Reglas del aprendizaje automático: Las mejores prácticas para la ingeniería de ML

Martin Zinkevich

Este documento pretende ayudar a aquellos con conocimientos básicos de aprendizaje automático a beneficiarse de las mejores prácticas de aprendizaje automático de todo Google. Presenta un estilo para el aprendizaje automático, similar a la Guía de estilo de C++ de Google y a otras guías populares de programación práctica. Si has asistido a una clase de aprendizaje automático o has creado o trabajado en un modelo de aprendizaje automático, dispones de los conocimientos necesarios para leer este documento.

Terminología

Los siguientes términos aparecerán repetidamente en nuestro debate sobre el aprendizaje automático eficaz:

Instancia: Aquello sobre lo que se quiere hacer una predicción. Por ejemplo, la instancia puede ser una página web que se quiere clasificar como "sobre gatos" o "no sobre gatos".
Etiqueta: Una respuesta para una tarea de predicción-- , ya sea la respuesta producida por un sistema de aprendizaje automático o la respuesta correcta proporcionada en los datos de entrenamiento. Por ejemplo, la etiqueta de una página web puede ser "sobre gatos".
Característica: Propiedad de una instancia utilizada en una tarea de predicción. Por ejemplo, una página web puede tener la característica "contiene la palabra 'gato'".
Columna de características[1]: Conjunto de características relacionadas, como el conjunto de todos los posibles países en los que pueden vivir los usuarios. Un ejemplo puede tener una o más características presentes en una columna de características. Una columna de características se denomina "espacio de nombres" en el sistema VW (en Yahoo/Microsoft), o campo.
Ejemplo: Una instancia (con sus características) y una etiqueta.
Modelo: Representación estadística de una tarea de predicción. *Se entrena* un modelo con ejemplos y luego se utiliza para hacer predicciones.
Métrica: Un número que le interesa. Puede optimizarse directamente o no.
Objetivo: Una métrica que el algoritmo intenta optimizar.
Canalización: La infraestructura que rodea a un algoritmo de aprendizaje automático. Incluye la recopilación de datos desde el front-end, su colocación en archivos de datos de entrenamiento, el entrenamiento de uno o más modelos y la exportación de los modelos a producción.

Visión general

Para crear grandes productos:
haz aprendizaje automático como el gran ingeniero que eres, no como el gran experto en aprendizaje automático que no eres.

[1]Terminología específica de Google.

La mayoría de los problemas a los que te enfrentarás son, de hecho, problemas de ingeniería. Incluso con todos los recursos de un gran experto en aprendizaje automático, la mayor parte de los beneficios proceden de grandes características, no de grandes algoritmos de aprendizaje automático. Por lo tanto, el enfoque básico es:

1. asegúrese de que su tubería es sólida de extremo a extremo
2. empezar con un objetivo razonable
3. añada funciones de sentido común de forma sencilla
4. asegúrese de que su proceso sigue siendo sólido.

Con este planteamiento ganarás mucho dinero y/o harás feliz a mucha gente durante mucho tiempo. Desvíese de este enfoque sólo cuando no haya más trucos sencillos que le lleven más lejos. Añadir complejidad ralentiza los futuros lanzamientos.

Una vez que haya agotado los trucos simples, el aprendizaje automático de vanguardia podría estar en su futuro. Consulte la sección sobre proyectos de aprendizaje automático de la Fase III.

Este documento se divide en cuatro partes:

1. La primera parte le ayudará a saber si ha llegado el momento de crear un sistema de aprendizaje automático.
2. La segunda parte trata sobre el despliegue del primer pipeline.
3. La tercera parte trata sobre el lanzamiento y la iteración mientras se añaden nuevas características a su canalización, cómo evaluar los modelos y el sesgo de servicio de formación.
4. La última parte trata sobre qué hacer cuando se llega a un punto muerto.
5. A continuación, se incluye una lista de trabajos relacionados y un apéndice con información general sobre los sistemas utilizados habitualmente como ejemplos en este documento.

Antes del aprendizaje automático

Regla nº 1: No tenga miedo de lanzar un producto sin aprendizaje automático.
El aprendizaje automático es genial, pero requiere datos. Teóricamente, se pueden tomar datos de un problema diferente y luego ajustar el modelo para un nuevo producto, pero es probable que el rendimiento de la heurística básica sea inferior. Si crees que el aprendizaje automático te dará un impulso del 100%, entonces una heurística te llevará al 50% del camino.

Por ejemplo, si está clasificando aplicaciones en un mercado de aplicaciones, podría utilizar la tasa de instalación o el número de instalaciones. Si está detectando spam, filtre los editores que hayan enviado spam anteriormente. Tampoco tengas miedo de utilizar la edición humana. Si necesita clasificar contactos, ordene los más usados recientemente (o incluso ordene alfabéticamente). Si el aprendizaje automático no es absolutamente necesario para su producto, no lo utilice hasta que disponga de datos.

Regla nº 2: Primero, diseñe y aplique métricas.
Antes de formalizar lo que hará su sistema de aprendizaje automático, realice un seguimiento de todo lo posible en su sistema actual. Hágalo por las siguientes razones:

1. Es más fácil obtener antes el permiso de los usuarios del sistema.
2. Si crees que algo puede ser preocupante en el futuro, es mejor obtener datos históricos ahora.
3. Si diseñas tu sistema teniendo en cuenta la instrumentación métrica, las cosas te irán mejor en el futuro. En concreto, ¡no querrás encontrarte buscando cadenas en los registros para instrumentar tus métricas!
4. Te darás cuenta de qué cosas cambian y qué permanece igual. Por ejemplo, supongamos que desea optimizar directamente los usuarios activos diarios. Sin embargo, durante tus primeras manipulaciones del sistema, puedes notar que las alteraciones drásticas de la experiencia del usuario no cambian notablemente esta métrica.

El equipo de Google Plus mide las expansiones por lectura, los reenvíos por lectura, los plusones por lectura, los comentarios por lectura, los comentarios por usuario, los reenvíos por usuario, etc., que utilizan para calcular la bondad de una publicación en el momento de servirla. **También es importante contar con un marco de experimentos que permita agrupar a los usuarios y agregar estadísticas por experimento**. Véase **la regla nº 12**.

Siendo más liberal en la recopilación de métricas, puedes obtener una imagen más amplia de tu sistema. ¿Notas un problema? Añade una métrica para seguirlo. ¿Te entusiasma algún cambio cuantitativo de la última versión? Añade una métrica para seguirlo.

Regla nº 3: Elija el aprendizaje automático en lugar de una heurística compleja.
Una heurística simple puede hacer que su producto salga al mercado. Una heurística compleja es imposible de mantener. Una vez que tengas datos y una idea básica de lo que estás tratando de lograr, pasa al aprendizaje automático. Como en la mayoría de las tareas de ingeniería de software, usted querrá estar constantemente actualizando su enfoque, ya sea una heurística o un modelo de aprendizaje automático, y usted encontrará que el modelo de aprendizaje automático es más fácil de actualizar y mantener (ver Regla # 16).

Fase I de ML: Su primera canalización

Concéntrese en la infraestructura de su sistema para su primera canalización. Si bien es divertido pensar en todo el aprendizaje automático imaginativo que va a hacer, será difícil averiguar lo que está sucediendo si primero no confía en su canalización.

Regla 4: Simplifique el primer modelo y cree la infraestructura adecuada.
El primer modelo proporciona el mayor impulso a su producto, por lo que no necesita ser lujoso. Pero se encontrará con muchos más problemas de infraestructura de los que espera. Antes de que nadie pueda utilizar tu nuevo y elegante sistema de aprendizaje automático, tienes que determinar:

1. Cómo hacer llegar ejemplos a su algoritmo de aprendizaje.
2. Qué significa "bueno" y "malo" para su sistema.
3. Cómo integrar su modelo en su aplicación. Puedes aplicar el modelo en vivo, o precalcular el modelo sobre ejemplos fuera de línea y almacenar los resultados en una tabla. Por ejemplo, puede preclasificar páginas web y almacenar los resultados en una tabla, pero también puede clasificar mensajes de chat en directo.

Elegir características sencillas hace que sea más fácil asegurarse de que

1. Las características llegan correctamente a su algoritmo de aprendizaje.
2. El modelo aprenda ponderaciones razonables.
3. Las características lleguen correctamente al modelo en el servidor.

Una vez que tenga un sistema que haga estas tres cosas de forma fiable, habrá hecho la mayor parte del trabajo. Su modelo simple le proporciona métricas de referencia y un comportamiento de referencia que puede utilizar para probar modelos más complejos. Algunos equipos tienen como objetivo un primer lanzamiento "neutral": un primer lanzamiento que explícitamente desprioriza las ganancias del aprendizaje automático, para evitar distraerse.

Regla 5: Pruebe la infraestructura independientemente del aprendizaje automático.
Asegúrese de que la infraestructura se puede probar y de que las partes de aprendizaje del sistema están encapsuladas para que pueda probar todo lo que le rodea. En concreto:

1. Prueba la introducción de datos en el algoritmo. Compruebe que las columnas de características que deben rellenarse están rellenas. Cuando la privacidad lo permita, inspeccione manualmente la entrada a su algoritmo de entrenamiento. Si es posible, compruebe las estadísticas de su algoritmo en comparación con otros, como RASTA.
2. Pruebe a sacar modelos del algoritmo de entrenamiento. Asegúrese de que el modelo en su entorno de formación da la misma puntuación que el modelo en su entorno de servicio (véase la regla 37).

El aprendizaje automático tiene un elemento de imprevisibilidad, así que asegúrate de que tienes pruebas para el código de creación de ejemplos en el entrenamiento y en el servicio, y de que puedes cargar y utilizar un modelo fijo durante el servicio. Además, es importante comprender sus datos: consulte Consejos prácticos para el análisis de conjuntos de datos grandes y complejos.

Regla nº 6: Tenga cuidado con los datos descartados al copiar canalizaciones.
A menudo creamos un pipeline copiando un pipeline existente (es decir, programación de culto a la carga), y el pipeline antiguo deja caer datos que necesitamos para el nuevo pipeline. Por ejemplo, la canalización de Google Plus What's Hot omite publicaciones antiguas (porque intenta clasificar publicaciones recientes). Esta canalización se copió para utilizarla en Google Plus Stream, donde las publicaciones antiguas siguen siendo significativas, pero la canalización seguía eliminando publicaciones antiguas. Otro patrón común es registrar únicamente los datos que ha visto el usuario. Por lo tanto, estos datos son inútiles si queremos modelar por qué una publicación en particular no fue vista por el usuario, porque todos los ejemplos negativos han sido descartados. Un problema similar ocurrió en Play. Mientras se trabajaba en Play Apps Home, se creó un nuevo pipeline que también contenía ejemplos de otras dos páginas de destino (Play Games Home y Play Home Home) sin ninguna característica para desambiguar de dónde procedía cada ejemplo.

Regla nº 7: Convierta la heurística en características o gestiónelas externamente.
Normalmente, los problemas que intenta resolver el aprendizaje automático no son completamente nuevos. Ya existe un sistema para clasificar, ordenar o resolver el problema que sea. Esto significa que hay un montón de reglas y heurística. **Esta misma heurística puede darte un empujón cuando se ajusta con el aprendizaje automático**. La heurística debe extraerse de cualquier información que contenga, por dos razones. En primer lugar, la transición a un sistema de aprendizaje automático será más suave. En segundo lugar, por lo general esas reglas contienen una gran cantidad de la intuición sobre el sistema que usted no quiere tirar a la basura. Hay cuatro maneras de utilizar una heurística existente:

1. **Preprocesar usando la heurística**. Si la función es increíblemente asombrosa, esta es una opción. Por ejemplo, si en un filtro de spam el remitente ya está en la lista negra, no intentes volver a aprender lo que significa "lista negra". Bloquee el mensaje. Este enfoque tiene más sentido en tareas de clasificación binaria.
2. **Crear una característica**. Crear directamente una característica a partir de la heurística es genial. Por ejemplo, si utiliza una heurística para calcular una puntuación de relevancia para un resultado de consulta, puede incluir la puntuación como valor de una característica. Más adelante, es posible que desee utilizar técnicas de aprendizaje automático para masajear el valor (por ejemplo, convirtiendo el valor en uno de un conjunto finito de valores discretos, o combinándolo con otras características), pero comience utilizando el valor en bruto producido por la heurística.
3. **Minar las entradas brutas de la heurística**. Si existe una heurística para aplicaciones que combina el número de instalaciones, el número de caracteres del texto y el día de la semana, considere la posibilidad de separar estas piezas e introducirlas en el aprendizaje por separado. Algunas técnicas que se aplican a los conjuntos son válidas en este caso (véase la regla 40).
4. **Modificar la etiqueta.** Es una opción si cree que la heurística capta información que la etiqueta no contiene. Por ejemplo, si está intentando maximizar el número de descargas, pero también desea un contenido de calidad, quizá la solución sea multiplicar la etiqueta por el número medio de estrellas que ha recibido la aplicación. Aquí hay mucho margen de maniobra. Consulte la sección "Su primer objetivo".

Tenga en cuenta la complejidad añadida al utilizar la heurística en un sistema ML. El uso de heurísticos antiguos en su nuevo algoritmo de aprendizaje automático puede ayudar a crear una transición suave, pero piense si hay una forma más sencilla de lograr el mismo efecto.

Supervisión

En general, practique una buena higiene de alertas, como hacer que las alertas sean procesables y tener una página de panel de control.

Regla 8: Conozca los requisitos de frescura de su sistema.
¿Cuánto se degrada el rendimiento si tienes un modelo de hace un día? ¿De hace una semana? ¿Un trimestre? Esta información puede ayudarle a comprender las prioridades de su supervisión. Si pierde un 10% de sus ingresos si el modelo no se actualiza durante un día, tiene sentido que un ingeniero lo vigile continuamente. La mayoría de los sistemas de publicación de anuncios tienen que gestionar anuncios nuevos todos los días y deben actualizarse a diario.

cada día, y deben actualizarse diariamente. Por ejemplo, si el modelo ML para Google Play Search no se actualiza, puede tener un impacto en los ingresos en menos de un mes. Algunos modelos para What's Hot en Google Plus no tienen identificador de post en su modelo, por lo que pueden exportar estos modelos con poca frecuencia. Otros modelos que tienen identificadores de post se actualizan con mucha más frecuencia. Ten en cuenta también que la frescura puede cambiar con el tiempo, especialmente cuando se añaden o eliminan columnas de características de tu modelo.

Regla nº 9: Detecte los problemas antes de exportar los modelos.
Muchos sistemas de aprendizaje automático tienen una etapa en la que se exporta el modelo para servir. Si hay un problema con un modelo exportado, es un problema de cara al usuario. Si hay un problema antes, entonces es un problema de formación, y los usuarios no lo notarán.

Realice comprobaciones de sanidad antes de exportar el modelo. En concreto, asegúrese de que el rendimiento del modelo es razonable en los datos retenidos. O bien, si tiene dudas sobre los datos, no exporte el modelo. Muchos equipos que despliegan modelos continuamente comprueban el área bajo la curva ROC (o AUC) antes de exportarlos. **Los problemas sobre modelos que no se han exportado requieren una alerta por correo electrónico, pero los problemas sobre un modelo orientado al usuario pueden requerir una página.** Así que es mejor esperar y estar seguro antes de afectar a los usuarios.

Regla nº 10: Vigile los fallos silenciosos.
Este es un problema que ocurre más en los sistemas de aprendizaje automático que en otros tipos de sistemas. Supongamos que una tabla concreta a la que se está uniendo ya no se actualiza. El sistema de aprendizaje automático se ajustará, y el comportamiento seguirá siendo razonablemente bueno, decayendo gradualmente. A veces se encuentran tablas que llevaban meses desactualizadas, ¡y una simple actualización mejoró el rendimiento más que cualquier otro lanzamiento de ese trimestre! Por ejemplo, la cobertura de una característica puede cambiar debido a cambios de implementación: por ejemplo, una columna de característica podría estar poblada en el 90% de los ejemplos, y de repente caer al 60% de los ejemplos. En una ocasión, Play tenía una tabla que estaba obsoleta desde hacía 6 meses, y sólo con actualizarla se consiguió un aumento del 2% en la tasa de instalación. Si realizas un seguimiento estadístico de los datos, así como una inspección manual de los datos en ocasiones, puedes reducir este tipo de fallos.

Regla nº 11: Proporcione propietarios de columnas de características y documentación.
Si el sistema es grande y hay muchas columnas de características, sepa quién creó o mantiene cada columna de características. Si se da cuenta de que la persona que entiende de una columna de características se va, asegúrese de que alguien tiene la información. Aunque muchas columnas de características tienen nombres descriptivos, es bueno tener una descripción más detallada de lo que es la característica, de dónde viene y cómo se espera que ayude.

Su primer objetivo

Usted tiene muchas métricas, o medidas sobre el sistema que le importan, pero su algoritmo de aprendizaje automático a menudo requerirá un único **objetivo, un número que su algoritmo**

intenta optimizar. Aquí distingo entre objetivos y métricas: **una métrica es cualquier número que informe su sistema**, que puede o no ser importante. Véase también la regla nº 2.

Regla nº 12: No piense demasiado qué objetivo elige optimizar directamente.
Quieres ganar dinero, hacer felices a tus usuarios y hacer del mundo un lugar mejor. Hay toneladas de métricas que te importan, y deberías medirlas todas (ver **Regla #2**). Sin embargo, al principio del proceso de aprendizaje automático, notará que todas suben, incluso aquellas que no optimiza directamente. Por ejemplo, supongamos que se preocupa por el número de clics, el tiempo de permanencia en el sitio y los usuarios activos diarios. Si optimiza el número de clics, es probable que vea aumentar el tiempo de permanencia.
Por lo tanto, manténgalo simple y no piense demasiado en equilibrar diferentes métricas cuando aún puede aumentar fácilmente todas las métricas. Pero no lleve esta regla demasiado lejos: no confunda su objetivo con la salud final del sistema (véase la regla nº 39). Y, **si te encuentras aumentando la métrica directamente optimizada, pero decides no lanzarla, puede que sea necesaria alguna revisión del objetivo.**

Regla nº 13: Elija una métrica sencilla, observable y atribuible para su primer objetivo. A menudo no se sabe cuál es el verdadero objetivo. Crees que lo sabes, pero luego, al observar los datos y los análisis paralelos de tu antiguo sistema y del nuevo sistema de ML, te das cuenta de que quieres modificarlo. Además, a menudo los distintos miembros del equipo no se ponen de acuerdo sobre el verdadero objetivo. **El objetivo de ML debe ser algo fácil de medir y un sustituto del "verdadero" objetivo**[2]. Así que entrénate en el objetivo simple de ML, y considera tener una "capa de política" encima que te permita añadir lógica adicional (con suerte, lógica muy simple) para hacer la clasificación final.

Lo más fácil de modelar es un comportamiento del usuario directamente observado y atribuible a una acción del sistema:

1. ¿Se ha hecho clic en este enlace clasificado?
2. ¿Se ha descargado este objeto clasificado?
3. ¿Se ha reenviado/respondido/enviado por correo electrónico este objeto clasificado?
4. ¿Se ha valorado este objeto clasificado?
5. ¿Se ha marcado este objeto como spam/pornografía/ofensivo? Evite

modelar efectos indirectos al principio:

1. ¿El usuario visitó el sitio al día siguiente?
2. ¿Cuánto tiempo visitó el usuario el sitio?
3. ¿Cuáles fueron los usuarios activos diarios?

Los efectos indirectos son una buena métrica y pueden utilizarse durante las pruebas A/B y durante las decisiones de lanzamiento.
Por último, no intentes que el aprendizaje automático lo averigüe:

1. ¿Está contento el usuario con el producto?
2. ¿Está satisfecho con la experiencia?
3. ¿Está el producto mejorando el bienestar general del usuario?

---

[2]A menudo no existe un objetivo "verdadero". Véase la regla nº 39.

4. ¿Cómo afectará a la salud general de la empresa?

Todo esto es importante, pero también increíblemente difícil. En su lugar, utilice aproximaciones: si el usuario está contento, permanecerá más tiempo en el sitio. Si el usuario está satisfecho, volverá a visitarnos mañana. En lo que respecta al bienestar y la salud de la empresa, se requiere el juicio humano para conectar cualquier objetivo aprendido por máquina con la naturaleza del producto que vendes y tu plan de negocio, para que no acabemos aquí.

Regla 14: Empezar con un modelo interpretable facilita la depuración.

La regresión lineal, la regresión logística y la regresión de Poisson están directamente motivadas por un modelo probabilístico. Cada predicción es interpretable como una probabilidad o un valor esperado. Esto hace que sean más fáciles de depurar que los modelos que utilizan objetivos (pérdida de cero uno, varias pérdidas de bisagra, etcétera) que intentan optimizar directamente la precisión de la clasificación o el rendimiento de la clasificación. Por ejemplo, si las probabilidades en el entrenamiento se desvían de las probabilidades predichas en los análisis secundarios o inspeccionando el sistema de producción, esta desviación podría revelar un problema.

Por ejemplo, en la regresión lineal, logística o de Poisson, **hay subconjuntos de los datos en los que la expectativa media predicha es igual a la etiqueta media (calibrada en 1momento, o simplemente calibrada)**[3]. Si tiene una característica que es 1 o 0 para cada ejemplo, entonces el conjunto de ejemplos donde esa característica es 1 está calibrado. Asimismo, si tiene una característica que es 1 para cada ejemplo, entonces el conjunto de todos los ejemplos está calibrado.

Con modelos sencillos, es más fácil tratar con bucles de retroalimentación (véase la regla 36).
A menudo, utilizamos estas predicciones probabilísticas para tomar una decisión: por ejemplo, clasificar los mensajes en función del valor esperado decreciente (es decir, la probabilidad de clic/descarga/etc.). **Sin embargo, recuerde que cuando llega el momento de elegir qué modelo utilizar, la decisión importa más que la probabilidad de los datos dado el modelo (véase la regla 27)**.

Regla nº 15: Separe el filtrado de spam y la clasificación de calidad en una capa de políticas.

La clasificación por calidad es un arte, pero el filtrado de spam es una guerra. Las señales que utilizas para determinar los mensajes de alta calidad serán obvias para aquellos que utilizan tu sistema, y ajustarán sus mensajes para que tengan estas propiedades. Por lo tanto, su clasificación de calidad debe centrarse en clasificar el contenido que se publica de buena fe. No debe descartar el aprendiz de clasificación de calidad para clasificar el spam. **Del mismo modo, el contenido "subido de tono" debe tratarse por separado de la clasificación de calidad**.
El filtrado de spam es otra historia. Debe contar con que las características que necesita generar cambiarán constantemente. A menudo, habrá reglas obvias que se introducen en el sistema (si un mensaje tiene más de tres votos de spam, no se recupera, etc.). Cualquier modelo aprendido tendrá que actualizarse a diario, si no más rápido. La reputación del creador del contenido desempeñará un gran papel.

En algún nivel, habrá que integrar los resultados de estos dos sistemas. Hay que tener en cuenta que el filtrado del spam en los resultados de las búsquedas debería ser probablemente más agresivo que el filtrado del spam en el correo electrónico

[3]Esto es cierto suponiendo que no haya regularización y que el algoritmo haya convergido. Es aproximadamente cierto en general.

mensajes. Además, es una práctica habitual eliminar el spam de los datos de entrenamiento para el clasificador de calidad.

Fase II de ML: Ingeniería de características

En la primera fase del ciclo de vida de un sistema de aprendizaje automático, lo importante es introducir los datos de entrenamiento en el sistema de aprendizaje, instrumentar cualquier métrica de interés y crear una infraestructura de servicio. **Después de tener un sistema de extremo a extremo con pruebas de unidad y de sistema instrumentadas, comienza la Fase II**.

En la segunda fase, hay mucha fruta al alcance de la mano. Hay una variedad de características obvias que podrían ser tirado en el sistema. Por lo tanto, la segunda fase del aprendizaje automático consiste en extraer tantas características como sea posible y combinarlas de forma intuitiva. Durante esta fase, todas las métricas deberían seguir aumentando. Habrá muchos lanzamientos, y es un gran momento para atraer a muchos ingenieros que puedan unir todos los datos que necesitas para crear un sistema de aprendizaje realmente impresionante.

Regla nº 16: Planifica el lanzamiento y la iteración.
No espere que el modelo en el que está trabajando ahora sea el último que lance, ni siquiera que vaya a dejar de lanzar modelos. Por tanto, considere si la complejidad que está añadiendo con este lanzamiento ralentizará futuros lanzamientos. Muchos equipos han lanzado un modelo por trimestre o más durante años. Hay tres razones básicas para lanzar nuevos modelos:

1. se crean nuevas funciones
2. está ajustando la regularización y combinando características antiguas de nuevas formas, y/o
3. ajustar el objetivo.

En cualquier caso, dar a un modelo un poco de amor puede ser bueno: examinar los datos que alimentan el ejemplo puede ayudar a encontrar nuevas señales, así como las viejas, rotas. Así que, cuando construya su modelo, piense en lo fácil que es añadir, eliminar o recombinar características. Piense en lo fácil que es crear una nueva copia del modelo y verificar su corrección. Piense si es posible tener dos o tres copias funcionando en paralelo. Por último, no te preocupes por si la función 16 de 35 entra en esta versión del canal. La tendrás el próximo trimestre.

Regla nº 17: Comience con características observadas y notificadas directamente, en lugar de con características aprendidas.
Puede que sea un punto controvertido, pero evita muchos escollos. En primer lugar, describamos qué es una característica aprendida. Una característica aprendida es una característica generada por un sistema externo (como un sistema de agrupación no supervisado) o por el propio alumno (por ejemplo, a través de un modelo factorizado o de aprendizaje profundo).

aprendizaje profundo). Ambos pueden ser útiles, pero pueden tener muchos problemas, por lo que no deberían estar en el primer modelo.

Si utiliza un sistema externo para crear una característica, recuerde que el sistema tiene su propio objetivo. Es posible que el objetivo del sistema externo sólo esté débilmente correlacionado con su objetivo actual. Si toma una instantánea del sistema externo, puede quedar desfasada. Si actualiza las características del sistema externo, los significados pueden cambiar. Si utiliza un sistema externo para proporcionar una característica, tenga en cuenta q u e requiere mucho cuidado.

El principal problema de los modelos factorizados y los modelos profundos es que no son convexos. Por lo tanto, no hay garantía de que se pueda aproximar o encontrar una solución óptima, y los mínimos locales encontrados en cada iteración pueden ser diferentes. Esta variación hace difícil juzgar si el impacto de un cambio en el sistema es significativo o aleatorio. Al crear un modelo sin características profundas, se puede obtener un excelente rendimiento de referencia. Una vez conseguida esta línea de base, puede probar enfoques más esotéricos.

Regla 18: Explore con características de contenido que se generalicen a través de contextos.
A menudo, un sistema de aprendizaje automático es una pequeña parte de una imagen mucho más amplia. Por ejemplo, si imaginas una publicación que podría utilizarse en What's Hot, muchas personas añadirán, volverán a compartir o comentarán una publicación antes de que aparezca en What's Hot. Si proporcionas esas estadísticas al aprendiz, puede promocionar nuevas publicaciones para las que no tiene datos en el contexto que está optimizando. YouTube Watch Next podría utilizar el número de visualizaciones o cowatches (recuentos de cuántas veces se ha visto un vídeo después de ver otro) de la búsqueda en YouTube. También puede utilizar valoraciones explícitas de los usuarios. Por último, si tienes una acción de usuario que estás utilizando como etiqueta, ver esa acción en el documento en un contexto diferente puede ser una gran característica. Todas estas funciones te permiten introducir nuevos contenidos en el contexto. Tenga en cuenta que no se trata de personalización: primero averigüe si a alguien le gusta el contenido en este contexto y luego averigüe a quién le gusta más o menos.

Regla 19: Utiliza funciones muy específicas siempre que puedas.
Con toneladas de datos, es más sencillo aprender millones de características simples que unas pocas características complejas. Los identificadores de los documentos que se recuperan y las consultas canonicalizadas no aportan mucha generalización, pero alinea tu clasificación con tus etiquetas en las cabeceras de las consultas.. Por lo tanto, no tenga miedo de los grupos de características en los que cada característica se aplica a una fracción muy pequeña de sus datos, pero la cobertura global es superior al 90%. Puede utilizar la regularización para eliminar las características que se aplican a muy pocos ejemplos.

Regla 20: Combine y modifique las funciones existentes para crear otras nuevas de forma comprensible.
Hay varias formas de combinar y modificar características. Los sistemas de aprendizaje automático como TensorFlow permiten preprocesar los datos mediante transformaciones. Los dos enfoques más estándar son las "discretizaciones" y los "cruces" .

La discretización consiste en tomar una característica continua y crear muchas características discretas a partir de ella. Considere una característica continua como la edad. Puede crear una característica que sea 1 cuando la edad es inferior a 18 años, otra característica que sea 1 cuando la edad está entre 18 y 35 años, etcétera. No piense demasiado en los límites de estos histogramas: los cuantiles básicos le darán la mayor parte del impacto.

Los cruces combinan dos o más columnas de características. Una columna de características, en la terminología de TensorFlow, es un conjunto de características homogéneas, (por ejemplo, {hombre, mujer}, {EE.UU., Canadá, México}, etcétera). Un cruce es una nueva columna de características con características en, por ejemplo, $\{male, female\} \times \{US, Canada, M\text{exico}\}$.
Esta nueva columna de características contendrá la característica (hombre, Canadá). Si está utilizando TensorFlow y le dice a TensorFlow que cree este cruce por usted, esta característica (hombre, Canadá) estará presente en los ejemplos que representen a hombres canadienses. Tenga en cuenta que se necesitan grandes cantidades de datos para aprender modelos con cruces de tres, cuatro o más columnas de características base.

Los cruces que producen columnas de características muy grandes pueden sobreajustarse. Por ejemplo, imagine que está realizando algún tipo de búsqueda y tiene una columna de características con palabras en la consulta y otra con palabras en el documento. Puede combinarlas con una cruz, pero acabará teniendo muchas características (véase la regla nº 21). Cuando se trabaja con texto hay dos alternativas. La más draconiana es el producto punto. En su forma más simple, un producto de puntos se limita a contar el número de palabras comunes entre la consulta y el documento. Esta característica puede discretizarse. Otro enfoque es la intersección: así, tendremos una característica que está presente si y sólo si la palabra "pony" está en el documento y en la consulta, y otra característica que está presente si y sólo si la palabra "the" está en el documento y en la consulta.

Regla 21: El número de ponderaciones de características que puede aprender en un modelo lineal es aproximadamente proporcional a la cantidad de datos que tiene.
Hay resultados fascinantes de la teoría del aprendizaje estadístico sobre el nivel adecuado de complejidad de un modelo, pero esta regla es básicamente todo lo que necesita saber. He tenido conversaciones en las que la gente dudaba de que se pudiera aprender algo a partir de mil ejemplos, o de que alguna vez se necesitaran más de un millón de ejemplos, porque se quedaban atascados en un determinado método de aprendizaje. La clave está en adaptar el aprendizaje al tamaño de los datos:

1. Si estás trabajando en un sistema de clasificación de búsquedas, y hay millones de palabras diferentes en los documentos y en la consulta, y tienes 1.000 ejemplos etiquetados, entonces deberías usar un producto punto entre las características del documento y la consulta, TFIDF, y media docena de otras características de ingeniería humana. 1000 ejemplos, una docena de características.
2. Si tiene un millón de ejemplos, intersecte las columnas de características del documento y de la consulta, utilizando regularización y posiblemente selección de características. Esto le dará millones de características, pero con la regularización tendrá menos. Diez millones de ejemplos, quizá cien mil características.
3. Si tiene miles o cientos de miles de millones de ejemplos, puede cruzar las columnas de características con los tokens del documento y la consulta, utilizando la selección de características y la regularización. Tendrá mil millones de ejemplos y 10 millones de características.

La teoría del aprendizaje estadístico rara vez proporciona límites estrictos, pero ofrece una gran orientación como punto de partida. Al final, utilice **la regla 28** para decidir qué características utilizar.

Regla #22: Limpie las características que ya no utiliza.
Las funciones no utilizadas crean deuda técnica. Si te das cuenta de que no utilizas una función y de que combinarla con otras no funciona, elimínala de tu infraestructura. Hay que mantener la infraestructura limpia para poder probar las funciones más prometedoras lo antes posible. Si es necesario, siempre se puede volver a añadir.

Ten en cuenta la cobertura cuando consideres qué funciones añadir o mantener. ¿Cuántos ejemplos cubre la función? Por ejemplo, si usted tiene algunas características de personalización, pero sólo el 8% de sus usuarios tienen alguna característica de personalización, no va a ser muy eficaz.

Al mismo tiempo, algunas funciones pueden estar por encima de sus posibilidades. Por ejemplo, si tienes una característica que sólo cubre el 1% de los datos, pero el 90% de los ejemplos que tienen la característica son positivos, entonces será una gran característica a añadir.

Análisis humano del sistema

Antes de pasar a la tercera fase del aprendizaje automático, es importante centrarse en algo que no se enseña en ninguna clase de aprendizaje automático: cómo analizar un modelo existente y mejorarlo. Esto es más un arte que una ciencia, y sin embargo hay varios antipatrones que ayuda a evitar.

Regla #23: Usted no es un usuario final típico.
Esta es quizá la forma más fácil de que un equipo se estanque. Aunque el fishfooding (usar un prototipo dentro de tu equipo) y el dogfooding (usar un prototipo dentro de tu empresa) tienen muchas ventajas, los empleados deben fijarse en si la actuación es correcta. Mientras que un cambio que es obviamente malo no debería utilizarse, cualquier cosa que parezca razonablemente cercana a la producción debería probarse más a fondo, bien pagando a legos para que respondan a preguntas en una plataforma de crowdsourcing, bien mediante un experimento en vivo con usuarios reales.

Hay dos razones para ello. La primera es que estás demasiado cerca del código. Puede que estés buscando un aspecto concreto de los mensajes, o que simplemente estés demasiado implicado emocionalmente (por ejemplo, sesgo de confirmación). La segunda es que tu tiempo es demasiado valioso. Considera el coste de 9 ingenieros sentados en una reunión de una hora, y piensa en cuántas etiquetas humanas contratadas compra eso en una plataforma de crowdsourcing.

Si realmente quieres tener feedback de los usuarios, **utiliza metodologías de experiencia de usuario**. Crea personajes de usuario (una descripción está en *Designing User Experiences* de Bill Buxton) al principio de un proceso y haz pruebas de usabilidad (una descripción está en *Don't Make Me* Think de Steve Krug) después. Los personajes de usuario consisten en crear un usuario hipotético. Por ejemplo, si en tu equipo todos son hombres, puede ser útil diseñar un personaje de usuario mujer de 35 años (con sus características) y observar los resultados que genera en lugar de 10 resultados para hombres de 25 a 40 años. Invitar a personas reales para

personas reales para que observen su reacción ante el sitio (de forma local o remota) en las pruebas de usabilidad.

Regla 24: Mida el delta entre modelos.
Una de las mediciones más fáciles, y a veces más útiles, que puede hacer antes de que ningún usuario haya visto su nuevo modelo es calcular la diferencia entre los nuevos resultados y los de producción. Por ejemplo, si tiene un problema de clasificación, ejecute ambos modelos en una muestra de consultas de todo el sistema y observe el tamaño de la diferencia simétrica de los resultados (ponderada por la posición en la clasificación). Si la diferencia es muy pequeña, sin necesidad de realizar un experimento se puede afirmar que habrá pocos cambios. Si la diferencia es muy grande, entonces querrá asegurarse de que el cambio es bueno. Revisar las consultas en las que la diferencia simétrica es alta puede ayudarle a entender cualitativamente cómo ha sido el cambio. Asegúrese, sin embargo, de que el sistema es estable. Asegúrese de que un modelo comparado consigo mismo tiene una diferencia simétrica baja (idealmente cero).

Regla nº 25: Al elegir modelos, el rendimiento utilitario triunfa sobre el poder predictivo. Su modelo puede intentar predecir la tasa de clics. Sin embargo, al final, la cuestión clave es qué se hace con esa predicción. Si lo utiliza para clasificar documentos, la calidad de la clasificación final es más importante que la predicción en sí. Si predice la probabilidad de que un documento sea spam y tiene un límite para bloquearlo, la precisión de lo que se permite es más importante. La mayor parte del tiempo, estas dos cosas deberían estar de acuerdo: cuando no lo estén, probablemente será por una pequeña ganancia. Así, si hay algún cambio que mejora la pérdida de registros pero degrada el rendimiento del sistema, busca otra característica. Cuando esto empiece a ocurrir más a menudo, es hora de revisar el objetivo de su modelo.

Regla 26: Busque patrones en los errores medidos y cree nuevas características.
Supongamos que ve un ejemplo de entrenamiento en el que el modelo se "equivocó". En una tarea de clasificación, podría tratarse de un falso positivo o un falso negativo. En una tarea de clasificación, podría tratarse de un par en el que un positivo se ha clasificado peor que un negativo. Lo más importante es que se trata de un ejemplo en el que el sistema de aprendizaje automático *sabe que se equivocó* y que le gustaría corregir si tuviera la oportunidad. Si le das al modelo una característica que le permita corregir el error, el modelo intentará utilizarla.

Por otro lado, si intentas crear una característica basada en ejemplos que el sistema no ve como errores, la característica será ignorada. Por ejemplo, supongamos que en Play Apps Search, alguien busca "juegos gratis". Supongamos que uno de los primeros resultados es una aplicación menos relevante. Así que creas una función para "aplicaciones de broma". Sin embargo, si quieres maximizar el número de instalaciones y la gente instala una aplicación de broma cuando busca juegos gratis, la función "aplicaciones de broma" no tendrá el efecto deseado.

Una vez que tenga ejemplos de que el modelo se equivocó, busque tendencias que estén fuera de su actual conjunto de funciones. Por ejemplo, si el sistema parece estar degradando los mensajes más largos, añade la longitud del mensaje. No seas demasiado específico sobre las características que añades. Si vas a añadir la longitud del post,

no intentes adivinar lo que significa largo, simplemente añade una docena de características y deja que el modelo decida qué hacer con ellas (ver Regla #21). Esa es la forma más fácil de conseguir lo que quieres.

Regla 27: Intente cuantificar el comportamiento indeseable observado.
Algunos miembros de su equipo comenzarán a sentirse frustrados con las propiedades del sistema que no les gustan y que no son capturadas por la función de pérdida existente. En este punto, deben hacer lo que sea necesario para convertir sus quejas en números sólidos. Por ejemplo, si creen que en Play Search se muestran demasiadas "aplicaciones de broma", pueden hacer que evaluadores humanos identifiquen las aplicaciones de broma. (Es factible utilizar datos etiquetados por humanos en este caso porque una fracción relativamente pequeña de las consultas representa una gran fracción del tráfico). Si tus problemas son medibles, entonces puedes empezar a utilizarlos como características, objetivos o métricas. La regla general es **"medir primero, optimizar después".**

Regla 28: Tenga en cuenta que un comportamiento idéntico a corto plazo no implica un comportamiento idéntico a largo plazo.
Imagina que tienes un nuevo sistema que mira cada doc_id y exact_query, y luego calcula la probabilidad de clic de cada doc para cada consulta. Descubres que su comportamiento es casi idéntico al de tu sistema actual tanto en las pruebas lado a lado como en las pruebas A/B, así que, dada su simplicidad, lo lanzas. Sin embargo, te das cuenta de que no se muestran nuevas aplicaciones. ¿Por qué? Bueno, como tu sistema sólo muestra un documento basándose en su propio historial con esa consulta, no hay forma de saber que debe mostrarse un nuevo documento.

La única manera de entender cómo funcionaría un sistema de este tipo a largo plazo es hacer que se entrene sólo con los datos adquiridos cuando el modelo estaba activo. Esto es muy difícil.

Desviación entrenamiento-servicio

La desviación entre entrenamiento y servicio es una diferencia entre el rendimiento durante el entrenamiento y el rendimiento durante el servicio. Esta desviación puede deberse a

- una discrepancia entre cómo se manejan los datos en los pipelines de entrenamiento y de servicio, o
- un cambio en los datos entre el momento en que se entrena y el momento en que se sirve, o
- un bucle de retroalimentación entre el modelo y el algoritmo.

Hemos observado sistemas de aprendizaje automático en producción en Google con un sesgo entre la formación y el servicio que afecta negativamente al rendimiento. La mejor solución es monitorizarlo explícitamente para que los cambios en el sistema y en los datos no introduzcan sesgos de forma inadvertida.

Regla #29: La mejor manera de asegurarse de que se entrena como se sirve es guardar el conjunto de características utilizadas en el momento de servir, y luego canalizar esas características a un registro para utilizarlas en el momento del entrenamiento.

Incluso si no puedes hacer esto para cada ejemplo, hazlo para una pequeña fracción, de manera que puedas verificar la consistencia entre el servicio y el entrenamiento (ver **Regla #37**). Los equipos que han realizado esta medición en Google a veces se han sorprendido por los resultados. La página de inicio de YouTube

cambió a funciones de registro en el momento de servir con importantes mejoras de calidad y una reducción de la complejidad del código, y muchos equipos están cambiando su infraestructura mientras hablamos.

Regla nº 30: Ponderar la importancia de los datos muestreados, ¡no abandonarlos arbitrariamente!
Cuando se tienen demasiados datos, existe la tentación de tomar los archivos 112 e ignorar los archivos 1399. Esto es un error: descartar datos en el entrenamiento ha causado problemas en el pasado a varios equipos (ver Regla #6). Aunque los datos que nunca se mostraron al usuario pueden descartarse, la ponderación de importancia es lo mejor para el resto. La ponderación de importancia significa que si decide que va a muestrear el ejemplo X con una probabilidad del 30%, entonces dele un peso de 10/3. Con la ponderación de importancia, todos los datos del ejemplo X se muestrean con una probabilidad del 30%. **Con la ponderación de importancia, todas las propiedades de calibración comentadas en la regla 14 siguen siendo válidas**.

Regla 31: Tenga en cuenta que si une datos de una tabla en el momento del entrenamiento y del servicio, los datos de la tabla pueden cambiar.
Digamos que une los ID de documentos con una tabla que contiene características de esos documentos (como el número de comentarios o clics). Entre el tiempo de entrenamiento y el tiempo de servicio, las características de la tabla pueden cambiar. La predicción de su modelo para el mismo documento puede diferir entre el entrenamiento y el servicio. La forma más sencilla de evitar este tipo de problema es registrar las características en el momento de servir (véase la regla 32). Si la tabla cambia lentamente, también puede tomar instantáneas de la tabla cada hora o cada día para obtener datos razonablemente aproximados. Tenga en cuenta que esto no resuelve completamente el problema.

Regla nº 32: Reutilice el código entre su canal de formación y su canal de servicio siempre que sea posible.
El procesamiento por lotes es diferente al procesamiento en línea. En el procesamiento en línea, debe manejar cada solicitud a medida que llega (por ejemplo, debe hacer una búsqueda separada para cada consulta), mientras que en el procesamiento por lotes, puede combinar tareas (por ejemplo, hacer una unión). A la hora de servir, estás haciendo procesamiento en línea, mientras que la formación es una tarea de procesamiento por lotes. Sin embargo, hay algunas cosas que puedes hacer para reutilizar código. Por ejemplo, puede crear un objeto específico para su sistema en el que el resultado de cualquier consulta o unión pueda almacenarse de forma legible para el ser humano, y los errores puedan comprobarse fácilmente. A continuación, una vez recopilada toda la información, durante el servicio o la formación, se ejecuta un método común para establecer un puente entre el objeto legible por humanos específico del sistema y el formato que espera el sistema de aprendizaje automático. **De este modo, se elimina una fuente de desviación en el servicio de formación**. Como corolario, intente no utilizar dos lenguajes de programación diferentes entre el entrenamiento y el servicio, ya que esa decisión hará casi imposible que comparta código.

Regla nº 33: Si elabora un modelo basado en los datos hasta el 5 de enero, pruebe el modelo con los datos del 6 de enero y posteriores.
En general, mida el rendimiento de un modelo en los datos recogidos después de los datos en los que entrenó el modelo, ya que esto refleja mejor lo que su sistema hará en la producción. Si produce un modelo basado en los datos hasta el 5 de enero, pruebe el modelo con los datos del 6 de enero. Es de esperar que el rendimiento no sea tan bueno con los nuevos datos, pero no debería ser radicalmente peor. Dado que puede haber efectos diarios, es posible que no pueda predecir la tasa media de clics o la tasa de conversión, pero sí el área bajo el modelo.

tasa de conversión, pero el área bajo la curva, que representa la probabilidad de dar al ejemplo positivo una puntuación más alta que a un ejemplo negativo, debería estar razonablemente cerca.

Regla nº 34: En la clasificación binaria para filtrado (como la detección de spam o la determinación de correos electrónicos interesantes), haga pequeños sacrificios a corto plazo en el rendimiento para obtener datos muy limpios. En una tarea de filtrado, los ejemplos marcados como negativos no se muestran al usuario. Supongamos que tiene un filtro que bloquea el 75% de los ejemplos negativos al servir. Podría verse tentado a extraer datos de entrenamiento adicionales de las instancias mostradas a los usuarios. Por ejemplo, si un usuario marca como spam un correo electrónico que su filtro ha dejado pasar, puede que quiera aprender de él.

Pero este enfoque introduce un sesgo de muestreo. Puede recopilar datos más limpios si, en lugar de servir, etiqueta el 1% de todo el tráfico como "retenido" y envía todos los ejemplos retenidos al usuario. Ahora su filtro bloquea al menos el 74% de los ejemplos negativos. Estos ejemplos retenidos pueden convertirse en sus datos de entrenamiento.

Tenga en cuenta que si su filtro bloquea el 95% de los ejemplos negativos o más, esto es menos viable. Aún así, si desea medir el rendimiento del servicio, puede hacer una muestra aún más pequeña (digamos 0,1% o 0,001%). Diez mil ejemplos son suficientes para estimar el rendimiento con bastante precisión.

Regla nº 35: Cuidado con el sesgo inherente a los problemas de clasificación.
Cuando cambia su algoritmo de clasificación de forma tan radical que aparecen resultados diferentes, en realidad ha cambiado los datos que su algoritmo va a ver en el futuro. Este tipo de sesgo aparecerá, y usted debe diseñar su modelo en torno a él. Existen varios enfoques diferentes. Todos ellos son formas de favorecer los datos que el modelo ya ha visto.

1. Aumente la regularización de las características que cubren más consultas en comparación con las características que sólo aparecen en una consulta. De esta forma, el modelo favorecerá las características que son específicas para una o unas pocas consultas frente a las características que se generalizan a todas las consultas. Este enfoque puede ayudar a evitar que resultados muy populares se filtren a consultas irrelevantes. Tenga en cuenta que esto se opone al consejo más convencional de tener más regularización en las columnas de características con más valores únicos.
2. Permita que las características sólo tengan pesos positivos. Así, cualquier característica buena será mejor que una característica "desconocida".
3. No tener características de sólo documento. Esta es una versión extrema de #1. Por ejemplo, incluso si una aplicación determinada es una descarga popular independientemente de la consulta, no querrás mostrarla en todas partes[4]. No tener funciones de "sólo documentos" simplifica las cosas.

[4]La razón por la que no quieres mostrar una aplicación popular específica en todas partes tiene que ver con la importancia de hacer *accesibles* todas las aplicaciones deseadas. Por ejemplo, si alguien busca "aplicación para ver pájaros", puede que descargue "angry birds", pero seguro que no era esa su intención. Mostrar una aplicación de este tipo podría mejorar la tasa de descargas, pero en última instancia dejaría insatisfechas las necesidades del usuario.

Regla 36: Evite los bucles de retroalimentación con funciones posicionales.
La posición de un contenido influye enormemente en la probabilidad de que el usuario interactúe con él. Si colocas una aplicación en la primera posición, se hará clic en ella más a menudo, y estarás convencido de que es más probable que se haga clic en ella. Una forma de abordar este problema es añadir características posicionales, es decir, características sobre la posición del contenido en la página. El modelo se entrena con características posicionales y aprende a dar más importancia, por ejemplo, a la característica "1ª posición". Así, el modelo da menos importancia a otros factores en los ejemplos con "1stposition=true". Entonces, al servir, no da a ninguna instancia la característica posicional, o les da a todas la misma característica por defecto, porque está puntuando candidatos *antes de* haber decidido el orden en que mostrarlos.

Tenga en cuenta que es importante mantener las características posicionales separadas del resto del modelo debido a la asimetría entre el entrenamiento y la prueba. Lo ideal es que el modelo sea la suma de una función de las características posicionales y una función del resto de las características. Por ejemplo, no cruce las características posicionales con ninguna característica del documento.

Regla #37: Medir el Sesgo de Entrenamiento/Servicio.
Hay varias cosas que pueden causar sesgo en el sentido más general. Además, se puede dividir en varias partes:

1. La diferencia entre el rendimiento en los datos de entrenamiento y los datos retenidos. En general, siempre existirá, y no siempre es mala.
2. La diferencia entre el rendimiento de los datos retenidos y los datos del "día siguiente". De nuevo, siempre existirá. **Debería ajustar su regularización para maximizar el rendimiento del día siguiente**. Sin embargo, grandes caídas en el rendimiento entre los datos retenidos y los del día siguiente pueden indicar que algunas características son sensibles al tiempo y posiblemente degradan el rendimiento del modelo.
3. La diferencia entre el rendimiento de los datos del "día siguiente" y los datos reales. Si se aplica un modelo a un ejemplo en los datos de entrenamiento y al mismo ejemplo en servicio, debería dar exactamente el mismo resultado (véase la regla nº 5). Por lo tanto, una discrepancia aquí probablemente indica un error de ingeniería.

Fase III del ML: Crecimiento lento, refinamiento de la optimización y modelos complejos

Habrá ciertos indicios de que la segunda fase está llegando a su fin. En primer lugar, sus ganancias mensuales empezarán a disminuir. Empezará a haber compensaciones entre las métricas: verá que algunas aumentan y otras disminuyen en algunos experimentos. Aquí es donde la cosa se pone interesante. Como las ganancias son más difíciles de conseguir, el aprendizaje automático tiene que ser más sofisticado.

Una advertencia: esta sección tiene más reglas bluesky que las secciones anteriores. Hemos visto a muchos equipos pasar por los tiempos felices de la Fase I y la Fase II de aprendizaje automático. Una vez alcanzada la Fase III, los equipos tienen que encontrar su propio camino.

Regla nº 38: No pierdas el tiempo en nuevas funciones si el problema son los objetivos no alineados.

A medida que sus mediciones se estanquen, su equipo empezará a analizar cuestiones que están fuera del alcance de los objetivos de su sistema actual de aprendizaje automático. Como se ha dicho antes, si los objetivos del producto no están cubiertos por el objetivo algorítmico existente, tendrá que cambiar su objetivo o los objetivos de su producto. Por ejemplo, puede optimizar los clics, los plusones o las descargas, pero tomar decisiones de lanzamiento basadas en parte en calificadores humanos.

Regla nº 39: Las decisiones de lanzamiento son un sustituto de los objetivos a largo plazo del producto.

Alice tiene una idea para reducir la pérdida logística de predecir instalaciones. Añade una función. La pérdida logística disminuye. Cuando hace un experimento en vivo, ve que la tasa de instalación aumenta. Sin embargo, cuando acude a una reunión de revisión del lanzamiento, alguien señala que el número de usuarios activos diarios desciende un 5%. El equipo decide no lanzar el modelo. Alice está decepcionada, pero ahora se da cuenta de que las decisiones de lanzamiento dependen de múltiples criterios, sólo algunos de los cuales se pueden optimizar directamente utilizando ML.

La verdad es que el mundo real no es mazmorras y dragones: no hay "puntos de impacto" que identifiquen la salud de tu producto. El equipo tiene que utilizar las estadísticas que recopila para intentar predecir eficazmente lo bueno que será el sistema en el futuro. Tienen que preocuparse por el compromiso, los usuarios activos de 1 día (DAU), los DAU de 30, los ingresos y el retorno de la inversión del anunciante. Estas métricas que se pueden medir en pruebas A/B en sí mismas son sólo una aproximación a objetivos a más largo plazo: satisfacer a los usuarios, aumentar el número de usuarios, satisfacer a los socios y obtener beneficios, que incluso entonces se podrían considerar aproximaciones para tener un producto útil y de alta calidad y una empresa próspera dentro de cinco años.

Las únicas decisiones de lanzamiento fáciles son aquellas en las que todas las métricas mejoran (o al menos no empeoran). Si el equipo tiene que elegir entre un sofisticado algoritmo de aprendizaje automático y una simple heurística, si la heurística simple hace un mejor trabajo en todas estas métricas, debe elegir la heurística. Además, no existe una clasificación explícita de todos los valores métricos posibles. En concreto, consideremos los dos escenarios siguientes:

Experimento	Usuarios activos diarios	Ingresos/día
A	1 millón	4 millones
B	2 millones	2 millones de dólares

Si el sistema actual es A, es improbable que el equipo cambie a B. Si el sistema actual es B, es improbable que el equipo cambie a A. Esto parece entrar en conflicto con el comportamiento racional: sin embargo, las predicciones de cambio de métricas pueden o no resultar, y por lo tanto hay un gran riesgo implicado en cualquiera de los dos cambios. Cada métrica cubre algún riesgo que preocupa al equipo.

Además, ninguna métrica cubre la preocupación última del equipo: "¿dónde estará mi producto dentro de cinco años?

Por otra parte, los individuos tienden a favorecer un objetivo que pueden optimizar directamente. La mayoría de las herramientas de aprendizaje automático favorecen este tipo de entorno. En un entorno de este tipo, un ingeniero que lanza nuevas funciones puede obtener un flujo constante de lanzamientos. Existe un tipo de aprendizaje automático, el aprendizaje multiobjetivo, que empieza a abordar este problema. Por ejemplo, se puede formular un problema de satisfacción de restricciones que tenga límites inferiores en cada métrica y optimice alguna combinación lineal de métricas. Sin embargo, incluso entonces, no todas las métricas se enmarcan fácilmente como objetivos de aprendizaje automático: si se hace clic en un documento o se instala una aplicación, es porque se ha mostrado el contenido. Pero es mucho más difícil averiguar por qué un usuario visita tu sitio. Predecir el éxito futuro de un sitio en su conjunto es una tarea de IAcompleta, tan difícil como la visión por ordenador o el procesamiento del lenguaje natural.

Regla 40: simplificar los conjuntos.
Los modelos unificados que toman características brutas y clasifican directamente el contenido son los más fáciles de depurar y comprender. Sin embargo, un conjunto de modelos (un "modelo" que combina las puntuaciones de otros modelos) puede funcionar mejor. **Para simplificar las cosas, cada modelo debe ser o bien un conjunto que sólo tome la entrada de otros modelos, o bien un modelo base que tome muchas características, pero no ambos.** Si tiene modelos sobre otros modelos que se han entrenado por separado, combinarlos puede dar lugar a un mal comportamiento.

Utilice un modelo simple para ensamblar que sólo tome la salida de sus modelos "base" como entrada. También querrá aplicar propiedades a estos modelos ensamblados. Por ejemplo, un aumento en la puntuación producida por un modelo base no debería disminuir la puntuación del conjunto. Además, es mejor que los modelos entrantes sean semánticamente interpretables (por ejemplo, calibrados) para que los cambios de los modelos subyacentes no confundan al modelo de conjunto. Además, **asegúrese de que el aumento de la probabilidad de predicción de un clasificador subyacente no disminuye la probabilidad de predicción del conjunto**.

Regla nº 41: Cuando el rendimiento se estanca, busque nuevas fuentes de información cualitativas que añadir en lugar de refinar las señales existentes.
Ha añadido información demográfica sobre el usuario. Ha añadido información sobre las palabras del documento. Has explorado plantillas y ajustado la regularización. No has visto un lanzamiento con más de un 1% de mejora en tus métricas clave en unos cuantos trimestres. ¿Y ahora qué?

Es hora de empezar a construir la infraestructura para características radicalmente diferentes, como el historial de documentos a los que este usuario ha accedido en el último día, semana o año, o datos de una propiedad diferente. Utiliza entidades de wikidata o algo interno de tu empresa (como el gráfico de conocimiento de Google). Utilice el aprendizaje profundo. Empieza a ajustar tus expectativas sobre cuánto retorno

de la inversión y amplíe sus esfuerzos en consecuencia. Como en cualquier proyecto de ingeniería, tienes que sopesar el beneficio de añadir nuevas funciones frente al coste de una mayor complejidad.

Regla nº 42: No espere que la diversidad, la personalización o la relevancia estén tan correlacionadas con la popularidad como cree.
La diversidad en un conjunto de contenidos puede significar muchas cosas, siendo la diversidad de la fuente del contenido una de las más comunes. Personalización implica que cada usuario obtiene sus propios resultados. Relevancia implica que los resultados para una consulta concreta son más apropiados para esa consulta que para cualquier otra. Así pues, estas tres propiedades se definen como diferentes de lo ordinario.

El problema es que lo ordinario tiende a ser difícil de superar.

Tenga en cuenta que si su sistema mide los clics, el tiempo de permanencia, las visitas, los +1, los reenvíos, etc., está midiendo **la popularidad** del contenido. Los equipos a veces intentan aprender un modelo personal con diversidad. Para personalizar, añaden características que permitirían al sistema personalizar (algunas características que representan el interés del usuario) o diversificar (características que indican si este documento tiene alguna característica en común con otros documentos devueltos, como el autor o el contenido), y descubren que esas características obtienen menos peso (o a veces un signo diferente) del que esperan.

Esto no significa que la diversidad, la personalización o la relevancia no sean valiosas. Como se señala en la regla anterior, puedes hacer postprocesamiento para aumentar la diversidad o la relevancia. Si ves que los objetivos a largo plazo aumentan, entonces puedes declarar que la diversidad/relevancia es valiosa, aparte de la popularidad. A continuación, puede seguir utilizando el postprocesamiento o modificar directamente el objetivo en función de la diversidad o la relevancia.

Regla nº 43: Sus amigos tienden a ser los mismos en los distintos productos. Tus intereses tienden a no serlo.
Los equipos de Google han conseguido mucha tracción al tomar un modelo que predice la cercanía de una conexión en un producto, y hacer que funcione bien en otro. Tus amigos son quienes son. Por otro lado, he visto a varios equipos luchar con las características de personalización a través de las divisiones de productos. Sí, parece que debería funcionar. Por ahora, no parece que funcione. Lo que a veces ha funcionado es utilizar datos brutos de una propiedad para predecir el comportamiento en otra. Además, tenga en cuenta que incluso saber que un usuario tiene un historial en otra propiedad puede ayudar. Por ejemplo, la presencia de actividad del usuario en dos productos puede ser indicativa por sí misma.

Trabajos relacionados

Existen muchos documentos sobre aprendizaje automático tanto en Google como fuera de la empresa.

- Curso acelerado aprendizaje automáticode : introducción al aprendizaje automático aplicado

- Aprendizaje automático: A Probabilistic Approach, de Kevin Murphy, para comprender el campo del aprendizaje automático.
- Practical Advice for the Analysis of Large, Complex Data Sets: un enfoque de ciencia de datos para pensar sobre conjuntos de datos.

- Deep Learning de Ian Goodfellow et al para el aprendizaje de modelos no lineales
- Documento de Google sobre la deuda técnica, que contiene muchos consejos generales.
- Documentación de Tensorflow

Agradecimientos

Gracias a David Westbrook, Peter Brandt, Samuel Ieong, Chenyu Zhao, Li Wei, Michalis Potamias, Evan Rosen, Barry Rosenberg, Christine Robson, James Pine, Tal Shaked, Tushar Chandra, Mustafa Ispir, Jeremiah Harmsen, Konstantinos Katsiapis, Glen Anderson, Dan Duckworth, Shishir Birmiwal, Gal Elidan, Su Lin Wu, Jaihui Liu, Fernando Pereira y Hrishikesh Aradhye por sus numerosas correcciones, sugerencias y ejemplos útiles para este documento. Gracias también a Kristen Lefevre, Suddha Basu y Chris Berg, que nos ayudaron con una versión anterior. Cualquier error, omisión u opinión impopular son de mi autoría.

Apéndice

En este documento hay varias referencias a productos de Google. Para proporcionar más contexto, a continuación ofrezco una breve descripción de los ejemplos más comunes.

Visión general de YouTube

YouTube es un servicio de streaming de vídeo. Tanto el equipo de YouTube Watch Next como el de YouTube Home Page utilizan modelos ML para clasificar las recomendaciones de vídeos. Watch Next recomienda vídeos para ver después del que se está reproduciendo, mientras que Home Page recomienda vídeos a los usuarios que navegan por la página de inicio.

Visión general de Google Play

Google Play cuenta con muchos modelos que resuelven diversos problemas. Las aplicaciones Play Search, Play Home Page Personalized Recommendations y "Users Also Installed" utilizan el aprendizaje automático.

Visión general de Google Plus

Google Plus utiliza el aprendizaje automático en diversas situaciones: clasificación de publicaciones en el "flujo" de publicaciones vistas por el usuario, clasificación de publicaciones "What's Hot" (publicaciones que son muy populares ahora), clasificación de personas que conoces, etc.