrategia intermedia: en lugar de tomar un lote completo o una sola muestra a la vez, tomamos un minilote de observaciones (Li et al., 2014). La elección concreta del tamaño de dicho minilote depende de muchos factores, como la cantidad de memoria, el número de aceleradores, la elección de capas y el tamaño total del conjunto de datos. A pesar de todo, un número entre 32 y 256, preferiblemente múltiplo de una gran potencia de 2 , es un buen comienzo. Esto nos lleva al descenso de gradiente estocástico por lotes.

En su forma más básica, en cada iteración (t), primero seleccionamos aleatoriamente un minilote (minibatch) que consiste en un número fijo de ejemplos de entrenamiento. A continuación, calculamos la derivada (gradiente) de la pérdida media en el mini lote con respecto a los parámetros del modelo. Por último, multiplicamos el gradiente por un pequeño valor positivo predeterminado (n), denominado tasa de aprendizaje, y restamos el término resultante de los valores actuales de los parámetros. Podemos expresar la actualización del siguiente modo: tBt|B|n



En resumen, el descenso de gradiente estocástico por mini-lotes procede de la siguiente manera: (i) inicializar los valores de los parámetros del modelo, típicamente de forma aleatoria; (ii) muestrear de manera iterativa minibatches aleatorios de los datos, actualizando los parámetros en la dirección del gradiente negativo. Para pérdidas cuadráticas y transformaciones afines, esto tiene una expansión en forma cerrada:

Word

Descripción generada automáticamente con confianza media

Dado que elegimos un minilote, tenemos que normalizar por su tamaño. Con frecuencia, el tamaño del minilote y la tasa de aprendizaje son definidos por el usuario. Estos parámetros ajustables que no se actualizan en el bucle de entrenamiento se denominan hiperparámetros. Pueden ajustarse automáticamente mediante una serie de técnicas, como la optimización bayesiana (Frazier, 2018). Al final, la calidad de la solución suele evaluarse en un conjunto de datos de validación independiente (o conjunto de validación). B|B|

Después de entrenar durante un número predeterminado de iteraciones (o hasta que se cumpla algún otro criterio de detención), registramos los parámetros del modelo estimado, denotados ^w y ^b. Tenga en cuenta que incluso si nuestra función es realmente lineal y sin ruido, estos parámetros no serán los minimizadores exactos de la pérdida, ni siquiera deterministas. Aunque el algoritmo converge lentamente hacia los minimizadores, normalmente no los encontrará exactamente en un número finito de pasos. Además, las minibatches utilizadas para actualizar los parámetros se eligen al azar. Esto rompe el determinismo. B

La regresión lineal resulta ser un problema de aprendizaje con un mínimo global (siempre que X^TX sea de rango completo o, de forma equivalente, siempre que sea invertible). Sin embargo, las superficies de pérdida de las redes profundas contienen muchos puntos de inflexión y mínimos. Afortunadamente, normalmente no nos importa encontrar un conjunto exacto de parámetros, sino simplemente cualquier conjunto de parámetros que conduzca a predicciones precisas (y, por tanto, a una pérdida baja). En la práctica, los profesionales del aprendizaje profundo rara vez tiene dificultades para encontrar parámetros que minimicen la pérdida en los conjuntos de entrenamiento (Frankle y Carbin, 2018, Izmailov et al., 2018). La tarea más formidable es encontrar parámetros que conduzcan a predicciones precisas sobre datos previamente no vistos, un desafío llamado generalización.

3.1.1.5. Predicciones

Dado el modelo, ahora podemos hacer predicciones para un nuevo ejemplo, por ejemplo, predecir el precio de venta de una casa no vista anteriormente dada su área y edad. Los profesionales del aprendizaje profundo han empezado a llamar a esta fase de predicción “inferencia”, pero se trata de un término un poco inapropiado, ya que la inferencia se refiere en general a cualquier conclusión alcanzada sobre la base de la evidencia, incluyendo tanto los valores de los parámetros como la etiqueta probable para un ejemplo no visto. En todo caso, en la literatura estadística inferencia denota más a menudo inferencia de parámetros y esta sobrecarga de terminología crea una confusión innecesaria cuando los profesionales del aprendizaje profundo hablan con estadísticos. En lo sucesivo nos ceñiremos a la predicción siempre que sea posible. ^w^Tx+^bx1x2

3.1.2. Vectorización para velocidad 21/02/2024

Cuando entrenamos nuestros modelos, normalmente queremos procesar minilotes enteros de ejemplos simultáneamente. Para hacerlo de forma eficiente, es necesario vectorizar los cálculos y aprovechar las bibliotecas rápidas de álgebra lineal en lugar de escribir costosos bucles for en Python.

Para ver por qué esto es tan importante, consideremos dos métodos para sumar vectores. Para empezar, instanciamos dos vectores de 10.000 dimensiones compuestos por todo 1s. En el primer método, recorremos los vectores con un bucle for de Python. En el segundo, nos basamos en una única llamada a +.

n = 10000

a = torch.ones(n)

b = torch.ones(n)

Ahora podemos comparar las cargas de trabajo. En primer lugar, las añadimos, coordenada a coordenada, mediante un bucle for.

c = torch.zeros(n)

t = time.time()

**for** i **in** range(n):

c[i] = a[i] + b[i]

f'*{*time.time() - t*:*.5f*}* sec'

'0.17802 sec'

Alternativamente, confiamos en el operador + recargado para calcular la suma elemental.

t = time.time()

d = a + b

f'*{*time.time() - t*:*.5f*}* sec'

'0.00036 sec'

El segundo método es mucho más rápido que el primero. Vectorizar el código a menudo produce aumentos de velocidad del orden de magnitud. Además, trasladamos una mayor parte de las matemáticas a la biblioteca, por lo que no tenemos que escribir tantos cálculos nosotros mismos, lo que reduce la posibilidad de errores y aumenta la portabilidad del código.

3.1.3. La distribución normal y las pérdidas al cuadrado

Hasta ahora hemos dado una motivación bastante funcional del objetivo de pérdida al cuadrado: los parámetros óptimos devuelven la expectativa condicional E[Y/X] siempre que el patrón subyacente sea realmente lineal, y la pérdida asigna grandes penalizaciones a los valores atípicos. También podemos proporcionar una motivación más formal para el objetivo de pérdida al cuadrado haciendo suposiciones probabilísticas sobre la distribución del ruido.

La regresión lineal se inventó a finales del siglo XIX. Aunque durante mucho tiempo se ha debatido si fue Gauss o Legendre el primero en concebir la idea, fue Gauss quien también descubrió la distribución normal (también llamada gaussiana). Resulta que la distribución normal y la regresión lineal con pérdida al cuadrado comparten una conexión más profunda que un parentesco común.

Para empezar, recordemos que una distribución normal con media μ y varianza σ2 (desviación típica σ) viene dada por:



A continuación, definimos una función para calcular la distribución normal:

**def** normal(x, mu, sigma):

p = 1 / math.sqrt(2 \* math.pi \* sigma\*\*2)

**return** p \* np.exp(-0.5 \* (x - mu)\*\*2 / sigma\*\*2)

Ahora podemos visualizar las distribuciones normales.

*# Use NumPy again for visualization*

x = np.arange(-7, 7, 0.01)

*# Mean and standard deviation pairs*

params = [(0, 1), (0, 2), (3, 1)]

d2l.plot(x, [normal(x, mu, sigma) **for** mu, sigma **in** params], xlabel='x',

ylabel='p(x)', figsize=(4.5, 2.5),

legend=[f'mean *{*mu*}*, std *{*sigma*}*' **for** mu, sigma **in** params])

Gráfico, Gráfico de líneas

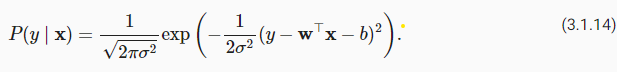
Descripción generada automáticamente

Obsérvese que el cambio de la media corresponde a un desplazamiento a lo largo del eje x, y que el aumento de la varianza extiende la distribución, reduciendo su pico.

Una forma de motivar la regresión lineal con pérdida al cuadrado es suponer que las observaciones proceden de mediciones ruidosas, en las que el ruido ‘e’ sigue la distribución normal N(0, σ2):



Por lo tanto, ahora podemos escribir la probabilidad de ver un determinado y para un determinado x mediante:



Como tal, la probabilidad se factoriza. Según el principio de máxima verosimilitud, los mejores valores de los parámetros w y b son los que maximizan la verosimilitud de todo el conjunto de datos:

La igualdad se deduce porque todos los pares (x^i, y^i) se han dibujado independientemente unos de otros. Los estimadores elegidos según el principio de máxima verosimilitud se denominan estimadores de máxima verosimilitud. Aunque maximizar el producto de muchas funciones exponenciales puede parecer difícil, podemos simplificar las cosas significativamente, sin cambiar el objetivo, maximizando el logaritmo de la probabilidad. Por razones históricas, las optimizaciones se expresan más a menudo como minimización que como maximización. Así que, sin cambiar nada, podemos minimizar la log-verosimilitud negativa, que podemos expresar de la siguiente manera:



Si suponemos que σ es fijo, podemos ignorar el primer término, porque no depende de w ni de b. El segundo término es idéntico a la pérdida por error al cuadrado introducida antes, salvo por la constante multiplicativa 1/ σ2. Afortunadamente, la solución tampoco depende de σ tampoco. De ello se deduce que la minimización del error cuadrático medio es equivalente a la estimación de máxima verosimilitud de un modelo lineal bajo el supuesto de ruido gaussiano aditivo.

3.1.4. La regresión lineal como red neuronal

Mientras que los modelos lineales no son lo suficientemente ricos como para expresar las muchas redes complicadas que introduciremos en este libro, las redes neuronales (artificiales) son lo suficientemente ricas como para subsumir los modelos lineales como redes en las que cada característica está representada por una neurona de entrada, todas las cuales están conectadas directamente a la salida.

La Fig. 3.1.2 representa la regresión lineal como una red neuronal. El diagrama destaca el patrón de conectividad, por ejemplo, cómo se conecta cada entrada a la salida, pero no los valores específicos que toman los pesos o los sesgos.

Diagrama

Descripción generada automáticamente

Las entradas son x1,...,xd. Nos referimos a d como el número de entradas o la dimensionalidad de las características en la capa de entrada. La salida de la red es ‘o1’ . Como sólo intentamos predecir un único valor numérico, sólo tenemos una neurona de salida. Observe que los valores de entrada están todos dados. Sólo hay una neurona computarizada. En resumen, podemos pensar en la regresión lineal como una red neuronal totalmente conectada de una sola capa. Encontraremos redes con muchas más capas en capítulos posteriores.