Instituto Tecnológico de Ciudad Madero

Miranda Martínez Diego Ismael

19071551

Inteligencia artificial

12:00 – 1:00 pm

Tarea 13. Machine Learning P1

5 de junio de 2023

El Panorama del Aprendizaje Automático

Con los libros electrónicos Early Release, obtiene libros en su forma más antigua: el contenido sin editar y sin editar del autor a medida que él o ella escribe, por lo que puede aprovechar estas tecnologías mucho antes de que el oficial lanzamiento de estos títulos. Lo siguiente será el Capítulo 1 en el final lanzamiento del libro.

Cuando la mayoría de las personas escuchan "Aprendizaje automático", se imaginan 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 ML que realmente se convirtió en la corriente principal, mejorando la vida de cientos de millones de personas, se apoderó del mundo allá por la década de 1990: era el filtro de spam. No es exactamente una Skynet consciente de sí misma, pero técnicamente califica como aprendizaje automático (En realidad, ha aprendido tan bien que rara vez necesita marcar un correo electrónico como spam). Le siguieron cientos de aplicaciones ML que ahora potencian silenciosamente cientos de productos y funciones que usa regularmente, a partir de mejores recomendaciones a la búsqueda por voz. ¿Dónde comienza el aprendizaje automático y dónde termina? que hace exactamente significa que una máquina aprenda algo? Si descargo una copia de Wikipedia, ¿tiene mi computadora realmente "aprendió" algo? ¿Es de repente más inteligente? En este capítulo, comenzaremos aclarando qué es el aprendizaje automático y por qué es posible que desee utilizarlo. Luego, antes de comenzar a explorar el continente de Machine Learning, tomaremos una mira el mapa y aprende sobre las principales regiones y los hitos más notables: aprendizaje supervisado versus no supervisado, aprendizaje en línea versus aprendizaje por lotes, aprendizaje basado en instancias versus aprendizaje basado en modelos. Luego veremos el flujo de trabajo de un ML típico proyecto, discuta los principales desafíos que puede enfrentar y cubra cómo evaluar y afinar un sistema de Machine Learning.

Este capítulo introduce una gran cantidad de conceptos fundamentales (y jerga) que cada dato científico debe saber de memoria. Será una descripción general de alto nivel (el único capítulo sin mucho código), todo bastante simple, pero debes asegurarte de que todo esté muy claro para usted antes de continuar con el resto del libro. Así que toma un café y vamos. ¡Empezar!

Si ya conoce todos los conceptos básicos de Machine Learning, es posible que desee para pasar directamente al Capítulo 2. Si no está seguro, intente contestar todas las preguntas enumeradas 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: [El aprendizaje automático es el] campo de estudio que le da a las computadoras la capacidad de aprender sin estar explícitamente programado.

—Arthur Samuel, 1959

Y uno más orientado a la ingeniería: Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna tarea T y alguna medida de desempeño P, si su desempeño en T, medido por P, mejora con experiencia e.

—Tom Mitchell, 1997

Por ejemplo, su filtro de spam es un programa de aprendizaje automático que puede aprender a marcar correo no deseado dados ejemplos de correos electrónicos no deseados (por ejemplo, marcados por los usuarios) y ejemplos de correos electrónicos regulares (no spam, también llamados "ham") correos electrónicos. Los ejemplos que utiliza el sistema para aprender son llamados conjunto de entrenamiento. Cada ejemplo de entrenamiento se denomina instancia de entrenamiento (o muestra). En este caso, la tarea T es marcar el spam para nuevos correos electrónicos, la experiencia E es la capacitación

datos, y la medida de desempeño P necesita ser definida; por ejemplo, puedes usar la proporción de correos electrónicos clasificados correctamente. Esta medida particular de desempeño se llama precisión y se utiliza a menudo en tareas de clasificación.

Si simplemente descarga una copia de Wikipedia, su computadora tiene muchos más datos, pero es no de repente mejor en cualquier tarea. Por lo tanto, no es Machine Learning.

¿Por qué utilizar el aprendizaje automático?

Considere cómo escribiría un filtro de spam usando técnicas de programación tradicionales:

1. Primero, vería cómo se ve típicamente el spam. Puede notar que algunas palabras o frases (como "4U", "tarjeta de crédito", "gratis" y "increíble") tienden a aparecer mucho en el tema. Quizás también notará algunos otros patrones en el nombre del remitente, el cuerpo del correo electrónico, etc.

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

3. Probaría su programa y repetiría los pasos 1 y 2 hasta que sea lo suficientemente bueno.

Dado que el problema no es trivial, 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 buenas predictoras de spam al detectar patrones inusualmente frecuentes de palabras en los ejemplos de spam en comparación con los ejemplos de jamón. El programa es mucho más corto, más fácil de mantener y probablemente más preciso.

Además, si los spammers notan que todos sus correos electrónicos que contienen "4U" están bloqueados, podrían comenzar a escribir "Para ti" en su lugar. Sería necesario actualizar un filtro de correo no deseado que utiliza técnicas de programación tradicionales para marcar los correos electrónicos "Para ti". Si los spammers siguen eludiendo su filtro de spam, tendrá que seguir escribiendo nuevas reglas para siempre. Por el contrario, un filtro de spam basado en técnicas de Machine Learning detecta automáticamente que "For U" se ha vuelto inusualmente frecuente en el spam marcado por los usuarios y comienza a marcarlos sin su intervención.

Otra área en la que brilla Machine Learning es para problemas que son demasiado complejos para los enfoques tradicionales o no tienen un algoritmo conocido. Por ejemplo, considere el reconocimiento de voz: suponga que desea comenzar de manera simple y escribir un programa capaz de distinguir las palabras "uno" y "dos". Puede notar que la palabra "dos" comienza con un sonido de tono alto ("T"), por lo que podría codificar un algoritmo que mide la intensidad del sonido de tono alto y usarlo para distinguir unos y dos. Obviamente, esta técnica no escalará a miles de palabras pronunciadas por millones de personas muy diferentes personas en entornos ruidosos y en decenas de idiomas. La mejor solución (al menos hoy) es escribir un algoritmo que aprenda por sí mismo, dadas muchas grabaciones de ejemplo para cada palabra.

Finalmente, Machine Learning puede ayudar a los humanos a aprender: los algoritmos de ML se pueden inspeccionar para ver lo que han aprendido (aunque para algunos algoritmos esto puede ser complicado). Por ejemplo, una vez que el filtro de spam ha sido entrenado en suficiente spam, puede inspeccionarse 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 ML para profundizar en grandes cantidades de datos puede ayudar a descubrir patrones que no eran evidentes de inmediato. Esto se llama minería de datos.

En resumen, el aprendizaje automático es ideal para:

- Problemas para los que las soluciones existentes requieren muchos ajustes manuales o largas listas de reglas: un algoritmo de aprendizaje automático a menudo puede simplificar el código y funcionar mejor.
- Problemas complejos para los que no existe una buena solución utilizando un enfoque tradicional: las mejores técnicas de Machine Learning pueden encontrar una solución.
- Entornos fluctuantes: un sistema de Machine Learning puede adaptarse a nuevos datos.
- Obtener información sobre problemas complejos y grandes cantidades de datos.

Tipos de sistemas de aprendizaje automático.

Hay tantos tipos diferentes de sistemas de aprendizaje automático que es útil clasificarlos en categorías amplias basadas en:

- Si están o no entrenados con supervisión humana (aprendizaje supervisado, no supervisado, semisupervisado y por refuerzo)
- Si pueden o no aprender gradualmente sobre la marcha (aprendizaje en línea versus aprendizaje por lotes)
- Si funcionan simplemente comparando nuevos puntos de datos con puntos de datos conocidos, o si detectan patrones en los datos de entrenamiento y construyen un modelo predictivo, como 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 por refuerzo.

Aprendizaje supervisado.

En el aprendizaje supervisado, los datos de entrenamiento que alimenta al algoritmo incluyen las soluciones deseadas, llamadas etiquetas.

Una 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 los nuevos correos electrónicos.

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

En Machine Learning, un atributo es un tipo de datos (por ejemplo, "Millas"), mientras que una característica tiene varios significados según el contexto, pero generalmente significa un atributo más su valor (por ejemplo, "Millas = 15 000"). Sin embargo, muchas personas usan las palabras atributo y característica indistintamente.

Tenga en cuenta que algunos algoritmos de regresión también se pueden usar 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).

Estos son algunos de los algoritmos de aprendizaje supervisado más importantes (cubiertos 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

Aprendizaje sin supervisión.

En el aprendizaje no supervisado, como puede suponer, los datos de entrenamiento no están etiquetados. El sistema trata de aprender sin un maestro.

Estos son algunos de los algoritmos de aprendizaje no supervisado más importantes (la mayoría de ellos se tratan en el Capítulo 8 y el Capítulo 9):

- Agrupación
  - K-medias
  - DBSCAN
  - Análisis de conglomerados jerárquicos (HCA)
- Detección de anomalías y detección de novedades
  - SVM de una clase
  - Fores de aislamiento
- Visualización y reducción de dimensionalidad
  - Análisis de componentes principales (PCA)

- PCA del núcleo
- Incrustación lineal local (LLE)
- Incrustación de vecinos estocásticos distribuidos en t (t-SNE)

• Aprendizaje de reglas de asociación

- A priori
- CEPAL

Por ejemplo, supongamos que tiene una gran cantidad de datos sobre los visitantes de su blog. Es posible que desee ejecutar un algoritmo de agrupación para tratar de detectar grupos de visitantes similares (Figura 1-8). En ningún momento le dices al algoritmo a qué grupo pertenece un visitante: encuentra esas conexiones sin tu ayuda. Por ejemplo, podría 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, etc. Si utiliza un algoritmo de agrupamiento jerárquico, también puede subdividir cada grupo en grupos más pequeños grupos Esto puede ayudarlo a orientar sus publicaciones para cada grupo.

Los algoritmos de visualización también 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 se puede trazar fácilmente (Figura 1-9). Estos algoritmos intentan preservar la mayor cantidad de estructura posible (por ejemplo, tratando de evitar que los grupos 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 inesperados.

Una tarea relacionada es la reducción de la dimensionalidad, en la que el objetivo es simplificar los datos sin perder demasiada información. Una forma de hacer esto es fusionar varias características correlacionadas en una sola. Por ejemplo, el kilometraje de un automóvil puede estar muy relacionado con su edad, por lo que el algoritmo de

reducción de dimensionalidad los fusionará en una característica que representa el desgaste del automóvil. Esto se llama extracción de características.

Suele ser una buena idea tratar de reducir la dimensión de los datos de entrenamiento mediante un algoritmo de reducción de dimensionalidad antes de enviarlos 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 no supervisada es la detección de anomalías, por ejemplo, detectar transacciones inusuales con 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 muestra principalmente instancias normales durante el entrenamiento, por lo que aprende a reconocerlas y, cuando ve una nueva instancia, puede determinar si parece normal o si es probable que sea una anomalía. Una tarea muy similar es la detección de novedades: la diferencia es que los algoritmos de detección de novedades esperan ver solo datos normales durante el entrenamiento, mientras que los algoritmos de detección de anomalías suelen ser más tolerantes, a menudo pueden funcionar bien incluso con un pequeño porcentaje de valores atípicos en el conjunto de entrenamiento.

Finalmente, otra tarea común no supervisada 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, supón que tienes un supermercado. Ejecutar 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 uno cerca del otro.

Aprendizaje semisupervisado.

Algunos algoritmos pueden manejar datos de entrenamiento parcialmente etiquetados, generalmente muchos datos sin etiquetar y un poco de datos etiquetados. Esto se llama aprendizaje semisupervisado.

Algunos servicios de alojamiento de fotos, como Google Photos, son buenos ejemplos de esto. Una vez que subes todas tus fotos familiares al servicio, automáticamente reconoce que la misma persona A aparece en las fotos, mientras que otra persona B. Esta es la parte no supervisada del algoritmo (agrupamiento). Ahora todo lo que necesita el sistema es que usted le diga quiénes son estas personas. Solo una etiqueta por persona,4 y puede 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. Por ejemplo, las redes de creencias profundas (DBN) se basan en componentes no supervisados llamados máquinas restringidas de Boltzmann (RBM) apiladas una encima de la otra. Los RBM 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 es una bestia muy diferente. El sistema de aprendizaje, denominado agente en este contexto, puede observar el entorno, seleccionar y realizar acciones y obtener recompensas a cambio (o sanciones en forma de recompensas negativas. Luego debe aprender por sí mismo cuál es la mejor estrategia, llamada política, para obtener la mayor recompensa con el 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 apagó 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 o no aprender de forma incremental a partir de un flujo de datos entrantes.

Aprendizaje por lotes.

En el 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 aprender más; simplemente aplica lo que ha aprendido. Esto se llama aprendizaje fuera de línea.

Si desea que un sistema de aprendizaje por lotes conozca nuevos datos (como un nuevo tipo de spam), debe entrenar una nueva versión del sistema desde cero en el conjunto de datos completo (no solo los datos nuevos, sino también los datos antiguos), luego detenga el sistema anterior y reemplácelo por uno nuevo.

Afortunadamente, todo el proceso de capacitación, evaluación y lanzamiento de un sistema de aprendizaje automático se puede automatizar con bastante facilidad, 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 sea necesaria.

Esta solución es simple y, a menudo, funciona bien, pero la capacitación con el conjunto completo de datos puede llevar muchas horas, por lo que normalmente entrenaría un nuevo sistema solo cada 24 horas o incluso 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 en 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, terminará costándote mucho dinero. Si la cantidad de datos es enorme, puede que incluso sea imposible utilizar un lote algoritmo de aprendizaje.

Finalmente, si su sistema necesita poder aprender de forma autónoma y tiene recursos limitados (por ejemplo, una aplicación de teléfono inteligente o un rover en Marte), entonces lleva consigo grandes cantidades de datos de entrenamiento y consume muchos recursos para entrenar durante horas todos los días. 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, el sistema se entrena de manera incremental alimentándolo con instancias de datos de forma secuencial, ya sea individualmente o en pequeños grupos llamados minilotes. 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.

El aprendizaje en línea es excelente para los sistemas que reciben datos como un flujo continuo (por ejemplo, precios de acciones) y necesitan adaptarse para cambiar de forma 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 las necesita, por lo que puede descartarlas (a menos que desee poder retroceder a un estado anterior y "reproducir" los datos). Esto puede ahorrar una gran cantidad de espacio.

Los algoritmos de aprendizaje en línea también se pueden usar para entrenar sistemas en grandes conjuntos de datos que no caben en la memoria principal de una máquina (esto se denomina 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 ejecuta en todos los datos.

El aprendizaje fuera del núcleo generalmente se realiza fuera de línea (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: esto se denomina tasa de aprendizaje. Si establece una tasa de aprendizaje alta, su sistema se adaptará rápidamente a los nuevos datos, pero también tenderá a olvidar rápidamente los datos antiguos (no desea que un filtro de correo no deseado marque solo los últimos tipos de correo no deseado que se muestran).

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 o a las secuencias de puntos de datos no representativos (outliers).

Un gran desafío con el aprendizaje en línea es que, si se alimentan datos incorrectos al sistema, el rendimiento del sistema disminuirá gradualmente. Si estamos hablando de un sistema en vivo, sus clientes lo notarán. Por ejemplo, los datos erróneos podrían 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 apagar rápidamente el aprendizaje (y posiblemente volver a un estado de funcionamiento 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 Machine Learning es por cómo se generalizan. La mayoría de las tareas de Machine Learning consisten en hacer predicciones. Esto significa que, dada una cantidad de ejemplos de capacitación, 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 desempeño en nuevas instancias.

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

Aprendizaje basado en instancias.

Posiblemente la forma más trivial de aprender es simplemente aprender de memoria. Si tuviera que crear un filtro de correo no deseado 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. Esto requiere 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 comparándolos con los ejemplos aprendidos (o un subconjunto de ellos), utilizando una medida de similitud. Por ejemplo, la nueva instancia se clasificaría como un triángulo porque la mayoría de las instancias más similares pertenecen a esa clase.

Aprendizaje basado en modelos.

Otra 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. Esto se llama aprendizaje basado en modelos.

Por ejemplo, suponga que desea saber si el dinero hace feliz a la gente, por lo que descarga los datos del Índice para una Vida Mejor del sitio web de la OCDE, así como las estadísticas sobre el PIB per cápita del sitio web del FMI. Luego une las tablas y ordena por PIB per cápita. Grafiquemos los datos de algunos países al azar.

¡Parece que hay una tendencia aquí! Aunque los datos son ruidosos (es decir, parcialmente aleatorios), parece que la satisfacción con la vida aumenta de forma más o menos lineal a medida que aumenta el PIB per cápita del país. Así que decide modelar la satisfacción con la vida como una función lineal del PIB per cápita. Este paso se llama selección de modelo: seleccionó un modelo lineal de satisfacción con la vida con solo un atributo, el PIB per cápita (Ecuación 1-1).

Ecuación 1-1. Un modelo lineal simple satisfacción_vida = $\theta 0 + \theta 1 \times$ PIB_per cápita

Este modelo tiene dos parámetros de modelo, $\theta 0$ y $\theta 15$. Al ajustar estos parámetros, puede hacer que su modelo represente cualquier función lineal.

Antes de que pueda usar su modelo, debe definir los valores de los parámetros $\theta 0$ y $\theta 1$.

¿Cómo puede saber qué valores harán que su modelo funcione mejor? Para responder a esta pregunta, debe especificar una medida de rendimiento. Puede definir 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 problemas de regresión lineal, la gente normalmente usa una función de costo 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.

A esto se le llama entrenar el modelo. En nuestro caso, el algoritmo encuentra que los valores óptimos de los parámetros son θ0 = 4.85 y θ1 = 4.91 × 10–5.

Ahora el modelo se ajusta lo más posible a los datos de entrenamiento (para un modelo lineal).

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 probablemente sea alrededor de 4.85 + 22,587 × 4.91 × 10-5 = 5.96.

Si en su lugar hubiera utilizado un algoritmo de aprendizaje basado en instancias, habría encontrado que Eslovenia tiene el PIB per cápita más cercano a el de Chipre ($20.732), y dado que los datos de la OCDE nos dicen que la satisfacción con la vida de los eslovenos es de 5,7, habrías predicho una vida satisfacción de 5,7 para Chipre.

Si te alejas un poco y miras los próximos dos países más cercanos,

encontrará Portugal y España con satisfacciones con la vida de 5,1 y 6,5, respectivamente. Promediando estos tres valores, obtienes 5.77, que está bastante cerca de tu modelo anterior basado en dicción. Este algoritmo simple se llama regresión de k-vecinos más cercanos (en este ejemplo, k = 3).

Reemplazo del modelo de regresión lineal con k-vecinos más cercanos la regresión en el código anterior es tan simple como reemplazar estas dos líneas:

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.
- Ha seleccionado un modelo.
- Lo entrenó con los datos de entrenamiento (es decir, el algoritmo de aprendizaje buscó el valor de los parámetros del modelo que minimizan una función de costo).
- Finalmente, aplicó 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 Machine Learning. En el Capítulo 2, experimentará esto de primera mano al pasar por un proyecto de extremo a extremo.

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 los sistemas ML y cómo es un flujo de trabajo de proyecto típico. Ahora veamos qué puede salir mal en el aprendizaje y evitar que hagas 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 qué es una manzana, todo lo que necesita es señalar una manzana y decir "manzana" (posiblemente repitiendo este procedimiento varias veces). Ahora el niño puede reconocer manzanas en todo tipo de colores y formas. Genio. El aprendizaje automático aún no ha llegado; 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, puede necesitar millones de ejemplos (a menos que pueda reutilizar partes de un modelo existente).

La efectividad irrazonable de los datos.

En un famoso artículo publicado en 2001, los investigadores de Microsoft Michele Banko y Eric Brill demostraron que algoritmos de aprendizaje automático muy diferentes, incluidos los bastante simples, se desempeñaron casi de manera idéntica en un problema complejo de desambiguación del lenguaje natural una vez que se les proporcionaron suficientes datos.

Como lo expresaron los autores: "estos resultados sugieren que es posible que deseemos reconsiderar el equilibrio 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. Cabe señalar, sin embargo, que los conjuntos de datos de tamaño pequeño y mediano siguen siendo muy comunes, y no siempre es fácil o barato obtener datos de entrenamiento adicionales, así que no abandones los algoritmos todavía.

Datos de entrenamiento no representativos.

Para poder generalizar bien, es crucial que sus datos de entrenamiento sean representativos de los nuevos casos a los que desea generalizar. Esto es cierto ya sea que utilice el aprendizaje basado en instancias o 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.

Si entrena un modelo lineal con estos datos, obtiene la línea continua, mientras que el modelo antiguo se representa con la línea punteada. Como puede ver, agregar algunos países que faltan 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 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 entrenamiento no representativo, entrenamos un modelo que es poco probable que haga predicciones precisas, especialmente para países muy pobres y muy ricos.

Es crucial usar un conjunto de entrenamiento 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 del azar), pero incluso muestras muy grandes pueden no ser representativas si el método de muestreo es defectuoso. Se llama sesgo de muestreo.

A Famous Example of Sampling Bias.

Quizás el ejemplo más famoso de sesgo de muestreo ocurrió durante las elecciones presidenciales de EE. UU. en 1936, que enfrentó a Landon contra Roosevelt: el Literary Digest realizó una encuesta muy grande y envió correos a unos 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 encuestas, el Literary Digest utilizó directorios telefónicos, listas de suscriptores de revistas, listas de miembros de clubes y la como. 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. Una vez más, esto introduce un sesgo de muestreo, al descartar a las personas a las que no les importa mucho la política, a las personas a las que no les gusta Literary Digest y otros grupos clave. Este es un tipo especial de sesgo de muestreo llamado sesgo de falta de respuesta.

Aquí hay otro ejemplo: supongamos que desea construir un sistema para reconocer videos de música funk. Una forma de crear su 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 obtendrá muchos videos de "funk carioca", que no se parecen en nada a James Brown).

Por otro lado, ¿de qué otra manera puedes obtener un gran conjunto de entrenamiento?

Datos de mala calidad

Obviamente, si sus datos de entrenamiento están llenos de errores, valores atípicos y ruido (por ejemplo, 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 pasan una parte importante de su tiempo haciendo precisamente eso. Por ejemplo:

- Si algunas instancias son claramente atípicas, puede ser útil simplemente descartarlas o intentar corregir los errores manualmente.

- Si a algunas instancias les faltan algunas características (p. ej., 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 (p. ej., con la mediana de edad), 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. Tu sistema solo será capaz de aprender si los datos de entrenamiento contienen suficientes características relevantes y no demasiadas irrelevantes. Una parte crítica del éxito de un proyecto de aprendizaje automático es crear un buen conjunto de funciones para entrenar. Este proceso, llamado ingeniería de características, implica:

- Selección de funciones: selección de 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).
- Creación de nuevas características mediante la recopilación de nuevos datos.

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

Superposición de los datos de entrenamiento.

Digamos que está visitando un país extranjero y el taxista lo estafa. Es posible que sienta la tentación de decir que todos los taxistas de ese país son ladrones. Generalizar en exceso 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 funciona bien con los datos de entrenamiento, pero no generaliza bien.

Un modelo polinomial de satisfacción con la vida de alto grado que sobre ajusta fuertemente los datos de entrenamiento. Aunque funciona mucho mejor con los datos de entrenamiento que el modelo lineal simple, ¿realmente confiaría en sus predicciones?

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 que no son 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 superior a 7:

Nueva Zelanda (7,3), Noruega (7,4), Suecia (7,2) y Suiza (7,5). ¿Qué tan seguro está de que la regla de satisfacción W se generaliza a Ruanda o Zimbabue?

Obviamente, este patrón ocurrió en los datos de entrenamiento por pura casualidad, pero el modelo no tiene forma de saber 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 el cantidad y ruido de los datos de entrenamiento. Las posibles soluciones son:

- Simplificar el modelo seleccionando uno con menos parámetros (p. ej., un modelo lineal en lugar de un modelo polinomial 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 los valores atípicos)

La restricción de un modelo para simplificarlo y reducir el riesgo de sobreajuste se denomina regularización. Por ejemplo, el modelo lineal que definimos anteriormente tiene dos parámetros, $\theta 0$ y $\theta 1$. 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 (θ0) como la pendiente (θ1) de la línea. Si forzamos θ1 = 0, el algoritmo tendría solo un grado de libertad y le resultaría mucho más difícil 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 de hecho! Si permitimos que el algoritmo modifique θ1 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 con dos grados de libertad, pero más complejo que con uno solo. Desea encontrar el equilibrio adecuado entre ajustar perfectamente los datos de entrenamiento y mantener el modelo lo suficientemente simple para garantizar que se generalice bien. Muestra tres modelos: la línea punteada representa el modelo original que se entrenó con algunos países faltantes, la línea discontinua es nuestro segundo modelo entrenado con todos los países, y la línea sólida 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, lo 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.

La cantidad de regularización que se aplicará 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 propio algoritmo de aprendizaje; debe establecerse 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 sobreajustará 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 próximo capítulo).

Comprender los datos de entrenamiento.

Como puede suponer, el ajuste insuficiente es lo opuesto al ajuste excesivo: 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 fallar; la realidad es simplemente más compleja que el modelo, por lo que sus predicciones están destinadas a ser inexactas, incluso en los ejemplos de entrenamiento.

Las principales opciones para solucionar este problema son:

- Seleccionar un modelo más potente, con más parámetros
- Alimentar mejores características al algoritmo de aprendizaje (ingeniería de características)
- Reducir las restricciones del modelo (p. ej., reducir el hiperparámetro de regularización)

Dar un paso atrás.

A estas alturas ya sabes mucho sobre Machine Learning. Sin embargo, analizamos tantos conceptos que es posible que se sienta un poco perdido, así que retrocedamos y veamos el panorama general:

- El aprendizaje automático se trata de 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 ML: supervisados o no, por lotes o en línea, basados en instancias o basados en modelos, etc.
- En un proyecto de ML, recopila datos en un conjunto de entrenamiento y alimenta el conjunto de entrenamiento a un algoritmo de aprendizaje. Si el algoritmo está basado en un modelo, ajusta algunos parámetros para ajustar el modelo al conjunto de entrenamiento (es decir, para hacer buenas predicciones sobre el propio conjunto de entrenamiento), y luego, con suerte, también podrá hacer buenas predicciones sobre

casos nuevos. Si el algoritmo está basado en instancias, simplemente aprende los ejemplos de memoria y generaliza a nuevas instancias comparándolas con las instancias aprendidas usando una medida de similitud.

- El sistema no funcionará bien si su conjunto de entrenamiento es demasiado pequeño o si los datos no son representativos, tienen ruido o están contaminados con características irrelevantes (entrada basura, salida basura). Por último, su modelo no debe ser ni demasiado simple (en cuyo caso se ajustará mal) ni demasiado complejo (en cuyo caso se ajustará demasiado).

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

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

Una mejor opción es dividir sus datos en dos conjuntos: el conjunto de entrenamiento y el conjunto de prueba. Como lo implican estos nombres, usted entrena su modelo usando el conjunto de entrenamiento y lo prueba usando el conjunto de prueba. La tasa de error en 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 se está sobre ajustando a los datos de entrenamiento.

Es común utilizar el 80 % de los datos para entrenamiento y reservar el 20 % para pruebas. Sin embargo, esto depende del tamaño del conjunto de datos: si contiene 10

millones de instancias, retener el 1 % significa que su conjunto de prueba contendrá 100 000 instancias: eso es probablemente más que suficiente para obtener una buena estimación del error de generalización.

Ajuste de hiperparámetros y selección de modelos.

Por lo tanto, evaluar un modelo es bastante simple: solo use un conjunto de prueba. Ahora suponga que está dudando entre dos modelos (por ejemplo, un modelo lineal y un modelo polinomial): ¿cómo puede decidir? Una opción es entrenar a ambos y comparar qué tan bien generalizan usando el conjunto de prueba.

Ahora suponga que el modelo lineal 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 error de generalización más bajo, digamos solo un 5% de error.

Entonces 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.

Una solución común a este problema se denomina validación de retención:

simplemente se mantiene una parte del conjunto de entrenamiento para evaluar varios modelos candidatos y seleccionar el mejor.

El nuevo conjunto reservado 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 funciona 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 da 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. Una 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 haberlo entrenado con el resto de los datos. Al promediar todas las evaluaciones de un modelo, obtenemos una medida mucho más precisa de su rendimiento.

Sin embargo, existe un inconveniente: el tiempo de entrenamiento se multiplica por el número de conjuntos de validación. Discrepancia de datos.

En algunos casos, es fácil obtener una gran cantidad de datos para el entrenamiento, pero no es perfectamente representativo de los datos que se utilizarán en la 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. Tal vez solo tenga 10 000 imágenes representativas (es decir, tomadas con la aplicación). En este

caso, la regla más importante que debe recordar es que el conjunto de validación y la prueba deben ser lo más representativos posible de los datos que espera usar en producción, por lo que deben estar compuestos exclusivamente de imágenes representativas: puede mezclarlas y colocar la mitad en el conjunto de validación y la otra mitad en el conjunto de prueba (asegurándose de que no haya duplicados o casi duplicados en ambos conjuntos). Después de entrenar su modelo en las imágenes web, si observa que el rendimiento de su 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 es solo debido a la falta de coincidencia entre las imágenes de la web y las imágenes de la aplicación móvil. Una solución es mostrar parte de las imágenes de entrenamiento (de la web) en otro conjunto que Andrew Ng llama conjunto de desarrollo de entrenamiento. Después de entrenar el modelo (en el conjunto de entrenamiento, no en el conjunto de entrenamiento-desarrollo), puede evaluarlo en el conjunto de entrenamiento-desarrollo: si funciona bien, entonces el modelo no se está sobreajustando al conjunto de entrenamiento, por lo que, si funciona mal en la validación establecido, el problema debe provenir de la falta de coincidencia de datos. Puede intentar solucionar este problema preprocesando las imágenes web para que se parezcan más a las imágenes que será tomado por la aplicación móvil y luego volverá a entrenar el modelo. Por el contrario, si el modelo funciona mal en el conjunto de entrenamiento y desarrollo, entonces el modelo 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, como se discutió anteriormente.

No hay teorema de 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. Sin embargo, 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 Wolpert demostró que si no se hace absolutamente ninguna suposición sobre los datos, entonces no hay razón para preferir un modelo sobre otro. Esto se llama el teorema 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 existe un modelo que a priori garantice un mejor funcionamiento (de ahí el nombre del teorema). La única forma de saber con seguridad qué modelo es el mejor es evaluarlos todos. Dado que esto no es posible, en la práctica hace algunas suposiciones razonables sobre los datos y evalúa solo unos pocos 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:

1. ¿Cómo definiría el aprendizaje automático?
2. ¿Puedes nombrar cuatro tipos de problemas donde brilla?
3. ¿Qué es un conjunto de entrenamiento etiquetado?
4. ¿Cuáles son las dos tareas supervisadas más comunes?
5. ¿Puedes nombrar cuatro tareas comunes sin supervisión?
6. ¿Qué tipo de algoritmo de aprendizaje automático usaría para permitir que un robot camine en varios terrenos desconocidos?
7. ¿Qué tipo de algoritmo usaría para segmentar a sus clientes en varios grupos?

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

9. ¿Qué es un sistema de aprendizaje en línea?

10. ¿Qué es el aprendizaje fuera del núcleo?

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

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

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

14. ¿Puede nombrar cuatro de los principales desafíos en Machine Learning?

15. Si su modelo funciona muy bien con los datos de entrenamiento, pero se generaliza mal a instancias nuevas, ¿qué está pasando? ¿Puedes nombrar tres posibles soluciones?

16. ¿Qué es un conjunto de prueba y por qué querría usarlo?

17. ¿Cuál es el propósito de un conjunto de validación?

18. ¿Qué puede salir mal si ajusta los hiperparámetros usando el conjunto de prueba?

19. ¿Qué es la validación cruzada repetida y por qué preferiría usar un solo conjunto de validación?

¿Las soluciones a estos ejercicios están disponibles en?

Diccionario.

- Machine Learning.

Es un aprendizaje de máquinas que utilizan muchos datos con el objetivo de ser cada vez más inteligentes a la hora de hacer cosas. El ML se alimenta de algoritmo e información, es algo así como llevar la inteligencia al dato.

- Algoritmos.

Un algoritmo es una serie de pasos matemáticos u operacionales específicas para resolver un problema o realizar una tarea. En el contexto del Machine Learning, un algoritmo transforma o analiza datos para llevar a cabo las siguientes tareas:

- Análisis de regresión.
- Clasificar a los clientes.
- Encontrar relaciones entre los SKU (códigos compuestos por letras y números que identifican las características de un producto).

- Modelo.

La definición más simple de un modelo es la representación matemática de las relaciones en un conjunto de datos. O lo que es lo mismo, es una forma simplificada y matemáticamente formalizada de aproximarse a la realidad y hacer predicciones a partir de esta aproximación. Un ejemplo simple que encontramos en el huffingtonpost es el siguiente:

- Características/ variables.

Objetivamente, las características son elementos o dimensiones de un conjunto de datos; por lo tanto, elegir las informativas, discriminatorias e independientes es un paso crucial para lograr algoritmos efectivos.

- Supervisión vs no supervisión.

El aprendizaje automático puede tener dos enfoques fundamentales. Por una parte, el aprendizaje supervisado, que es una forma de enseñar a un algoritmo cómo hacer su trabajo cuando tiene un conjunto de datos para los que sabe «su respuesta». Por ejemplo, para crear un modelo que pueda reconocer imágenes de datos a través de este proceso, el sistema ya tendría las imágenes etiquetadas como «gato» o «no gato».

Por otra parte, se denomina aprendizaje no supervisado cuando un algoritmo analiza el dato que no ha sido etiquetado con una respuesta para identificar patrones o correlaciones. Este sistema puede analizar un gran conjunto de datos de un cliente y obtener resultados que indiquen que este pertenece a siete grupos grandes o doce pequeños. A continuación, el científico de datos puede analizar estos resultados para averiguar qué define a cada grupo y cómo podría impactar en tu negocio.

- Deep Learning.

El Deep Learning es un tipo de aprendizaje automático, utiliza múltiples capas de cálculo y otras más avanzadas con características abstractas y de alto nivel. En el ejemplo que mencionábamos antes de las fotos de gato, la primera capa podría referirse a un conjunto de líneas que pudieran crear una figura, y las capas posteriores pudieran buscar elementos más concretos, como ojos o una cara completa.