En Machine Learning un *atributo* es un tipo de datos (p. ej., "Kilometraje"), mientras que un *característica* tiene varios significados según el contexto, pero generalmente significa un atributo más su valor (p. ej., "Kilometraje = 15.000 "). Mucha gente usa las palabras *atributo* y *característica* aunque de manera intercambiable.

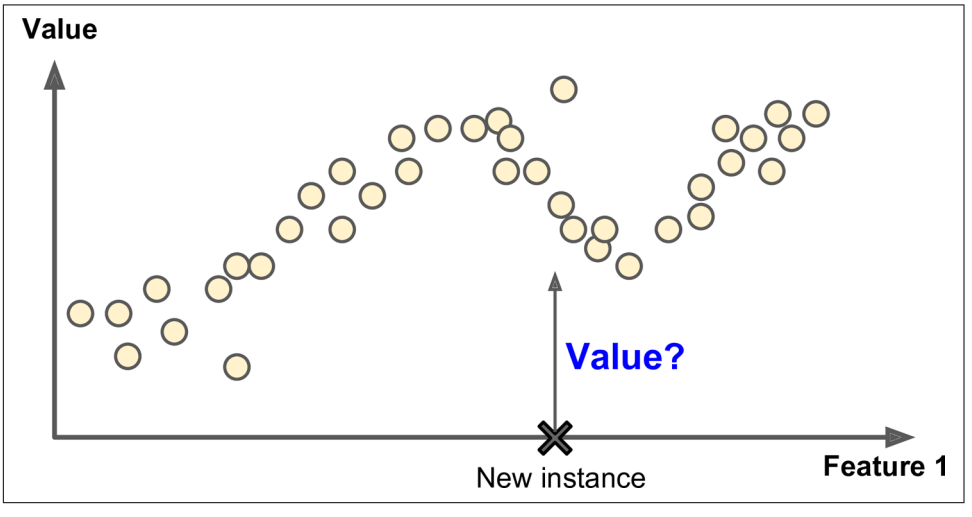

Figura 1-6. Regresión

Tenga en cuenta que algunos algoritmos de regresión también se pueden utilizar para la clasificación y viceversa. Por ejemplo, *Regresión logística* se usa comúnmente para la clasificación, ya que puede generar un valor que corresponde a la probabilidad de pertenecer a una clase determinada (por ejemplo, 20% de probabilidad de ser spam).

Estos son algunos de los algoritmos de aprendizaje supervisado más importantes (que se tratan en este libro):

- k-Vecinos más cercanos
- Regresión lineal
- Regresión logística
- Soporta máquinas vectoriales (SVM)
- Árboles de decisión y bosques aleatorios
- Redes neuronales [2]

2 Algunas arquitecturas de redes neuronales pueden no estar supervisadas, como los codificadores automáticos y las máquinas Boltzmann restringidas. También pueden estar semisupervisados, como en las redes de creencias profundas y en la formación previa no supervisada.

Aprendizaje sin supervisión

En *aprendizaje sin supervisión,* como puede imaginar, los datos de entrenamiento no están etiquetados (Figura 1-7). El sistema intenta aprender sin un maestro.

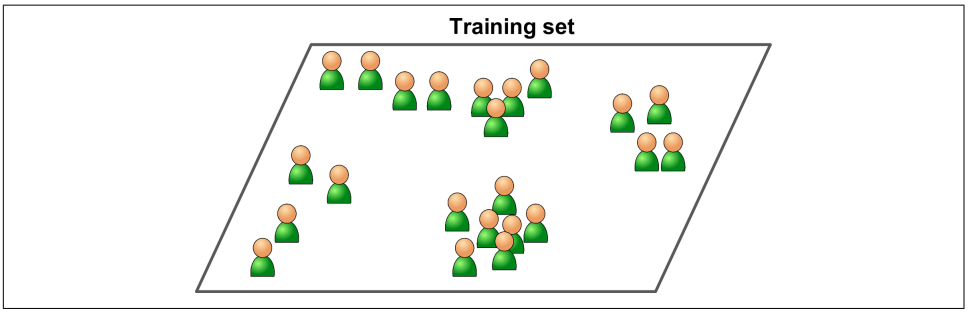

Figura 1-7. Un conjunto de formación sin etiquetar para el aprendizaje sin supervisión

Estos son algunos de los algoritmos de aprendizaje no supervisados más importantes (cubriremos la reducción de dimensionalidad en Capítulo 8):

- Agrupación
  - k-medias
  - Análisis jerárquico de conglomerados (HCA)
  - Maximización de expectativas
- Visualización y reducción de dimensionalidad
  - Análisis de componentes principales (PCA)
  - Kernel PCA
  - Incrustación localmente lineal (LLE)
  - Incrustación de vecinos estocásticos distribuidos en t (t-SNE)
- Aprendizaje de reglas de asociación
  - A priori
  - Eclat

Por ejemplo, supongamos que tiene muchos datos sobre los visitantes de su blog. Es posible que desee ejecutar un *agrupamiento* algoritmo para intentar detectar grupos de visitantes similares (Figura 1-8). En ningún momento le dice al algoritmo a qué grupo pertenece un visitante: encuentra esas conexiones sin su ayuda. Por ejemplo, puede notar que el 40% de sus visitantes son hombres que aman las historietas y generalmente leen su blog por la noche, mientras que el 20% son jóvenes amantes de la ciencia ficción que visitan los fines de semana, y así sucesivamente. Si usa un

agrupación jerárquica algoritmo, también puede subdividir cada grupo en grupos más pequeños. Esto puede ayudarlo a orientar sus publicaciones para cada grupo.

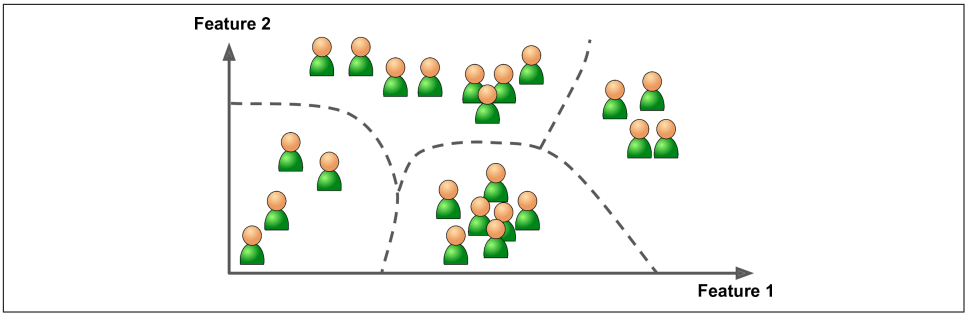

Figura 1-8. Agrupación

Visualización Los algoritmos también son buenos ejemplos de algoritmos de aprendizaje no supervisados: les proporcionas una gran cantidad de datos complejos y sin etiquetar, y generan una representación 2D o 3D de tus datos que se puede trazar fácilmente (Figura 1-9). Estos algoritmos intentan preservar tanta estructura como puedan (por ejemplo, tratando de evitar que los clústeres separados en el espacio de entrada se superpongan en la visualización), para que pueda comprender cómo se organizan los datos y tal vez identificar patrones insospechados.

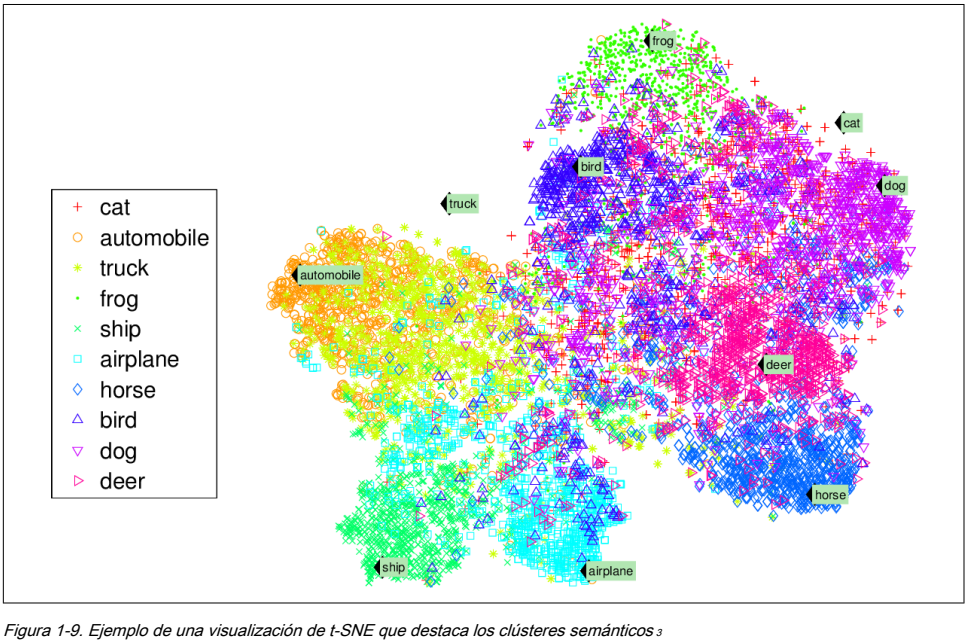

Figura 1-9. Ejemplo de una visualización de t-SNE que destaca los clústeres semánticos [3]

3 Observe cómo los animales están bastante bien separados de los vehículos, cómo los caballos están cerca de los ciervos pero lejos de los pájaros, etc. Figura reproducida con permiso de Socher, Ganjoo, Manning y Ng (2013), "Visualización en T-SNE del espacio semántico de palabras".

Una tarea relacionada es *reducción de dimensionalidad,* en el que el objetivo es simplificar los datos sin perder demasiada información. Una forma de hacer esto es fusionar varias características correlacionadas en una. Por ejemplo, el kilometraje de un automóvil puede estar muy correlacionado con su edad, por lo que el algoritmo de reducción de dimensionalidad los fusionará en una característica que represente el desgaste del automóvil. Se llama *extracción de características.*

A menudo, es una buena idea intentar reducir la dimensión de sus datos de entrenamiento utilizando un algoritmo de reducción de dimensionalidad antes de alimentarlo a otro algoritmo de aprendizaje automático (como un algoritmo de aprendizaje supervisado). Se ejecutará mucho más rápido, los datos ocuparán menos espacio en disco y memoria y, en algunos casos, también puede funcionar mejor.

Otra tarea importante sin supervisión es *Detección de anomalías* —Por ejemplo, detectar transacciones inusuales con tarjetas de crédito para evitar fraudes, detectar defectos de fabricación o eliminar automáticamente valores atípicos de un conjunto de datos antes de enviarlos a otro algoritmo de aprendizaje. El sistema está entrenado con instancias normales, y cuando ve una nueva instancia, puede decir si se ve como una normal o si es probable que sea una anomalía (ver Figura 1-10).

Figura 1-10. Detección de anomalías

Finalmente, otra tarea común sin supervisión es *aprendizaje de reglas de asociación,* en el que el objetivo es profundizar en grandes cantidades de datos y descubrir relaciones interesantes entre atributos. Por ejemplo, suponga que tiene un supermercado. La ejecución de una regla de asociación en sus registros de ventas puede revelar que las personas que compran salsa barbacoa y papas fritas también tienden a comprar bistec. Por lo tanto, es posible que desee colocar estos elementos cerca unos de otros.

Aprendizaje semisupervisado

Algunos algoritmos pueden trabajar con datos de entrenamiento parcialmente etiquetados, generalmente muchos datos sin etiquetar y un poco de datos etiquetados. Se llama *aprendizaje semisupervisado* (Figura 1-11).

Algunos servicios de alojamiento de fotografías, como Google Photos, son buenos ejemplos de esto. Una vez que subes todas las fotos de tu familia al servicio, este reconoce automáticamente que la misma persona A aparece en las fotos 1, 5 y 11, mientras que otra persona B aparece en las fotos 2, 5 y 7. Esta es la parte sin supervisión del algoritmo (agrupamiento). Ahora todo lo que necesita el sistema es que le digas quiénes son estas personas. Solo una etiqueta por persona [4]

y es capaz de nombrar a todos en cada foto, lo que es útil para buscar fotos.

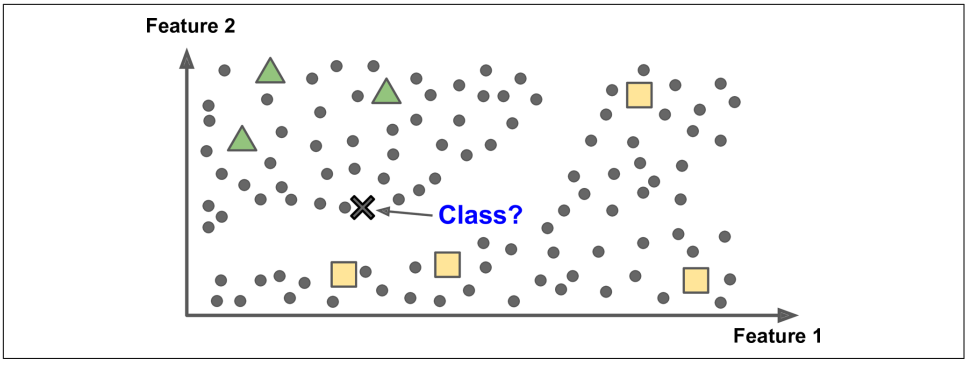

Figura 1-11. Aprendizaje semisupervisado

La mayoría de los algoritmos de aprendizaje semisupervisados son combinaciones de algoritmos supervisados y no supervisados. Por ejemplo, *redes de creencias profundas* DBN) se basan en componentes no supervisados llamados *máquinas Boltzmann restringidas (* RBM) apilados uno encima del otro. Los GBR se entrenan secuencialmente de manera no supervisada, y luego todo el sistema se ajusta mediante técnicas de aprendizaje supervisado.

Aprendizaje reforzado

Aprendizaje reforzado es una bestia muy diferente. El sistema de aprendizaje, llamado *agente* en este contexto, puede observar el entorno, seleccionar y realizar acciones, y obtener *recompensas* a cambio (o *sanciones* en forma de recompensas negativas, como en Figura 1-12). Entonces debe aprender por sí mismo cuál es la mejor estrategia, llamada *política,* para obtener la mayor recompensa a lo largo del tiempo. Una política define qué acción debe elegir el agente cuando se encuentra en una situación determinada.

4 Ahí es cuando el sistema funciona perfectamente. En la práctica, a menudo crea algunos clústeres por persona y, a veces, mezcla a dos personas que se parecen, por lo que es necesario proporcionar algunas etiquetas por persona y limpiar manualmente algunos clústeres.

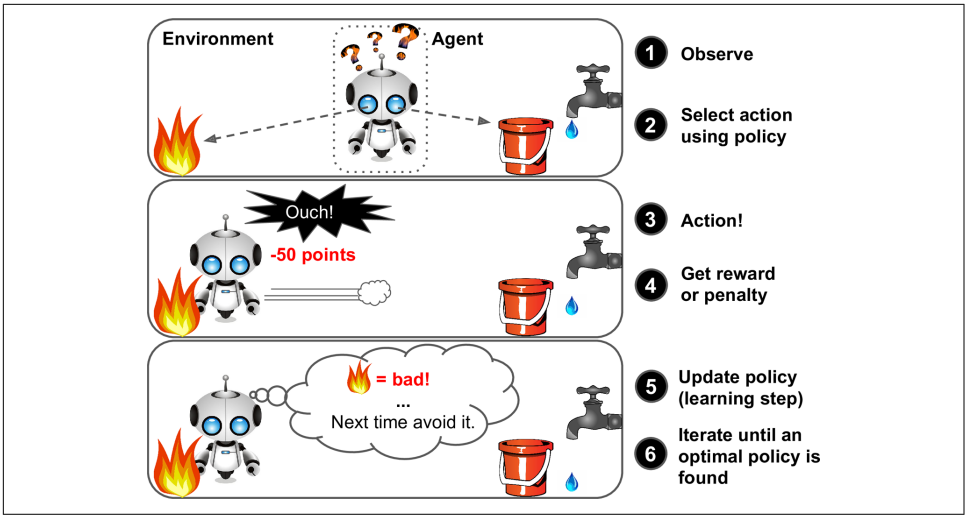

Figura 1-12. Aprendizaje reforzado

Por ejemplo, muchos robots implementan algoritmos de aprendizaje por refuerzo para aprender a caminar. El programa AlphaGo de DeepMind también es un buen ejemplo de aprendizaje por refuerzo: apareció en los titulares en marzo de 2016 cuando venció al campeón mundial Lee Sedol en el juego de *Vamos*. Aprendió su política ganadora analizando millones de juegos y luego jugando muchos juegos contra sí mismo. Tenga en cuenta que el aprendizaje se desactivó durante los juegos contra el campeón; AlphaGo solo estaba aplicando la política que había aprendido.

Aprendizaje por lotes y en línea

Otro criterio utilizado para clasificar los sistemas de aprendizaje automático es si el sistema puede aprender de forma incremental a partir de un flujo de datos entrantes.

Aprendizaje por lotes

En *aprendizaje por lotes,* el sistema es incapaz de aprender de forma incremental: debe entrenarse utilizando todos los datos disponibles. Por lo general, esto requerirá mucho tiempo y recursos informáticos, por lo que generalmente se realiza sin conexión. Primero se entrena el sistema, luego se lanza a producción y se ejecuta sin más aprendizaje; simplemente aplica lo que ha aprendido. Se llama *aprendizaje fuera de línea.*

Si desea que un sistema de aprendizaje por lotes conozca datos nuevos (como un nuevo tipo de correo no deseado), debe entrenar una nueva versión del sistema desde cero en el conjunto de datos completo (no solo los nuevos datos, sino también los antiguos) y, a continuación, detenga el sistema antiguo y sustitúyalo por el nuevo.

Afortunadamente, todo el proceso de capacitación, evaluación y lanzamiento de un sistema de aprendizaje automático se puede automatizar con bastante facilidad (como se muestra en Figura 1-3), así que incluso un

El sistema de aprendizaje por lotes puede adaptarse al cambio. Simplemente actualice los datos y entrene una nueva versión del sistema desde cero con la frecuencia que necesite.

Esta solución es simple y, a menudo, funciona bien, pero el entrenamiento con el conjunto completo de datos puede llevar muchas horas, por lo que normalmente entrenaría un nuevo sistema solo cada 24 horas o incluso solo semanalmente. Si su sistema necesita adaptarse a datos que cambian rápidamente (por ejemplo, para predecir los precios de las acciones), entonces necesita una solución más reactiva.

Además, el entrenamiento con el conjunto completo de datos requiere una gran cantidad de recursos informáticos (CPU, espacio de memoria, espacio en disco, E / S de disco, E / S de red, etc.). Si tienes muchos datos y automatizas tu sistema para entrenar desde cero todos los días, te acabará costando mucho dinero. Si la cantidad de datos es enorme, incluso puede resultar imposible utilizar un algoritmo de aprendizaje por lotes.

Finalmente, si su sistema necesita ser capaz de aprender de forma autónoma y tiene recursos limitados (por ejemplo, una aplicación de teléfono inteligente o un rover en Marte), entonces lleve consigo grandes cantidades de datos de entrenamiento y consuma muchos recursos para entrenar durante horas cada día. El día es espectacular.

Afortunadamente, una mejor opción en todos estos casos es utilizar algoritmos que sean capaces de aprender de forma incremental.

Aprender en línea

En *aprender en línea,* entrena el sistema de forma incremental alimentándolo con instancias de datos secuencialmente, ya sea individualmente o por grupos pequeños *mini lotes.* Cada paso de aprendizaje es rápido y económico, por lo que el sistema puede aprender sobre nuevos datos sobre la marcha, a medida que llegan (consulte Figura 1-13).

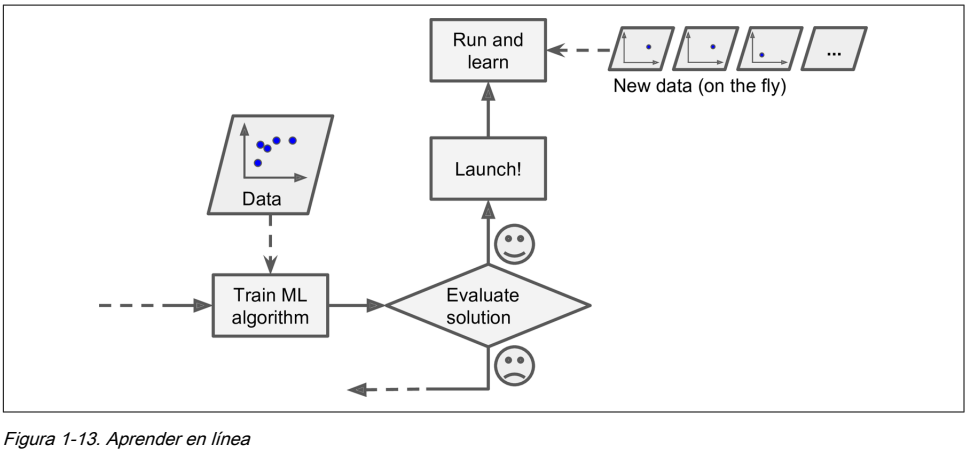

Figura 1-13. Aprender en línea

El aprendizaje en línea es excelente para los sistemas que reciben datos como un flujo continuo (por ejemplo, precios de las acciones) y necesitan adaptarse a los cambios de manera rápida o autónoma. Tambien es una buena opcion

si tiene recursos informáticos limitados: una vez que un sistema de aprendizaje en línea ha aprendido sobre nuevas instancias de datos, ya no los necesita, por lo que puede descartarlos (a menos que desee poder retroceder a un estado anterior y "reproducir" el datos). Esto puede ahorrar una gran cantidad de espacio.

Los algoritmos de aprendizaje en línea también se pueden utilizar para entrenar sistemas en enormes conjuntos de datos que no pueden caber en la memoria principal de una máquina (esto se llama *fuera de núcleo* aprendizaje). El algoritmo carga parte de los datos, ejecuta un paso de entrenamiento en esos datos y repite el proceso hasta que se ha ejecutado en todos los datos (consulte Figura 1-14).

Todo este proceso generalmente se realiza fuera de línea (es decir, no en el sistema en vivo), por lo que *aprender en línea* puede ser un nombre confuso. Piense en ello como *aprendizaje incremental.*

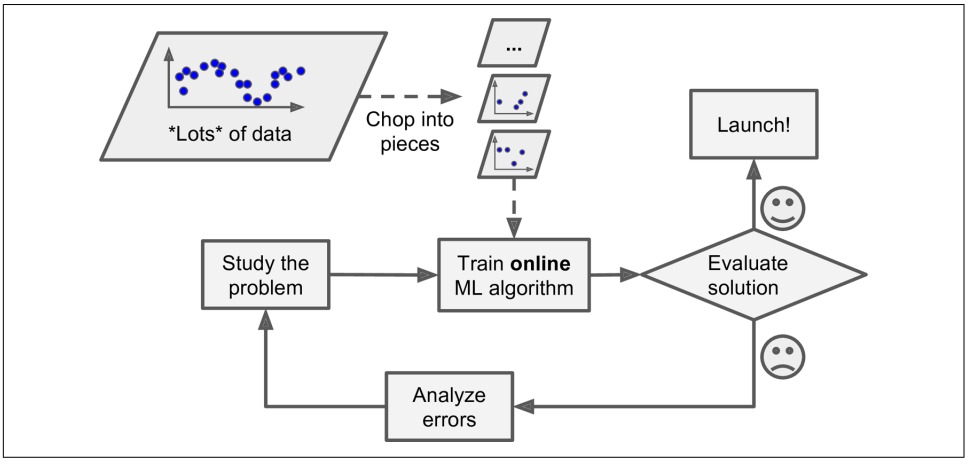

Figura 1-14. Usar el aprendizaje en línea para manejar grandes conjuntos de datos

Un parámetro importante de los sistemas de aprendizaje en línea es qué tan rápido deben adaptarse a los datos cambiantes: esto se llama el *tasa de aprendizaje*. Si establece una alta tasa de aprendizaje, su sistema se adaptará rápidamente a los nuevos datos, pero también tenderá a olvidar rápidamente los datos antiguos (no desea que un filtro de spam marque solo los últimos tipos de spam que se mostró) . Por el contrario, si establece una tasa de aprendizaje baja, el sistema tendrá más inercia; es decir, aprenderá más lentamente, pero también será menos sensible al ruido en los nuevos datos oa secuencias de puntos de datos no representativos.

Un gran desafío con el aprendizaje en línea es que si se introducen datos incorrectos en el sistema, el rendimiento del sistema disminuirá gradualmente. Si hablamos de un sistema en vivo, sus clientes lo notarán. Por ejemplo, los datos incorrectos pueden provenir de un sensor que funciona mal en un robot o de alguien que envía spam a un motor de búsqueda para tratar de obtener una clasificación alta en la búsqueda.

resultados. Para reducir este riesgo, debe monitorear su sistema de cerca y apagar rápidamente el aprendizaje (y posiblemente volver a un estado de trabajo anterior) si detecta una caída en el rendimiento. También es posible que desee monitorear los datos de entrada y reaccionar ante datos anormales (por ejemplo, utilizando un algoritmo de detección de anomalías).

Aprendizaje basado en instancias versus aprendizaje basado en modelos

Una forma más de categorizar los sistemas de aprendizaje automático es por cómo *generalizar.*
La mayoría de las tareas de aprendizaje automático se tratan de hacer predicciones. Esto significa que, dados una serie de ejemplos de formación, el sistema debe poder generalizar a ejemplos que nunca antes había visto. Tener una buena medida de rendimiento en los datos de entrenamiento es bueno, pero insuficiente; el verdadero objetivo es tener un buen rendimiento en nuevas instancias.

Hay dos enfoques principales para la generalización: aprendizaje basado en instancias y aprendizaje basado en modelos.

Aprendizaje basado en instancias

Posiblemente, la forma más trivial de aprendizaje es simplemente aprender de memoria. Si creara un filtro de spam de esta manera, solo marcaría todos los correos electrónicos que son idénticos a los correos electrónicos que ya han sido marcados por los usuarios; no es la peor solución, pero ciertamente no es la mejor.

En lugar de solo marcar los correos electrónicos que son idénticos a los correos electrónicos no deseados conocidos, su filtro de correo no deseado podría programarse para marcar también los correos electrónicos que son muy similares a los correos electrónicos no deseados conocidos. Esto requiere un *medida de similitud* entre dos correos electrónicos. Una medida de similitud (muy básica) entre dos correos electrónicos podría ser contar el número de palabras que tienen en común. El sistema marcaría un correo electrónico como spam si tiene muchas palabras en común con un correo electrónico spam conocido.

Se llama *aprendizaje basado en instancias:* el sistema aprende los ejemplos de memoria, luego generaliza a nuevos casos usando una medida de similitud (Figura 1-15).

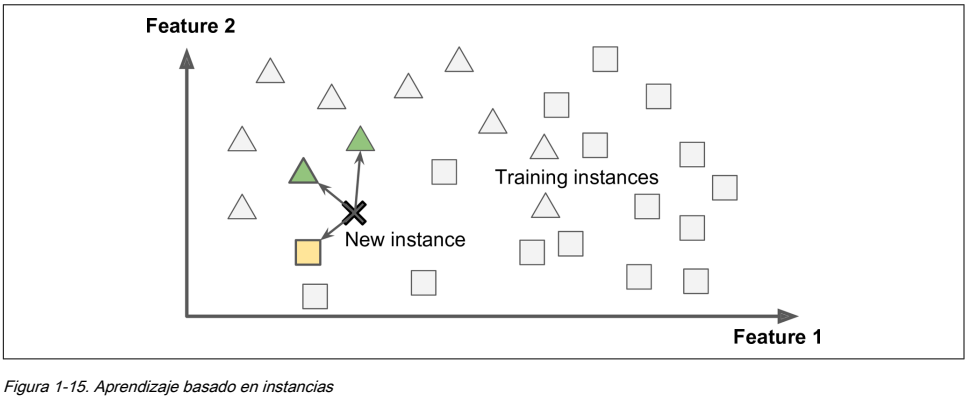

Figura 1-15. Aprendizaje basado en instancias

Aprendizaje basado en modelos

Otra forma de generalizar a partir de un conjunto de ejemplos es construir un modelo de estos ejemplos, luego usar ese modelo para hacer *predicciones.* Se llama *aprendizaje basado en modelos* (Figura 1-16).

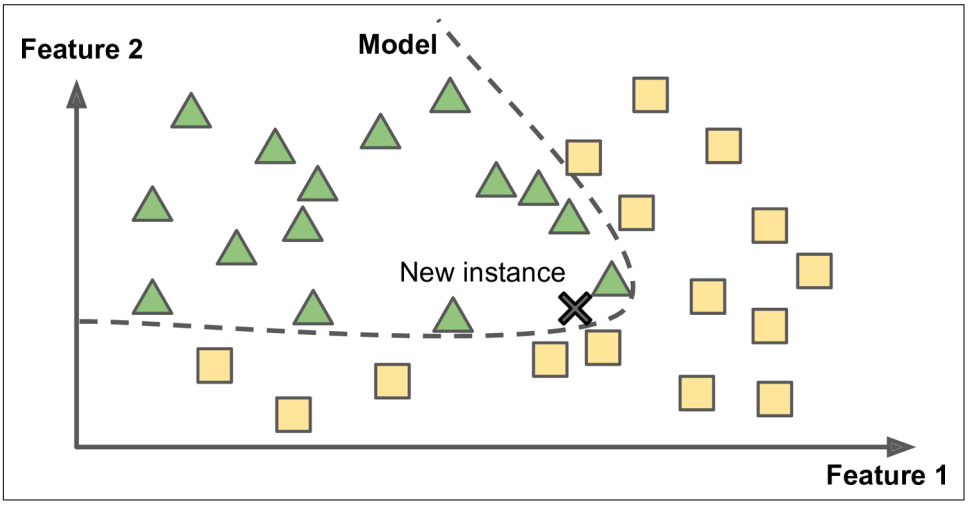

Figura 1-16. Aprendizaje basado en modelos

Por ejemplo, suponga que quiere saber si el dinero hace feliz a la gente, así que descarga el *Índice de vida mejor* datos del Sitio web de la OCDE así como estadísticas sobre el PIB per cápita del Sitio web del FMI . Luego se une a las tablas y ordena por PIB per cápita. Tabla 1-1 muestra un extracto de lo que obtiene.

Tabla 1-1. ¿El dinero hace a la gente más feliz?

País	PIB per cápita (USD)	Satisfacción con la vida
Hungría	12,240	4.9
Corea	27.195	5.8
Francia	37,675	6.5
Australia	50,962	7.3
Estados Unidos	55,805	7.2

Tracemos los datos de algunos países aleatorios (Figura 1-17).

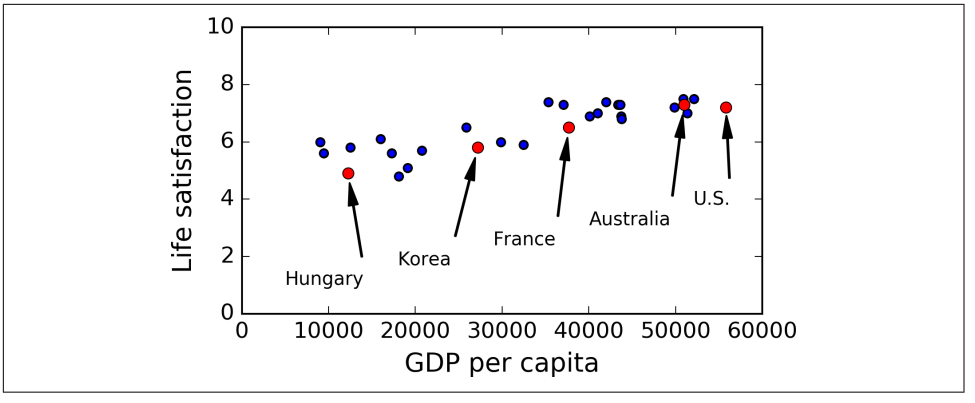

Figura 1-17. ¿Ve una tendencia aquí?

¡Parece haber una tendencia aquí! Aunque los datos son *ruidoso (* es decir, parcialmente aleatorio), parece que la satisfacción con la vida aumenta más o menos linealmente a medida que aumenta el PIB per cápita del país. Entonces decide modelar la satisfacción con la vida como una función lineal del PIB per cápita. Este paso se llama *selección de modelo:* seleccionaste un *Modelo lineal* de satisfacción vital con un solo atributo, el PIB per cápita (Ecuación 1-1).

Ecuación 1-1. Un modelo lineal simple li f e_satis f action =

$\theta_0 + \theta_1 \times$ *PIB per cápita*

Este modelo tiene dos *parámetros del modelo,* θ_0 y θ_1[5] Al ajustar estos parámetros, puede hacer que su modelo represente cualquier función lineal, como se muestra en Figura 1-18 .

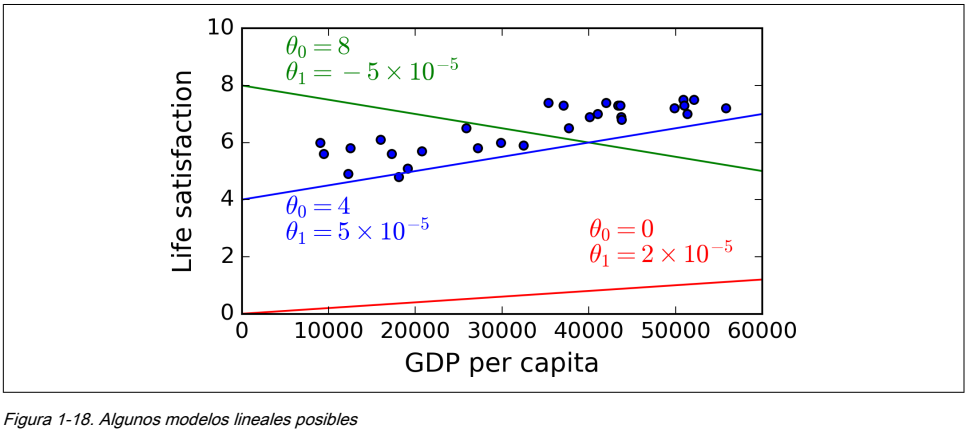

Figura 1-18. Algunos modelos lineales posibles

5 Por convención, la letra griega θ (theta) se usa con frecuencia para representar los parámetros del modelo.

Antes de que pueda utilizar su modelo, debe definir los valores de los parámetros θ_0 y θ_1.
¿Cómo puede saber qué valores harán que su modelo funcione mejor? Para responder a esto
pregunta, debe especificar una medida de rendimiento. Puede definir un *función de utilidad (* o *función de fitness)* que mide como *bueno* su modelo es, o puede definir un *función de costo* que mide como *malo* es. Para problemas de regresión lineal, la gente suele utilizar una función de coste que mide la distancia entre las predicciones del modelo lineal y los ejemplos de formación; el objetivo es minimizar esta distancia.

Aquí es donde entra en juego el algoritmo de regresión lineal: lo alimenta con sus ejemplos de entrenamiento y encuentra los parámetros que hacen que el modelo lineal se ajuste mejor a sus datos. Se llama *formación* el modelo. En nuestro caso, el algoritmo encuentra que el óptimo
los valores de los parámetros son $\theta_0 = 4.85$ y $\theta_1 = 4{,}91 \times 10^{-5}$.

Ahora el modelo se ajusta lo más posible a los datos de entrenamiento (para un modelo lineal), como puede ver en Figura 1-19 .

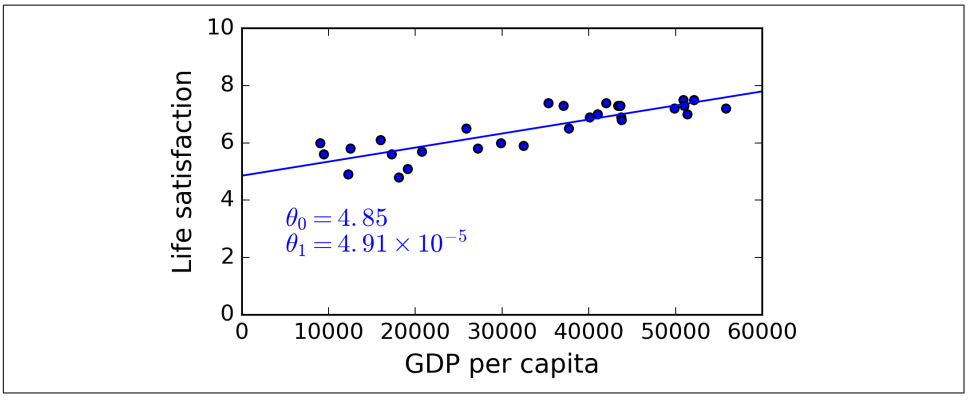

Figura 1-19. El modelo lineal que mejor se ajusta a los datos de entrenamiento

Finalmente está listo para ejecutar el modelo para hacer predicciones. Por ejemplo, digamos que quiere saber qué tan felices son los chipriotas, y los datos de la OCDE no tienen la respuesta. Afortunadamente, puede usar su modelo para hacer una buena predicción: busca el PIB per cápita de Chipre, encuentra $ 22,587 y luego aplica su modelo y encuentra que es probable que la satisfacción con la vida esté en algún lugar alrededor de 4.85 + 22.587 × 4.91 × 10-5 = 5.96.

Para abrir tu apetito Ejemplo 1-1 muestra el código Python que carga los datos, lo prepara, [6] crea un diagrama de dispersión para la visualización y luego entrena un modelo lineal y hace una predicción. [7]

6 El código asume que prepare_country_stats () ya está definido: fusiona el PIB y los datos de satisfacción con la vida en un solo marco de datos de Pandas.

7 Está bien si aún no comprende todo el código; presentaremos Scikit-Learn en los siguientes capítulos.

Ejemplo 1-1. Entrenamiento y ejecución de un modelo lineal usando Scikit-Learn

```
importar matplotlib
importar matplotlib.pyplot como plt
importar numpy como notario público
importar pandas como pd
importar sklearn

# Cargar los datos
oecd_bli = pd . read_csv ( "oecd_bli_2015.csv" , miles = ',' )
PIB per cápita = pd . read_csv ( "gdp_per_capita.csv" , miles = ',' , delimitador = '\ t ' ,
                                                    codificacion = 'latin1' , na_values = "n / A" )

# Preparar los datos
country_stats = prepare_country_stats ( oecd_bli , PIB per cápita )
X = notario público . C_ [ country_stats [ "PIB per cápita" ]]
y = notario público . C_ [ country_stats [ "Satisfacción de vida" ]]

# Visualiza los datos
country_stats . trama ( tipo = 'dispersión' , X = "PIB per cápita" , y = 'Satisfacción de vida' )
plt . show ()

# Seleccione un modelo lineal
lin_reg_model = sklearn . Modelo lineal . Regresión lineal ()

# Entrena al modelo
lin_reg_model . ajuste ( X , y )

# Haga una predicción para Chipre
X_new = [[ 22587 ]] # PIB per cápita de Chipre
impresión ( lin_reg_model . predecir ( X_new )) # salidas [[5.96242338]]
```

Si hubiera utilizado un algoritmo de aprendizaje basado en instancias en su lugar, habría descubierto que Eslovenia tiene el PIB per cápita más cercano al de Chipre ($ 20,732), y dado que los datos de la OCDE nos dicen que la satisfacción con la vida de los eslovenos es 5.7, usted habría predijo una satisfacción con la vida de 5,7 para Chipre. Si se aleja un poco y mira los dos siguientes países más cercanos, encontrará Portugal y España con satisfacciones de vida de 5.1 y 6.5, respectivamente. Al promediar estos tres valores, obtiene 5.77, que está bastante cerca de su predicción basada en el modelo. Este simple algoritmo se llama *k-Vecinos más cercanos* regresión (en este ejemplo, $k = 3$).

Reemplazar el modelo de regresión lineal con la regresión de k-vecinos más cercanos en el código anterior es tan simple como reemplazar esta línea:

```
clf = sklearn . Modelo lineal . Regresión lineal ()
```

Con este:

```
clf = sklearn . vecinos . KVecinosRegresor ( n_vecinos = 3 )
```

Si todo salió bien, su modelo hará buenas predicciones. De lo contrario, es posible que deba usar más atributos (tasa de empleo, salud, contaminación del aire, etc.), obtener más o mejores datos de capacitación de calidad, o quizás seleccionar un modelo más poderoso (por ejemplo, un modelo de regresión polinomial) .

En resumen:

- Estudiaste los datos.
- Seleccionaste un modelo.
- Lo entrenó en los datos de entrenamiento (es decir, el algoritmo de aprendizaje buscó los valores de los parámetros del modelo que minimizan una función de costo).
- Finalmente, aplicó el modelo para hacer predicciones en nuevos casos (esto se llama *inferencia),* esperando que este modelo se generalice bien.

Así es como se ve un proyecto típico de aprendizaje automático. En Capitulo 2 Experimentarás esto de primera mano al pasar por un proyecto de principio a fin.

Hemos cubierto mucho terreno hasta ahora: ahora sabe de qué se trata realmente el aprendizaje automático, por qué es útil, cuáles son algunas de las categorías más comunes de sistemas de aprendizaje automático y cómo es un flujo de trabajo de proyecto típico. Ahora veamos qué puede salir mal en el aprendizaje y evitar que haga predicciones precisas.

Principales desafíos del aprendizaje automático

En resumen, dado que su tarea principal es seleccionar un algoritmo de aprendizaje y entrenarlo con algunos datos, las dos cosas que pueden salir mal son "algoritmo incorrecto" y "datos incorrectos". Comencemos con ejemplos de datos incorrectos.

Cantidad insuficiente de datos de entrenamiento

Para que un niño pequeño aprenda lo que es una manzana, todo lo que necesita es que usted señale una manzana y diga “manzana” (posiblemente repita este procedimiento varias veces). Ahora el niño puede reconocer manzanas en todo tipo de colores y formas. Genio.

El aprendizaje automático no ha llegado todavía; Se necesitan muchos datos para que la mayoría de los algoritmos de aprendizaje automático funcionen correctamente. Incluso para problemas muy simples, normalmente necesita miles de ejemplos, y para problemas complejos, como el reconocimiento de imágenes o voz, es posible que necesite millones de ejemplos (a menos que pueda reutilizar partes de un modelo existente).

La efectividad irrazonable de los datos

en un papel famoso publicado en 2001, los investigadores de Microsoft Michele Banko y Eric Brill demostraron que algoritmos de aprendizaje automático muy diferentes, incluidos los bastante simples, funcionaban casi de manera idéntica en un problema complejo de desambiguación del lenguaje natural[8] una vez que recibieron suficientes datos (como puede ver en Figura 1-20).

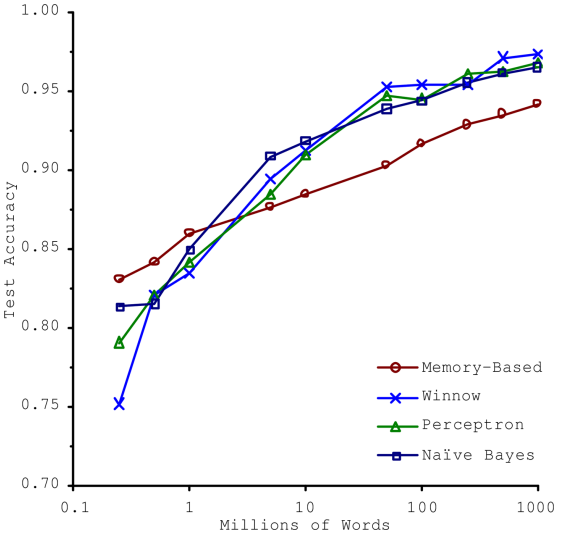

Figura 1-20. La importancia de los datos frente a los algoritmos[9]

Como lo expresaron los autores: "estos resultados sugieren que es posible que deseemos reconsiderar el compromiso entre gastar tiempo y dinero en el desarrollo de algoritmos versus gastarlo en el desarrollo de corpus".

La idea de que los datos importan más que los algoritmos para problemas complejos fue popularizada aún más por Peter Norvig et al. en un documento titulado "La efectividad irrazonable de los datos" publicado en 2009.[10] Sin embargo, debe tenerse en cuenta que los conjuntos de datos pequeños y medianos siguen siendo muy comunes, y no siempre es fácil o barato obtener datos de entrenamiento adicionales, así que no abandone los algoritmos todavía.

8 Por ejemplo, saber si escribir “a”, “dos” o “demasiado” según el contexto.

9 Figura reproducida con permiso de Banko y Brill (2001), “Learning Curves for Confusion Set Disambiguation”.

10 “La efectividad irrazonable de los datos”, Peter Norvig et al. (2009).

Datos de formación no representativos

Para generalizar bien, es crucial que sus datos de entrenamiento sean representativos de los nuevos casos a los que desea generalizar. Esto es cierto tanto si utiliza el aprendizaje basado en instancias como el aprendizaje basado en modelos.

Por ejemplo, el conjunto de países que usamos anteriormente para entrenar el modelo lineal no era perfectamente representativo; faltaban algunos países. Figura 1-21 muestra cómo se ven los datos cuando agrega los países que faltan.

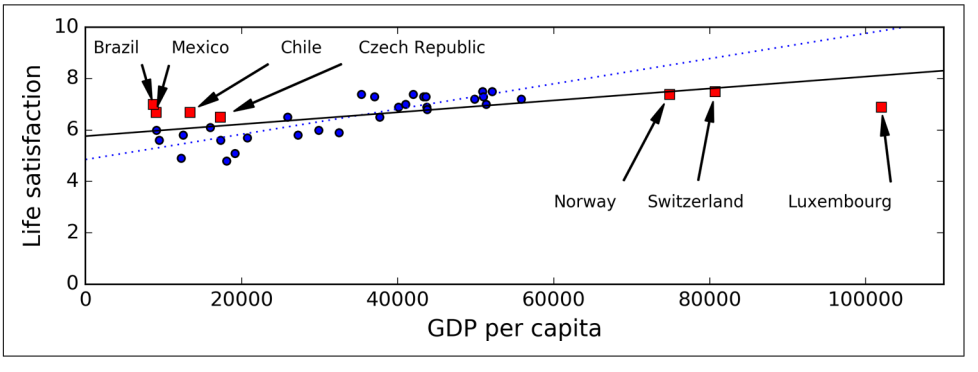

Figura 1-21. Una muestra de formación más representativa

Si entrena un modelo lineal con estos datos, obtiene la línea sólida, mientras que el modelo anterior está representado por la línea de puntos. Como puede ver, agregar algunos países faltantes no solo altera significativamente el modelo, sino que deja en claro que un modelo lineal tan simple probablemente nunca funcionará bien. Parece que los países muy ricos no son más felices que los países moderadamente ricos (de hecho, parecen más infelices) y, a la inversa, algunos países pobres parecen más felices que muchos países ricos.

Mediante el uso de un conjunto de capacitación no representativo, entrenamos un modelo que es poco probable que haga predicciones precisas, especialmente para países muy pobres y muy ricos.

Es fundamental utilizar un conjunto de formación que sea representativo de los casos a los que desea generalizar. Esto suele ser más difícil de lo que parece: si la muestra es demasiado pequeña, tendrá *ruido de muestreo* es decir, datos no representativos como resultado de la casualidad), pero incluso muestras muy grandes pueden no ser representativas si el método de muestreo es defectuoso. Se llama *sesgo de muestreo.*

Un ejemplo famoso de sesgo muestral

Quizás el ejemplo más famoso de sesgo de muestreo ocurrió durante las elecciones presidenciales de Estados Unidos en 1936, que enfrentó a Landon contra Roosevelt: *Compendio literario* realizó una encuesta muy grande, enviando correo a cerca de 10 millones de personas. Obtuvo 2,4 millones de respuestas y predijo con gran confianza que Landon obtendría el 57% de los votos.

En cambio, Roosevelt ganó con el 62% de los votos. El defecto estaba en el *Compendio literario* método de muestreo:

- Primero, para obtener las direcciones a las que enviar las encuestas, el *Compendio literario* utilizaba directorios telefónicos, listas de suscriptores de revistas, listas de miembros de clubes y similares. Todas estas listas tienden a favorecer a las personas más ricas, que tienen más probabilidades de votar por los republicanos (de ahí Landon).

- Segundo, menos del 25% de las personas que recibieron la encuesta respondieron. Nuevamente, esto introduce un sesgo de muestreo, al descartar a las personas a las que no les importa mucho la política, las personas a las que no les gusta la política. *Compendio literario,* y otros grupos clave. Este es un tipo especial de sesgo de muestreo llamado *Sesgo de falta de respuesta.*

Aquí hay otro ejemplo: digamos que quiere construir un sistema para reconocer videos de música funk. Una forma de crear tu conjunto de entrenamiento es buscar "música funk" en YouTube y usar los videos resultantes. Pero esto supone que el motor de búsqueda de YouTube devuelve un conjunto de videos que son representativos de todos los videos de música funk en YouTube. En realidad, es probable que los resultados de la búsqueda estén sesgados hacia artistas populares (y si vives en Brasil, obtendrás muchos videos “funk carioca”, que no suenan en nada a James Brown). Por otro lado, ¿de qué otra manera puede obtener un gran conjunto de entrenamiento?

Datos de mala calidad

Obviamente, si sus datos de entrenamiento están llenos de errores, valores atípicos y ruido (p. Ej., Debido a mediciones de baja calidad), será más difícil para el sistema detectar los patrones subyacentes, por lo que es menos probable que su sistema funcione bien. A menudo, vale la pena dedicar tiempo a limpiar los datos de entrenamiento. La verdad es que la mayoría de los científicos de datos dedican una parte importante de su tiempo a hacerlo. Por ejemplo:

- Si algunos casos son claramente valores atípicos, puede ser útil simplemente descartarlos o intentar corregir los errores manualmente.

- Si a algunas instancias les faltan algunas características (por ejemplo, el 5% de sus clientes no especificaron su edad), debe decidir si desea ignorar este atributo por completo, ignorar estas instancias, completar los valores faltantes (por ejemplo, con la edad media), o entrenar un modelo con la característica y un modelo sin ella, y así sucesivamente.

Características irrelevantes

Como dice el refrán: basura entra, basura sale. Su sistema solo podrá aprender si los datos de entrenamiento contienen suficientes características relevantes y no demasiadas irrelevantes. Una parte fundamental del éxito de un proyecto de aprendizaje automático es crear un buen conjunto de características para entrenar. Este proceso, llamado *ingeniería de características,* implica:

- *Selección de características:* seleccionar las funciones más útiles para entrenar entre las funciones existentes.

- *Extracción de características:* combinar características existentes para producir una más útil (como vimos anteriormente, los algoritmos de reducción de dimensionalidad pueden ayudar).

• Crear nuevas funciones mediante la recopilación de nuevos datos.

Ahora que hemos visto muchos ejemplos de datos incorrectos, veamos un par de ejemplos de algoritmos incorrectos.

Sobreajuste de los datos de entrenamiento

Digamos que estás de visita en un país extranjero y el taxista te estafa. Podrías tener la tentación de decir eso *todos* Los taxistas de ese país son ladrones. Sobregeneralizar es algo que los humanos hacemos con demasiada frecuencia y, lamentablemente, las máquinas pueden caer en la misma trampa si no tenemos cuidado. En Machine Learning esto se llama *sobreajuste:* significa que el modelo funciona bien en los datos de entrenamiento, pero no generaliza bien.

Figura 1-22 muestra un ejemplo de un modelo polinomial de satisfacción con la vida de alto grado que se superpone en gran medida a los datos de entrenamiento. Aunque funciona mucho mejor en los datos de entrenamiento que el modelo lineal simple, ¿realmente confiaría en sus predicciones?

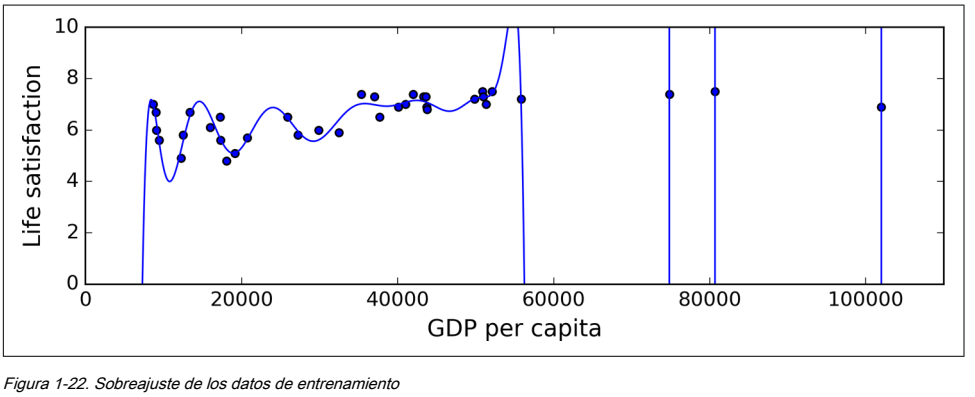

Figura 1-22. Sobreajuste de los datos de entrenamiento

Los modelos complejos, como las redes neuronales profundas, pueden detectar patrones sutiles en los datos, pero si el conjunto de entrenamiento es ruidoso o si es demasiado pequeño (lo que introduce ruido de muestreo), es probable que el modelo detecte patrones en el ruido mismo. Obviamente, estos patrones no se generalizarán a nuevas instancias. Por ejemplo, supongamos que alimenta su modelo de satisfacción con la vida con muchos más atributos, incluidos los poco informativos como el nombre del país. En ese caso, un modelo complejo puede detectar patrones como el hecho de que todos los países en los datos de entrenamiento con un *w* en su nombre tienen una satisfacción con la vida superior a 7: Nueva Zelanda (7,3), Noruega (7,4), Suecia (7,2) y Suiza (7,5). Que confiado

¿Es usted que la regla de satisfacción W se generaliza a Ruanda o Zimbabwe? Obviamente, este patrón ocurrió en los datos de entrenamiento por pura casualidad, pero el modelo no tiene forma de decir si un patrón es real o simplemente el resultado del ruido en los datos.

El sobreajuste ocurre cuando el modelo es demasiado complejo en relación con la cantidad y el ruido de los datos de entrenamiento. Las posibles soluciones son:

- Simplificar el modelo seleccionando uno con menos parámetros (por ejemplo, un modelo lineal en lugar de un modelo polinómico de alto grado), reduciendo el número de atributos en los datos de entrenamiento o restringiendo el modelo.
- Para recopilar más datos de entrenamiento
- Para reducir el ruido en los datos de entrenamiento (p. Ej., Corregir errores de datos y eliminar valores atípicos)

Restringir un modelo para hacerlo más simple y reducir el riesgo de sobreajuste se llama *regularización*. Por ejemplo, el modelo lineal que definimos anteriormente tiene dos parámetros, θ_0 y θ_1. Esto le da al algoritmo de aprendizaje dos *grados de libertad* para adaptar el modelo a los datos de entrenamiento: puede ajustar tanto la altura (θ_0) y la pendiente θ_1) de la línea. Si nosotros forzamos $\theta_1 = 0$, el algoritmo tendría solo un grado de libertad y le costaría mucho más ajustar los datos correctamente: todo lo que podría hacer es mover la línea hacia arriba o hacia abajo para acercarse lo más posible a las instancias de entrenamiento, por lo que terminaría alrededor de la media. ¡Un modelo muy simple en verdad! Si permitimos que el algoritmo modifique θ_1 pero lo obligamos a mantenerlo pequeño, entonces el algoritmo de aprendizaje efectivamente tendrá algunos donde entre uno y dos grados de libertad. Producirá un modelo más simple que con dos grados de libertad, pero más complejo que con solo uno. Desea encontrar el equilibrio adecuado entre ajustar los datos perfectamente y mantener el modelo lo suficientemente simple como para asegurarse de que se generalice bien.

Figura 1-23 muestra tres modelos: la línea de puntos representa el modelo original que fue entrenado con algunos países faltantes, la línea de puntos es nuestro segundo modelo entrenado con todos los países y la línea continua es un modelo lineal entrenado con los mismos datos que el primer modelo pero con una restricción de regularización. Puede ver que la regularización obligó al modelo a tener una pendiente más pequeña, que se ajusta un poco menos a los datos de entrenamiento en los que se entrenó el modelo, pero en realidad le permite generalizar mejor a nuevos ejemplos.

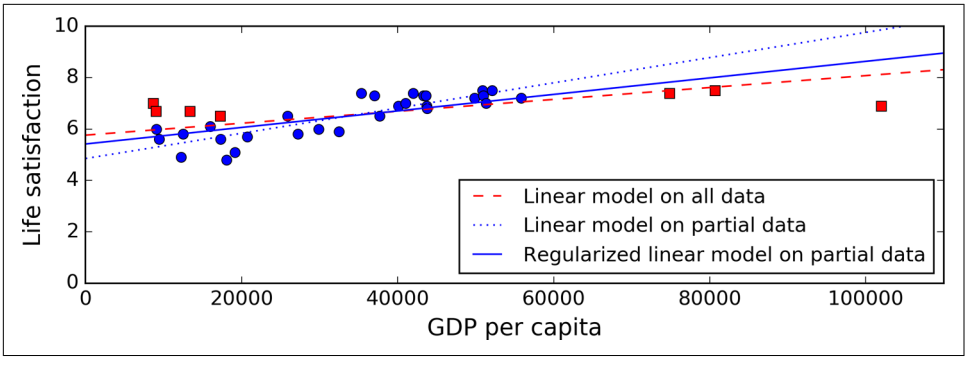

Figura 1-23. La regularización reduce el riesgo de sobreajuste

La cantidad de regularización que se debe aplicar durante el aprendizaje se puede controlar mediante un *hiper- parámetro*. Un hiperparámetro es un parámetro de un algoritmo de aprendizaje (no del modelo). Como tal, no se ve afectado por el algoritmo de aprendizaje en sí; debe configurarse antes del entrenamiento y permanece constante durante el entrenamiento. Si establece el hiperparámetro de regularización en un valor muy grande, obtendrá un modelo casi plano (una pendiente cercana a cero); Es casi seguro que el algoritmo de aprendizaje no se adaptará demasiado a los datos de entrenamiento, pero será menos probable que encuentre una buena solución. Ajustar los hiperparámetros es una parte importante de la construcción de un sistema de aprendizaje automático (verá un ejemplo detallado en el siguiente capítulo).

Adecuar los datos de entrenamiento

Como puedes adivinar *desajuste* es lo contrario de sobreajuste: ocurre cuando su modelo es demasiado simple para aprender la estructura subyacente de los datos. Por ejemplo, un modelo lineal de satisfacción con la vida tiende a no encajar; La realidad es simplemente más compleja que el modelo, por lo que sus predicciones seguramente serán inexactas, incluso en los ejemplos de capacitación.

Las principales opciones para solucionar este problema son:

- Seleccionar un modelo más potente, con más parámetros
- Mejorar las funciones del algoritmo de aprendizaje (ingeniería de funciones)
- Reducir las restricciones del modelo (por ejemplo, reducir el hiperparámetro de regularización)

Dar un paso atrás

A estas alturas ya sabes mucho sobre Machine Learning. Sin embargo, pasamos por tantos conceptos que puede que te sientas un poco perdido, así que retrocedamos y veamos el panorama general: