Aprendizaje automático práctico con Scikit-Learn, Keras y TensorFlow, segunda edición por Aurélien Géron

Capítulo 1. El panorama del aprendizaje automático

Cuandola mayoría de la gente escucha "Machine Learning", se imagina un robot: un mayordomo confiable o un Terminator mortal, según a quién le preguntes. Pero el aprendizaje automático no es solo una fantasía futurista; ya está aquí. De hecho, ha existido durante décadas en algunas aplicaciones especializadas,como el reconocimiento óptico de caracteres (OCR). Pero la primera aplicación de ML que realmente se volvió popular, mejorando las vidas de cientos de millones de personas, se apoderó del mundo en la década de 1990:el filtro de spam . No es exactamente un Skynet consciente de sí mismo, pero técnicamente califica como Machine Learning (en realidad ha aprendido tan bien que ya rara vez es necesario marcar un correo electrónico como spam). Le siguieron cientos de aplicaciones de aprendizaje automático que ahora impulsan silenciosamente cientos de productos y funciones que usa regularmente, desde mejores recomendaciones hasta búsquedas por voz.

¿Dónde comienza el aprendizaje automático y dónde termina? ¿Qué significa exactamente que una máquina aprenda algo? Si descargo una copia de Wikipedia, ¿mi computadora realmente ha aprendido algo? ¿Es de repente más inteligente? En este capítulo, comenzaremos por aclarar qué es el aprendizaje automático y por qué es posible que desee utilizarlo.

Luego, antes de comenzar a explorar el continente del aprendizaje automático, echaremos un vistazo al mapa y aprenderemos sobre las regiones principales y los puntos de referencia más notables: aprendizaje supervisado versus no supervisado, aprendizaje en línea versus aprendizaje por lotes, basado en instancias versus basado en modelos. aprendizaje. Luego, analizaremos el flujo de trabajo de un proyecto de aprendizaje automático típico, discutiremos los principales desafíos que puede enfrentar y cubriremos cómo evaluar y ajustar un sistema de aprendizaje automático.

Este capítulo presenta muchos conceptos fundamentales (y jerga) que todo científico de datos debería saber de memoria. Será una descripción general de alto nivel (es el único capítulo sin mucho código), todo bastante simple, pero debe asegurarse de que todo sea claro antes de continuar con el resto del libro. ¡Así que tómate un café y comencemos!

PROPINA

Si ya conoce todos los conceptos básicos del aprendizaje automático, es posible que desee pasar directamente al Capítulo 2 . Si no está seguro, intente responder todas las preguntas que se enumeran al final del capítulo antes de continuar.

¿Qué es el aprendizaje automático?

El aprendizaje automático es la ciencia (y el arte) de programar computadoras para que puedan aprender de los datos .

Aquí hay una definición un poco más general:

[Machine Learning es el] campo de estudio que brinda a las computadoras la capacidad de aprender sin ser programadas explícitamente.

Arthur Samuel, 1959

Y uno más orientado a la ingeniería:

Un programa informático se dice que aprender de la experiencia E con respecto a alguna tarea T y alguna medida de rendimiento P , si su desempeño en T , medido por P , mejora con la experiencia E .

Tom Mitchell, 1997

Su filtro de correo no deseado es un programa de aprendizaje automático que, con ejemplos de correos electrónicos no deseados (p. Ej., Marcados por usuarios) y ejemplos de correos electrónicos regulares (no spam, también llamados "ham"), puede aprender a marcar el spam. Los ejemplos que usa el sistema para aprender sonllamado el conjunto de entrenamiento . Cada ejemplo de entrenamiento esllamada instancia de entrenamiento (o muestra). En este caso, la tarea T es marcar spam para nuevos correos electrónicos, la experiencia E eses necesario definir los datos de entrenamiento y la medida de desempeño P ; por ejemplo, puede utilizar la proporción de correos electrónicos clasificados correctamente. EstaUna medida de desempeño particular se llama precisión y se usa a menudo en tareas de clasificación.

Si acaba de descargar una copia de Wikipedia, su computadora tiene muchos más datos, pero de repente no es mejor en ninguna tarea. Por lo tanto, descargar una copia de Wikipedia no es aprendizaje automático.

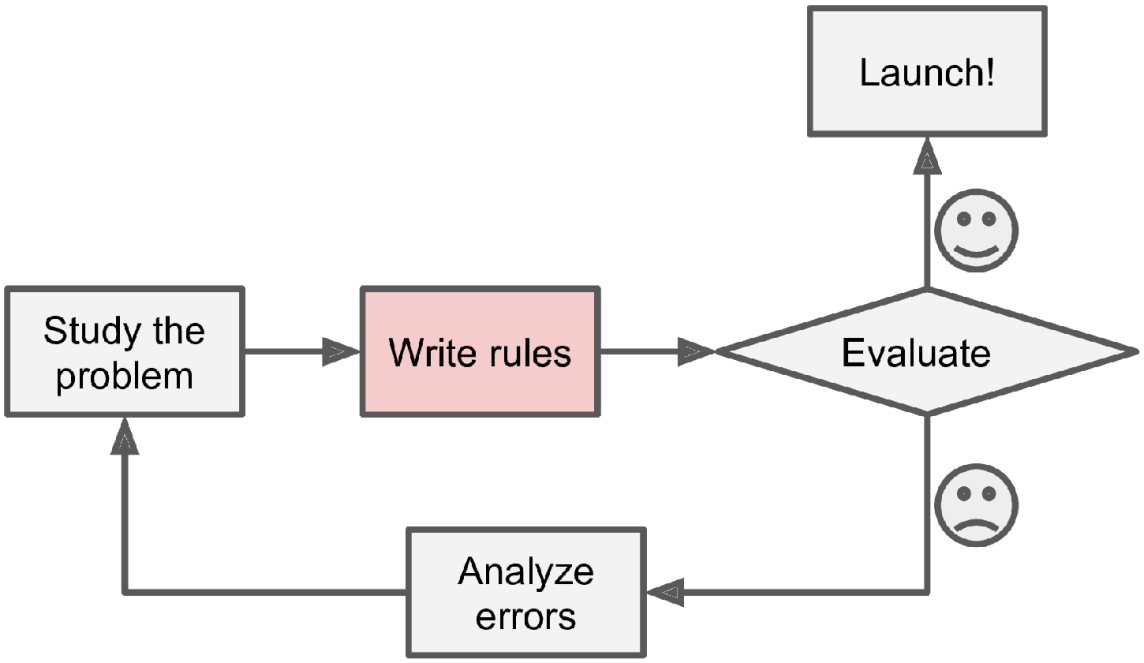
¿Por qué utilizar el aprendizaje automático?

Considerarcómo escribiría un filtro de spam utilizando técnicas de programación tradicionales (Figura 1-1):

Primero, debería considerar cómo se ve normalmente el spam. Es posible que observe que algunas palabras o frases (como "4U", "tarjeta de crédito", "gratis" e "increíble") tienden a aparecer mucho en la línea de asunto. Quizás también notaría algunos otros patrones en el nombre del remitente, el cuerpo del correo electrónico y otras partes del correo electrónico.

Escribiría un algoritmo de detección para cada uno de los patrones que notó, y su programa marcaría los correos electrónicos como spam si se detectaran varios de estos patrones.

Probaría su programa y repetiría los pasos 1 y 2 hasta que fuera lo suficientemente bueno para iniciarse.

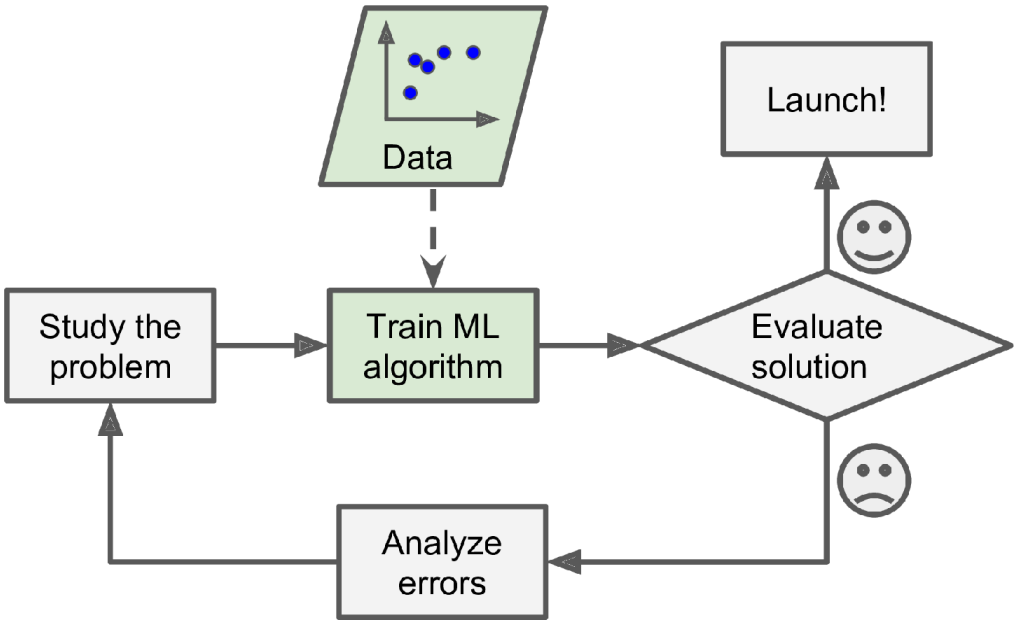


Dado que el problema es difícil, es probable que su programa se convierta en una larga lista de reglas complejas, bastante difíciles de mantener.

Por el contrario, un filtro de spam basado en técnicas de aprendizaje automático aprende automáticamente qué palabras y frases son buenos predictores de spam al detectar patrones de palabras inusualmente frecuentes en los ejemplos de spam en comparación con los ejemplos de radioaficionados (Figura 1-2). El programa es mucho más corto, más fácil de mantener y probablemente más preciso.

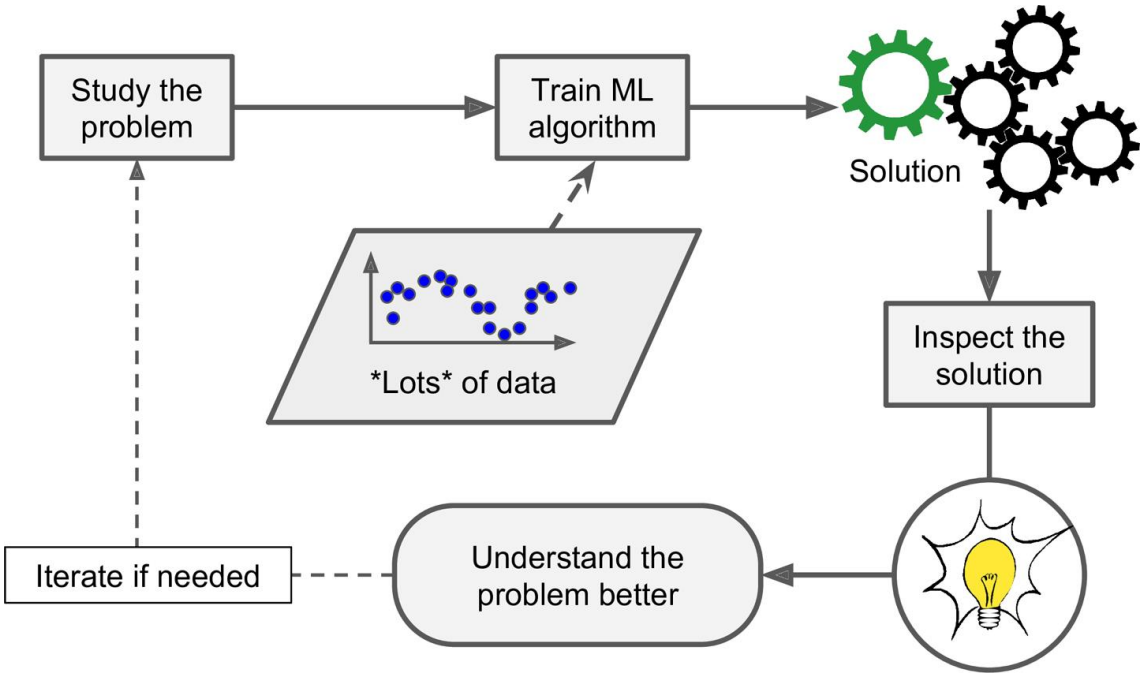
¿Qué pasa si los spammers notan que todos sus correos electrónicos que contienen "4U" están bloqueados? En su lugar, podrían empezar a escribir "For U". Un filtro de spam que utiliza técnicas de programación tradicionales debería actualizarse para marcar los correos electrónicos "For U". Si los spammers siguen trabajando alrededor de su filtro de spam, tendrá que seguir escribiendo nuevas reglas para siempre.

Por el contrario, un filtro de spam basado en técnicas de aprendizaje automático detecta automáticamente que "For U" se ha vuelto inusualmente frecuente en el spam marcado por los usuarios, y comienza a marcarlos sin su intervención (Figura 1-3).



Otra área en la que destaca el aprendizaje automático son los problemas que son demasiado complejos para los enfoques tradicionales o que no tienen un algoritmo conocido. Por ejemplo, considere el reconocimiento de voz. Supongamos que desea comenzar de manera simple y escribir un programa capaz de distinguir las palabras "uno" y "dos". Es posible que notes que la palabra "dos" comienza con un sonido de tono alto ("T"), por lo que podrías codificar un algoritmo que mida la intensidad del sonido de tono alto y usarlo para distinguir unos y dos, pero obviamente esta técnica no escalar a miles de palabras pronunciadas por millones de personas muy diferentes en entornos ruidosos y en docenas de idiomas. La mejor solución (al menos hoy) es escribir un algoritmo que aprenda por sí mismo, dados muchos ejemplos de grabaciones para cada palabra.

Finalmente, el aprendizaje automático puede ayudar a los humanos a aprender (Figura 1-4). Los algoritmos de ML se pueden inspeccionar para ver qué han aprendido (aunque para algunos algoritmos esto puede ser complicado). Por ejemplo, una vez que un filtro de spam ha sido entrenado con suficiente spam, se puede inspeccionar fácilmente para revelar la lista de palabras y combinaciones de palabras que cree que son los mejores predictores de spam. A veces, esto revelará correlaciones insospechadas o nuevas tendencias y, por lo tanto, conducirá a una mejor comprensión del problema. La aplicación de técnicas de aprendizaje automático para profundizar en grandes cantidades de datos puede ayudar a descubrir patrones que no eran evidentes de inmediato. A esto se le llama minería de datos .



En resumen, el aprendizaje automático es ideal para:

Problemas para los que las soluciones existentes requieren muchos ajustes o listas largas de reglas: un algoritmo de aprendizaje automático a menudo puede simplificar el código y funcionar mejor que el enfoque tradicional.

Problemas complejos para los que el uso de un enfoque tradicional no ofrece una buena solución: las mejores técnicas de Machine Learning quizás puedan encontrar una solución.

Entornos fluctuantes: un sistema de aprendizaje automático puede adaptarse a nuevos datos.

Obtener conocimientos sobre problemas complejos y grandes cantidades de datos.

Ejemplos de aplicaciones

Miremos en algunos ejemplos concretos de tareas de Machine Learning, junto con las técnicas que pueden abordarlas:

Analizar imágenes de productos en una línea de producción para clasificarlos automáticamente

Esta es la clasificación de imágenes, que normalmente se realiza mediante redes neuronales convolucionales (CNN; consulte el Capítulo 14).

Detectar tumores en escáneres cerebrales

Esta es la segmentación semántica, en la que se clasifica cada píxel de la imagen (ya que queremos determinar la ubicación exacta y la forma de los tumores), normalmente también utilizando CNN.

Clasificación automática de artículos de noticias

Este es el procesamiento del lenguaje natural (NLP) y, más específicamente, la clasificación de texto, que se puede abordar utilizando redes neuronales recurrentes (RNN), CNN o Transformers (consulte el Capítulo 16).

Marcar automáticamente comentarios ofensivos en foros de discusión

Esta también es una clasificación de texto, utilizando las mismas herramientas de PNL.

Resumir documentos largos automáticamente

Esta es una rama de la PNL llamada resumen de texto, nuevamente usando las mismas herramientas.

Crear un chatbot o un asistente personal

Esto involucra muchos componentes de la PNL, incluida la comprensión del lenguaje natural (NLU) y los módulos de respuesta a preguntas.

Pronosticar los ingresos de su empresa el próximo año, en base a muchas métricas de desempeño

Esta es una tarea de regresión (es decir, predecir valores) que se puede abordar utilizando cualquier modelo de regresión, como un modelo de regresión lineal o de regresión polinomial (ver Capítulo 4), una SVM de regresión (ver Capítulo 5), un bosque aleatorio de regresión (ver Capítulo 7), o una red neuronal artificial (ver Capítulo 10). Si desea tener en cuenta secuencias de métricas de rendimiento pasadas, es posible que desee utilizar RNN, CNN o Transformers (consulte los capítulos 15 y 16).

Hacer que su aplicación reaccione a los comandos de voz

Este es el reconocimiento de voz, que requiere el procesamiento de muestras de audio: dado que son secuencias largas y complejas, generalmente se procesan utilizando RNN, CNN o Transformers (consulte los capítulos 15 y 16).

Detectar fraudes con tarjetas de crédito

Esta es la detección de anomalías (consulte el Capítulo 9).

Segmentar clientes en función de sus compras para que puedas diseñar una estrategia de marketing diferente para cada segmento

Esto es agrupación (consulte el Capítulo 9).

Representar un conjunto de datos complejo y de alta dimensión en un diagrama claro y profundo

Esta es la visualización de datos, que a menudo implica técnicas de reducción de dimensionalidad (consulte el Capítulo 8).

Recomendar un producto que pueda interesar a un cliente, basado en compras pasadas

Este es un sistema de recomendación. Un enfoque consiste en alimentar las compras anteriores (y otra información sobre el cliente) a una red neuronal artificial (consulte el Capítulo 10) y hacer que produzca la próxima compra más probable. Esta red neuronal normalmente se entrenaría en secuencias pasadas de compras en todos los clientes.

Construyendo un bot inteligente para un juego

Esto a menudo se aborda mediante el aprendizaje por refuerzo (RL; consulte el capítulo 18), que es una rama del aprendizaje automático que capacita a los agentes (como los bots) para elegir las acciones que maximizarán sus recompensas con el tiempo (por ejemplo, un bot puede obtener una recompensa cada vez que el jugador pierde algunos puntos de vida), dentro de un entorno determinado (como el juego). El famoso programa AlphaGo que venció al campeón mundial en el juego de Go fue construido usando RL.

Esta lista podría seguir y seguir, pero es de esperar que le dé una idea de la increíble amplitud y complejidad de las tareas que puede abordar el aprendizaje automático y los tipos de técnicas que usaría para cada tarea.

Tipos de sistemas de aprendizaje automático

Ahí Hay tantos tipos diferentes de sistemas de aprendizaje automático que es útil clasificarlos en categorías amplias, según los siguientes criterios:

Si están capacitados o no con supervisión humana (supervisados, no supervisados, semisupervisados y aprendizaje reforzado)

Si pueden aprender o no de forma incremental sobre la marcha (aprendizaje en línea o por lotes)

Ya sea que funcionen simplemente comparando nuevos puntos de datos con puntos de datos conocidos o, en cambio, detectando patrones en los datos de entrenamiento y construyendo un modelo predictivo, al igual que lo hacen los científicos (aprendizaje basado en instancias versus aprendizaje basado en modelos)

Estos criterios no son exclusivos; puedes combinarlos de la forma que quieras. Por ejemplo, un filtro de spam de última generación puede aprender sobre la marcha utilizando un modelo de red neuronal profunda entrenado con ejemplos de spam y ham; esto lo convierte en un sistema de aprendizaje supervisado, basado en modelos y en línea.

Veamos cada uno de estos criterios un poco más de cerca.

Aprendizaje supervisado / no supervisado

Los sistemas de aprendizaje automático se pueden clasificar según la cantidad y el tipo de supervisión que reciben durante el entrenamiento. Hay cuatro categorías principales: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semisupervisado y aprendizaje reforzado .

Aprendizaje supervisado

En el aprendizaje supervisado ,El conjunto de entrenamiento que alimenta al algoritmo incluye las soluciones deseadas, llamadas etiquetas (Figura 1-5).

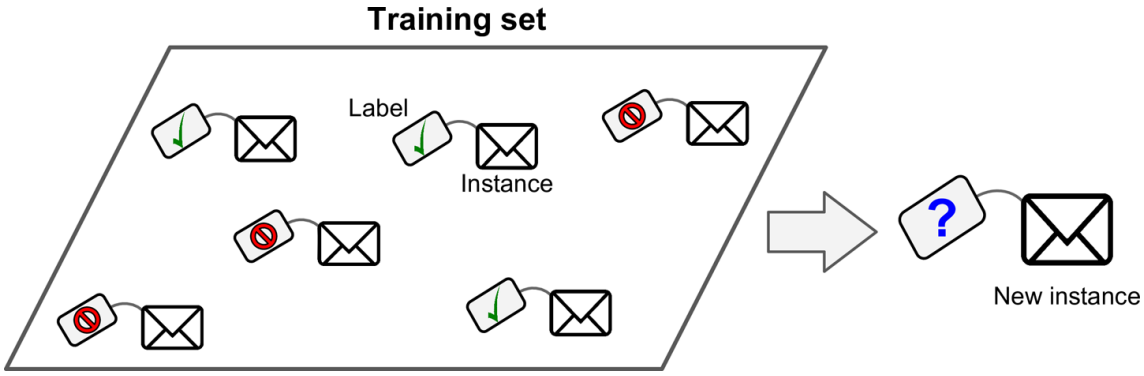


Figura 1-5. Un conjunto de formación etiquetado para la clasificación de spam (un ejemplo de aprendizaje supervisado)

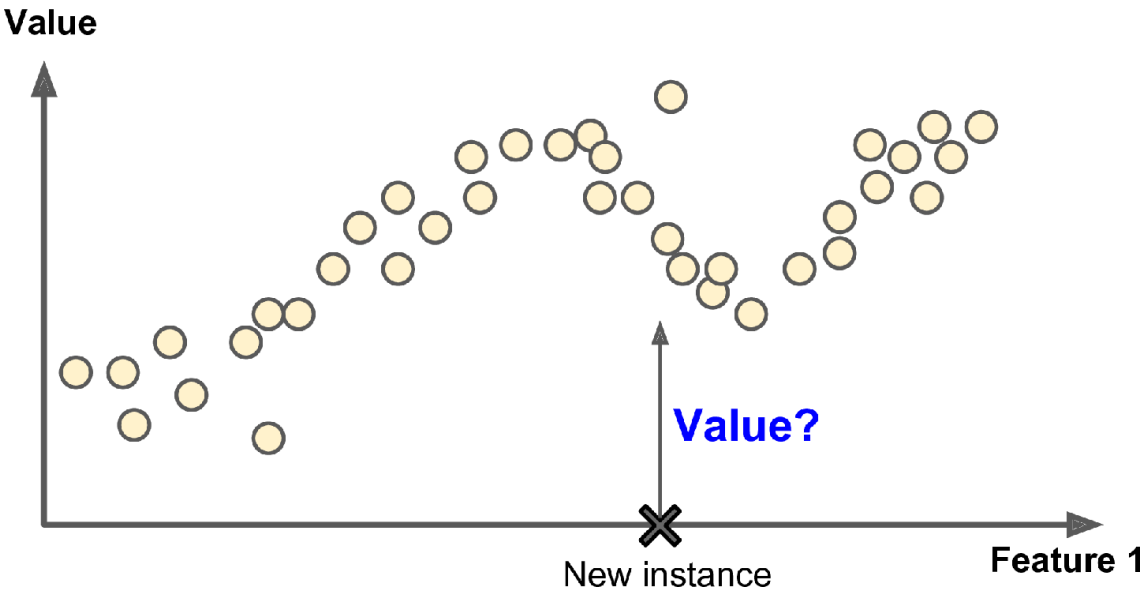
UNALa tarea típica de aprendizaje supervisado es la clasificación . El filtro de spam es un buen ejemplo de esto: está entrenado con muchos correos electrónicos de ejemplo junto con su clase (spam o ham), y debe aprender a clasificar nuevos correos electrónicos.

OtroLa tarea típica es predecir un valor numérico objetivo , como el precio de un automóvil, dado un conjunto de características (kilometraje, edad, marca, etc.) llamadas predictores . EstaEste tipo de tarea se llama regresión (Figura 1-6). 1 Para entrenar el sistema, debe proporcionarle muchos ejemplos de automóviles, incluidos sus predictores y sus etiquetas (es decir, sus precios).

NOTA

En la máquinaAprender un atributo es un tipo de datos (por ejemplo, "kilometraje"), mientras que una característica tiene varios significados, según el contexto, pero generalmente significa un atributo más su valor (por ejemplo, "kilometraje = 15.000"). Mucha gente usa las palabras atributo y función indistintamente.

Notaque algunos algoritmos de regresión también se pueden utilizar para la clasificación, y viceversa. Por ejemplo, la 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).



mls2 0106

Figura 1-6. Un problema de regresión: predice un valor, dada una característica de entrada (generalmente hay múltiples características de entrada y, a veces, múltiples valores de salida)

aquí son algunos de los algoritmos de aprendizaje supervisado más importantes (tratados en este libro):

k-Vecinos más cercanos

Regresión lineal

Regresión logística

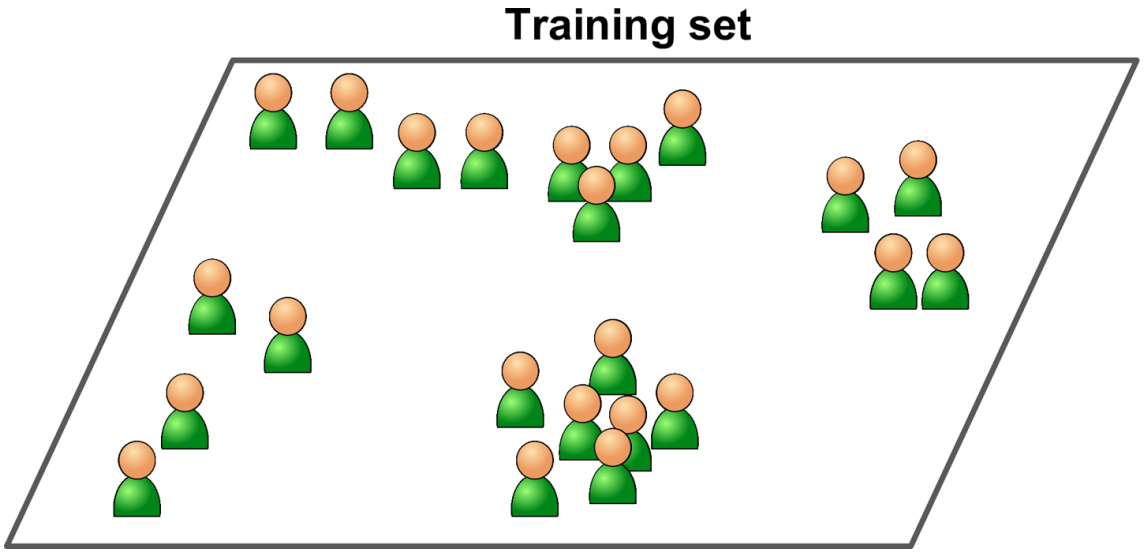
Máquinas de vectores de soporte (SVM)

Árboles de decisión y bosques aleatorios

Redes neuronales 2

Aprendizaje sin supervisión

En el aprendizaje no supervisado , como ustedpodría adivinar, los datos de entrenamiento no están etiquetados (Figura 1-7). El sistema intenta aprender sin un maestro.



aquíson algunos de los algoritmos de aprendizaje no supervisado más importantes (la mayoría de ellos se tratan en los capítulos 8 y 9):

Clustering

K-medias

DBSCAN

Análisis jerárquico de conglomerados (HCA)

Detección de anomalías y detección de novedades

SVM de una clase

Bosque de aislamiento

Reducción de visualización y dimensionalidad

Análisis de componentes principales (PCA)

Kernel PCA

Incrustación localmente lineal (LLE)

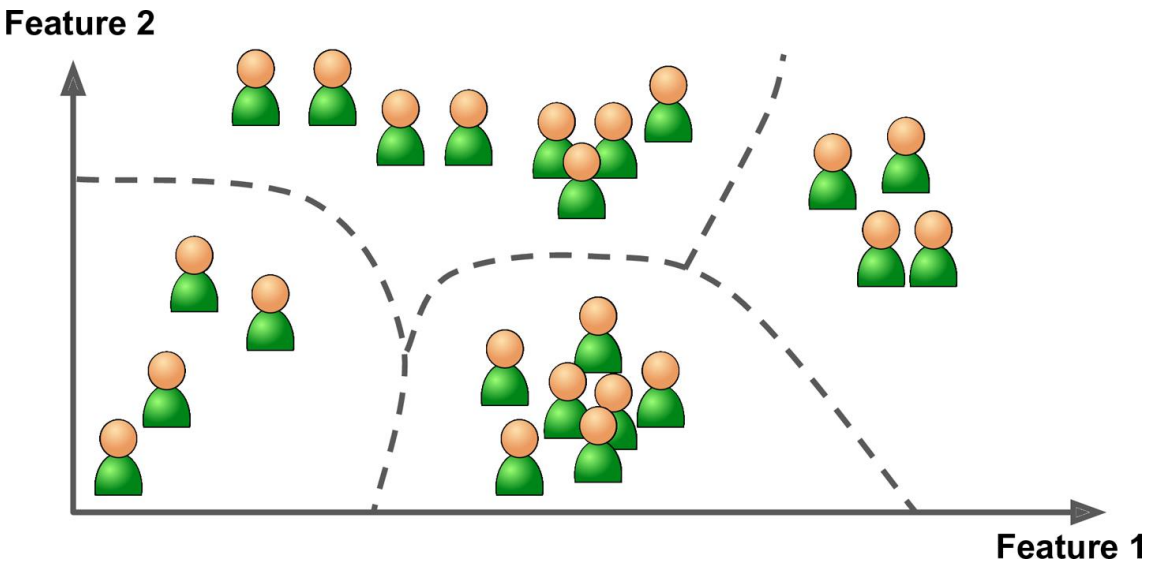
Incorporación de vecinos estocásticos distribuidos en t (t-SNE)

Aprendizaje de reglas de asociación

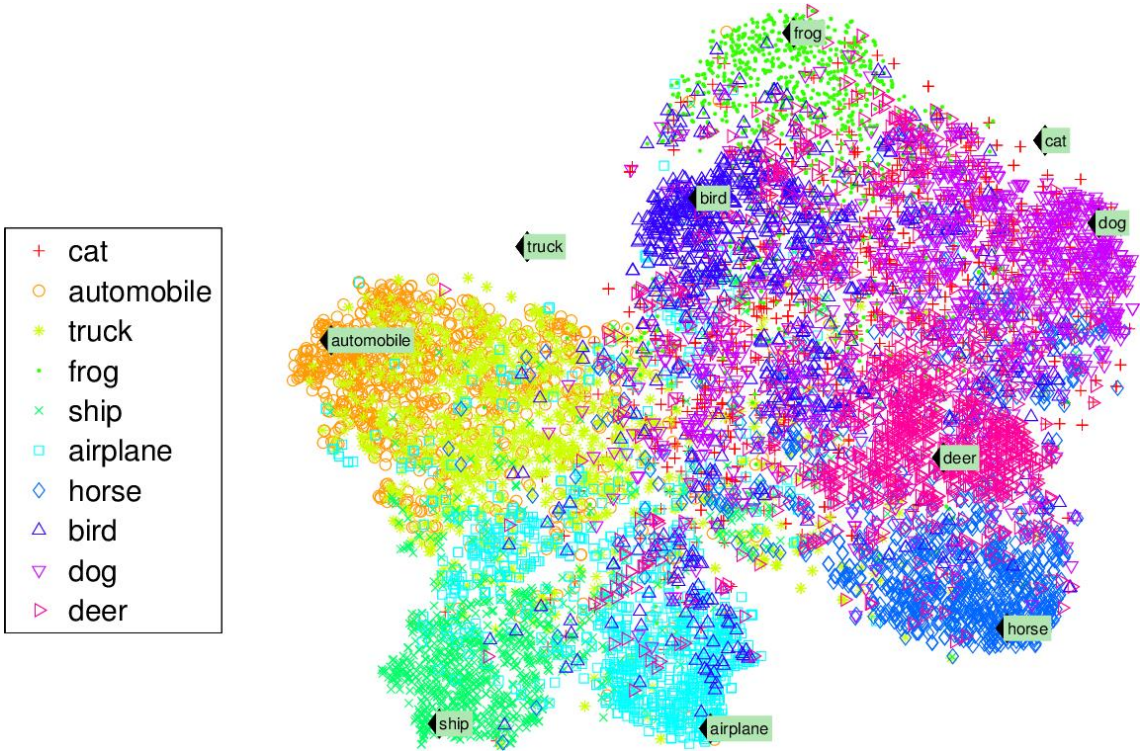
A priori

Brillo

porPor ejemplo, supongamos que tiene muchos datos sobre los visitantes de su blog. Es posible que desee ejecutar un algoritmo de agrupación en clústeres 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 los cómics 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. Si tuUtilice un algoritmo de agrupamiento jerárquico , también puede subdividir cada grupo en grupos más pequeños. Esto puede ayudarlo a orientar sus publicaciones para cada grupo.



Algoritmos de visualizacióntambién son buenos ejemplos de algoritmos de aprendizaje no supervisados: les proporciona una gran cantidad de datos complejos y sin etiquetar, y generan una representación 2D o 3D de sus datos que puede trazarse fácilmente (Figura 1-9). Estos algoritmos intentan preservar toda la estructura posible (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.



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

PROPINA

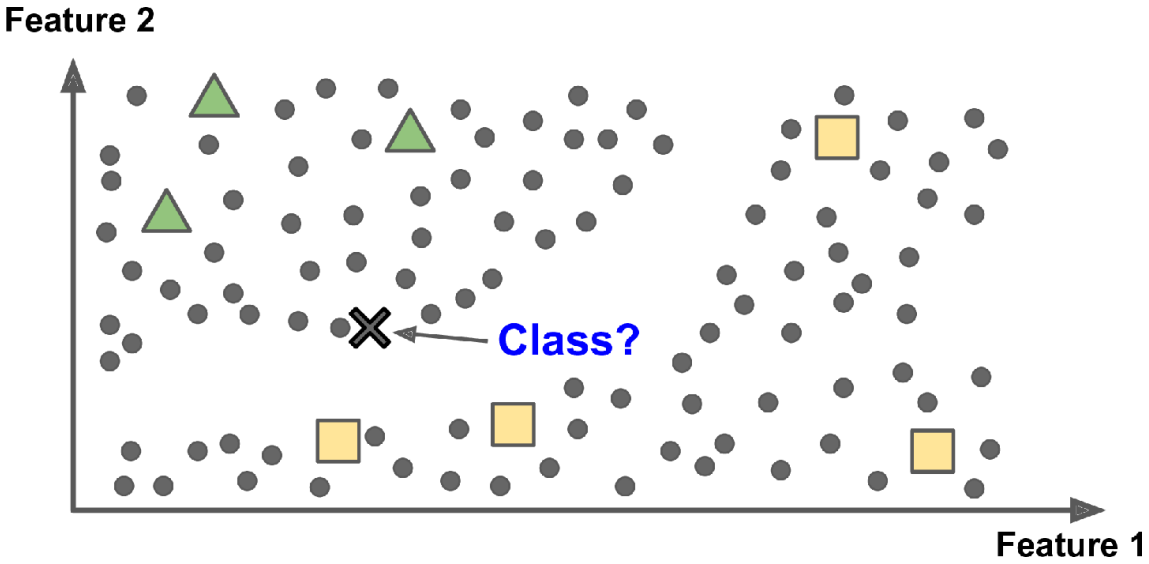
A menudo, es una buena idea intentar reducir la dimensión de sus datos de entrenamiento utilizando un algoritmo de reducción de dimensionalidad antes de alimentarlo a otro algoritmo de aprendizaje automático (como un algoritmo de aprendizaje supervisado). Se ejecutará mucho más rápido, los datos ocuparán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

TodavíaOtra tarea importante sin supervisión es la detección de anomalías, por ejemplo, detectar transacciones inusuales de tarjetas de crédito para evitar fraudes, detectar defectos de fabricación o eliminar automáticamente los valores atípicos de un conjunto de datos antes de enviarlos a otro algoritmo de aprendizaje. El sistema se muestra principalmente en instancias normales durante el entrenamiento, por lo que aprende a reconocerlas; luego, cuando ve una nueva instancia, puede decir si se ve como una normal o si es probable que sea una anomalía (vea la Figura 1-10). Una tarea muy similares la detección de novedades : tiene como objetivo detectar nuevas instancias que se vean diferentes de todas las instancias en el conjunto de entrenamiento. Esto requiere tener un conjunto de entrenamiento muy "limpio", desprovisto de cualquier instancia que le gustaría que detecte el algoritmo. Por ejemplo, si tiene miles de imágenes de perros, y el 1% de estas imágenes representan chihuahuas, entonces un algoritmo de detección de novedades no debería tratar las nuevas imágenes de chihuahuas como novedades. Por otro lado, los algoritmos de detección de anomalías pueden considerar a estos perros como tan raros y tan diferentes de otros perros que probablemente los clasificarían como anomalías (sin ofender a los chihuahuas).

Finalmente, otroUna tarea común sin supervisión es el 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

Desde el etiquetadoPor lo general, los datos requieren mucho tiempo y son costosos, a menudo tendrá muchas instancias sin etiquetar y pocas instancias etiquetadas. Algunos algoritmos pueden manejar datos que están parcialmente etiquetados. A esto se le llama aprendizaje semisupervisado (Figura 1-11).



mls2 0111

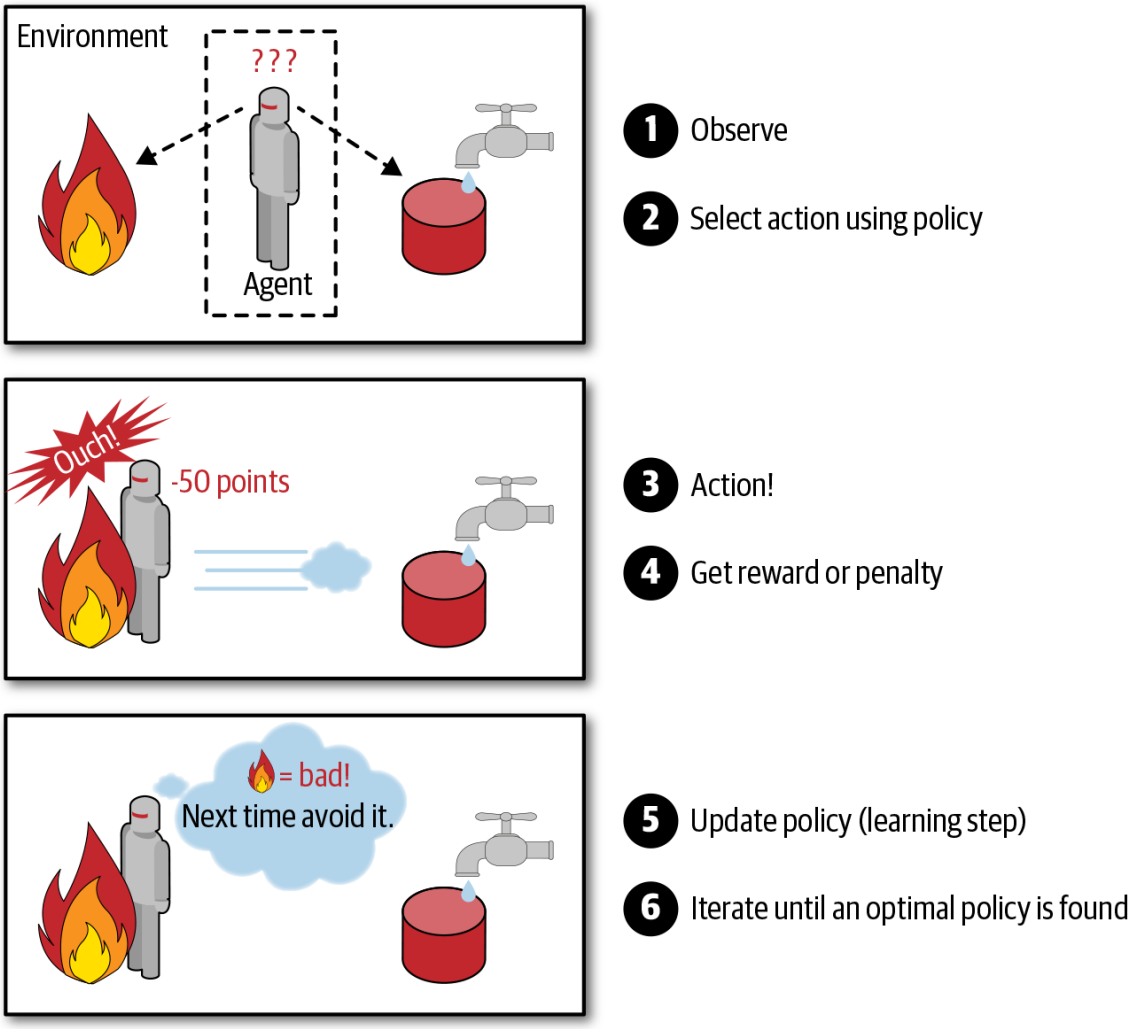
Figura 1-11. Aprendizaje semisupervisado con dos clases (triángulos y cuadrados): los ejemplos sin etiquetar (círculos) ayudan a clasificar una nueva instancia (la cruz) en la clase triángulo en lugar de la clase cuadrada, aunque está más cerca de los cuadrados etiquetados

AlgunosLos 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. Simplemente agregue una etiqueta por persona 4 y podrá nombrar a todos en cada foto, lo cual es útil para buscar fotos.

La mayoría de los algoritmos de aprendizaje semisupervisados son combinaciones de algoritmos supervisados y no supervisados. porejemplo, las redes de creencias profundas (DBN) se basan en componentes no supervisadosllamadas máquinas de Boltzmann restringidas (RBM) apiladas una encima de la otra. Los GBR se entrenan secuencialmente de manera no supervisada, y luego todo el sistema se ajusta utilizando técnicas de aprendizaje supervisado.

Aprendizaje reforzado

El aprendizaje por refuerzo esuna 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 penalizaciones en forma de recompensas negativas, como se muestra en la Figura 1-12). Luego, 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.



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 mayo de 2017 cuando venció al campeón mundial Ke Jie en el juego de Go. 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 el aprendizaje por lotes , elEl 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. Esto esllamado aprendizaje fuera de línea .

Si desea que un sistema de aprendizaje por lotes conozca datos nuevos (como un nuevo tipo de spam), 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 la Figura 1-3), por lo que incluso un 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 debe transportar grandes cantidades de datos de entrenamiento y consumir 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 el aprendizaje en línea , entrenar el sistema de manera incremental alimentándolo con instancias de datos secuencialmente, ya sea individualmente o en grupos pequeñosllamados 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 la Figura 1-13).

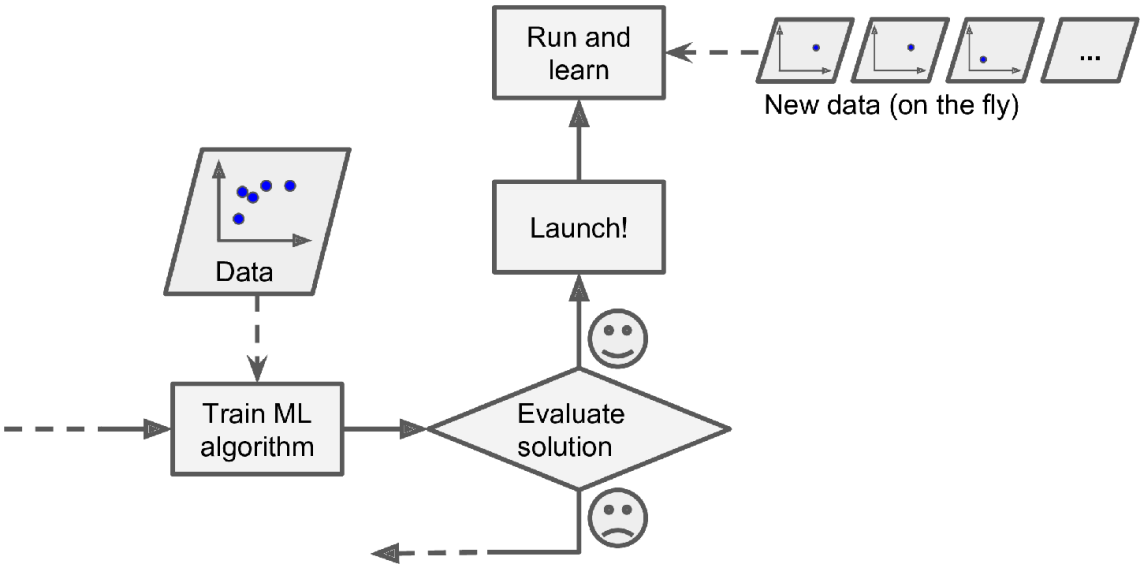


Figura 1-13. En el aprendizaje en línea, un modelo se capacita y se lanza a la producción, y luego sigue aprendiendo a medida que ingresan nuevos datos.

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 al cambio de manera rápida o autónoma. También es una buena opción 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 una estado y "reproducir" los datos). Esto puede ahorrar una gran cantidad de espacio.

En líneaLos algoritmos de aprendizaje también se pueden usar para entrenar sistemas en enormes conjuntos de datos que no pueden caber en la memoria principal de una máquina (esto se llama aprendizaje fuera del núcleo). 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 la Figura 1-14).

ADVERTENCIA

Fuera de núcleoEl aprendizaje generalmente se realiza sin conexión (es decir, no en el sistema en vivo), por lo que el aprendizaje en línea puede ser un nombre confuso. Piense en ello como un aprendizaje incremental .

Un parámetro importante de los sistemas de aprendizaje en línea es qué tan rápido deben adaptarse a los datos cambiantes: estose llama 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 (valores atípicos).

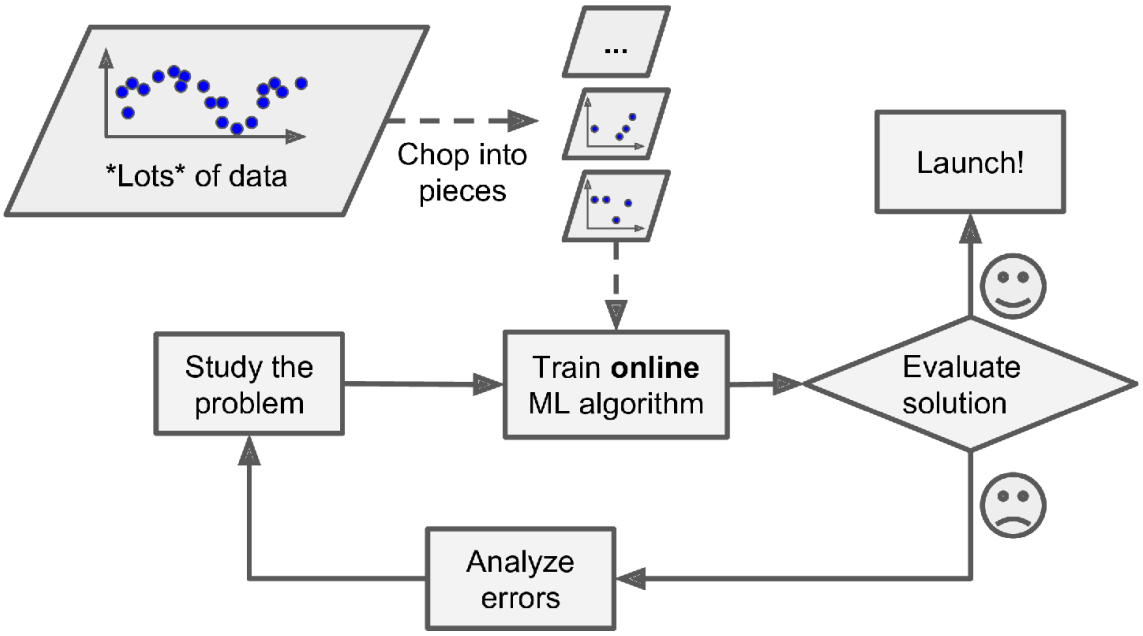


Figura 1-14. Usar el aprendizaje en línea para manejar grandes conjuntos de datos

Un gran desafío con el aprendizaje en línea es que si se ingresan datos incorrectos al sistema, el rendimiento del sistema disminuirá gradualmente. Si es 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 los resultados de búsqueda. Para reducir este riesgo, debe monitorear su sistema de cerca y desactivar rápidamente el aprendizaje (y posiblemente volver a un estado de trabajo anterior) si detecta una caída en el rendimiento. También puede querer monitorear los datos de entrada y reaccionar ante datos anormales (por ejemplo, usando 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 cómo se generalizan . La mayoría de las tareas de aprendizaje automático tratan de hacerpredicciones. Esto significa que, dados una serie de ejemplos de entrenamiento, el sistema debe poder hacer buenas predicciones para (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

Posiblementela forma más trivial de aprendizaje es simplemente aprender de memoria. Si creara un filtro de spam de esta manera, simplemente 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. Estarequiere una medida de similitud entre dos correos electrónicos. Una medida de similitud (muy básica) entre dos correos electrónicos podría ser contar la cantidad 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.

Esto se llama aprendizaje basado en instancias : el sistema aprende los ejemplos de memoria, luego generaliza a nuevos casos usando una medida de similitud para compararlos con los ejemplos aprendidos (o un subconjunto de ellos). Por ejemplo, en la Figura 1-15, la nueva instancia se clasificaría como un triángulo porque la mayoría de las instancias más similares pertenecen a esa clase.

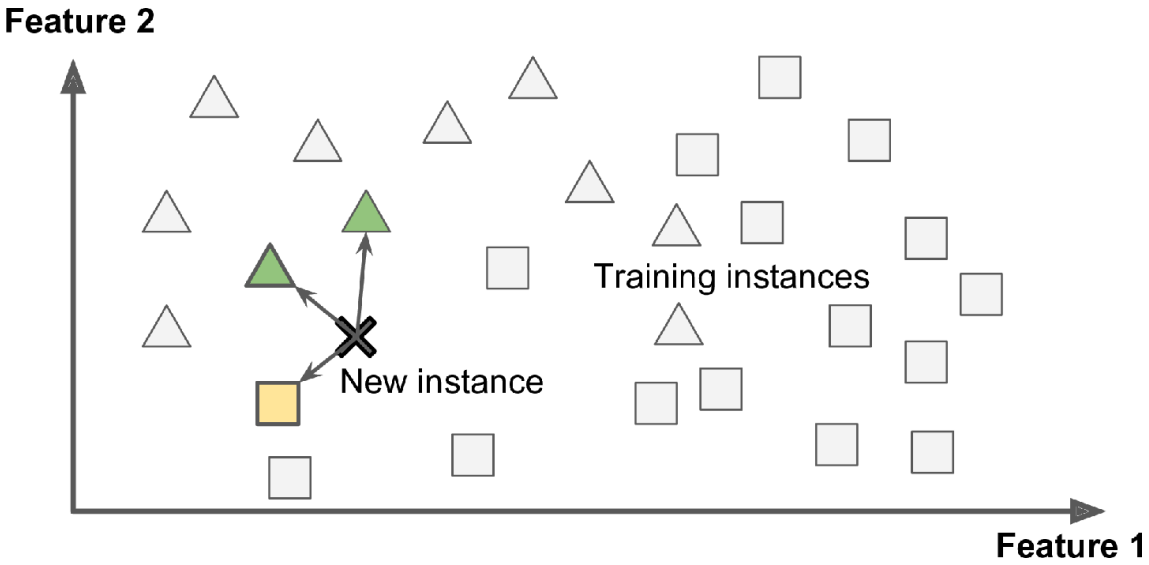
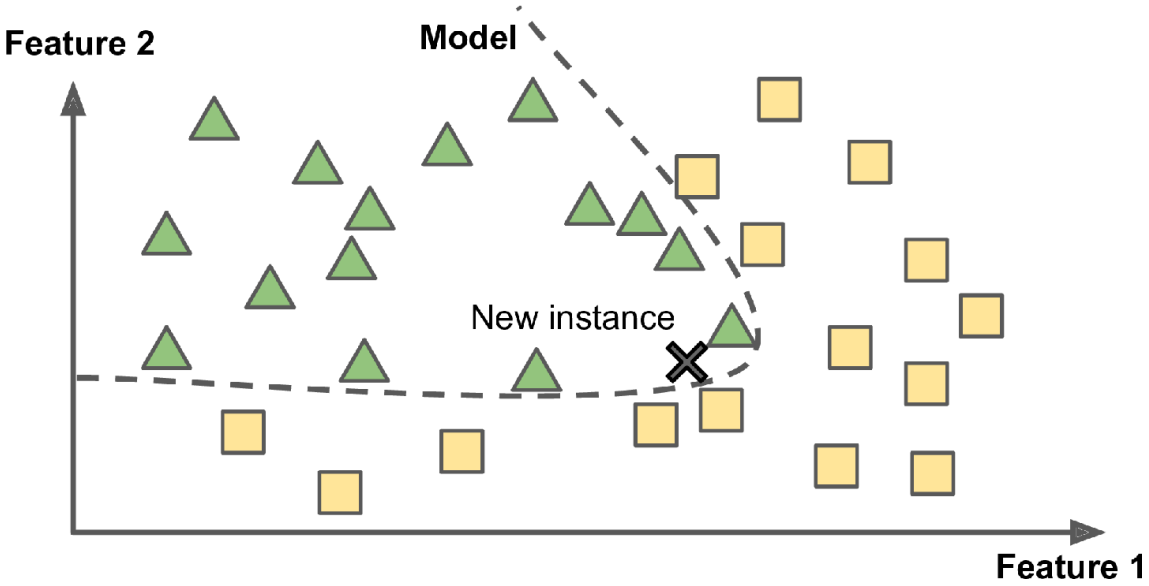


Figura 1-15. Aprendizaje basado en instancias

Aprendizaje basado en modelos

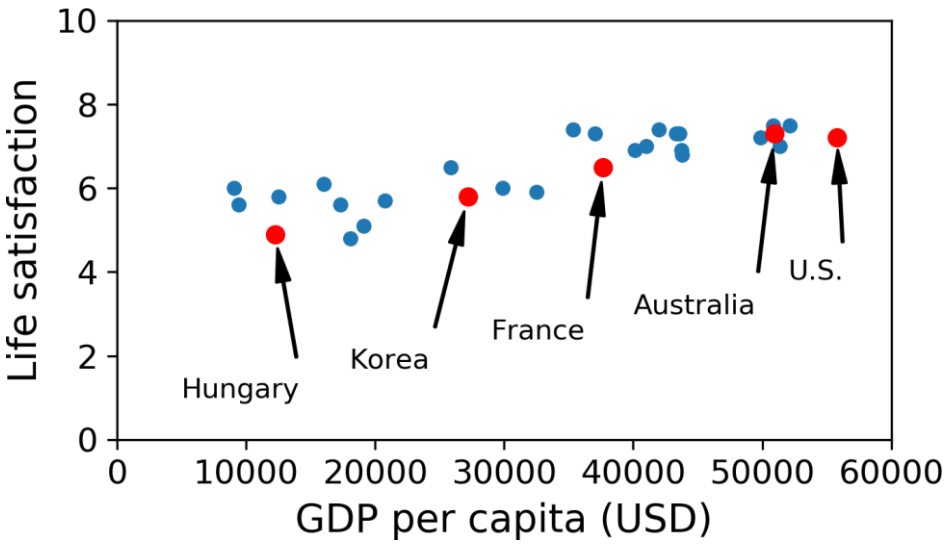
OtroUna forma de generalizar a partir de un conjunto de ejemplos es construir un modelo de estos ejemplos y luego usar ese modelo para hacer predicciones . A esto se le llama aprendizaje basado en modelos (Figura 1-16).



Por ejemplo, supongadesea saber si el dinero hace feliz a la gente, por lo que descarga los datos del Índice de Vida Mejor del sitio web de la OCDE y las estadísticas sobre el producto interno bruto (PIB) per cápita del sitio web del FMI . Luego se une a las tablas y ordena por PIB per cápita. La 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 de 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

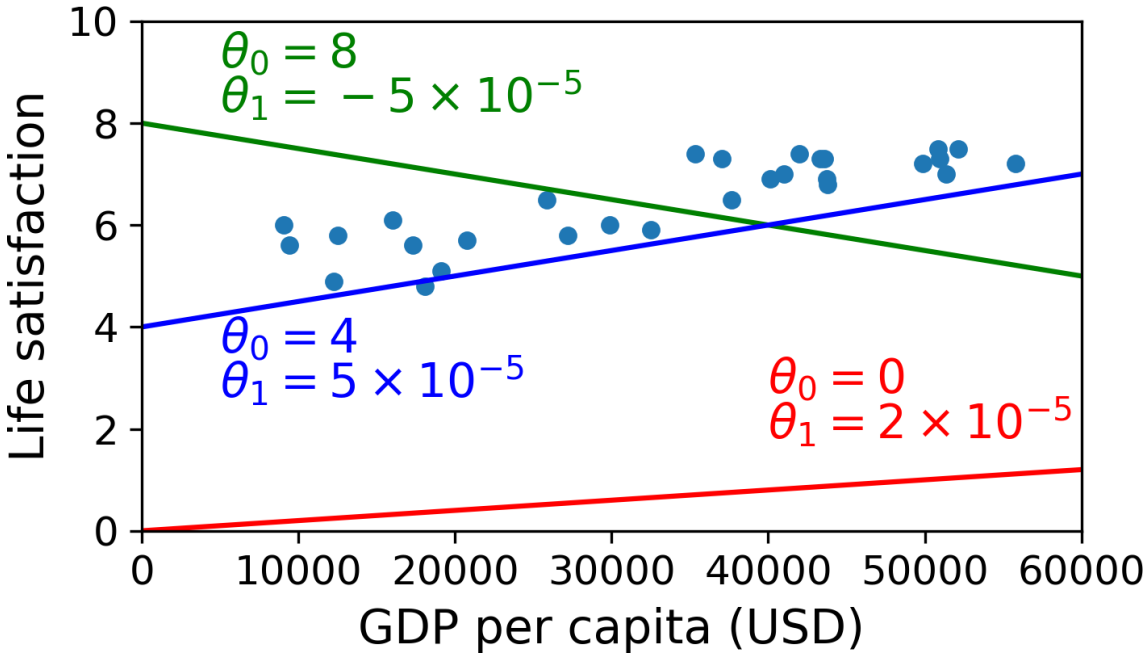


¡Parece haber una tendencia aquí! A pesar de quelos datos son ruidosos (es decir, en parte aleatorios), parece que la satisfacción con la vida aumenta más o menos linealmente a medida que aumenta el PIB per cápita del país. Entonces decide modelar la satisfacción con la vida como una función lineal del PIB per cápita. EstaEste paso se llama selección de modelo : seleccionó un modelo lineal de satisfacción con la vida con un solo atributo, el PIB per cápita (Ecuación 1-1).

Ecuación 1-1. Un modelo lineal simple

satisfacción de vida = θ 0 + θ 1 × PIB per cápita

Estamodelo tiene dos parámetros de modelo , θ 0 y θ 1 . 5 Al ajustar estos parámetros, puede hacer que su modelo represente cualquier función lineal, como se muestra en la Figura 1-18 .



Antes de que pueda usar 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 esta pregunta, debe especificar una medida de desempeño. Tu también puedesdefina una función de utilidad (o función de aptitud) que mida qué tan bueno es su modelo, o puede definir una función de costo que mida qué tan malo es. Para los problemas de regresión lineal, las personas suelen utilizar una función de coste que mide la distancia entre las predicciones del modelo lineal y los ejemplos de entrenamiento; el objetivo es minimizar esta distancia.

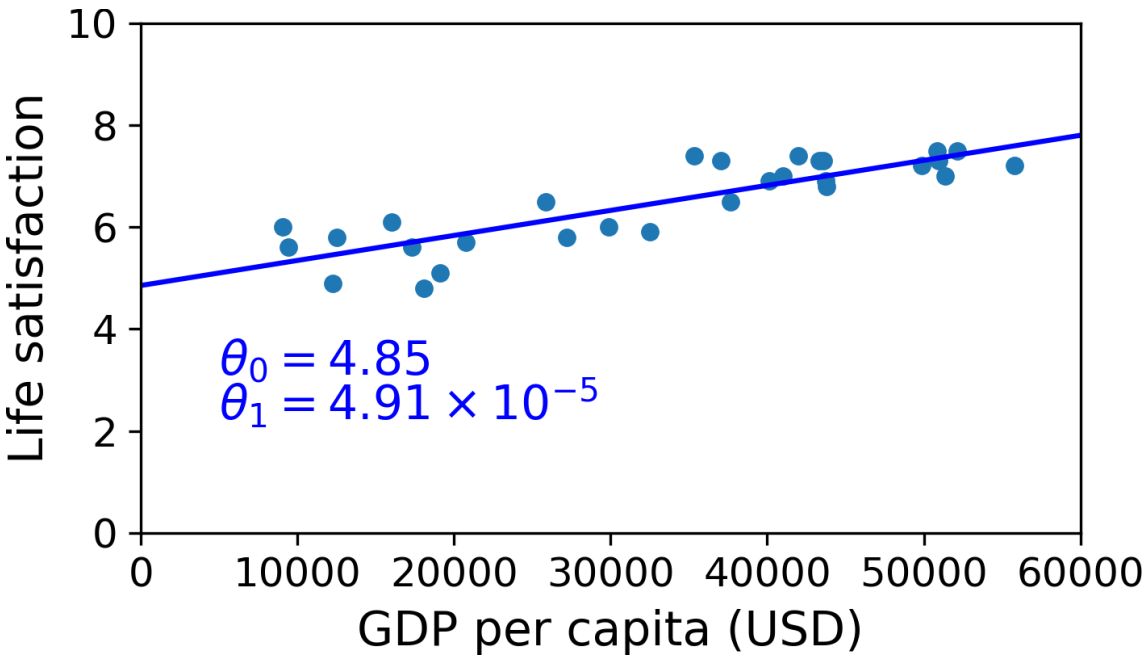
Aquí es donde entra en juego el algoritmo de regresión lineal: lo alimenta con sus ejemplos de entrenamiento y encuentra los parámetros que hacen que el modelo lineal se ajuste mejor a sus datos. Estase llama entrenar el modelo. En nuestro caso, el algoritmo encuentra que los valores óptimos de los parámetros son θ 0 = 4.85 y θ 1 = 4.91 × 10 –5 .

ADVERTENCIA

Confusamente, lo mismola palabra "modelo" puede referirse a un tipo de modelo (por ejemplo, Regresión lineal),a una arquitectura de modelo completamente especificada (por

ejemplo, regresión lineal con una entrada y una salida), oal modelo entrenado final listo para usarse en predicciones (por ejemplo, Regresión lineal con una entrada y una salida, usando θ 0 = 4.85 y θ 1 = 4.91 × 10 –5). La selección del modelo consiste en elegir el tipo de modelo y especificar completamente su arquitectura. Entrenar un modelo significa ejecutar un algoritmo para encontrar los parámetros del modelo que harán que se ajuste mejor a los datos de entrenamiento (y, con suerte, hacer buenas predicciones sobre nuevos datos).

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



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 la satisfacción con la vida es probable que sea alrededor de 4.85 + 22.587 × 4.91 × 10 -5 = 5.96.

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

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

```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import sklearn.linear_model


# Load the data
```

```
oecd_bli = pd.read_csv("oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv("gdp_per_capita.csv",thousands=',',delimiter='\t',
                             encoding='latin1', na_values="n/a")



# Prepare the data
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Visualize the data
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Select a linear model
model = sklearn.linear_model.LinearRegression()

# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[22587]]  # Cyprus's GDP per capita
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```

NOTA

Si hubiera usado unEn su lugar, con un algoritmo de aprendizaje basado en instancias, 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, habría predicho una satisfacción con la vida de 5.7 para Chipre. Si te alejas un poco y miras los dos países siguientes más cercanos, encontrarás 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. EstaEl algoritmo simple se llama regresión k-Vecinos más cercanos (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 estas dos líneas:

```
import sklearn.linear_model
model = sklearn.linear_model.LinearRegression()
```

con estos dos:

```
import sklearn.neighbors
model = sklearn.neighbors.KNeighborsRegressor(
    n_neighbors=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 datos de capacitación o de mejor 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 tuaplicó el modelo para hacer predicciones sobre 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 el Capítulo 2 , experimentará 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 se ve 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

EnEn 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.

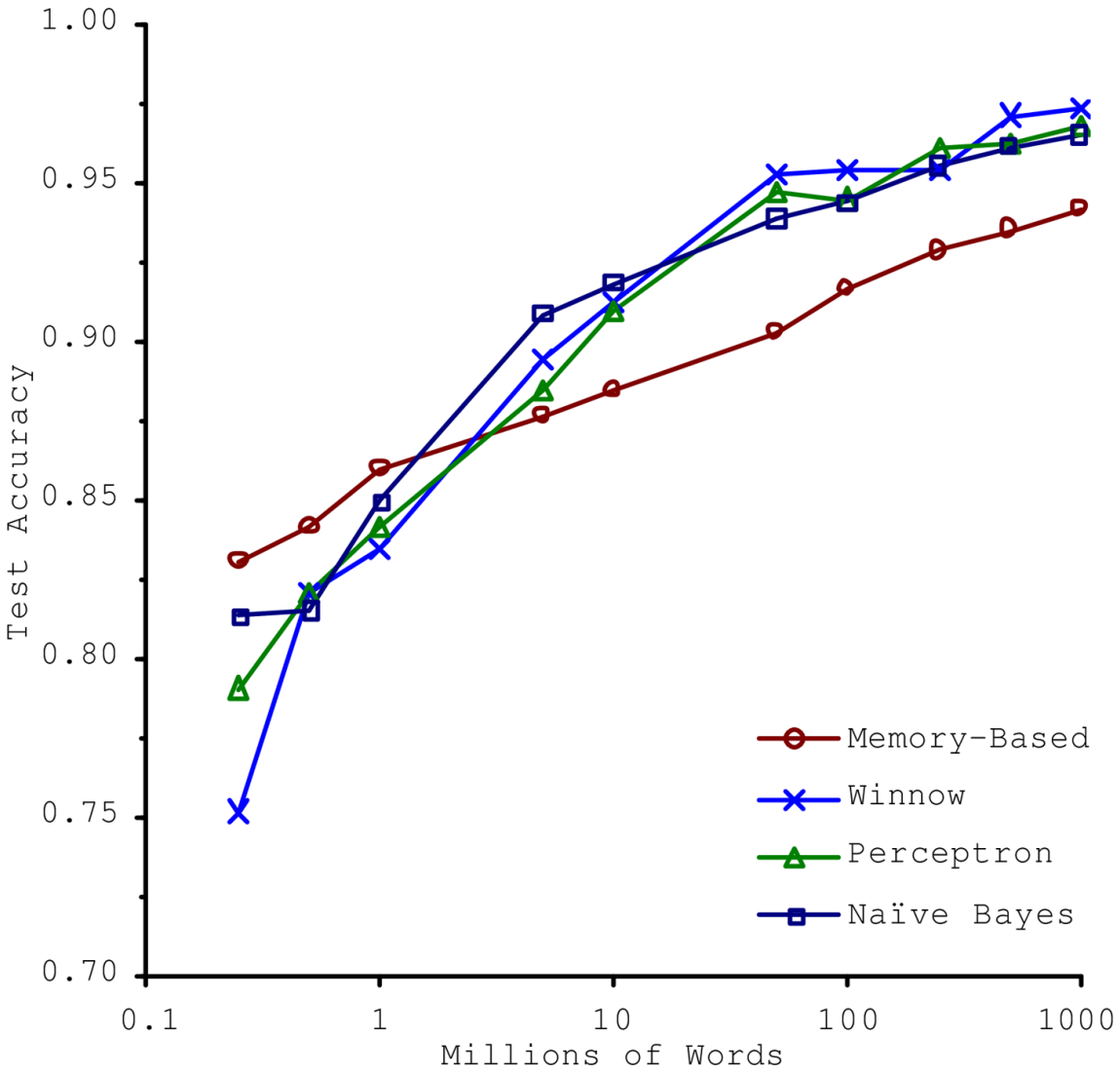
Cantidad insuficiente de datos de entrenamiento

porun niño pequeño para aprender 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 de 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 famoso artículo publicadoEn 2001, los investigadores de Microsoft Michele Banko y Eric Brill demostraron que algoritmos de aprendizaje automático muy diferentes, incluidos los bastante simples, se desempeñaban casi de manera idéntica en un problema complejo de desambiguación del lenguaje natural 8 una vez que se les daban suficientes datos (como puede ver en la Figura 1 -20).



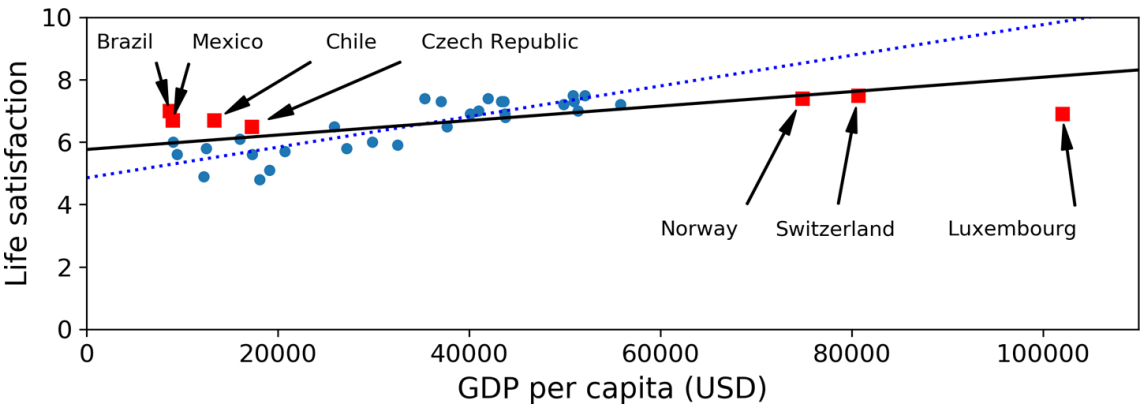
Como Los autores lo expresan, "estos resultados sugieren que es posible que deseemos reconsiderar el compromiso entre gastar tiempo y dinero en el desarrollo de algoritmos versus gastarlo en el desarrollo de corpus".

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 artículo titulado "La eficacia irrazonable de los datos" , publicado en 2009. 10 Sin embargo, cabe señalar que los conjuntos de datos pequeños y medianos siguen siendo muy comunes, y no siempre es fácil o barato obtener datos de formación adicionales , así que no abandone los algoritmos todavía.

Datos de formación no representativos

EnPara 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. La Figura 1-21 muestra cómo se ven los datos cuando agrega los países que faltan.



mls2 0121

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 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,tienen ruido de muestreo (es decir, datos no representativos como resultado de la casualidad), pero incluso muestras muy grandes pueden ser no representativas si el método de muestreo es defectuoso. Estase llama sesgo de muestreo .

EJEMPLOS DE SESGO DE MUESTREO

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: el Literary Digest realizó una encuesta muy grande, enviando correo a aproximadamente 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. La falla estaba en el método de muestreo del Literary Digest :

En primer lugar, para obtener las direcciones a las que enviar las urnas, el Literary Digest utilizó directorios telefónicos, listas de suscriptores de revistas, listas de miembros de clubes, etc.

Todas estas listas tendían a favorecer a las personas más ricas, que tenían más probabilidades de votar por los republicanos (de ahí Landon).

En segundo lugar, menos del 25% de las personas encuestadas respondieron. Una vez más, esto introdujo un sesgo de muestreo, al descartar potencialmente a las personas a las que no les importaba mucho la política, las personas a las que no les gustaba el Literary Digest y otros grupos clave. Este es un tipo especial de sesgo de muestreo llamado sesgo por falta de respuesta .

Aquí hay otro ejemplo: digamos que desea crear un sistema para reconocer videos musicales 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 de “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 entrenamientoestá lleno de errores, valores atípicos y ruido (por ejemplo, debido a mediciones de baja calidad), hará que sea 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. A continuación, se muestran algunos ejemplos de cuándo querría limpiar los datos de entrenamiento:

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 mediana age), o entrene un modelo con la función y un modelo sin ella.

Características irrelevantes

Comodice 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. EstaEl proceso, llamado ingeniería de características , implica los siguientes pasos:

Selección de funciones (seleccionando las funciones más útiles para entrenar entre las funciones existentes)

Extracción de características (combinando 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

Decirestás visitando un país extranjero y el taxista te estafa. Puede sentirse tentado a decir que 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 denomina sobreajuste : significa que el modelo se desempeña bien en los datos de entrenamiento, pero no se generaliza bien.

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

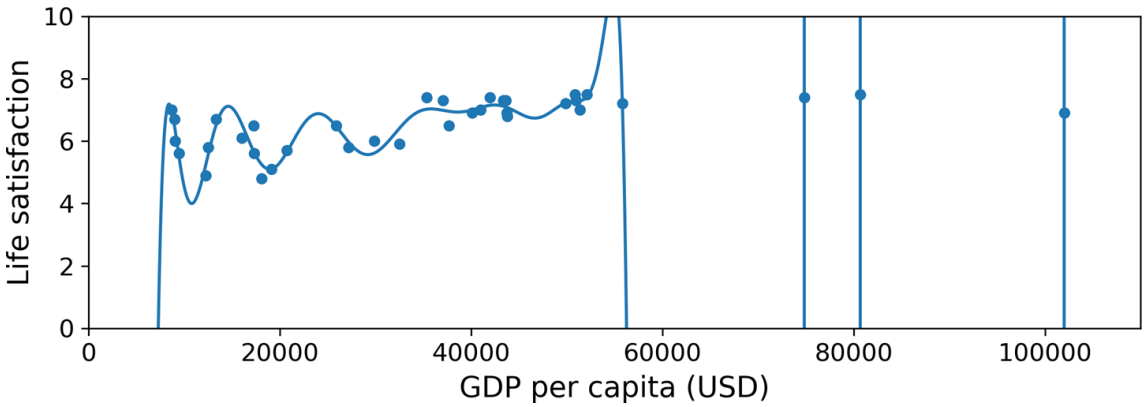


Figura 1-22. Sobreajuste de los datos de entrenamiento

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 nuevos casos. 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 una w en su nombre tienen una satisfacción con la vida mayor que 7:

Nueva Zelanda (7.3), Noruega (7.4), Suecia (7.2), y Suiza (7,5). ¿Qué tan seguro estás de que el w-¿La regla de satisfacción 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.

ADVERTENCIA

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

Simplifique 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.

Reúna más datos de entrenamiento.

Reducir el ruido en los datos de entrenamiento (por ejemplo, corregir errores de datos y eliminar valores atípicos).

Restringirun 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) como la pendiente (θ 1) de la línea. Si forzamos θ 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 tendrá efectivamente entre uno y dos grados de libertad. Producirá un modelo más simple que uno con dos grados de libertad, pero más complejo que uno con solo uno. Desea encontrar el equilibrio adecuado entre ajustar los datos de entrenamiento a la perfección y mantener el modelo lo suficientemente simple como para asegurarse de que se generalice bien.

La figura 1-23 muestra tres modelos. La línea punteada representa el modelo original que se entrenó en los países representados como círculos (sin los países representados como cuadrados), la línea punteada es nuestro segundo modelo entrenado con todos los países (círculos y cuadrados) y la línea continua es un modelo 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: este modelo no se ajusta a los datos de entrenamiento (círculos) tan bien como al primer modelo, pero en realidad se generaliza mejor a nuevos ejemplos que no vio durante el entrenamiento (cuadrados) .

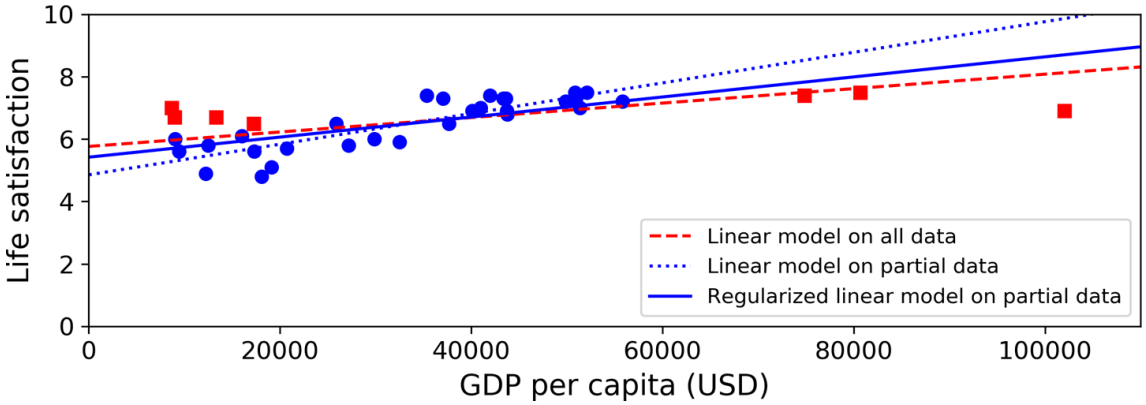


Figura 1-23. La regularización reduce el riesgo de sobreajuste

losLa cantidad de regularización a aplicar durante el aprendizaje se puede controlar mediante un hiperparámetro . 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).

Adecuación de los datos de entrenamiento

ComoPuede adivinar que el desajuste es lo opuesto al 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 más compleja que el modelo, por lo que sus predicciones seguramente serán inexactas, incluso en los ejemplos de entrenamiento .

Estas son las principales opciones para solucionar este problema:

Seleccione un modelo más potente, con más parámetros.

Proporcione mejores funciones al algoritmo de aprendizaje (ingeniería de funciones).

Reducir las limitaciones del modelo (por ejemplo, reducir el hiperparámetro de regularización).

Dar un paso atrás

Porahora 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 :

El aprendizaje automático consiste en hacer que las máquinas mejoren en alguna tarea aprendiendo de los datos, en lugar de tener que codificar reglas explícitamente.

Hay muchos tipos diferentes de sistemas de AA: supervisados o no, por lotes o en línea, basados en instancias o basados en modelos.

En un proyecto de AA, recopila datos en un conjunto de entrenamiento y alimenta el conjunto de entrenamiento a un algoritmo de aprendizaje. Si el algoritmo está basado en modelos, sintoniza algunos parámetros para ajustar el modelo al conjunto de entrenamiento (es decir, para hacer buenas predicciones sobre el conjunto de entrenamiento en sí), y luego, con suerte, también podrá hacer buenas predicciones en casos nuevos. . Si el algoritmo está basado en instancias, simplemente aprende los ejemplos de memoria y los generaliza a nuevas instancias usando una medida de similitud para compararlos con las instancias aprendidas.

El sistema no funcionará bien si su conjunto de entrenamiento es demasiado pequeño, o si los datos no son representativos, son ruidosos o están contaminados con características irrelevantes (basura dentro, basura fuera). Por último, su modelo no debe ser ni demasiado simple (en cuyo caso no encajará) ni demasiado complejo (en cuyo caso se sobreajustará).

Solo hay un último tema importante que cubrir: una vez que haya entrenado un modelo, no querrá simplemente "esperar" que se generalice a nuevos casos. Quiere evaluarlo y ajustarlo si es necesario. Veamos cómo hacer eso.

Prueba y validación

losLa única forma de saber qué tan bien se generalizará un modelo a casos nuevos es probarlo en casos nuevos. Una forma de hacerlo es poner su modelo en producción y controlar su rendimiento. Esto funciona bien, pero si su modelo es terriblemente malo, sus usuarios se quejarán, no es la mejor idea.

UNALa mejor opción es dividir sus datos en dos conjuntos: el conjunto de entrenamiento y el conjunto de prueba . Como indican estos nombres, entrena su modelo con el conjunto de entrenamiento y lo prueba con el conjunto de prueba. La tasa de erroren casos nuevos se denomina error de generalización (o error fuera de la muestra) y, al evaluar su modelo en el conjunto de prueba, obtiene una estimación de este error. Este valor le indica qué tan bien funcionará su modelo en instancias que nunca antes había visto.

Si el error de entrenamiento es bajo (es decir, su modelo comete pocos errores en el conjunto de entrenamiento) pero el error de generalización es alto, significa que su modelo está sobreajustando los datos de entrenamiento.

PROPINA

Es común usar el 80% de los datos para entrenamientoy aguante el 20% para la prueba. Sin embargo, esto depende del tamaño del conjunto de datos: si contiene 10 millones de instancias, entonces mantener el 1% significa que su conjunto de prueba contendrá 100,000 instancias, probablemente más que suficiente para obtener una buena estimación del error de generalización.

Ajuste de hiperparámetros y selección de modelos

Evaluar un modeloes bastante simple: solo use un conjunto de prueba. Pero suponga que está dudando entre dos tipos de modelos (digamos, un modelo lineal y un modelo polinomial): ¿cómo puede decidir entre ellos? Una opción es entrenar a ambos y comparar qué tan bien generalizan usando el conjunto de prueba.

Ahora suponga que el modelo lineal se generaliza mejor, pero desea aplicar cierta regularización para evitar el sobreajuste. La pregunta es, ¿cómo se elige el valor del hiperparámetro de regularización? Una opción es entrenar 100 modelos diferentes usando 100 valores diferentes para este hiperparámetro. Suponga que encuentra el mejor valor de hiperparámetro que produce un modelo con el menor error de generalización, digamos, solo un error del 5%. Lanza este modelo a producción, pero desafortunadamente no funciona tan bien como se esperaba y produce un 15% de errores. ¿Lo que acaba de suceder?

El problema es que midió el error de generalización varias veces en el conjunto de prueba y adaptó el modelo y los hiperparámetros para producir el mejor modelo para ese conjunto en particular . Esto significa que es poco probable que el modelo funcione tan bien con datos nuevos.

UNALa solución común a este problema se llama validación de exclusión : simplemente se muestra parte del conjunto de entrenamiento para evaluar varios modelos candidatos y seleccionar el mejor. El nuevo conjunto excluido se denomina conjunto de validación (o, a veces , conjunto de desarrollo o conjunto de desarrollo).). Más específicamente, entrena varios modelos con varios hiperparámetros en el conjunto de entrenamiento reducido (es decir, el conjunto de entrenamiento completo menos el conjunto de validación) y selecciona el modelo que se desempeña mejor en el conjunto de validación. Después de este proceso de validación de reserva, entrena el mejor modelo en el conjunto de entrenamiento completo (incluido el conjunto de validación), y esto le brinda el modelo final. Por último, evalúa este modelo final en el conjunto de prueba para obtener una estimación del error de generalización.

Esta solución suele funcionar bastante bien. Sin embargo, si el conjunto de validación es demasiado pequeño, las evaluaciones del modelo serán imprecisas: puede terminar seleccionando un modelo subóptimo por error. Por el contrario, si el conjunto de validación es demasiado grande, el conjunto de entrenamiento restante será mucho más pequeño que el conjunto de entrenamiento completo. ¿Por qué es esto malo? Bueno, dado que el modelo

final se entrenará en el conjunto de entrenamiento completo, no es ideal comparar modelos candidatos entrenados en un conjunto de entrenamiento mucho más pequeño. Sería como seleccionar al velocista más rápido para participar en un maratón. UnoUna forma de resolver este problema es realizar una validación cruzada repetida , utilizando muchos conjuntos de validación pequeños. Cada modelo se evalúa una vez por conjunto de validación después de que se entrena con el resto de los datos. Al promediar todas las evaluaciones de un modelo, obtiene una medida mucho más precisa de su desempeño. Sin embargo, existe un inconveniente: el tiempo de entrenamiento se multiplica por el número de conjuntos de validación.

Discrepancia de datos

EnEn algunos casos, es fácil obtener una gran cantidad de datos para entrenamiento, pero estos datos probablemente no serán perfectamente representativos de los datos que se usarán en producción. Por ejemplo, suponga que desea crear una aplicación móvil para tomar fotografías de flores y determinar automáticamente su especie. Puede descargar fácilmente millones de imágenes de flores en la web, pero no serán perfectamente representativas de las imágenes que realmente se tomarán con la aplicación en un dispositivo móvil. Quizás solo tenga 10,000 imágenes representativas (es decir, realmente tomadas con la aplicación). En este caso, la regla más importante a recordar es que el conjunto de validación y el conjunto de prueba deben ser lo más representativos posible de los datos que espera usar en producción, por lo que deben estar compuestos exclusivamente por imágenes representativas: puede mezclarlos y poner la mitad en el conjunto de validación y la mitad en el conjunto de prueba (asegurándose de que no haya duplicados o casi duplicados en ambos conjuntos). Pero después de entrenar su modelo en las imágenes web, si observa que el rendimiento del modelo en el conjunto de validación es decepcionante, no sabrá si esto se debe a que su modelo se ha sobreajustado al conjunto de entrenamiento, o si esto se debe solo a la desajuste entre las imágenes web y las imágenes de la aplicación móvil. Uno o si esto se debe simplemente a la falta de coincidencia entre las imágenes web y las imágenes de la aplicación móvil. Uno o si esto se debe simplemente a la falta de coincidencia entre las imágenes web y las imágenes de la aplicación móvil. UnoLa solución es mostrar algunas de las imágenes de entrenamiento (de la web) en otro conjunto que Andrew Ng llama el conjunto train-dev . Una vez entrenado el modelo (en el conjunto de entrenamiento, noen el conjunto train-dev), puede evaluarlo en el conjunto train-dev. Si funciona bien, entonces el modelo no se adapta demasiado al conjunto de entrenamiento. Si tiene un desempeño deficiente en el conjunto de validación, el problema debe provenir de la falta de coincidencia de datos. Puede intentar abordar este problema procesando previamente las imágenes web para que se parezcan más a las imágenes que se tomarán con la aplicación móvil y, luego, reentrenar al modelo. Por el contrario, si el modelo tiene un rendimiento deficiente en el conjunto train-dev, entonces debe haber sobreajustado el conjunto de entrenamiento, por lo que debe intentar simplificar o regularizar el modelo, obtener más datos de entrenamiento y limpiar los datos de entrenamiento.

TEOREMA SIN ALMUERZO GRATIS

Un modelo es una versión simplificada de las observaciones. Las simplificaciones están destinadas a descartar los detalles superfluos que es poco probable que se generalicen a nuevas instancias. Para decidir qué datos descartar y qué datos conservar, debe hacer suposiciones . Por ejemplo, un modelo lineal supone que los datos son fundamentalmente

lineales y que la distancia entre las instancias y la línea recta es solo ruido, que puede ignorarse con seguridad.

En un famoso artículo de 1996 , 11 David Wolpertdemostró que si no hace absolutamente ninguna suposición sobre los datos, entonces no hay razón para preferir un modelo sobre cualquier otro. Esto se llama el teorema de No Free Lunch (NFL). Para algunos conjuntos de datos, el mejor modelo es un modelo lineal, mientras que para otros conjuntos de datos es una red neuronal. No hay ningún modelo que esté garantizado a priori para funcionar mejor (de ahí el nombre del teorema). La única forma de saber con certeza qué modelo es el mejor es evaluarlos todos. Dado que esto no es posible, en la práctica usted hace algunas suposiciones razonables sobre los datos y evalúa solo algunos modelos razonables. Por ejemplo, para tareas simples, puede evaluar modelos lineales con varios niveles de regularización y, para un problema complejo, puede evaluar varias redes neuronales.

Ejercicios

En este capítulo hemos cubierto algunos de los conceptos más importantes en Machine Learning. En los próximos capítulos profundizaremos y escribiremos más código, pero antes de hacerlo, asegúrese de saber cómo responder las siguientes preguntas:

¿Cómo definiría el aprendizaje automático?

¿Puedes nombrar cuatro tipos de problemas en los que brilla?

¿Qué es un conjunto de entrenamiento etiquetado?

¿Cuáles son las dos tareas supervisadas más comunes?

¿Puedes nombrar cuatro tareas comunes sin supervisión?

¿Qué tipo de algoritmo de aprendizaje automático usaría para permitir que un robot camine en varios terrenos desconocidos?

¿Qué tipo de algoritmo utilizaría para segmentar a sus clientes en varios grupos?

¿Enmarcaría el problema de la detección de spam como un problema de aprendizaje supervisado o un problema de aprendizaje no supervisado?

¿Qué es un sistema de aprendizaje en línea?

¿Qué es el aprendizaje fuera de núcleo?

¿Qué tipo de algoritmo de aprendizaje se basa en una medida de similitud para hacer predicciones?

¿Cuál es la diferencia entre un parámetro de modelo y el hiperparámetro de un algoritmo de aprendizaje?

¿Qué buscan los algoritmos de aprendizaje basados en modelos? ¿Cuál es la estrategia más común que utilizan para tener éxito? ¿Cómo hacen predicciones?

¿Puedes nombrar cuatro de los principales desafíos del aprendizaje automático?

Si su modelo tiene un excelente rendimiento en los datos de entrenamiento, pero se generaliza mal a nuevas instancias, ¿qué está sucediendo? ¿Puedes nombrar tres posibles soluciones?

¿Qué es un equipo de prueba y por qué querría usarlo?

¿Cuál es el propósito de un conjunto de validación?

¿Qué es el conjunto train-dev, cuándo lo necesita y cómo lo usa?

¿Qué puede salir mal si ajusta los hiperparámetros con el conjunto de prueba?

Las soluciones a estos ejercicios están disponibles en el Apéndice A .

1Dato curioso: este nombre que suena extraño es un término de estadística introducido por Francis Galton mientras estudiaba el hecho de que los hijos de personas altas tienden a ser más bajos que sus padres. Dado que los niños eran más bajos, llamó a esta regresión a la media . Luego, este nombre se aplicó a los métodos que utilizó para analizar las correlaciones entre variables.

2Algunas 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.

3Observe cómo los animales están bastante bien separados de los vehículos y cómo los caballos están cerca de los ciervos pero lejos de las aves. Figura reproducida con permiso de Richard Socher et al., “Aprendizaje de acción cero a través de la transferencia entre modos”, Actas de la 26ª Conferencia Internacional sobre Sistemas de Procesamiento de Información Neural 1 (2013): 935–943.

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

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

6La prepare_country_stats()definición de la función no se muestra aquí (consulte el cuaderno de Jupyter de este capítulo si desea todos los detalles sangrientos). Es solo un código aburrido de pandas que une los datos de satisfacción con la vida de la OCDE con los datos del PIB per cápita del FMI.

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

8 Por ejemplo, saber si escribir "a", "dos" o "también", según el contexto.

9Figura reproducida con permiso de Michele Banko y Eric Brill, “Escala a cuerpos muy muy grandes para la desambiguación del lenguaje natural”, Actas de la 39ª Reunión Anual de la Asociación de Lingüística Computacional (2001): 26–33.

10Peter Norvig et al., “La efectividad irrazonable de los datos”, IEEE Intelligent Systems 24, no. 2 (2009): 8–12.

11David Wolpert, "La falta de distinciones a priori entre algoritmos de aprendizaje", Computación neuronal 8, no. 7 (1996): 1341-1390.