<https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/RMRSp/utilidad-de-los-modelos-predictivos-en-las-organizaciones>

MÚSICA]

[MÚSICA] Hola.

Bienvenidos al curso de introducción al Machine Learning.

En este primer video, definiremos en forma general que es Machine Learning.

A través de lo que se puede hacer,

se mostrarán algunos ejemplos de su aplicación y relevancia.

Para empezar de manera muy general,

Machine Learning usa datos u observaciones de realizaciones, basados en un fenómeno

de interés con comportamiento aleatorio para generalizar su funcionamiento.

Como resultado práctico, al aprender estas reglas de funcionamiento,

se puede predecir el resultado completo de una instancia del fenómeno

si se conoce parcialmente la información del mismo.

Es decir, se puede inferir un resultado completo a partir de información parcial.

Esto se suele definir como el proceso de aprender de la experiencia

para inferir comportamientos a futuro.

Por ejemplo,

después de realizar muchas veces el recorrido desde su casa a su oficina,

una persona puede estimar el tiempo que tardará en llegar en un día determinado.

Este tiempo puede depender de otras variables como si llueve o no,

la hora del día a la que inicia el recorrido, la temperatura, entre otras.

La experiencia de la persona lo puede llevar a atener en su mente

la generalización de cómo se producen las variables simultáneamente.

Como consecuencia, para un día determinado, observa si llueve, la hora,

la temperatura, etcétera y la persona

puede inferir cuánto se demora el recorrido al salir de su casa.

En Machine Learning,

la idea es registrar muchas observaciones como datos, luego un algoritmo

computacional puede procesar esos datos para aprender o generalizar el fenómeno.

De esta forma, una computadora podría aprender el tiempo de viaje,

dependiendo del valor de las otras variables de interés,

tal como lo haría la persona pero posiblemente, con mucha más precisión.

Una forma de entender este proceso es entender los datos como resultados de dos

tipos de variables.

Unas variables de entrada o inputs y unas variables de salida o outputs.

La diferencia entre los dos tipos de variables es que después de generalizar el

fenómeno, dados unos valores particulares para las variables de entrada,

se puede predecir o relacionar cuál es la salida correspondiente.

Por supuesto, muchos datos con posibles relaciones de entradas y salidas,

no son observadas.

Por ejemplo,

en los datos del tiempo de recorrido no hay una observación de un día que llueva,

salga a las 8 de la mañana y la temperatura sea de 39 grados centígrados.

El propósito de Machine Learning es aprender todas las relaciones

posibles de entradas con salidas incluso aquéllas que no son observadas.

[MÚSICA] Un ejemplo muy básico,

es relacionar imágenes digitales con su contenido.

Las variables de entrada son los niveles de intensidad de cada uno de los pixeles

de la imagen en formato RGB o tres capas de color.

Las variables de salida son relacionadas con el contenido.

Por ejemplo, si en la imagen se representa una flor,

una vez se implementa un algoritmo de Machine Learning,

será posible determinar si una imagen representa la flor o no la representa.

En este tipo de aplicaciones,

una máquina sería capaz de ver la imagen y descifrar su contenido.

Claramente, esto da la sensación de que las máquinas pueden aprender aunque

posiblemente esto sea una afirmación bastante [INAUDIBLE] En la aplicación de

algoritmos de Machine Learning, se aprende en el fenómeno de interés a partir de

observaciones particulares, se reconocen tres tipos de tareas fundamentales.

[MÚSICA] La primera es el aprendizaje supervisado, en

el cual los datos que se usan, contienen datos de las entradas y de las salidas.

El algoritmo debe construir un mapa que se pueda generalizar,

incluso posibles realizaciones no observadas.

El ejemplo anterior de reconocimiento del contenido de las imágenes

es un ejemplo típico de este caso.

La segunda es el aprendizaje no supervisado, que se usa

cuando no se tienen explícitamente registradas las variables de salida.

En este caso, existen varios problemas que se pueden resolver.

Por ejemplo, encontrar grupos de individuos agrupando las

observaciones por similitud, que se conoce detección de grupos o clusters.

Otro problema puede ser encontrar observaciones que son atípicas al resto,

lo cual puede representar una anomalía que se debe estudiar.

Esto se conoce como detección de datos atípico.

En el aprendizaje no supervisado, el propósito es inferir

las variables de salida asociadas a cada dato las cuales no han sido observadas.

Por ejemplo, a qué grupo pertenece el dato en el caso del clustering

o si el dato es atípico o no.

Una tercera tarea es el aprendizaje reforzado o Reinforcement Learning,

en donde los modelos supervisados se usan para encontrar decisiones dinámicas

óptimas que determinan como actuar dadas las condiciones actuales de un sistema.

En este curso nos concentraremos en el aprendizaje supervisado,

es decir, contamos con datos

que tienen tanto las entradas como las salidas y nuestro objetivo fundamental

es encontrar procedimientos de análisis de datos que generen mapas entre los

dos conjuntos con la menor probabilidad de cometer errores en las predicciones.

Las aplicaciones posibles de modelos supervisados son muchas.

El auge de estas metodologías está fundamentado en los numerosos casos de

éxito que han generado el uso de diferentes contactos, como el de negocios,

salud, finanzas, etcétera.

Por ejemplo, a partir de imágenes médicas

es posible mejorar el diagnóstico de ciertas enfermedades de manera automática.

O a partir de datos genéticos humanos se puede determinar el tratamiento exacto

que genera mayor beneficio para un paciente.

[MÚSICA] En diversos negocios

el uso de Machine Learning ha generado apreciables ventajas competitivas.

Principalmente, esto se ha dado usando los datos generados

por el uso de transacciones electrónicas a través de internet.

Así por ejemplo,

se podría determinar qué tipo de productos es más probable que compre un cliente.

Sistemas de recomendación o cuál es la probabilidad de que abandone el uso de

cierto servicio.

Detección de Churn.

En el sector financiero es importante poder predecir si el tomador de un

préstamo estará en capacidad de cumplir sus obligaciones de pago o no.

Probabilidad de Default.

[MÚSICA] Machine Learning ha soportado además el avance tecnológico de

muchas aplicaciones modernas incluyendo la inteligencia artificial,

en la cuál se busca que las máquinas puedan replicar las

decisiones cognitivas de las personas en ambientes limitados.

Por ejemplo,

sistemas complejos de seguridad urbana con mecanismos de reconocimiento facial,

usan algoritmos de Machine Learning para hacer sus predicciones.

Las posibilidades van mucho más allá de estos ejemplos.

La importancia del Machine Learning no está en su evidente belleza conceptual y

matemática, sino en la capacidad de producir soluciones a problemas

que generan valor para las organizaciones o para las personas en general.

[MÚSICA] Por otra parte,

los problemas de aprendizaje supervisado se pueden clasificar en dos tipos,

dependiendo de las características de variables de salida.

Primero, los problemas de regresión en donde el output o variable de respuesta

es una variable continua que se puede representar como un número real,

por ejemplo cuando se quiere predecir

el nivel de una represa de agua en función de variables climatológicas

o la nota que se obtiene en una clase en función de las horas de estudio semanal.

La segunda, son los problemas de clasificación en donde la variable de

respuesta es categórica.

Es decir, se busca predecir cuál clase particular de un grupo de opciones

pertenecen a observaciones.

Por ejemplo cuando se quiere determinar si una transacción de compra no presencial

como el uso de tarjetas de crédito en compras por internet,

es un fraude o no lo es.

O también como en el ejemplo you visto,

cuando se busca predecir si la imagen es de una flor o no lo es.

Las clases son sí es o no es.

Esta separación en los tipos de problemas es útil para el desarrollo y comprensión

de ciertos algoritmos.

Así como para generar diferentes métricas que midan el desempeño de los mismos.

[MÚSICA] En este video vimos una introducción al Machine Learning

en particular al aprendizaje supervisado que estudiaremos en el curso.

En el próximo video daremos los conceptos fundamentales para comprender y

desarrollar algoritmos en problemas de clasificación y de regresión.

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

https://www.coursera.org/learn/introduccion-al-machine-learning/supplement/PQUFg/modelos-de-machine-learning-en-contextos-supervisados

**Modelos de Machine Learning en contextos supervisados**

El aprendizaje automático o machine learning, es una disciplina que se encarga de diseñar e implementar técnicas computacionales para que las máquinas “aprendan”. Por ejemplo, se puede entrenar a una máquina para que aprenda a:

1. Identificar los grupos de clientes a los que se les debe aplicar la misma estrategia de marketing.
2. Agrupar imágenes que tengan similitud entre ellas.
3. Identificar si una transacción bancaria con ciertas características es fraudulenta o no.
4. Identificar si en una imagen aparece un perro o no.
5. Predecir, a partir de información histórica, las ventas de una empresa para cierto mes.

Este aprendizaje se produce como resultado de la observación de muchas realizaciones pasadas en las cuales se ve el resultado de la variable de interés en función de otras variables que se usan como predictores. Estas observaciones se registran como datos y se usan para generalizar el comportamiento del fenómeno real.

En general, los contextos para aplicar aprendizaje automático se pueden dividir en problemas no supervisados y supervisados. En particular, los problemas uno y dos del listado anterior corresponden a problemas no supervisados, ya que los datos no tienen etiquetas (para cada cliente o cada imagen) que establezcan a qué grupo deben pertenecer, sino que la pretensión que se tiene es que precisamente sea la máquina quien sea capaz de generar estas relaciones entre clientes o imágenes sin que el modelador las conozca a priori.

Por otro lado, los ejemplos tres al cinco del listado corresponden a problemas supervisados, ya que sí se debe contar con información histórica sobre la etiqueta de cada observación, etiqueta a la que formalmente llamaremos “variable de respuesta”. Es decir, para el tercer problema del listado, se debe tener un registro histórico de transacciones que fueron fraudulentas y transacciones que no lo fueron y sus características (información transaccional); para el cuarto, registros históricos de imágenes en las que sepamos si hay un perro o no y sus características (pixeles de la imagen); y para el quinto, información histórica del valor de las ventas mensuales de la empresa. De esta forma, se entrena al algoritmo con estos datos para que reconozca las relaciones entre las características y la variable de respuesta de cada contexto (fraude o no, hay un perro en la imagen o no, y el valor de las ventas) para posteriormente, con datos distintos a los de entrenamiento, evaluar si el algoritmo aprendió correctamente. Así las cosas, en este curso nos encargaremos de implementar machine learning en problemas supervisados.

A su vez, los problemas supervisados se pueden dividir en dos tipos: problemas de regresión y problemas de clasificación. En los problemas de regresión, se quiere que la máquina o algoritmo realice predicciones sobre una variable de respuesta de naturaleza continua, mientras que en los problemas de clasificación se busca que la máquina realice predicciones sobre una variable de respuesta de naturaleza discreta. De esta manera, el tercer y cuarto ejemplo corresponden a problemas de clasificación (pues se habla de las variables de respuesta discretas de pertenencia o no a un grupo) y el quinto ejemplo se trata de un problema de regresión, ya que la variable de respuesta puede tomar cualquier valor en una recta numérica continua (cualquier valor real positivo).

Formalmente, que una máquina aprenda se refiere a la implementación de técnicas estadísticas, matemáticas y de optimización que permitan obtener la mejor aproximación posible de una función que relaciona la información de entrada (variables predictoras) con la salida de interés (variable de respuesta). Así, algebraicamente, se tiene el siguiente paradigma en un problema de regresión: yi=f(xi1,xi2,xi3,xi4,xi5,…,xik)+ϵiyi​=f(xi1​,xi2​,xi3​,xi4​,xi5​,…,xik​)+ϵi​

Donde yiyi​ es la observación ii de la variable de respuesta, xijxij​ corresponde a la observación ii de la variable predictora jj, ϵiϵi​ corresponde al error aleatorio de la observación ii, y f()f() es la función (desconocida) que relaciona las variables predictoras con la variable de respuesta. En particular, se busca que la máquina aproxime de la mejor forma a la función f()f() por medio de técnicas estadísticas y de optimización, estimación que representaremos como f^()f^​().

Por otro lado, en un problema de clasificación se busca estimar la función g()g() que aproxima la relación entre las variables predictoras xijxij​ y la clase a la que pertenece la variable yiyi​. Generalmente, para crear esta relación, un algoritmo supervisado de clasificación calcula la probabilidad de que yiyi​ pertenezca a la clase de interés (que por notación se escribirá como (yi=1)(yi​=1) o no (yi=0)(yi​=0)). Es decir:

P(yi=1∣xi1,xi2,…,xik)=1−P(yi=0∣xi1,xi2,…,xik)=g(xi1,xi2,…,xik)P(yi​=1∣xi1​,xi2​,…,xik​)=1−P(yi​=0∣xi1​,xi2​,…,xik​)=g(xi1​,xi2​,…,xik​)

Siguiendo la misma notación que se presentó anteriormente, la estimación realizada se representará como g^()g^​()

<https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/bBDoW/principios-generales-de-machine-learning>

Principios generales de Machine Learning

Hola. En este video explicaremos los principios fundamentales

que soportan los modelos y algoritmos usados para el aprendizaje supervisado,

es decir, para predecir una variable de interés.

Recordemos que en el aprendizaje supervisado se tiene un conjunto de variables de

entrada o inputs que se quieren

relacionar con una o varias variables de salida o outputs.

La idea de construir esta relación es que para cada

realización posible de las entradas

se pueda asignar un valor correspondiente de la salida.

Si se conoce esta relación, es

posible predecir la salida para un conjunto particular de entradas,

que es justamente la habilidad de predecir.

De manera general, a las entradas las llamaremos X,

las cuales para cada individuo observado son un grupo de p valores.

Por ejemplo, el i décimo individuo tiene observaciones x\_1i hasta x\_pi.

En el caso de las imágenes para reconocer flores que vimos en el video anterior,

x\_ ji representa el valor de intensidad en el i décimo pixel de la j décima imagen.

Supongamos que X es el

conjunto de todas las observaciones posibles de variables de entrada.

Si, por ejemplo, todas estas variables son registros numéricos,

entonces X está contenido en los reales en p dimensión.

Asimismo, podemos llamar Y el espacio

de todos los posibles valores de la variable de salida.

Puesto así, la relación mencionada corresponde al concepto de función,

la cual es una regla que para cada observación o

punto en X se debe asociar un punto en Y.

En este sentido, se define el propósito del aprendizaje

supervisado como el de estimar o aprender una función.

Usaremos ayudas visuales que permitan representar esta función con dominio en X,

que se diferenciará a la regresión o clasificación.

Por ejemplo, para regresión usaremos la representación en números reales,

en la cual se crea una relación en un plano cartesiano, como muestra la figura.

Para un punto particular x estrella,

el output correspondiente es f de x estrella.

Para el problema de clasificación,

puede ser más diciente una representación de la partición en X

que genera la función que va hacia el espacio de salida de Y. Es decir,

dado que es categórico, por ejemplo,

sí o no, podemos representar la función como muestra gráfica.

Los puntos en X relacionados con sí y no,

respectivamente, generan una partición en dos.

Aunque ambas representaciones implican que los inputs son variables continuas,

como números reales,

esto no es necesario para construir los modelos predictivos.

Ahora bien, el objetivo es aprender la función

f que va X a Y. ¿Cómo lo podemos aprender?

La respuesta se encuentra en el tipo de información

que tenemos para llegar a n. En el caso de regresión,

por ejemplo, el problema se ve así.

Lo que tenemos son observaciones particulares,

pasadas, de parejas en entradas y salidas.

Esto lo podemos representar como x\_i,

y\_i para i igual 1,

2 hasta n. Es decir,

n es el número de observaciones que tenemos.

En general, el espacio X es demasiado grande para observar todas las parejas posibles.

Lo cierto es que observamos una porción mínima de dupla.

El problema es entonces reconstruir la función para todo el espacio x,

aunque solo se tenga esta información parcial.

Esto con el fin de que el modelo se pueda usar en nuevos valores de las variables x.

Este problema de completar la función en todo el

espacio se conoce como el problema de generalización,

dado que a partir de un número finito de

observaciones se busca generalizar el comportamiento del fenómeno completo.

Sin embargo, la solución al problema no es

interpolar o unir los puntos para generar una función continua.

Falta un ingrediente fundamental para construir buenos algoritmos predictivos,

que es justamente lo que dificulta el aprendizaje de la función.

Ese elemento que faltaba considerar es la aleatoriedad de los datos.

Por ejemplo, en el caso de regresión,

supongamos que se quiere predecir el salario que

recibe alguien que acaba de obtener su título de maestría.

Esa es la variable en y,

en función de su promedio acumulado,

GPA, y de su edad.

Supongamos que la muestra observada tiene dos

personas con exactamente el mismo GPA y la misma edad.

Esto es, tienen el mismo registro para las entradas x\_1 y x\_2.

¿Deben entonces tener la misma observación en y?

La respuesta natural es: no.

Existe variabilidad inherente en los datos,

que no es controlable o explicable.

Este es el comportamiento natural de los fenómenos de interés que queremos predecir.

Para un valor de x,

en las entradas no se observa un único valor de y.

Esto parece contradecir el concepto de función para un modelo predictivo.

La realidad de estos modelos es que lo que se puede aprender es una función.

Sin embargo, esta única predicción para x no

necesariamente coincide con la respuesta que se va a observar.

En regresión, por ejemplo,

lo más probable es que esta coincidencia no se dé.

La idea de los modelos predictivos es construir

funciones que para cada x relacionen un valor de y,

de forma tal que la probabilidad de equivocarse en la predicción,

en los casos de clasificación,

o que la distancia media con observación en el

caso de regresión sea lo más pequeñas posible.

Una de las observaciones fundamentales que necesitamos para construir modelos

predictivos es que los datos son

realizaciones de variables aleatorias idénticamente distribuidas y,

por lo general, independientes.

Así, se asume que cada observación es una pareja x,

y seleccionada aleatoriamente de un modelo con distribución de probabilidad p de x y.

Gráficamente, en el problema de regresión,

pongamos en el eje x al promedio y en el eje y al salario del recién egresado.

El conjunto de puntos de la muestra se ve así.

Ahora pensemos en el primer dato como una realización de las variables aleatorias x,

y. Para el segundo dato,

independiente del resultado del primero,

se repite una realización de las mismas variables aleatorias y así sucesivamente.

El principio fundamental que vamos a estudiar es que en todos los datos,

el proceso aleatorio está regido por las mismas reglas de probabilidad p de x,

y. Este es el mismo supuesto fundamental que se hace en estadística,

aunque generalmente no con fines de predicción.

Esta idea nos permite formalizar las bases

matemáticas y numéricas para construir y evaluar algoritmos predictivos.

Antes de formalizar los modelos y su evaluación,

hablaremos de un concepto fundamental para predecir bien,

algo que podemos definir como flexibilidad en la función estimada.

Los fenómenos reales, por lo general,

implican relaciones complejas y llenas de detalles.

Es difícil que una función muy simple logre ser buen modelo predictivo.

Un modelo flexible debe permitir que

la función estimada se adapte a las irregularidades de la vida real.

Por otro lado, un modelo muy flexible puede predecir bien,

pero puede ser difícil de interpretar.

Pongamos un ejemplo concreto.

x es el precio de una camisa y y es el número de unidades vendidas por mes.

Supongamos que las unidades que se observan son diferentes tiendas de ropa.

Por supuesto, existe una relación entre estas variables

relacionadas con la elasticidad de la demanda.

Si con los datos se estima un modelo de regresión lineal,

y igual a Beta\_0 más Beta\_1 por x más el error aleatorio,

entonces con el parámetro

Beta\_1 se puede estimar la elasticidad del precio de las camisas.

Con esto se pueden tomar decisiones

estratégicas conociendo si es un bien altamente elástico o no.

Se puede estimar cuánto cae el número de camisas vendidas,

si el precio va a aumentar el 10

por ciento debido a un incremento en el costo de la tela.

Este modelo es altamente interpretativo,

dado que todo se concentra en un solo parámetro de interés.

Este tipo de modelos en los cuales la función que relaciona

a x y y se puede comprender en términos de pocos parámetros lo llamamos interpretables.

El efecto de la variable x se conoce en términos de la pendiente.

Sin embargo, es muy posible que en términos predictivos este no sea el mejor modelo.

La vida real se comporta de una manera más compleja y,

seguramente, no es representable por una línea recta.

Existen particularidades que a veces pueden ser sutiles.

Por ejemplo, si el precio es 2,99 en lugar de 3,

existe un efecto de percepción que introduce no linealidad,

que, por cierto, es usado repetidamente por almacenes de retail.

Un modelo que seguramente predice mejor debe ser más flexible.

Por ejemplo, el famoso algoritmo de k vecinos, que funciona así.

Para un x estrella sobre el cual se quiere predecir,

se buscan los k x\_i en la muestra que sean más cercanos a x estrella.

Y la predicción es f de x estrella

como el promedio de las y correspondientes a las x seleccionadas.

Esto se conoce como un promedio local.

Al mover x estrella,

la función f puede ser muy flexible.

Entre más pequeño sea k, más flexible la función.

k puede tomar valores enteros entre 1 y n. Si k es igual a n,

la función es completamente plana, poco flexible.

Mientras si k es igual a 1,

la función está en su grado más alto de flexibilidad,

en donde f de x\_i es igual a y\_i.

Si bien la función de k vecinos puede predecir mucho mejor,

está definida punto a punto y no tiene una forma analítica que se pueda interpretar.

Aunque se pueda visualizar,

no será posible explicar el efecto de las variables predictoras x.

En general, los modelos que mejor predicen no son interpretables,

por el alto grado de flexibilidad que requieren.

La flexibilidad, sin embargo, tiene un precio.

Si tenemos un modelo muy flexible,

como, por ejemplo, un k muy pequeño,

en el caso de k vecinos,

es posible que no se prediga bien.

Esto porque la función estimada se va

a parecer mucho a los datos observados, casi como interpolación.

No obstante, para un dato nuevo,

esa predicción puede ser errónea.

Este problema se conoce como sobreajuste o overfitting.

Pongamos el siguiente caso gráfico de regresión.

Supongamos que la función que mejor predice es la que se dibuja en color verde.

Ahora, pensemos en el algoritmo k vecinos.

Si k es muy grande, la función estimada es muy suave,

como lo muestra la curva de color azul.

Esto se conoce como sesgo,

el mismo concepto de la estadística paramétrica.

Para un k pequeño, se obtiene la curva de color roja,

la cual, como ya vimos, tiene overfitting.

Si ampliamos la imagen para un punto particular x estrella,

ni la función con sesgo ni la función con overfitting predicen bien.

Recordemos que la idea es predecir un conjunto nuevo.

k es un parámetro con el cual se puede cambiar la flexibilidad de la función estimada.

En general, todos los algoritmos de machine learning para

aprendizaje supervisado se pueden entender así

: un procedimiento que estima una función del espacio x al espacio y,

para el cual es posible ajustar el grado de flexibilidad.

El proceso de encontrar el grado óptimo de flexibilidad de un algoritmo

para unos datos particulares se conoce como calibración.

Una representación de ese problema es como un control de flexibilidad,

tune, en el cual se debe buscar el punto óptimo para la función.

Casi todos los algoritmos, sino todos,

pueden ser vistos desde este marco conceptual genérico.

Adicionalmente, el efecto del número de datos o tamaño de muestra es interesante.

Para un k pequeño, el algoritmo ya no hace overfitting,

dado que el promedio estabiliza la variabilidad.

Es decir, entre más datos,

el algoritmo va predecir mejor,

porque se puede tener más flexibilidad.

El grado de flexibilidad óptima depende del tamaño de la muestra.

Esto dictamina qué tanta flexibilidad se puede comprar.

En este video comprendimos los

principios básicos de los algoritmos de machine learning que sirven para predecir,

teniendo en cuenta que los datos son realizaciones de variables aleatorias.

Además, vimos que el concepto de la

flexibilidad es fundamental para que la función estimada prediga bien,

que el grado de flexibilidad que podemos

comprar está determinado por el tamaño de la muestra.

En el próximo video,

formalizaremos la idea de la flexibilidad y

desarrollaremos las métricas necesarias para evaluar

y calibrar modelos supervisados de machine learning.

<https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/bBDoW/principios-generales-de-machine-learning>

Principios generales de Machine Learning

Hola. En este video explicaremos los principios fundamentales

que soportan los modelos y algoritmos usados para el aprendizaje supervisado,

es decir, para predecir una variable de interés.

Recordemos que en el aprendizaje supervisado se tiene un conjunto de variables de

entrada o inputs que se quieren

relacionar con una o varias variables de salida o outputs.

La idea de construir esta relación es que para cada

realización posible de las entradas

se pueda asignar un valor correspondiente de la salida.

Si se conoce esta relación, es

posible predecir la salida para un conjunto particular de entradas,

que es justamente la habilidad de predecir.

De manera general, a las entradas las llamaremos X,

las cuales para cada individuo observado son un grupo de p valores.

Por ejemplo, el i décimo individuo tiene observaciones x\_1i hasta x\_pi.

En el caso de las imágenes para reconocer flores que vimos en el video anterior,

x\_ ji representa el valor de intensidad en el i décimo pixel de la j décima imagen.

Supongamos que X es el

conjunto de todas las observaciones posibles de variables de entrada.

Si, por ejemplo, todas estas variables son registros numéricos,

entonces X está contenido en los reales en p dimensión.

Asimismo, podemos llamar Y el espacio

de todos los posibles valores de la variable de salida.

Puesto así, la relación mencionada corresponde al concepto de función,

la cual es una regla que para cada observación o

punto en X se debe asociar un punto en Y.

En este sentido, se define el propósito del aprendizaje

supervisado como el de estimar o aprender una función.

Usaremos ayudas visuales que permitan representar esta función con dominio en X,

que se diferenciará a la regresión o clasificación.

Por ejemplo, para regresión usaremos la representación en números reales,

en la cual se crea una relación en un plano cartesiano, como muestra la figura.

Para un punto particular x estrella,

el output correspondiente es f de x estrella.

Para el problema de clasificación,

puede ser más diciente una representación de la partición en X

que genera la función que va hacia el espacio de salida de Y. Es decir,

dado que es categórico, por ejemplo,

sí o no, podemos representar la función como muestra gráfica.

Los puntos en X relacionados con sí y no,

respectivamente, generan una partición en dos.

Aunque ambas representaciones implican que los inputs son variables continuas,

como números reales,

esto no es necesario para construir los modelos predictivos.

Ahora bien, el objetivo es aprender la función

f que va X a Y. ¿Cómo lo podemos aprender?

La respuesta se encuentra en el tipo de información

que tenemos para llegar a n. En el caso de regresión,

por ejemplo, el problema se ve así.

Lo que tenemos son observaciones particulares,

pasadas, de parejas en entradas y salidas.

Esto lo podemos representar como x\_i,

y\_i para i igual 1,

2 hasta n. Es decir,

n es el número de observaciones que tenemos.

En general, el espacio X es demasiado grande para observar todas las parejas posibles.

Lo cierto es que observamos una porción mínima de dupla.

El problema es entonces reconstruir la función para todo el espacio x,

aunque solo se tenga esta información parcial.

Esto con el fin de que el modelo se pueda usar en nuevos valores de las variables x.

Este problema de completar la función en todo el

espacio se conoce como el problema de generalización,

dado que a partir de un número finito de

observaciones se busca generalizar el comportamiento del fenómeno completo.

Sin embargo, la solución al problema no es

interpolar o unir los puntos para generar una función continua.

Falta un ingrediente fundamental para construir buenos algoritmos predictivos,

que es justamente lo que dificulta el aprendizaje de la función.

Ese elemento que faltaba considerar es la aleatoriedad de los datos.

Por ejemplo, en el caso de regresión,

supongamos que se quiere predecir el salario que

recibe alguien que acaba de obtener su título de maestría.

Esa es la variable en y,

en función de su promedio acumulado,

GPA, y de su edad.

Supongamos que la muestra observada tiene dos

personas con exactamente el mismo GPA y la misma edad.

Esto es, tienen el mismo registro para las entradas x\_1 y x\_2.

¿Deben entonces tener la misma observación en y?

La respuesta natural es: no.

Existe variabilidad inherente en los datos,

que no es controlable o explicable.

Este es el comportamiento natural de los fenómenos de interés que queremos predecir.

Para un valor de x,

en las entradas no se observa un único valor de y.

Esto parece contradecir el concepto de función para un modelo predictivo.

La realidad de estos modelos es que lo que se puede aprender es una función.

Sin embargo, esta única predicción para x no

necesariamente coincide con la respuesta que se va a observar.

En regresión, por ejemplo,

lo más probable es que esta coincidencia no se dé.

La idea de los modelos predictivos es construir

funciones que para cada x relacionen un valor de y,

de forma tal que la probabilidad de equivocarse en la predicción,

en los casos de clasificación,

o que la distancia media con observación en el

caso de regresión sea lo más pequeñas posible.

Una de las observaciones fundamentales que necesitamos para construir modelos

predictivos es que los datos son

realizaciones de variables aleatorias idénticamente distribuidas y,

por lo general, independientes.

Así, se asume que cada observación es una pareja x,

y seleccionada aleatoriamente de un modelo con distribución de probabilidad p de x y.

Gráficamente, en el problema de regresión,

pongamos en el eje x al promedio y en el eje y al salario del recién egresado.

El conjunto de puntos de la muestra se ve así.

Ahora pensemos en el primer dato como una realización de las variables aleatorias x,

y. Para el segundo dato,

independiente del resultado del primero,

se repite una realización de las mismas variables aleatorias y así sucesivamente.

El principio fundamental que vamos a estudiar es que en todos los datos,

el proceso aleatorio está regido por las mismas reglas de probabilidad p de x,

y. Este es el mismo supuesto fundamental que se hace en estadística,

aunque generalmente no con fines de predicción.

Esta idea nos permite formalizar las bases

matemáticas y numéricas para construir y evaluar algoritmos predictivos.

Antes de formalizar los modelos y su evaluación,

hablaremos de un concepto fundamental para predecir bien,

algo que podemos definir como flexibilidad en la función estimada.

Los fenómenos reales, por lo general,

implican relaciones complejas y llenas de detalles.

Es difícil que una función muy simple logre ser buen modelo predictivo.

Un modelo flexible debe permitir que

la función estimada se adapte a las irregularidades de la vida real.

Por otro lado, un modelo muy flexible puede predecir bien,

pero puede ser difícil de interpretar.

Pongamos un ejemplo concreto.

x es el precio de una camisa y y es el número de unidades vendidas por mes.

Supongamos que las unidades que se observan son diferentes tiendas de ropa.

Por supuesto, existe una relación entre estas variables

relacionadas con la elasticidad de la demanda.

Si con los datos se estima un modelo de regresión lineal,

y igual a Beta\_0 más Beta\_1 por x más el error aleatorio,

entonces con el parámetro

Beta\_1 se puede estimar la elasticidad del precio de las camisas.

Con esto se pueden tomar decisiones

estratégicas conociendo si es un bien altamente elástico o no.

Se puede estimar cuánto cae el número de camisas vendidas,

si el precio va a aumentar el 10

por ciento debido a un incremento en el costo de la tela.

Este modelo es altamente interpretativo,

dado que todo se concentra en un solo parámetro de interés.

Este tipo de modelos en los cuales la función que relaciona

a x y y se puede comprender en términos de pocos parámetros lo llamamos interpretables.

El efecto de la variable x se conoce en términos de la pendiente.

Sin embargo, es muy posible que en términos predictivos este no sea el mejor modelo.

La vida real se comporta de una manera más compleja y,

seguramente, no es representable por una línea recta.

Existen particularidades que a veces pueden ser sutiles.

Por ejemplo, si el precio es 2,99 en lugar de 3,

existe un efecto de percepción que introduce no linealidad,

que, por cierto, es usado repetidamente por almacenes de retail.

Un modelo que seguramente predice mejor debe ser más flexible.

Por ejemplo, el famoso algoritmo de k vecinos, que funciona así.

Para un x estrella sobre el cual se quiere predecir,

se buscan los k x\_i en la muestra que sean más cercanos a x estrella.

Y la predicción es f de x estrella

como el promedio de las y correspondientes a las x seleccionadas.

Esto se conoce como un promedio local.

Al mover x estrella,

la función f puede ser muy flexible.

Entre más pequeño sea k, más flexible la función.

k puede tomar valores enteros entre 1 y n. Si k es igual a n,

la función es completamente plana, poco flexible.

Mientras si k es igual a 1,

la función está en su grado más alto de flexibilidad,

en donde f de x\_i es igual a y\_i.

Si bien la función de k vecinos puede predecir mucho mejor,

está definida punto a punto y no tiene una forma analítica que se pueda interpretar.

Aunque se pueda visualizar,

no será posible explicar el efecto de las variables predictoras x.

En general, los modelos que mejor predicen no son interpretables,

por el alto grado de flexibilidad que requieren.

La flexibilidad, sin embargo, tiene un precio.

Si tenemos un modelo muy flexible,

como, por ejemplo, un k muy pequeño,

en el caso de k vecinos,

es posible que no se prediga bien.

Esto porque la función estimada se va

a parecer mucho a los datos observados, casi como interpolación.

No obstante, para un dato nuevo,

esa predicción puede ser errónea.

Este problema se conoce como sobreajuste o overfitting.

Pongamos el siguiente caso gráfico de regresión.

Supongamos que la función que mejor predice es la que se dibuja en color verde.

Ahora, pensemos en el algoritmo k vecinos.

Si k es muy grande, la función estimada es muy suave,

como lo muestra la curva de color azul.

Esto se conoce como sesgo,

el mismo concepto de la estadística paramétrica.

Para un k pequeño, se obtiene la curva de color roja,

la cual, como ya vimos, tiene overfitting.

Si ampliamos la imagen para un punto particular x estrella,

ni la función con sesgo ni la función con overfitting predicen bien.

Recordemos que la idea es predecir un conjunto nuevo.

k es un parámetro con el cual se puede cambiar la flexibilidad de la función estimada.

En general, todos los algoritmos de machine learning para

aprendizaje supervisado se pueden entender así

: un procedimiento que estima una función del espacio x al espacio y,

para el cual es posible ajustar el grado de flexibilidad.

El proceso de encontrar el grado óptimo de flexibilidad de un algoritmo

para unos datos particulares se conoce como calibración.

Una representación de ese problema es como un control de flexibilidad,

tune, en el cual se debe buscar el punto óptimo para la función.

Casi todos los algoritmos, sino todos,

pueden ser vistos desde este marco conceptual genérico.

Adicionalmente, el efecto del número de datos o tamaño de muestra es interesante.

Para un k pequeño, el algoritmo ya no hace overfitting,

dado que el promedio estabiliza la variabilidad.

Es decir, entre más datos,

el algoritmo va predecir mejor,

porque se puede tener más flexibilidad.

El grado de flexibilidad óptima depende del tamaño de la muestra.

Esto dictamina qué tanta flexibilidad se puede comprar.

En este video comprendimos los

principios básicos de los algoritmos de machine learning que sirven para predecir,

teniendo en cuenta que los datos son realizaciones de variables aleatorias.

Además, vimos que el concepto de la

flexibilidad es fundamental para que la función estimada prediga bien,

que el grado de flexibilidad que podemos

comprar está determinado por el tamaño de la muestra.

En el próximo video,

formalizaremos la idea de la flexibilidad y

desarrollaremos las métricas necesarias para evaluar

y calibrar modelos supervisados de machine learning.

<https://www.coursera.org/learn/introduccion-al-machine-learning/supplement/AYM9j/diferencias-entre-modelos-explicativos-y-predictivos-contexto-etico-y-casos-de>

# Diferencias entre modelos explicativos y predictivos: contexto ético y casos de estudio

Machine learning aplicado al modelamiento predictivo estadístico, como su nombre lo indica, tiene como objetivo fundamental obtener la mejor predicción posible para la toma de decisiones en contextos como salud, finanzas, marketing o cadena de suministros. En ese orden de ideas, en un problema de regresión, la primera aproximación para solucionarlo sería estimar una regresión lineal del siguiente estilo:

yi=∑j=1kβjxij+ϵiyi​=j=1∑k​βj​xij​+ϵi​

y^i=∑j=1k𝛽^jxij

Donde el subíndice ii representa la i-ésima unidad (o sujeto) observada, y el subíndice jj es para representar la variable, con j=1,2,…,pj=1,2,…,p. Así, implementando y estimando el anterior modelo, es posible realizar una interpretación de la forma en la que se está procesando la información de entrada (xij)(xij​) para obtener la estimación de salida (y^i)(y^​i​). En particular, esta interpretación tiene como pieza fundamental a 𝛽^j, donde este representa la estimación del cambio en y^iy^​i​ dado un cambio en una unidad en xijxij​.

Sin embargo, el modelo de regresión lineal naturalmente se queda corto en describir relaciones no lineales entre las variables de entrada y la variable de respuesta, razón por la cual sería pertinente usar un modelo no lineal para tener una mejor predicción. En este orden de ideas, se puede aplicar un modelo aditivo de la siguiente forma:

yi=∑j=1kfj(xij)+𝜖i

Donde fj()fj​() es una función analítica suave de la variable xijxij​. En este tipo de modelos, es posible interpretar el efecto que tiene xijxij​ sobre y^iy^​i​ utilizando las derivadas de cada una de las funciones. Cada una de estas funciones se puede visualizar gráficamente, haciendo el modelo interpretable por márgenes, o por variables. Para cada variable, sin embargo, no se tiene una pendiente para medir el efecto de la variable. Este tipo de modelos aditivos es medianamente interpretable. Sin embargo, muchas veces, las formas aditivas se quedan cortas para representar la realidad, y es posible que no se encuentren entre los mejores predictores.

Otra forma de interpretación en modelos predictivos se presenta en los de árboles de decisión (para regresión y clasificación), los cuales funcionan como un modelo no paramétrico en donde se realizan particiones en el espacio de las xijxij​ para realizar la predicción correspondiente. Este tipo de modelos no son interpretables en la manera tradicional en la que lo son los modelos aditivos analíticos y los modelos lineales. La interpretabilidad de los modelos basados en árboles se hace por reglas, entendiendo las particiones hechas en el espacio de las xijxij​ para saber la manera en la que se realizará la predicción. Cuando se usan ensamblajes de árboles, entonces se pierde este tipo de interpretabilidad, sin embargo, es posible obtener métricas de la importancia de las variables, algo que podría ser equivalente a los p-valores en los modelos de regresión lineal. Los modelos de ensamblaje, por otro lado, tienen una capacidad predictiva mucho mayor.

Finalmente, es posible que la relación entre xijxij​ y yiyi​ no corresponda a un modelo de particiones y se requiera un modelo con mayor poder de predicción y flexibilidad. Así llegamos a modelos como las redes neuronales profundas, las cuales corresponden a múltiples capas de transformaciones lineales y no lineales de las variables de entrada para finalmente obtener y^iy^​i​. Las redes neuronales profundas poseen un poder predictivo muy alto en ciertos contextos con una topología de datos adecuada (por ejemplo, para el reconocimiento de imágenes). Sin embargo, al no ser posible interpretar la relación de cada xijxij​ y y^iy^​i​, este tipo de modelos reciben el nombre de “modelos caja negra”.

De lo anterior resulta evidente que, en el aprendizaje automático, para obtener un mayor poder de predicción, es necesario sacrificar la interpretación del modelo, en donde a un lado del espectro tenemos modelos con relativamente poco poder predictivo pero alta interpretabilidad, como la regresión lineal, y al otro lado del espectro tenemos modelos con relativamente alto poder predictivo pero nula interpretabilidad, como las redes neuronales profundas.

Una vez se ha presentado la idea de los contrastes existentes entre interpretación y predicción en modelos de machine learning, resulta natural verse motivado a implementar el gran poder predictivo de los modelos caja negra para todo tipo de problemas, sin embargo, es necesario tener especial precaución con las consecuencias éticas de aplicarlos en ciertos contextos. De hecho, como lo establece el artículo “Explaining Explanations: An Overview of Interpretability of Machine Learning” [1], uno de los retos más grandes actualmente en esta disciplina consiste en poder realizar cada vez mejores explicaciones de los modelos caja negra para poder utilizar satisfactoriamente estos modelos en contextos éticamente complejos.

Por ejemplo, en contextos de justicia penal, resultaría natural utilizar información de la investigación de un caso para resolver el problema de clasificación sobre si un acusado es culpable o inocente. De esta manera, se manifiesta un inconveniente ético que consiste en decidir sobre la libertad de un ser humano sin poder argumentar sobre las razones por las cuales la información de entrada ha llevado a tomar la decisión de determinar su culpabilidad. En particular, el artículo “Machine Bias” [2], pone en evidencia un sesgo racial en los modelos predictivos que se pretenden utilizar para determinar futuros criminales.

Por último, para utilizar correctamente el modelamiento predictivo, también es necesario entender las limitaciones de los modelos utilizados para discriminar los contextos en los que es adecuado aplicar machine learning. Por ejemplo, el artículo “Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images” [3] muestra cómo las redes neuronales profundas pueden dar resultados equivocados con un nivel de confianza bastante alto, en particular, aplicado al contexto de reconocimiento de imágenes.

En conclusión, en el modelamiento estadístico predictivo es fundamentalmente importante reconocer el intercambio existente entre la interpretabilidad de un modelo y su poder predictivo. Sin embargo, de la misma manera, es necesario entender en qué contextos es adecuado sacrificar la interpretabilidad en favor de aumentar su poder predictivo, y en qué contextos hacer esto sería inadecuado dadas las condiciones éticas del problema para, así, actuar responsablemente como profesionales en la toma de decisiones y la implementación de modelos predictivos.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/4OP3D/medicion-de-desempeno-en-modelos-predictivos-sesgo-vs-varianza

# Medición de desempeño en modelos predictivos: sesgo vs. varianza

Hola, en este video explicaremos cómo evaluar

un modelo predictivo para un conjunto particular de datos.

En términos del video anterior,

esta validación sirve para calibrar o determinar el grado

óptimo de flexibilidad y para comparar y seleccionar modelos.

Como ya hemos visto,

la flexibilidad es la propiedad fundamental para representar

las irregularidades de los fenómenos reales y para predecir mejor.

Sin embargo, al estimar una función flexible,

se requiere un tamaño de muestra suficientemente grande.

Dado que si tratamos de estimar una función muy flexible con datos insuficientes,

haremos sobreajuste o overfitting,

lo que impedirá que el modelo prediga correctamente.

El elemento que falta para calibrar y seleccionar modelos,

es una métrica adecuada que se pueda obtener a partir de los datos.

Por facilidad, esta métrica la desarrollaremos para los números de regresión.

Sin embargo, después veremos que para los de clasificación,

aunque se pueden usar otras métricas,

el principio es el mismo.

Recordemos que los datos de la muestra los denotamos como las duplas x\_1,

y\_1 hasta x\_n, y\_n.

Sin pérdida de generalidad,

suponemos que cada x\_i pertenece a los números reales con p dimensiones.

El modelo predictivo,

que también llamaremos algoritmo o función

estimada de ahora en adelante la notaremos como f gorro.

Esto es para representar que es una función de los datos y que,

por lo tanto, es una variable aleatoria.

Como el objetivo es predecir la respuesta y para un nuevo x,

llamaremos x estrella a esta observación

particular y y estrella a la respuesta correspondiente no observada aún.

Una métrica adecuada para determinar si el modelo predice bien es el EPE de f gorro,

que es igual al valor esperado de la diferencia

entre y estrella y f gorro evaluado en x estrella.

Esta diferencia, al cuadrado.

El EPE corresponde a la sigla en inglés del valor esperado de [inaudible] predicción.

Este valor esperado se toma con respecto a f gorro y con respecto a y estrella.

En este caso, como de costumbre,

se asume que x estrella

y y estrella son variables aleatorias independientes de la muestra.

Para entender mejor esa métrica,

veremos cuál es la función que mejor puede predecir,

es decir, la que minimiza el EPE.

Esta función la podemos llamar f\_opt y se define para cada punto x estrella

como el argumento que minimiza el EPE de f con respecto a todos los [inaudible].

Después de algunas derivaciones que no mostraremos en este video,

se obtiene que el f óptimo en x estrella es

igual al valor esperado y estrella dado x estrella,

es decir, el valor esperado de y estrella.

De esta forma, la función que mejor predice y bajo la métrica del EPE

es el valor esperado adicional de y dado x. Por supuesto,

esta función es desconocida,

dado que conocemos los datos,

pero no las reglas de probabilidad que lo producen.

Usando esta función óptima teórica,

el EPE se puede descomponer como:

el EPE de f gorro es

igual al valor esperado de y estrella menos el f óptimo al cuadrado,

más el valor esperado de f gorro,

menos el f óptimo al cuadrado más dos veces el producto cruzado,

donde se puede mostrar que este producto cruzado es 0,

dado que y estrella es independiente de los datos de la muestra y,

por lo tanto, de f gorro.

La parte de la izquierda en la ecuación no depende de f gorro.

Esto es el valor esperado de y estrella menos

f óptimo al cuadrado es igual a la varianza de y estrella es igual a Sigma cuadrado,

como la llamaremos de ahora en adelante y se conoce como el error irreducible.

Dado que el error irreducible no depende de f gorro,

este valor corresponde al error de

predicción del mejor predictor posible, de ahí su nombre.

Ningún algoritmo puede predecir mejor que ese.

Este es el precio que se tiene que pagar por tratar de predecir el valor

de una variable aleatoria con un número fijo y es importante,

entonces, considerar que ningún algoritmo podrá predecir de manera exacta.

La parte de la derecha del EPE es el error reducido y es el que

queremos hacer lo más pequeño posible escogiendo el mejor f gorro que encontremos.

Haciendo algunas derivaciones adicionales y definiendo el valor esperado de f gorro,

entonces tenemos que,

el error reducible se puede representar

como el valor esperado de f gorro menos su valor esperado al cuadrado,

más el valor esperado del valor esperado de f gorro,

menos el f óptimo al cuadrado.

La parte de la izquierda,

corresponde a la varianza de f gorro y la parte de la

derecha corresponde al sesgo al cuadrado.

Estas dos propiedades de f gorro corresponden a

mediciones de exceso de flexibilidad por defecto de este.

De esta forma, la varianza de f gorro corresponde a la variabilidad de

la variable aleatoria f gorro como función de los datos. Miremos los gráficos.

Para una muestra tomada aleatoriamente,

definimos los tres modelos: f óptimo en verde,

el modelo flexible en rojo y el modelo con sesgo en azul.

Si tomáramos otro muestra del mismo tamaño,

el modelo flexible cambia mucho,

dado que se apega a los datos nuevos.

Esto es lo que hace que el algoritmo sea muy volátil.

Al cambiar la muestra el f gorro en x asterisco,

puede ser completamente diferente.

Por otro lado, cuando el modelo es poco flexible,

representado en azul, tiene sesgo,

dado que tiende a estar centrado alrededor de una función muy rígida,

como por ejemplo, la función lineal en la gráfica.

Si se tomaron muchas muestras diferentes,

la distribución de las estimaciones

por x asterisco del algoritmo flexible y del sesgado serían muy diferentes.

La curva azul representa la distribución

de las predicciones para el algoritmo sesgado para muchas muestras,

tiene poca varianza, pero tiene sesgo.

Por otro lado, en rojo está la

distribución de las predicciones por el algoritmo flexible,

tiene mucha varianza, pero está centrado en f\_opt.

Una forma de entender la dicotomía entre sesgo y

varianza es una analogía con el lanzamiento de dardos.

En la siguiente gráfica, suponga que un lanzador está interesado en atinarle al centro,

el cual para nosotros corresponde a la constante de interés f óptimo,

evaluado en x estrella.

Un lanzador se comporta como una variable aleatoria,

dado que cada vez que repite un lanzamiento,

no cae en el mismo lugar.

Esto es, un lanzador es equivalente a f gorro.

En rojo tenemos al lanzador 1,

el cual representamos como f gorro 1 y en azul representamos al lanzador 2, f gorro 2.

El estimador lanzador rojo tiene mucha varianza,

dado que los resultados de sus lanzamientos son dispersos, aunque están centrados.

Al contrario el lanzador azul tiene poca varianza, pero no está centrado.

Seleccionar el mejor de estos dos estimadores es responder

a cuál de los dos lanzadores le confiaría un solo lanzamiento,

si se quiere que el resultado esté lo más cercano posible al centro.

¿Cuál sería mejor, el rojo o el azul?

Por supuesto, en este caso no hay una respuesta satisfactoria.

Los algoritmos de machine learning

funcionan de tal manera que no tenemos que seleccionar uno de los dos,

sino que cambiando los calibración podemos combinarlos,

algo así como poner proporciones de cada lanzador y mezclarlos

en una batidora para crear uno solo con el grado óptimo de flexibilidad.

El EPE es la suma del error irreducible y el reducible.

A su vez, el error reducible es,

entonces, la suma de dos cantidades positivas.

Una no cuenta con la flexibilidad con la

varianza y otra que disminuye con lo mismo o el sesgo al cuadrado.

Esto quiere decir que si miramos el EPE en función de algún parámetro

que cambie la flexibilidad del algoritmo debe dar una curva con forma de U,

es decir, convexa.

Hacia la izquierda, donde tenemos menos flexibilidad,

el EPE aumenta, dado que tenemos más sesgo.

Hacia la derecha el EPE aumenta,

dado que tenemos más varianza.

De acuerdo a eso, debe existir un mínimo

que corresponde al grado óptimo de calibración que estamos buscando.

Por cierto, la curva siempre debe estar por encima del error irreducible.

Hasta ahora, el EPE parece funcionar bien

como una métrica teórica para calibrar y comparar modelos.

Sin embargo, tenemos un grave problema.

El EPE es desconocido y no lo podemos medir directamente.

De alguna forma lo debemos estimar a partir de los datos,

pero, eso será un tema de nuestro siguiente video.

En este video,

comprendimos cómo evaluar un modelo predictivo,

determinando su desempeño a partir de una métrica

teórica que recoge el concepto que necesitábamos.

Para un tamaño de muestra dado,

debe existir un grado óptimo de flexibilidad.

En el próximo video, desarrollaremos el mecanismo para

estimar el EPE a partir de los datos y entenderemos que

esta medición nos lleva al requerimiento de separación de muestras

para entrenamiento y para evaluación.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/6n788/medicion-de-desempeno-en-modelos-predictivos-mse-y-separacion-de-muestras

# Medición de desempeño en modelos predictivos: MSE y separación de muestras

0:06/8:57

#  Medición de desempeño en modelos predictivos: MSE y separación de muestras

### Interactive Transcript - Enable basic transcript mode by pressing the escape key

You may navigate through the transcript using tab. To save a note for a section of text press ⌘ + S. To expand your selection you may use ⌘ + arrow key. You may contract your selection using shift + ⌘ + arrow key. For screen readers that are incompatible with using arrow keys for shortcuts, you can replace them with the H J K L keys. Some screen readers may require using ⌘ in conjunction with the alt key

[MÚSICA]

[MÚSICA] Hola.

En este video continuaremos con el desarrollo de métricas que sirvan

para evaluar un modelo predictivo.

En particular, veremos la definición del error cuadrático medio o MSE.

Además entenderemos que la medición de desempeño con lleva el concepto de

separación de la muestra en dos, una para entrenamiento y otra para prueba.

La idea fundamental será estimar el EPE a partir de los datos disponibles en la

muestra.

Recordemos que el EPE funcionaría bien como métrica de medición del desempeño,

sin embargo es una medida teórica que no puede ser calculada directamente.

Empecemos por retomar el concepto del EPE de f gorro, el cual se descompone en

el error irreducible más la varianza más el sesgo al cuadrado de f

gorro.Para desarrollar una aproximación del EPE basado en los datos, debemos

hacer énfasis en dos detalles claros que son muy sutiles, pero muy importantes.

El primero es que la definición del EPE se hace sobre un punto x estrella,

y estrella independiente de la muestra.

El segundo es que al ser el EPE un valor esperado,

cabe la posibilidad de ser estimado por medio de un promedio muestrario.

Empecemos a detallar el primer punto.

Cuando se define la aplicabilidad de un modelo proyectivo se hace sobre una

observación x estrella, y estrella en donde dado que x es igual a x estrella, se

quiere predecir a y estrella por medio de la posición evaluada f gorro, x estrella.

Gráficamente esto lo podemos ver así.

Con los puntos de la muestra se estima f gorro y luego, esta se evalúa en

x estrella, sin que x estrella y, y estrella afecten la estimación de f gorro.

[MÚSICA] Notemos entonces que x estrella,

y estrella es un punto independiente de los puntos de la muestra.

Esto es lo que permitió que el producto cruzado se cancelara

en la separación del EPE en una parte reducible y otra parte irreducible.

Esto es muy importante.

Si en lugar de x estrella, y estrella,

se usara uno de los datos de la muestra que podemos llamar x sub i, y sub i,

entonces el EPE de f gorro en x sub i sería igual al valor

esperado de la diferencia de y sub i menos f gorro de x sub i al cuadrado.

Sin embargo, como you lo definimos a medida que aumenta la flexibilidad

de f gorro, la posición estimada se acerca más y más a los datos de la muestra.

Por ejemplo, tenemos en la gráfica el caso de un modelo de regresión por k vecinos.

Si k es muy pequeño, la función se acerca a los datos.

En el caso de k igual a 1, f gorro de x sub i es igual a y sub i.

Esto implica que a medida de que la flexibilidad aumenta

el EPE de f gorro se aproxima a 0.

La curva es decreciente por lo cual no puede ser usada para calibrar el modelo.

Este concepto lo usaremos un poco más adelante.

Ahora hablemos del segundo punto clave para sacar el EPE.

El EPE es una medida teórica desconocida definida por un valor esperado.

Esto puede ser usado para estimarlo por medio de un promedio muestral.

Por ley de grandes números,

sin el número de datos con los que se hace este promedio tiende a infinito,

el promedio muestral converge al valor esperado correspondiente.

De esta forma la versión empírica del EPE o el promedio, la llamamos el MSE.

Error cuadrático medio.

Media square error, siglas en inglés.

Y puede ser definido como el promedio de los y sub i menos

el f gorro de x sub i al cuadrado.

El MSE puede ser entendido como un estimado del EPE.

Note que por claridad nada explicación omitimos este detalle técnico, el MSE se

está estimando como la integral del EPE para todos los valores posibles de x.

Esto no nos afecta los conceptos you descritos.

Tal como lo dijimos, el MSE no se puede calcular con los mismo datos con los

cuales fue estimada la función f gorro.

Esto haría que a mayor flexibilidad el MSE se va a ir a 0.

Para estimar correctamente el EPE,

el MSE debe ser calculado con unos datos diferentes.

En la práctica sin embargo, nosotros tenemos un conjunto de datos de muestra.

Por lo tanto la muestra debe ser separada en dos partes.

Una parte es usada para calcular f gorro,

que se conoce como el conjunto de datos de entrenamiento, o train

y la otra parte se usa para evaluar el modelo mediante métricas como el MSE.

Este último conjunto de datos se conoce como la muestra de evaluación o test.

Esta separación de muestras,

necesaria para calibrar y comparar el desempeño de diferentes modelos, es una de

las características diferenciadoras en la práctica de [INAUDIBLE]

Por supuesto quedan inquietudes acerca de cómo se deben separar esas muestras.

Como you habíamos aprendido, entre más datos usemos para estimar f gorro,

se podrá predecir mejor.

Esta es la propiedad de consistencia que se conoce también en estadística.

Partir la muestra parece indicar que el número de datos para entrenar se reduce,

lo cual puede crear ineficiencias en las predicciones.

[MÚSICA] Antes de continuar con estrategias para separación de muestras,

quisiera hablar un poco sobre el caso de la medida de desempeño para clasificación.

Recordemos que en este caso la variable de respuesta y, es categórica.

Por supuesto, no tiene mucho sentido usar el MSE como métrica de desempeño.

Dado que esta [INAUDIBLE] la diferencia entre el valor

de y estrella y la proyección de f gorro, y estrella.

Para variables categóricas esta operación de diferencia no se puede usar.

El EPE para clasificación se puede definir como EPE de gorro en x estrella

es igual al valor esperado de una función y, sobre si

y estrella es diferente a f gorro de x estrella, donde la función y

evaluada en a, toma el valor de 1 si la función a es cierta y 0 si no lo es.

Es decir, se busca minimizar la probabilidad

de que la proyección sea incorrecta.

De manera análoga, es posible determinar el mejor predictor posible por f óptimo.

Por ejemplo, si tomamos al espacio y como el conjunto de todas las clases posibles,

entonces f óptimo en x estrella, es la clase en y, para la cual la

probabilidad de que y estrella sea igual a esta clase, sea mayor a cualquier otra.

La descomposición sesgo varianza reproducible funciona de manera similar.

No la hacemos acá para no repetir el mismo proceso.

Aunque se había dejado implícito hasta ahora,

no está de más aclarar que la partición de los datos entre un conjunto para entrenar

y otro para evaluar, se debe realizar de manera completamente aleatoria.

Si no es así, entonces se pueden generar sesgos de selección

que pueden favorecer ciertos modelos o terminar en calibraciones erróneas.

Para verlo gráficamente,

suponga que se tiene el problema de regresión con estos datos.

Al partir la muestra aleatoriamente,

se generan dos muestras independientes de la misma población.

Si no se hace así, entonces se puede sobre estimar el error

o obtener un grado de calibración muy diferente, como por ejemplo,

si se seleccionan puntos cuyos x están en el rango azul.

En este video, desarrollamos el MSE como estimación del EPE

para evaluar modelos predictivos de regresión a partir de los datos.

Dado que se busca predecir sobre datos nuevos.

Para que esta medida sea válida, se debe aplicar sobre datos diferentes e

independientes a los datos que se usen para estimar el modelo de gorro.

Esto conlleva a la necesidad de separar la muestra entre entrenamiento y evaluación.

En el próximo video revisaremos posibles estrategias para partir la muestra

de datos de forma tal que al usar un conjunto de entrenamiento,

no se pierda demasiada información para predecir.

[MÚSICA] [MÚSICA]

[MÚSICA] [MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/n6NgH/estrategias-de-particion-de-muestras-validacion-cruzada

# Estrategias de partición de muestras: validación cruzada

Hola. En este video desarrollaremos estrategias para la separación de muestras,

de forma que medir el error de predicción en una muestra,

separada de los datos de entrenamiento,

no genere problemas de ineficiencia en el poder

predictivo de los modelos o errores en las calificaciones de los mismos.

Recordemos que para medir el desempeño del modelo de regresión o clasificación,

es necesario aplicar la métrica que aproxima

el EPE en datos diferentes a los usados para estimar el modelo.

Antes de revisar estrategias

eficientes para realizar el proceso de separación de muestras,

miremos los objetivos que se buscan al medir el desempeño en un modelo.

El primero ya lo conocemos, la calibración.

Esto es, seleccionar el grado óptimo

de flexibilidad para que el modelo prediga lo mejor posible.

Es decir, el balance entre el sesgo y la varianza del error reducido.

Un segundo objetivo tiene que ver con la

estimación del error de predicción de un modelo ya calibrado.

En este caso,

este error también se conoce como error de generalización.

Se usa para reportar el poder predictivo del modelo,

independientemente de la calibración,

y también para comparar diferentes modelos.

Por ejemplo, suponga que con los mismos

datos se estiman y calibran dos modelos diferentes.

Modelo uno, de regresión lineal y modelo dos, de K vecinos.

Una vez se tienen los dos modelos calibrados,

se pueden comparar los dos para determinar cuál predice mejor,

midiendo el MSE, en la muestra de prueba.

Note que en este caso no se estima una curva,

sino los valores estimados para los dos modelos.

Esto hace que la estrategia de partición de

muestras deba tener en cuenta los objetivos antes descrito.

Por lo tanto, las operaciones se hacen en tres partes: una muestra de entrenamiento,

una muestra de validación y una muestra de prueba,

como se puede ver en el gráfico.

La muestra de validación se usa para calibrar el

modelo y la muestra de prueba se usa para medir el error de generalización.

No existen reglas claras de cómo repartir las proporciones del

total de la muestra y esto depende de varios factores,

como por ejemplo, el número total de datos.

Es común, en la práctica,

usar una muestra de prueba entre el 20 y el 30 por ciento del total de datos.

En algunos estudios,

es posible que no se requiera tener una muestra de pruebas.

Por ejemplo, cuando el propósito en el estudio es

estimar un solo modelo y no busca el mejor desempeño predictivo.

La muestra de prueba se debe separar desde el principio,

por lo que es en el proceso de

validación en donde podemos desarrollar estrategias más eficientes.

Para simplificar la notación,

llamamos n\_total al número total de datos en la muestra.

Así mismo, definimos a n\_train,

a n\_val y a n\_test,

los datos usados para entrenamiento,

validación y prueba, respectivamente.

De ahora en adelante, en este video,

llamaremos a n,

la suma de n\_train y n\_test,

esto es el tamaño de muestra disponible para estimar y calibrar el modelo.

Consideremos el principio de eficiencia estadística.

Entre más datos se usen para estimar el modelo,

mejor será la predicción.

Esto se debe cumplir para todos los modelos,

tanto estadísticos como de "machine learning".

Como consecuencia, el proceso de

calibración no debería disminuir el poder predictivo del

modelo al reducir el tamaño de muestra.

¿Tiene sentido no usar los datos de validación para estimar el modelo?

Esto parece ser un proceso ineficiente.

Algo con más sentido puede ser usar la muestra de validación para calibrar y una vez,

calibrado el modelo, usar los n datos para predecir.

Es decir, los datos de entrenamiento y validación.

Entendiéndolo así, seleccionar la proporción usada en la

muestra de validación se transforma en un problema de sesgo y varianza.

Estamos interesados en calibrar el modelo final que usa n datos.

Sin embargo, al sacar un porcentaje,

el 50 por ciento para validación,

estamos calibrando el modelo que usa el 50 por ciento de los datos.

Es decir, con una MSE mayor al que buscamos estimar.

Este es el mismo problema de sesgo que ya conocemos.

Entre más grande es la proporción de datos llevados a la validación,

entonces más sesgada está la curva del MSE.

De acuerdo a lo que hemos visto,

la curva del modelo con n datos sería el azul,

mientras que la curva con los datos de n\_train sería la roja.

Está más arriba y se calibra con un grado menor de flexibilidad.

Es mejor usar una proporción pequeña para n\_val.

Por otro lado, si la proporción de datos de validación es pequeña,

entonces tenemos otro problema: mucha variabilidad.

Supongamos que el 90 por ciento de los datos en

n son para entrenamiento y 10 por ciento para validación.

Si seleccionamos aleatoriamente este 10 por ciento, varias veces,

de manera independiente, entonces las curvas pueden ser muy diferentes.

De acuerdo a esto, es mejor usar una proporción grande para n\_val.

Claramente, la selección de los tamaños de muestra para validar

parece estar guiada por los mismos principios de sesgo y varianza.

Sin embargo, en lugar de encontrar una proporción óptima, buscaremos otras estrategias.

Una opción para reducir la varianza es generar diferentes particiones,

manteniendo las mismas proporciones para entrenamiento y validación.

Luego de esto, se promedian las curvas correspondientes de MSE.

Al mantener la proporción fija,

entonces las curvas que se promedian tienen el mismo sesgo.

Por ejemplo, si se usan 50 por ciento de los datos para cada una de las dos muestras,

después se pueden intercambiar los roles de cada conjunto,

una vez para entrenar y otra vez para validar.

La curva del MSE estimada es el promedio de las dos curvas individuales.

Esa estrategia se conoce como validación cruzada o "cross validation".

En particular, "k fold cross validation".

La idea es dividir la muestra de tamaño n,

en "k" pliegues o "folds",

de igual tamaño cada uno;

es decir, n sobre k.

Luego, se hacen k modelos diferentes,

en los cuales se deja uno de los pliegues como muestra de validación.

Note que al promediar las k curvas del MSE,

se reduce la varianza.

La proporción de datos usados para la validación es 1 sobre k, en cada iteración.

Note que el caso más extremo es cuando se usan n pliegues o k igual a

n. En este caso se usa un dato para validación en cada iteración.

Este procedimiento recibe el nombre especial de validación cruzada,

dejando uno afuera: "only one out cross validation".

Aunque este mecanismo tiene menor sesgo posible,

puede ser muy ineficiente de implementar computacionalmente,

dado que se requieren estimar n modelos.

Sin embargo, algunas implementaciones

se pueden calcular directamente sin repetir las estimaciones,

como lo veremos más adelante.

Lo más usado en la práctica,

aunque tal vez sin ninguna justificación teórica,

son 5 o 10 pliegues.

En este video,

explicamos los conceptos de la separación de muestras en conjuntos excluyentes,

usados en la práctica de "machine learning".

El proceso de calibración

usando una muestra de validación puede ser

ineficiente y crea un dilema de sesgo y varianza sobre la curva del MSE.

Por último, vimos cómo la validación cruzada

resulta una estrategia conveniente para obtener mejores calibraciones.

El próximo video, usaremos los conceptos adquiridos

esta semana sobre "machine learning", en general,

para empezar a desarrollar algoritmos particulares.

https://www.coursera.org/learn/introduccion-al-machine-learning/supplement/7E4qn/particion-de-muestras

# Partición de muestras

Como se ha explicado anteriormente en el curso, la evaluación del desempeño de un modelo de *machine learning* no puede ser medida en el mismo conjunto de datos que se utilizaron para entrenar dicho modelo. Esto ocurre por el problema de sobreajuste u *overfitting*: ajustar excesivamente el modelo para que represente bien los datos de entrenamiento pero que sea un mal predictor de datos nuevos. Por esta razón, la primera forma que se tiene para solucionar esto es segmentar en dos conjuntos la totalidad de los datos que se tienen, donde uno de estos será el conjunto de entrenamiento y el otro será el conjunto de evaluación del desempeño del modelo.

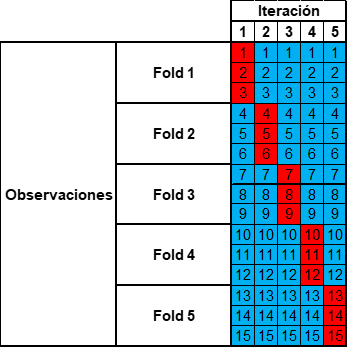
Sin embargo, debido a que los datos son un recurso limitado, realizar la partición anterior resulta costoso ya que, en el entrenamiento del modelo, se estaría desconociendo un alto porcentaje de los datos. Por tal motivo, se han desarrollado otras técnicas de partición de muestras para hacer más eficiente el uso del recurso escaso que representan los datos sin incurrir en el problema de sobreajuste.

La primera de estas técnicas que se explicarán en esta lectura corresponde a la validación cruzada o *cross-validation*. Esta técnica tiene diferentes variantes dependiendo del número de pliegues o *folds* con los que se realice. En general la técnica funciona de la siguiente manera:

1. Divida el total de N observaciones en k pliegues, esto es, realizar una partición de las observaciones totales en k grupos con aproximadamente NkkN​ observaciones por grupo.
2. Seleccione uno de estos pliegues y asígnelo como conjunto de evaluación.
3. Los restantes k-1 pliegues utilícelos como un solo conjunto de entrenamiento.
4. Posteriormente, estime el modelo de machine learning con el conjunto de entrenamiento y mida su desempeño en el pliegue asignado como conjunto de evaluación.
5. Guarde el valor de la métrica de desempeño correspondiente en un arreglo.
6. Ejecute el paso 2 y el paso 5 cambiando el conjunto de evaluación (sin repetirlos). Esto llevará a que en total haga el anterior proceso k veces.
7. Finalmente, promedie el valor de las métricas de desempeño obtenidas y esa será la estimación del desempeño del modelo.

El anterior procedimiento recibe el nombre de validación cruzada de k pliegues o *k-fold cross-validation.* De esta forma, si el número de pliegues que se utilizan son 10, se dice que la técnica de partición de muestras que se está utilizando es *10*-*fold cross-validation* y si, por ejemplo, son 5 los pliegues entonces se dice que se está utilizando *5-fold cross-validation.*

Es importante resaltar que, en caso de que k sea igual a N, se estaría utilizando N-1 datos para entrenamiento, y un dato se deja por fuera para la evaluación de desempeño, y finalmente se estarían promediando N métricas. Este tipo de validación cruzada recibe el nombre de validación cruzada dejando un dato afuera o *leave-one-out cross-validation* (LOOCV)*.* A continuación, se presenta gráficamente un ejemplo donde N es igual a 15 y se utilizan 5 pliegues:





Como se puede observar, se tienen tantas iteraciones como pliegues y en cada iteración se considera un pliegue para la evaluación del modelo que se estima con los pliegues restantes.

Ahora bien, es importante aclarar que el método de validación cruzada para partición de muestras posee algunas desventajas. En particular, aumentar el número de pliegues aumenta el sesgo de la estimación del desempeño del modelo y disminuirlo aumenta su varianza, características que serán importantes a tener en cuenta para el modelador en el momento de seleccionar la estrategia de partición de muestras adecuadas.

# SEMANA 2

<https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/AQhgL/modelos-de-regresion-lineal-y-metodos-de-seleccion-de-variables>

Hola. En este video

explicaremos cómo los modelos de regresión

lineal se pueden ver como algoritmos para predecir.

Además, entenderemos cómo calibrar este tipo de

modelos y cómo se resuelve el problema de selección de variables.

Lo más probable es que todos conozcamos los modelos de regresión lineal,

en los cuales se tiene una variable de respuesta o salida continua Y,

la cual se modela como Y igual a Beta\_0 más

Beta\_1 por X\_1 hasta Beta\_p por X\_p más un error aleatorio;

donde el error aleatorio tiene valor esperado 0;

es decir, la función predictora f gorro tiene forma lineal.

Esto quiere decir que formó un hiperplano en el espacio de red.

Por ejemplo, si p es 1,

entonces la función tiene forma de línea recta.

Si es p es 2,

entonces es un plano.

A Mayores dimensiones podemos imaginar la forma de hiperplano que se genera.

Una de las características de esos modelos es que son altamente interpretables.

Esto es el efecto de la variable X\_j sobre el valor

esperado de Y se define directamente sobre la pendiente Beta\_j.

Por ejemplo, si Y es la nota de un estudiante en una

materia de maestría y X son las horas de estudio semanales dedicadas al curso,

entonces Beta indica cuánto aumenta el valor esperado de la nota,

si se decide estudiar 1 hora más cada semana.

Así mismo, como lo habíamos dicho en el segundo video de la semana pasada,

un modelo de regresión lineal es muy poco flexible

y por eso su facilidad de interpretación.

Sin embargo, debemos considerar mejor las cosas.

Si fijamos el número de variables predictoras a p,

entonces el modelo de regresión lineal es sesgado con respecto a un modelo,

como por ejemplo, K-vecinos.

Lo que debemos considerar es que si aumenta el número de variables predictoras,

entonces el modelo puede ganar flexibilidad.

Por ejemplo, suponga que se

tiene una variable de salida Y y una variable de entrada o predictora

X. Si se define el modelo Y igual a Beta\_ 0 más Beta\_1 por X más el error aleatorio,

entonces el modelo es sesgado.

En esta gráfica vemos que este modelo no se ajusta bien a los datos.

Supongamos que a partir de la variable X calculamos 2 nuevas variables,

X al cuadrado y X al cubo;

con esto podemos ajustar el modelo Y igual a Beta\_0,

más Beta\_1 por X,

más Beta\_2 por X al cuadrado,

más Beta\_3 por X al cubo más un error aleatorio.

Si se ve el modelo en términos de 3 variables; es decir,

en 3 dimensiones, el modelo sigue siendo lineal.

Sin embargo, si se mira el modelo en función de los términos de la misma variable X,

entonces el modelo adquiere una forma polinomial cúbica que empieza a ser flexible.

El concepto fundamental en los modelos de regresión lineal,

cuando se miran como algoritmos predictivos,

es que se calibran con el número de variables que se usan para predecir.

Esto también se conoce como la dimensionalidad del modelo lineal.

Por supuesto, es importante considerar que al calibrar

usando como parámetro de calibración el número de variables,

solamente puede tomar valores discretos;

es decir, en los números naturales.

Llamemos p al número de variables totales disponibles para estimar el modelo,

esto es el modelo con p variable sería el de máxima flexibilidad.

El parámetro de calibración que determina el número de variables en

un modelo particular lo llamaremos K. Quiere decir que K va desde 0,

en el modelo con solo intercepto,

hasta p. La naturaleza de los modelos de regresión lineal

hace que seleccionando el valor de K no se determine el modelo a usar,

dado que diferentes combinaciones de variables pueden tener el mismo K. Por ejemplo,

si p es igual a 4,

entonces se tienen X\_1, X\_2 hasta X\_4.

Un modelo con X\_1 y X\_3 tiene K igual a 2,

igual a otro modelo que usa X\_2 y X\_3.

Esto quiere decir que si bien la dimensión

funciona como el mecanismo para determinar la flexibilidad,

seleccionar K no determina el mejor modelo,

sino que se debe realizar una selección de variables.

Antes de determinar cómo se hace esta selección,

repasemos un poco el mecanismo de estimación por mínimos cuadrados.

Recordemos que el algoritmo de estimación de la

función f igual a Beta\_0 más Beta\_1 por X\_1,

hasta Beta\_p por X\_p,

busca aproximar el valor de los parámetros Beta.

Estos se suelen estimar por mínimos cuadrados ordinarios;

es decir, si se tienen estas observaciones,

se buscan los parámetros Beta que minimizan la

suma de las distancias de cada variable de

respuesta Y\_i a la función estimada en el X correspondiente;

es decir, f gorro de X\_1i y hasta X\_pi.

De manera analítica,

el estimador de Beta se obtiene como Beta gorro\_0,

Beta\_1 gorro hasta Beta\_p gorro,

es el argumento que minimiza con respecto a B,

la sumatoria de los Y\_i menos B\_0 hasta B\_p por X\_pi,

todas estas al cuadrado.

Por ejemplo, si p es igual a 2 se buscan los

valores de Beta\_0 gorro Beta\_1 gorro y Beta\_2 gorro,

para los cuales se minimizan las distancias verticales de

los puntos observados al hiperplano correspondiente.

Cuando los datos se presentan de manera matricial, como Y,

un vector columna de n por 1,

con las observaciones de Y\_x,

una matriz de n por p más 1,

donde cada columna corresponde a las n observaciones de la variable X correspondiente,

y adicionalmente se agrega una columna de unos para que multiplique el intercepto.

De esta forma, el sistema de datos,

Y\_i igual a Beta\_0 más Beta\_1 por X\_1i,

hasta Beta\_p por X\_pi más el error aleatorio i,

para todo i igual 1, 2 hasta n,

se puede expresar de manera compacta como Y igual a X por Beta más el error;

donde el error es el vector con los n errores correspondientes.

Cuando se usa esta notación matricial,

la solución a los estimadores Beta por mínimos cuadrados,

se puede representar de manera más compacta como Beta gorro igual a X transpuesto

por X a la menos 1 por X transpuesto por Y.

Ahora retomemos el problema de calibración.

Dado que para cada número de variables en el

modelo K se pueden tener diferentes combinaciones,

entonces la curva del EPE en términos de la flexibilidad se ve más como unos puntos así.

Para cada K se tienen diferentes modelos.

Lo que se busca son los puntos para cada dimensión K que estén lo más abajo posible.

Esta es la curva que buscamos para ser análoga a lo que ya conocemos.

Estos puntos de la parte de abajo forman una curva convexa,

de forma tal que se pueda seleccionar el mejor modelo de todos.

En la práctica, el EPE se reemplaza por el MSE, como ya lo vimos.

Así es cómo vamos a calibrar los modelos de regresión lineal para predecir.

El problema, sin embargo,

es encontrar el punto en la parte inferior para cada K. Supongamos que K es igual a 1,

seleccionar la variable que mejor predice,

implica ajustar p modelos de una variable de cada uno.

La variable que más aporte a la función objetivo; es decir,

aquella para la cual el MSE es menor, es la que se selecciona.

Notemos que dado que no cambiamos la dimensión del modelo,

la flexibilidad es la misma y podemos

usar el MSE en la muestra de entrenamiento o "training".

Por supuesto, esto no funciona si

comparamos modelos con diferentes números de variables.

Una vez se selecciona la mejor variable,

pasamos a mirar todos los modelos de 2 variables.

El número de combinaciones para K igual a 2 es p combinado de 2;

es decir, p menos 1 por p sobre 2.

Por ejemplo, si p es igual a 10,

hay 45 combinaciones posibles de 2 variables.

Si la variable que ya seleccionamos cuando K es

igual a 1 se mantiene en el modelo de 2 variables,

entonces no se deben revisar 45 modelos, sino 9.

La pregunta es si esto ocurre.

Lamentablemente, la respuesta es,

en general, no.

Resulta que los modelos de regresión lineal tienen una

característica muy interesante: cuando se tienen variables en un modelo,

la información que aporta cada variable para predecir a Y,

es solo la información nueva que no está contenida en las otras variables.

Supongamos que se tienen dos variables x\_1 y x\_2.

Si se estima un modelo con x\_1,

la disminución de la función objetivo a SSE es A\_1.

Si se estima un modelo con x\_2,

la disminución de la función objetivo SSE es A\_2.

Sin embargo, si se estima el modelo con x\_1 y x\_2,

entonces el aporte de cada variable disminuye.

Cuando se habla de información nueva,

quiere decir que entre las variables puede haber información redundante.

Pongamos un ejemplo. Se quiere predecir a Y,

que es el gasto de las familias en términos de x\_1,

que es el ingreso medido en dólares,

y x\_2, que es el mismo ingreso pero medido en euros.

Cuando se miran las variables x\_1 y x\_2,

las dos tienen la misma información.

Si se ingresa una al modelo,

la otra no tiene nada nuevo que aportar.

Si el modelo 1 usa x\_1 y el modelo 2 usa x\_2 y el modelo 3 usa x\_1 y x\_2,

entonces las predicciones de los tres modelos son exactamente las mismas.

No obstante, el modelo de dos variables,

o sea el modelo 3, aunque tiene el mismo nivel de sesgo, aumenta su varianza.

El resultado de esto es que cuando se pasa de K igual a 1 a K igual a 2,

se deben ajustar los 45 modelos en lugar de solo 9.

En general, las variables del mejor modelo con K variables,

no necesariamente están incluidos en los mejores modelos con K más 1,

K más 2, hasta P variables.

La complicación generada es que el problema se vuelve combinatorio.

Para encontrar el mejor modelo,

debemos revisar todas las combinaciones posibles.

El número de combinaciones totales es 2 a la P. Por ejemplo,

si P es igual a 5,

entonces se deben revisar 32 modelos.

Si P es igual a 10,

son 1.024 modelos,

y si P es igual a 100,

entonces el número de modelos a ajustar son más que billones de billones,

aproximadamente, 1,26 por 10 a la 30.

Esto no se puede hacer computacionalmente en el tiempo realista.

Tener 100 variables predictoras o más,

no es algo fuera de lo común en las aplicaciones modernas de "machine learning".

Esta es la gran dificultad de los modelos de regresión lineal para predecir,

son difíciles de calibrar.

La buena noticia es que en los modelos lineales

no es necesario partir la muestra para calcular el MSE en datos de validación.

Esto es debido a que existen estimadores directos

del EPE en tests con los mismos datos de entrenamiento.

Aunque no miraremos muy a fondo de dónde salen estas métricas,

es suficiente decir que,

dado que en los modelos lineales la flexibilidad hace el número de variables,

es posible estimar el aumento en la varianza si se aumenta K en una dimensión más,

sin introducir ninguna información nueva.

Las métricas más usadas son el R cuadrado ajustado,

el AIC, Akaike Information Criterion,

y el BIC, el Bayesian Information Criterion.

Mientras el R cuadrado ajustado buscamos que sea lo más alto posible,

el AIC y el BIC son estimadores directos del EPE,

y se busca que sean lo más pequeños posibles.

El MSE en validación puede ser entonces reemplazado por cualquiera de estas métricas.

Si bien el problema de calibración es combinatorio,

existen estrategias para resolver el problema;

una exacta, que se conoce como "selección exhaustiva de variables" y otra aproximada,

que son las selecciones secuenciales.

Miremos la primera, la selección exhaustiva de variables,

cuya idea es revisar todas las combinaciones,

pero teniendo en cuenta que si la curva que generan los puntos de abajo es convexa,

entonces existe un K para el cual la curva sigue subiendo.

Todos los modelos de ahí en adelante son subóptimos y no vale la pena revisarlos.

La estrategia es, entonces,

seleccionar un K máximo y revisar las combinaciones para todo K menor o igual a K máximo.

Por ejemplo, si P es igual a 5 pero K máximo es igual a 2,

entonces se deben ajustar los modelos con 0, con 1 y con 2, es decir,

1 más 5 más 10 igual a 16 modelos en lugar de 32.

Si la curva que generan los puntos inferiores medidos

con el AIC o el BIC ya están en ascenso,

con esto es suficiente para encontrar el mejor modelo.

Esta es una estrategia de corte,

como se conoce en optimización combinatoria.

Cuando P es grande, es posible que la selección exhaustiva sea ineficiente,

dado que se requiere un K máximo grande.

En estos casos, es mejor usar otra alternativa que,

a pesar de ser subóptima,

puede aproximarse mucho al desempeño del mejor modelo.

Estas estrategias son de selección o eliminación secuencial de variables.

Por ejemplo, la selección tipo "Forward",

escoge la variable que mejor predice cuando K es igual a 1.

Luego, cuando se selecciona el mejor modelo con dos variables,

mantiene la primera que ya seleccionó.

Esto reduce muchísimo el número de modelos a revisar.

Por supuesto, como ya lo vimos,

el modelo seleccionado, por lo general, no es el mejor.

La curva generada con este procedimiento está

por arriba de la de la selección exhaustiva,

en términos de la métrica seleccionada.

Si el procedimiento secuencial va eliminando variables,

entonces empieza con el modelo de P variables y va hacia atrás

y se conoce como "Backward".

En este video aprendimos a usar los modelos

de regresión lineal como algoritmos predictivos.

El número de variables se comporta como el parámetro de calibración.

Sin embargo, seleccionar el

mejor modelo resulta en un proceso combinatorio difícil de resolver.

En el próximo video,

revisaremos estrategias alternativas para usar modelos lineales para predecir,

pero transformando las variables.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/39XkX/transformacion-de-variables-en-modelos-de-regresion-lineal

# Transformación de variables en modelos de regresión lineal

Hola. En este video

veremos cómo al transformar las variables predictoras en un modelo de regresión lineal,

sería posible mejorar el desempeño y facilitar el proceso de calibración.

Recordemos que la dificultad al calibrar los modelos

lineales es que resulta un problema combinatorio,

dada la información redundante que existe entre las variables.

En consecuencia, los procedimientos

secuenciales de ir introduciendo variables son su [inaudible].

Esto es, si entre las variables no existiera información redundante,

si todas fueran independientes entre sí,

los procedimientos secuenciales como "forward"

encontrarían el mejor modelo de manera mucho más eficiente.

La estrategia de transformar variables es

justamente reorganizar la información contenida en x\_1,

x\_2 hasta x\_p,

de forma tal que se creen nuevas variables z\_1,

z\_2 hasta z\_p,

que contengan la misma información en total,

pero distribuidas de manera diferente.

Cuando hablamos de transformar variables,

nos referimos a transformaciones lineales, de forma tal,

que cada una de las nuevas variables z\_j para j igual 1, 2, hasta p,

es calculada como una combinación lineal de las x. Esto es,

z\_j es igual a\_1j por x\_1 más a\_2j por x\_2 hasta a\_pj por x\_p.

En términos de la matriz de datos X que están los reales de n por p,

si ignoramos la columna de 1 para el intercepto,

los nuevos datos de la variable Z son: Z igual a X por A,

donde A es una matriz de p por p con las transformaciones.

Se asume que A tiene rango completo.

Es importante notar que al ser una transformación lineal de p variables a p variables,

entonces no se crea ni se pierde información adicional para predecir a Y. Es decir,

los modelos Y igual a Beta\_1 por x\_1 hasta Beta\_p por x\_p y y

igual a Alfa\_0 más Alfa\_1 por z\_1 hasta Alfa\_p por z\_p predicen exactamente igual.

La diferencia está en que cuando se usan k menor a p variables,

en este caso la información puede quedar organizada de manera diferente.

Ahora bien, en términos de un modelo de regresión para predecir qué

características queremos que tengan las nuevas variables z, fundamentalmente hay dos.

La primera es que no haya información redundante o correlaciones entre las variables

z. Esto hace que el aporte de cada

variable al modelo dependa de las demás variables a intuir.

La segunda es que la información que sirve

para predecir a Y quede contenida en un grupo pequeño de variables.

De esta forma, no se requiere un k muy alto y el modelo tendrá menos varianza.

En general, hay dos alternativas para encontrar la

matriz Z y resolver el modelo de regresión.

Una es regresión por componentes principales y

la otra es mínimos cuadrados parciales o partial least squares.

Empecemos con regresión por componentes principales.

En este caso,

la estrategia para encontrar Z es que sus columnas tengan

correlación 0 y por lo tanto no haya información redundante.

La idea es encontrar p componentes,

de forma tal que para el primer componente o z\_1 se maximice la varianza, es decir,

que de todas las posibles combinaciones lineales de x\_1, x\_2, x\_p,

en donde z\_1 es igual a a\_11 por x\_1 más a\_21 por x\_2 hasta a a\_p1 por x\_p,

se busque la que maximice la varianza de la variable z\_1 resultante,

bajo la restricción de que a\_11 al cuadrado más

a\_21 al cuadrado más hasta a\_p1 al cuadrado sea igual a 1.

Para mirarlo gráficamente,

supongamos que p igual a 2.

En este caso, se busca el vector con longitud 1,

para el cual al proyectar los puntos X sobre este vector,

la variable resultante tenga la varianza máxima.

Esos puntos proyectados corresponden a la variable z\_1.

Para encontrar la segunda componente z\_2,

se busca la combinación que tenga correlación 0 con z\_1

y que tenga varianza máxima bajo esta condición.

El procedimiento sigue así sucesivamente hasta que se obtienen las p variables z\_j.

Este procedimiento es equivalente a encontrar los vectores y los

valores propios de la matriz de varianza covarianza de X. Es decir,

si ese s\_x es la matriz de varianza de X,

donde los elementos de la diagonal son las varianza estimadas de las variables

x\_1 hasta x\_p y los elementos fuera de la diagonal son las covarianzas,

entonces, las columnas de la matriz A corresponden a los vectores propios de s\_x.

Los valores propios correspondientes son los varianzas de cada componente z\_j.

Note que las restricciones impuestas sobre las columnas de la matriz

A implican que el proceso de obtener los

componentes principales es equivalente a un proceso de rotación,

en donde las nuevas coordenadas son dictadas por

las combinaciones lineales que maximicen la varianza.

Sin pérdida de generalidad,

podemos asumir que los datos se encuentran centrados.

Es decir, que el promedio de cada columna de X es 0.

Una vez que se tienen las nuevas variables z\_1 hasta z\_p,

dado que son no correlacionadas,

es posible hacer un procedimiento secuencial tipo forward para encontrar el mejor modelo.

Otra opción más usada en la práctica es

usar los k primeros componentes y calibrar con respecto a

k. Es importante notar que al obtener las variables

z\_1 hasta z\_p por medio de componentes principales,

no se usan la variable de respuesta Y. Esto es,

solamente depende de los X. Por supuesto,

esta es una de las limitaciones de la regresión por componentes principales,

aunque muchas veces puede funcionar muy bien,

dado que los primeros componentes conservan la mayor parte de información contenida en X.

La otra estrategia, además de componentes principales,

es usar mínimos cuadrados parciales PLS.

Esa estrategia también obtiene variables z\_1 hasta z\_p que son no correlacionadas,

pero en lugar de concentrar la información en x,

busca concentrar la información que se tiene para predecir a y.

De cierto modo,

busca corregir el problema de componentes principales,

al no considerar la variable de respuesta.

La idea es descomponer la matriz X como X igual a Z por P transpuesta más un error.

De tal forma que la matriz Z contiene las proyecciones de X y P es una matriz de cargas.

Se busca maximizar la covarianza entre las columnas de Z y la respuesta Y,

contrario a regresión por componentes principales.

En partial least squares no se generan p variables Z para hacer selección de variables,

sino que para un cada k determinado se hace la regresión sobre z\_1 hasta z\_k.

Aunque en teoría partial least squares

debería predecir mejor que componentes principales.

En la práctica es muy común usar ambos algoritmos y

seleccionar el mejor de los modelos en una muestra de test.

Para concluir, las estrategias de transformación

de variables en modelos de regresión lineal,

consideremos que es muy probable que se encuentre un mejor modelo de

desempeño predictivo con las nuevas variables Z. Es decir,

que la curva LP se sitúe un poco por debajo

de la que se obtendría en un modelo con las variables originales

X. También es importante considerar

que al estimar los parámetros del hiper plano con las variables Z,

la interpretación no es la misma que en el modelo original,

dado que se tienen pendientes diferentes.

En ocasiones las variables Z son interpretables,

dado que son combinaciones lineales de los variables originales.

Por ejemplo, si para una muestra de un grupo de

personas x\_1 es la circunferencia del pecho,

x\_2 es la circunferencia de la cintura y x\_3 es la circunferencia de las caderas,

entonces sí z\_1 es igual a x\_1,

más x\_2 más x\_3.

Y z\_2 es igual a x\_1 menos x\_2 más x\_3, entonces,

z\_1 se puede interpretar como

el tamaño de las personas y z\_2 como la forma en proporción de pecho,

caderas con respecto a la cintura.

Sin embargo, esta interpretación no siempre está garantizada.

Al obtener mejor desempeño se da un paso más hacia los modelos menos interpretables.

En este vídeo estudiamos las estrategias de

transformación de variables en modelos de regresión

lineal para mejorar el desempeño y hacer

más fácil los procedimientos de selección de variables.

En el próximo vídeo,

estudiaremos estrategias de

regularización en donde se modifica directamente la función objetivo de mínimos cuadrados.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/A2KKo/metodos-de-regularizacion-en-modelos-de-regresion-lineal

# Métodos de regularización en modelos de regresión lineal

[MUSIC]

Hola, en este video explicaremos las estrategias de penalización en los modelos

de regresión lineal.

Con el fin de encontrar un balance entre el sesgo y la varianza del modelo.

Adicional a la selección de variables en el modelo original y a la transformación

de variables que vimos en el video pasado.

Existe una tercera alternativa para calibrar mejor y más fácil

los modelos de regresión lineal.

Esta estrategia es justamente modificar el problema de optimización por mínimos

cuadrados.

Para controlar la varianza del modelo resultante de una forma inteligente.

La manera de hacerlo es agregar un término a la función objetivo,

en la cual se penaliza el valor de los betas estimados en el hiperplano.

De forma tal que no puedan tomar valores demasiado grandes.

De forma general, cuando se usa penalización,

el problema de estimar los parámetros del hiperplano expresado como vector.

Queda como el beta gorro es igual al argumento que minimiza la

sumatoria de los y sub i menos beta transpuesto por x sub i al cuadrado.

Más lambda por el término de penalización,

donde la penalización puede ser una métrica que mida la magnitud de los betas.

Esto es una norma en el vector beta.

Por ejemplo, la norma estándar euclidiana, en donde la penalización de beta

es la norma al cuadrado, o sea, la suma de los betas al cuadrado.

Esto es lo que va a regular la solución para que algunos betas no tomen valores

demasiado grandes.

El parámetro lambda se convierte en el parámetro de calibración con el cual

regulamos qué tanto se penaliza el modelo.

[MUSIC]

Empecemos con la penalización de norma dos, esto es la norma euclidiana normal.

En este caso, la penalización de beta es la norma al cuadrado,

la suma de los betas al cuadrado.

Y la estimación de los betas queda como.

Beta gorro es el argumento que minimiza la sumatoria de los ys menos beta

transpuesto por x al cuadrado más lambda por la suma de los betas al cuadrado.

En general, el intercepto no es necesario de penalizar,

por lo que es mejor usar variables centradas en los modelos sin intercepto.

Cuando se usa un modelo con esta penalización,

se conoce como el estimador Ridge, o Ridge regression.

O también como una aplicación directa de regularización de Tikhonov.

La solución puede ser calculada de manera analítica, en forma matricial.

Como el beta gorro Ridge es x transpuesto por x más lambda por la matriz

identidad a la menos1 por x transpuesto por y.

Donde Ip es la matriz identidad de p por p.

A simple vista, no es tan obvio ver por qué esta

estrategia permite encontrar un balance entre el sesgo y la varianza.

¿Cómo funciona?

¿Cómo se controla la flexibilidad del modelo?

Para entender el mecanismo de regularización por penalización.

Revisemos el problema de la información redundante que nos llevó a la

selección exhaustiva de variables.

Supongamos que tenemos un modelo con tres variables predictores, x1, x2 y x3.

Supongamos que entre x1 y x2 existe alta correlación, o información redundante,

mientras que x3 no está correlacionada con las dos primeras variables.

Esto es, existe información redundante entre las dos primeras variables.

[MUSIC]

Para simplificar la notación, supongamos que todas las variables están centradas,

se le resta el promedio a cada una, incluyendo y.

Por lo cual se puede definir el modelo sin intercepto,

y igual a beta 1 por x1 más beta 2 por x2 más beta 3 por x3 más el error aleatorio.

Recordemos que en el modelo de regresión lineal, la varianza de los estimadores es.

Varianza de beta gorro es igual a sigma cuadrada por x transpuesto por x

a la menos 1.

Miremos entonces la matriz x transpuesto por x, que una vez se invierte,

es proporcional a la varianza.

En x transpuesto por x, en la diagonal de esta matriz,

están las sumas de los valores al cuadrado de cada columna de x.

Los elementos por fuera de la diagonal son los productos punto de la columnas de x,

y la matriz es simétrica.

Por facilidad, supongamos que la sumatoria de los x sub ij al cuadrado,

para j igual 1, 2, y 3 es igual a 1.

Y como las variables están centradas,

entonces x transpuesto por x se puede definir como la matriz de correlaciones.

[MUSIC]

En nuestro caso particular, como solo hay correlación entre x1 y x2,

entonces x transpuesto por x toma esta forma.

Supongamos, por ejemplo, que la correlación entre x1 y x2,

rho, sea de 0.5.

En este caso, la inversa de la matriz es x transpuesto por x a la menos 1.

En la diagonal, toma estos valores.

[MUSIC]

Si la correlación aumenta a 0.9, es decir,

mucha más información redundante, entonces la inversa

es aproximadamente 5 5 1 y menos4.5 en las correlaciones.

Como se puede apreciar, la covarianza de los estimadores beta 1 y beta 2 aumenta

drásticamente cuando aumenta la correlación.

[MUSIC]

El efecto de usar Ridge regression es que la solución de los betas estimados usa la

matriz inversa x transpuesto por x más lambda por la matriz

identidad a la menos 1.

Esto hace que la varianza de los betas se reduzca, pero sobre todo,

las varianzas que se encuentran infladas por la información redundante.

De esta forma, la penalización tiende a encoger en valor absoluto alos betas.

Pero este efecto está focalizado en las variables que no aportan tanta información

nueva al modelo.

Es decir, donde se usan más dimensiones sin necesidad.

Por supuesto, este encogimiento también produce sesgo.

El resultado es que, al aumentar el parámetro lambda,

se reduce la varianza de las predicciones y se aumenta el sesgo.

Por lo tanto, la flexibilidad se puede entender como inversamente

proporcional al parámetro lambda.

Notemos que si lambda va a infinito, entonces los estimadores de betas se

van a 0, y el F gorro será una constante, tiene máximo sesgo y varianza 0.

La curva del MSE, en términos de lambda, se ve así.

Contrario a la calibración por selección o transformación de variables,

el parámetro lambda es una variable continua.

[MUSIC]

En general, el desempeño predictivo

del modelo Ridge puede ser muy bueno.

Sin embargo, es importante tener en cuenta ciertas condiciones para que esto pase.

La primera, es que dado que se penaliza todas las metas por igual,

el algoritmo Ridge no es inmune a los cambios de escala en las variables.

Por ejemplo, cuando se pasa de centímetros a metros.

De esa forma, las variables deben estar estandarizadas en la escala.

La segunda es que si lambda es 0, se tiene un modelo con todas las variables,

el que tiene máxima flexibilidad.

Cuando se hace selección de variables, se eliminan las variables que no

son informativas, es decir, que no tienen información para predecir a y.

El modelo Ridge no tiene esta capacidad.

Las variables no informativas pueden perjudicar el desempeño del modelo.

Un procedimiento recomendado es hacer filtro de las variables para quedar con

aquellas que sí tienen información.

[MUSIC]

Antes de explicar otras formas de penalizar,

es bueno que miremos la geometría de la penalización.

Supongamos que se tienen dos variables x1 y x2.

La función objetivo que se busca minimizar, es decir,

la sumatoria de los cuadrados de los errores.

Es una función cuadrática en términos de las dos variables a optimizar.

La proyección de esta función se puede ver con estas curvas de nivel.

El estimado por mínimos cuadrados es el argumento que minimiza esta función,

que se ve así.

Al poner la penalización de la norma de beta al cuadrado, es equivalente

a restringir que la solución de los betas debe estar dentro de un círculo.

Cuyo radio depende del valor de lambda.

Entre más grande lambda, más pequeño el círculo.

La solución al problema penalizado está en el punto en el cual se encuentran las

curvas de nivel de la función de mínimos cuadrados con el círculo de la

penalización.

Si miramos el recorrido de las soluciones para varios valores de lambda,

entonces los betas estimados generan este recorrido a medida que se van encogiendo.

[MUSIC]

Ahora consideremos la penalización en norma 1.

La cual tiene la forma de penalización de beta es la sumatoria de los valores

absolutos de los beta sub j,

y se agrega de la misma forma a la sumatoria de cuadrados.

Este tipo de modelo se conoce como regresión tipo Lasso,

por least absolute shrinkage and selection operator.

Para entender esa penalización,

es más fácil revisar directamente la forma geométrica.

En este caso, la restricción asociada a la penalización tiene esta forma de diamante.

Si pensamos en el recorrido de encogimiento que se produce en el valor

estimado a los betas, vemos que es parecido al modelo Ridge.

Sin embargo,

en algún momento los vértices del diamante llevan a uno de los betas a 0.

Esto es equivalente a eliminar estas variables del modelo.

Esta es la gran diferencia entre el modelo Ridge y el modelo Lasso.

Este último tiene la capacidad de eliminar variables,

lo cual lo hace más robusto a las variables no informativas.

En términos de desempeño, también se calibra en función de lambda,

estimando la curva del MSE.

[MUSIC]

Una de las diferencias de los modelos penalizados con los de selección o

transformación de variables.

Es que en el modelo penalizado no es tan fácil definir los estimadores

intramuestrales del EPE, tales como el AIC o el BIC.

Generalmente, para los modelos Ridge y Lasso se usan validación

cruzada para estimar la curva del MSE y encontrar el valor óptimo de lambda.

Otra opción de penalización es combinar las normas vistas.

En este caso se hace una ponderación entre las dos a través de un parámetro

adicional alfa.

El beta gorro queda como el argumento que minimiza la sumatoria de los

cuadrados más la penalización.

Que es lambda veces alfa por la norma cuadrática más 1

menos alfa por la norma 1.

En este modelo, se conoce como red elástica o elastic net.

En donde la norma tiene esta forma, con la capacidad también de seleccionar

variables, pero no de manera tan rápida como lo hace el modelo Lasso.

[MUSIC]

En este video estudiamos la estrategia de penalización para calibrar modelos de

regresión lineal.

Estudiamos dos formas de penalización, Ridge y Lasso,

así como la combinación de los dos.

En general, estos modelos pueden tener muy buen desempeño.

Con la ventaja de que no es necesario resolver un problema combinatorio para

seleccionar variables, y que el parámetro de calibración es continuo.

En el próximo video, estudiaremos los modelos de clasificación lineales.

[MUSIC]

https://www.coursera.org/learn/introduccion-al-machine-learning/supplement/1wXVX/preprocesamiento-de-datos

# Preprocesamiento de datos

El preprocesamiento de datos es un elemento fundamental en el modelamiento predictivo debido a que, dependiendo del modelo, algunas características de los datos de entrada pueden afectar significativamente el desempeño de un algoritmo de machine learning, por lo cual resulta relevante la identificación del tipo de preprocesamiento de datos que necesita cada modelo y la manera de desarrollarlo.

Por lo anterior, en este documento se hará un breve repaso sobre los tipos de preprocesamiento de datos más relevantes: transformación de un solo predictor, transformación de varios predictores, gestión de datos faltantes y la eliminación de predictores.

**Transformación de un solo predictor**

Para algunos modelos de machine learning es necesario que los datos tengan la misma escala (por ejemplo, PCR), por lo cual, la manera más común de transformar cada predictor para lograr tal uniformidad es centrar y escalar los datos. Este procedimiento consiste en realizar una transformación lineal a cada predictor, de tal manera que tengan promedio igual a cero y desviación estándar igual a uno. Algebraicamente se debe realizar la siguiente transformación a cada predictor:

x~ij=xij−xˉjSj∀i∈{1,2,...n},∀j∈{1,2,...,p}x~ij​=Sj​xij​−xˉj​​∀i∈{1,2,...n},∀j∈{1,2,...,p}

xˉj=1n∑i=1nxij∀j∈{1,2,...,p}xˉj​=n1​i=1∑n​xij​∀j∈{1,2,...,p}

Sj2=1n−1∑i=1n(xij−xˉ)2∀j∈{1,2,...,p}Sj2​=n−11​i=1∑n​(xij​−xˉ)2∀j∈{1,2,...,p}

Donde:

* xijxij​ corresponde a la observación ii del predictor jj.
* xˉjxˉj​ corresponde al promedio aritmético de las observaciones del predictor jj.
* SjSj​ corresponde a la desviación estándar de las observaciones del predictor jj.
* x~ijx~ij​ corresponde a la transformación centrada y escalada de la observación ii del predictor jj.
* nn corresponde al número de observaciones.
* pp corresponde al número de predictores.

**Transformación de varios predictores**

También es posible que sea necesario combinar distintos predictores para disminuir la dimensión del espacio de salida de la función a estimar, o eliminar la redundancia de información en los datos. Los métodos más utilizados en este aspecto suelen ser las combinaciones lineales entre predictores. Por ejemplo, si en un caso particular se quieren combinar las variables predictoras x4,x5,x7yx8x4​,x5​,x7​yx8​, entonces se tendrá una nueva variable combinada z1z1​ que tendría la siguiente forma:

zi1=a4xi4+a5xi5+a7xi7+a8xi8∀i∈{1,2,...,n}zi1​=a4​xi4​+a5​xi5​+a7​xi7​+a8​xi8​∀i∈{1,2,...,n}

a4,a5,a7,a8∈Ra4​,a5​,a7​,a8​∈R

La forma más tradicional de reducir la dimensión del espacio de predictores es el principal component analysis (PCA), donde se utiliza la combinación lineal que condensa la mayor cantidad de variabilidad de los predictores en la menor cantidad de variables, evitando la redundancia mencionada anteriormente.

**Gestionar datos faltante**

Este es uno de los aspectos de preprocesamiento más comunes e importantes. En general, lo más importante para manejar situaciones con datos faltantes es entender la razón por la cual los datos no están y, para ello, se discutirán tres tipos de mecanismos de datos faltantes:

* Datos perdidos completamente aleatorios (MCAR), donde los datos perdidos corresponden a celdas totalmente aleatorias sin ningún patrón identificable.
* Datos perdidos aleatorios (MAR), donde los datos faltantes pueden estar relacionados con los valores de otras variables distintas a la que falta.
* Datos perdidos no aleatorios (NMAR), donde los datos faltantes dependen precisamente del valor de la variable que falta.

En general, eliminar las observaciones con datos faltantes puede aumentar el sesgo o disminuir la eficiencia de los estimadores que utilicemos en el modelamiento predictivo. Así, la solución de esta situación dependerá del tipo de datos faltantes con los que se esté lidiando y las estrategias se discutirán a fondo en la semana 6 de este curso.

Finalmente, es importante aclarar que los datos faltantes no afectan a todos los modelos de machine learning por igual. Por ejemplo, esta situación no es un problema para la estimación de árboles de decisión (modelo que se estudiará en la semana 5 de este curso), mientras que para una regresión lineal sí lo es.

**Eliminar predictores**

Esta es otra manera de eliminar la redundancia de información en las variables predictoras y disminuir el impacto de la maldición de la dimensionalidad (concepto que se estudiará en la semana 4). Para la eliminación de variables predictoras se utilizan algoritmos de selección de variables que serán vistos a profundidad en la semana 6. En general, estos algoritmos se dividen en dos:

* Algoritmos de filtro, los cuales utilizan métricas sobre la redundancia de información entre los predictores (por ejemplo, la matriz de correlaciones), antes de estimar un modelo de machine learning.
* Algoritmos de envoltura, los cuales calibran el modelo predictivo (por ejemplo, una regresión lineal) con diferentes subconjuntos de predictores y los comparan para identificar las variables que tienen mayor información para aproximar la variable de respuesta.

La gran diferencia entre estos dos enfoques de selección de variables consiste en la eficiencia computacional, ya que la calibración de varios modelos por parte de los algoritmos de envoltura implica un uso mayor de recursos computacionales a medida que aumenta considerablemente el número de variables.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/QSD88/conceptos-fundamentales-del-problema-de-clasificacion-modelos-de-clasificacion

# Conceptos fundamentales del problema de clasificación. Modelos de clasificación lineales

Hola, en este video introduciremos los conceptos fundamentales para el problema

de clasificación en machine learning, y veremos el modelo de regresión logística.

[MUSIC]

Recordemos que los modelos de clasificación se caracterizan porque

la variable de respuesta Y es categórica, es decir, toma valores

en un conjunto de clases excluyentes que no tienen una métrica numérica definida.

Así, por ejemplo, se quiere predecir si una transacción electrónica es fraude o no

a partir de la información de la transacción.

O, por ejemplo, se quiere determinar si la persona que pide un préstamo

bancario cumplirá o no con la deuda.

[MUSIC]

El número de clases puede ser más de dos, por ejemplo, si se quiere predecir el

dígito escrito a mano por una persona a través de la digitalización de la imagen,

con pixeles como variable.

En este caso, la variable de respuesta tiene diez clases.

Recordemos que la visualización más conveniente para el problema de

clasificación es la partición generada por la función f gorro sobre el espacio X.

Por ejemplo, en el caso de fraude, con las variables predictoras X1 y X2,

una función que clasifica, genera la partición que se muestra en la gráfica.

[MUSIC]

Como you habíamos visto, en el problema de clasificación, el error teórico

de predicción se puede definir como el EPE(f) es el valor esperado de una

función indicadora de cuando el Y estrella es diferente del f(X) estrella.

Esto es, la probabilidad de que Y estrella sea diferente del f(X) estrella,

donde I es la función indicadora que toma el valor de 1 cuando la

condición que contiene es cierta.

Con esto es posible definir el mejor predictor posible,

el f óptima de X estrella,

como la clase para la cual la probabilidad que Y estrella tome este valor sea máxima.

Por ejemplo, si en el punto X estrella,

la probabilidad que Y estrella sea fraude es 0.75,

y de que sea no fraude es 0.25, entonces la función f óptima predice fraude.

Por supuesto, esta función es desconocida.

Así como el MSE estima el EPE en el problema de regresión,

en el caso de clasificación podemos definir la versión empírica del EPE como

el promedio de la función indicadora de cuando el Y sub i es diferente al f(X1).

Este error está entre 0 y 1, y también presenta la forma convexa en función de la

flexibilidad, con lo cual se puede calibrar los modelos.

También es posible usar la exactitud del modelo,

o el accuracy, el cual se define como 1 menos el error.

En clasificación, es posible usar otras métricas para medir el desempeño de los

modelos, teniendo en cuenta que hay varias formas en las cuales se puede producir

el error en la predicción.

Por ejemplo, en el caso de fraude en la transacción, una forma del error es

predecir fraude cuando no lo es, y la otra es predecir no fraude cuando sí lo es.

Sin embargo, estas métricas serán definidas más adelante.

[MUSIC]

Por facilidad en las explicaciones de los conceptos de clasificación,

nos concentraremos en el caso de dos clases, es decir, cuando el espacio es Y,

tiene dos categorías, como por ejemplo, fraude o no fraude.

Si bien la función que predice va al espacio Y, es más conveniente

numéricamente si la función que predice está en los número reales.

Por ejemplo, puede ser la probabilidad o la verosimilitud de que Y estrella sea una

clase particular.

La función a estimar podría ser f gorro de X estrella,

es igual a la probabilidad estimada de que y estrella sea fraude dado que X

es igual a X estrella, donde la probabilidad estimada p gorro es para

determinar la probabilidad a partir de los datos.

Este enfoque de estimar una función numérica,

se conoce como estimación de función discriminante.

Una vez se estima la función discriminante, si esta es mayor a un

límite o threshold, se predice una clase, y si no, se predice la otra.

Por ejemplo, en el caso de la probabilidad,

se predice fraude cuando la probabilidad estimada es mayor a 0.5.

[MUSIC]

Como ejemplo inicial,

miremos el algoritmo de K vecinos en el problema de clasificación.

En este caso, las clases de Y se pueden codificar numéricamente como 1 y 0.

Por ejemplo, puede tomar el valor de 1 si la transacción es fraude y 0 si no lo es.

Las valores de 1 y 0 no significan ninguna medición,

pero se escogen por conveniencia numérica.

Veremos que para una X estrella, estimamos f gorro de X estrella como la

proporción de puntos en el vecindario que toman el valor de 1.

Esta proporción es un estimador directo de la probabilidad de que Y sea igual a 1.

En el ejemplo,

se estima directamente la probabilidad de que la transacción sea fraude.

Esta probabilidad es la función discriminante.

Cuando se va a predecir si en una transacción en particular hay fraude o no,

se mira si el f gorro de X es 3, es mayor a 0.5.

Notemos que a medida que el número de vecinos K se hace más pequeño,

entonces la función estimada se hace más flexible.

También miremos que en este caso, dado que se codifica numéricamente la variable Y,

el problema lo podemos observar como un problema de regresión.

En la representación usual de partición, una función flexible se podría ver como

esto, y un modelo lineal de clasificación, por supuesto, genera una partición lineal.

Hablando de los modelos lineales de clasificación, veremos uno de los enfoques

más usados, que se conoce como el modelo de regresión logística.

[MUSIC]

El modelo de regresión logística pertenece a la familia de los modelos lineales

generalizados, los cuales funcionan como un modelo de regresión lineal,

pero en los que la variable de respuesta Y tiene una distribución particular en la

familia de distribuciones de probabilidad exponenciales.

En regresión logística, la variable Y se codifica también como 1 o 0.

De esta forma, se puede decir que Y se distribuye Bernoulli con parámetro P,

que es igual a la probabilidad de que Y sea igual a 1.

Recordemos que la función de probabilidad de una variable

Bernoulli es P a la a, por (1-p) a la a, para a igual a 0 o 1.

La idea es usar el parámetro P, que la probabilidad que Y sea

igual a 1 en función de las variables predictoras X.

En particular, esta relación se debe dar a través de una función lineal.

Es decir, relacionar Px con beta 0 + beta 1 por X1, hasta beta p por Xp.

Por supuesto, el parámetro Px no se puede conectar directamente con el hiperplano,

esto generaría muchos problemas en el modelo, como vemos en la gráfica.

[MUSIC]

El primero de ellos es obvio.

El hiperplano está en el rango de lo real,

mientras que la probabilidad está en el intervalo de 0 a 1.

El segundo es un poco más sutil, y es el problema en el cual,

cuando el valor esperado de Y está muy cercano a 0 o a 1,

la varianza de Y está muy cercana a 0, dado que se define como Px por 1-Px.

[MUSIC]

La solución más directa, que es justamente la propuesta de los modelos generalizados,

es vincular Px con el hiperplano a través de una función de link,

la cual, para efectos de interpretación y de estimación, debe ser invertible,

es decir, monótona y suave.

El modelo de regresión logística se caracteriza porque

la función que relaciona la probabilidad de que Y sea igual a 1,

con el hiperplano beta transpuesto X es la función logística, de forma tal que,

la probabilidad de que Y sea igual a 1 dado X, es el exponente de beta

transpuesto por X, sobre uno más el exponente de beta transpuesto por X.

Veamos que la función logística, para un número real U,

se define como e a la u sobre 1, más e a la 1.

Y se ve así, obteniendo la inversa de la función logic,

se tiene que beta transpuesto por X es igual a beta 0 más beta 1 por X1,

hasta beta p por X sub p, que es igual al logaritmo de Px sobre 1- Px.

Esto se conoce como logaritmo del odds ratio, que está definido por P sobre 1-P.

Así es como se pueden interpretar los betas asociados a un modelo de regresión

logística.

[MUSIC]

Los parámetros del plano se estiman por máxima verosimilitud.

Esto es, se maximiza el logaritmo de la función de verosimilitud de los

datos dados por l(beta) es igual a la suma de Y sub i por el logaritmo de P

sub i más (1- Y sub i) por el logaritmo de (1-P sub i), donde P sub i

es la probabilidad de que Y sub i sea igual a 1, dado que X es igual a X sub i.

Esta estimación no es posible de realizar analíticamente, es decir,

derivando e igualando a 0.

Se usan métodos numéricos iterativos para estimar.

En general,

es un problema computacionalmente sencillo donde es posible utilizar un método de

descenso como Newton-Raphson o gradiente descendiente.

En la mayoría de aplicaciones, se usa una modificación

del método Newton-Raphson llamada mínimos cuadrados ponderados iterativos, quadratic

weighted least squares, los cuales tienen algunas ventajas numéricas y estadísticas.

Aunque en machine learning el interés no está en hacer inferencias sobre

los parámetros beta, si el tamaño de muestra es suficientemente grande,

es posible hacer inferencias sobre estos usando una distribución normal.

En la cual los estimadores tienden a tener esta distribución.

[MUSIC]

Una vez se tienen los parámetros de la recta estimados,

es posible hacer predicciones para puntos particulares X estrella.

Miremos que en este caso,

f gorro en X estrella es la probabilidad estimada que Y estrella sea igual a 1.

Esta probabilidad es mayor a 0.5

si el beta estimado transpuesto por X es mayor a 0.

Es decir, que el algoritmo predice que Y estrella es igual a 1 si el

plano evaluado en X estrella es positivo.

Por supuesto,

esto implica que la partición generada sobre el espacio X es lineal.

De este lado, el plano es positivo y este lado es negativo.

[MUSIC]

En este video, estudiamos los conceptos fundamentales de los modelos predictivos

en el contexto de clasificación.

Además, explicamos el modelo de regresión logística como la forma más directa de

adaptar los modelos lineales a este contexto.

En el próximo video, estudiaremos otros

algoritmos de clasificación lineales.

[MUSIC]

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/ldrR9/otros-modelos-de-clasificacion-lineales-naive-bayes-lda-y-qda

# Otros modelos de clasificación lineales: Naive Bayes, LDA y QDA

Hola. En este video introduciremos algunos algoritmos clasificadores lineales

que funcionan basados en la función de distribución de probabilidad

de las variables predictoras X,

condicionada a la clase particular de Y.

En particular veremos Naive Bayes, LDA y QDA.

Siendo completamente estrictos,

no todos estos algoritmos son lineales,

solamente el LDA lo es,

pero todos ellos funcionan con el principio fundamental del teorema de Bayes.

La idea fundamental es considerar la distribución de probabilidad de un predictor X\_j,

condicional a la clase particular en la cual puede estar Y. Por ejemplo,

supongamos que Y es Fraude o No Fraude en una transacción electrónica.

Si X\_1 es el monto de la transacción, entonces,

si Y es Fraude, la distribución de X\_1 se ve como lo muestra la función densidad azul.

Si Y es No Fraude, entonces la densidad de probabilidad X\_1 se ve como la curva roja.

Estas funciones de densidad corresponden a los modelos de

probabilidad de X\_1 condicionales a Y. Es decir,

existe g de X\_1,

dado que Y es Fraude y g de X\_1 dado que Y es No Fraude.

El propósito es inferir lo contrario; es decir,

la probabilidad de que Y sea Fraude,

dado que X\_1 toma un valor particular X estrella.

Para lograr ese tipo de probabilidades se puede usar el teorema de Bayes

como la probabilidad de que Y sea Fraude dado que X\_1 es igual a X estrella,

es g de X estrella,

dado que Y es igual a Fraude por la

probabilidad de que Y sea Fraude sobre el g de X estrella,

donde la probabilidad que Y sea igual

a Fraude corresponde a las probabilidades a priori de que una transacción sea

Fraude y g de X estrella a la densidad de

probabilidad no condicional de X\_1 evaluada en X estrella,

donde por ley de probabilidades totales,

g de X\_1 es igual a g de X\_1,

dado que Y es igual a Fraude por la probabilidad de que Y se sea Fraude,

más g de X\_1,

dado que Y es No Fraude, por la probabilidad de que Y sea No Fraude.

Empezaremos con la aplicación más directa y sencilla de este tipo de algoritmos.

Se conoce como el clasificador ingenuo de Bayes o Naive Bayes.

En este caso, se extiende la definición que vimos del teorema de Bayes,

para cuando se tienen múltiples predictores X\_1 hasta X\_p.

Antes de usar Bayes,

recordemos que se requieren de las probabilidades condicionales de X\_1 hasta X\_p dado Y,

para las cuales se tiene que la probabilidad

de X\_1 hasta X\_p dado Y es igual a la probabilidad X\_1 dado Y

por la probabilidad de X\_2 dado X\_1 y Y por la probabilidad de X\_3 dado X\_2, X\_1 y Y

hasta llegar a la probabilidad X\_p dado X\_1, X\_2 hasta X\_p menos 1 y Y.

Para simplificar el algoritmo y hacerlo más fácil de estimar y usar,

Naive Bayes asume que los predictores son independientes entre sí.

Esto es para todo k menor igual a p,

la probabilidad de X\_k dado X\_1,

X\_2 hasta X\_k menos 1 y Y,

es igual a la probabilidad X\_k dado Y.

De esta forma se puede trabajar con cada proyector X\_j condicional a Y,

independientemente de los demás.

Al usar el teorema de Bayes,

en el caso que p es igual a dos se obtiene

que la probabilidad de que Y sea igual a c dado X\_1, X\_2

es igual a g de X\_1 dado que Y es igual a c, por g de X\_2

dado que Y es igual a c por la probabilidad de Y sea igual a c,

sobre g de X\_1 por g de X\_2.

Otra forma de verlo es que Naive Bayes

ignora la distribución conjunta de los predictores.

Por ejemplo, en el caso de Fraude,

supongamos que además de X\_1 que es el monto de la transacción,

se tiene X\_2 como el tiempo desde la última transacción.

Supongamos las distribuciones de conjuntos

de X\_1 y X\_2 condicionadas a los 2 valores de Y,

la realización de los puntos tienen estas formas para cada clase.

Como se ve en el gráfico,

X\_1 y X\_2 pueden estar correlacionadas.

Naive Bayes usa las distribuciones marginales de manera independiente.

Las ventajas de usar Naive Bayes es la simplicidad de

su uso y la facilidad de escalar a grandes números de proyectores.

Existen muchas alternativas para estimar las distribuciones de

probabilidad de cada X\_1 y condicional a Y. Una opción,

cuando los predictores son continuos, por ejemplo,

es suponer que tienen distribución normal,

para lo cual se deben estimar la media y la varianza.

En este caso, la clasificación obtenida será

lineal como un plano ortogonal al vector de diferencia de medias.

Otra opción es usar estimaciones no paramétricas,

como por ejemplo, estimación de densidad por Kernel.

Para evitar posibles errores generados por el supuesto que

los predictores son independientes, el algoritmo LDA,

por sus siglas en inglés de "linear discriminant analysis"

o análisis discriminante lineal,

si tiene en cuenta la distribución conjunta de X\_1 hasta X\_p.

Sin embargo, dado que las distribuciones generales pueden dificultar la estimación,

se asume que los predictores son continuos y tienen distribución normal multivariada,

cuando Y se fija a una clase particular.

Por ejemplo, supongamos que Y puede tener 2 valores, clase 1 y clase 2.

Si Y pertenece a la clase 1,

entonces el vector de proyectores Y,

X\_1 hasta X\_p se distribuye como normal p variado con media μ \_1 y varianza Σ\_1,

donde μ \_1 es el vector de medias y

Σ\_1 es la matriz de varianza covarianza de X cuando Y es igual a 1.

Cuando Y es igual a 2, tenemos también una distribución normal,

pero con Naive μ \_2 y varianza Σ\_2.

Si p es igual a 2,

entonces las curvas de nivel de las funciones de densidad respectiva se ven así.

En el centro de la elipse está miu,

mientras que sigma da la forma.

Cuando hay correlación entre X\_1 y X\_2,

entonces los ejes de la elipse son oblicuos y no caen paralelos a las coordenadas.

En este caso, por ejemplo,

hay correlación positiva y en este otro hay correlación negativa.

LDA hace un supuesto adicional y es que Σ\_1 es igual a Σ\_2,

es decir, que las matrices de varianza-covarianza son iguales.

Después de resolver el teorema de Bayes usando estas

distribuciones para una X particular en rp,

se obtiene que la probabilidad que Y estrella sea

igual a 1 es mayor a la probabilidad que Y estrella sea igual a 2,

si la inversa de la matriz Σ por μ \_1

menos μ \_2 transpuesto por X estrella es mayor a una constante c,

donde el valor de c depende de las probabilidades a priori de Y, p\_1 y p\_2.

Notemos que contrario a Naive Bayes,

la inversa de la matriz sigma

hace que se tenga en cuenta la correlación entre los predictores.

Esta forma de clasificación corresponde a la participación lineal en el espacio X.

En la gráfica veremos las dos distribuciones.

Mientras Naive Bayes genera la partición ortogonal a la diferencia de medias,

LDA tiene en cuenta la forma de las elipses.

Como los parámetros miu y sigma son desconocidos,

pueden ser estimados a partir de los datos.

Si X\_ik representa los datos X\_i para cuando Y es igual a k,

entonces el miu estimado 1 es el promedio de los datos en el grupo 1

y el miu estimado 2 es el promedio de los datos en el grupo 2.

Además, la matriz de varianza se puede estimar como un

promedio ponderado de forma tal que sigma gorro es

igual a n\_1 menos 1 por S\_1 más n\_2 menos 1 por S\_2 sobre n\_1 más n\_2 menos 2.

Donde n\_1 y n\_2 son el número de datos en las clases 1 y 2 respectivamente.

Notemos que si en realidad los datos condicionales

a la clase de Y tienen una distribución normal multivariada,

entonces el procedimiento de clasificación con LDA es óptimo,

siempre y cuando las matrices de varianza sean iguales para cada clase.

El clasificador no será forzado a que sea lineal,

sino que esto es un subproducto de los supuestos de normalidad y de varianzas iguales.

De esta forma, obtener clasificadores más flexibles

implica relajar un poco estos supuestos.

Una aproximación bastante simple

es quitar el supuesto de que las matrices de varianza serán iguales.

¿Qué pasa si usamos el teorema de Bayes en este supuesto?

El resultado se conoce como el modelo QDA o análisis discriminante cuadrático.

Lo que va a pasar en este caso,

es que la función discriminante no será lineal,

sino que será una función cuadrática de los inputs.

Por esta razón, la partición generada en el espacio X tendrá una curvatura,

como lo muestro en la gráfica.

Cuando el método LDA,

visto como un algoritmo de clasificación lineal,

se compara con la regresión logística,

es difícil saber cuál funciona mejor para un conjunto particular de datos.

En general, es algo que se debe decidir por medio de una muestra de test.

Sin embargo, es importante notar que el desempeño de LDA se

va a ver afectado cuando los datos se alejan de cumplir los supuestos realizados.

Mientras que en regresión logística el desempeño se afecta

porque la función F óptima no es lineal,

es decir que queda mal especificado.

Otra diferencia entre estos dos modelos lineales

es que el LDA solo permite predictores continuos,

mientras que en el modelo logístico se pueden tener predictores

categóricos codificados como variables binarias.

En este video explicamos otras alternativas para generar modelos de clasificación

basados en el modelaje probabilístico de los predictores X,

condicionados a las clases de Y.

En particular, el algoritmo LDA resulta

ser un clasificador lineal como regresión logística.

En el próximo video estudiaremos otras métricas de evaluación de desempeño

para los modelos de clasificación.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/3ulge/medidas-de-desempeno-en-modelos-de-clasificacion-matriz-de-confusion-roc-auc-y

# Medidas de desempeño en modelos de clasificación: matriz de confusión, ROC, AUC y otras métricas

Hola, en este video

veremos que en el problema de clasificación se pueden

definir otras métricas para evaluar el desempeño de los modelos predictivos.

Esto no es solo una opción para tener más métricas, sino que a veces,

dependiendo de los datos,

es necesario utilizar otras alternativas.

Supongamos por ahora que y tiene dos clases,

las cuales podemos codificar como 1 o 0.

También por motivos de semántica podríamos llamar al 1 éxito y al 0 fracaso.

Por ejemplo, cuando se quiere reconocer si una transacción es fraude,

podemos llamar a la categoría fraude como 1 o éxito,

y esto facilita la comprensión de las métricas.

Lo que debemos considerar, es que la métrica del

error simplemente toma en cuenta cuando la predicción es incorrecta.

Sin embargo, pueden ocurrir dos clases de error.

Un error tipo 1, que es predecir éxito,

f gorro igual a 1,

cuando en realidad es fracaso,

y es igual a 0.

Un error tipo 2, que es predecir fracaso,

f gorro igual a 0,

cuando en realidad y es igual a 1.

De esta forma podemos contabilizar los dos errores para medir el desempeño del modelo.

La razón de esto es tener más información de cómo se comporta un algoritmo,

pero también prevenir ciertos errores que pueden ocurrir en las muestras desbalanceadas.

Por ejemplo, qué tan útil sería

un modelo que no cometa errores al predecir que una transacción es no fraude,

pero que al predecir si la transacción es fraude,

se equivoca el 70 por ciento de las veces.

¿Sería el modelo que queremos? Empecemos entonces definiendo la matriz de confusión,

en este caso se organizan las predicciones

f gorro de x\_i con respecto a las observaciones reales de y\_i.

En las columnas pondremos los valores predichos.

La columna de la izquierda tiene los valores predichos

como éxito y la derecha los fracasos.

Asimismo, en las filas de la matriz pondremos los valores reales de y.

En la fila superior están los casos cuando y es

éxito y en la inferior cuando y es fracaso.

Al cruzar la matriz tenemos las cuatro posibilidades.

En esta celda tenemos el número de casos que en realidad son 1 y predecimos como 1.

Estos casos se conocen como verdaderos positivos,

es decir, quedaron bien predichos.

Ahora, en esta celda,

los reales 0 que predecimos bien,

los llamamos verdaderos negativos.

En esta esquina ponemos los falsos positivos,

es decir, los que predecimos como 1 pero son 0.

Y en esta esquina ponemos los falsos negativos.

Por supuesto, lo que se busca es que todos los

casos queden contenidos en las celdas de la diagonal.

Todos los 1 los predicen bien y todos los 0 los predice bien.

En la matriz, es posible ver

algunas proporciones de los números en la diagonal con respecto al resto.

Sin embargo, es más útil desarrollar algunas métricas derivadas.

La primera será la probabilidad de detectar éxitos o 1,

esto es, la probabilidad de predecir 1,

dado que y es igual a 1.

Esta medida también recibe el nombre de sensibilidad.

Se calcula como los verdaderos positivos sobre el total de positivos.

Pongamos un ejemplo numérico.

Supongamos que se hace un algoritmo para predecir

fraude y se evalúan 100 datos en una muestra de test,

obteniendo la siguiente matriz de confusión.

En este caso, la sensibilidad del modelo es la probabilidad de detectar fraude,

cuando en realidad es fraude, es decir,

30 sobre 40, que es igual a 0,75.

Otra medida de interés es la probabilidad de detectar que no es fraude o no éxito,

cuando en realidad no lo es.

Esta métrica se conoce como especificidad

o tasa de verdaderos negativos y se calcula como,

los verdaderos negativos sobre el número total de negativos.

En nuestro ejemplo, la especificidad es 48 sobre 60,

que es igual a 0,8.

De manera similar, se puede calcular la probabilidad de que sea éxito,

dado que se predice éxito.

Esto es, dado que se predice fraude,

¿cuál es la probabilidad de que realmente sea fraude?

Y esta medida se conoce como precisión,

que se define como verdaderos positivos sobre verdaderos positivos más falsos positivos.

1 menos la precisión,

se conoce como la tasa de descubrimiento de falsos: false discovery rate.

En el caso del ejemplo tenemos que la precisión es 30 sobre 42,

5 séptimos y el false discovery rate es 2 séptimos.

De la matriz de confusión,

también se puede calcular el error de predicción y la

exactitud o el accuracy como los verdaderos

positivos más los verdaderos negativos sobre el número

total de datos y el error es 1 menos el accuracy.

Ahora bien, es cierto que métricas como la sensibilidad

y la especificidad nos dan la idea de cómo se están cometiendo los errores en el modelo.

Pero una pregunta válida es si en realidad lo necesitamos para calibrar,

¿por qué no simplemente usar el error para calibrar?

Para responder esta pregunta, pongamos un ejemplo.

En el mismo caso de detectar fraudes en compras no presenciales,

se estima un modelo de clasificación, de regresión logística.

Al evaluar el modelo en una muestra de test de tamaño 2.100,

se obtiene la siguiente matriz de confusión.

Notemos que el error de predicción es 20 por ciento

y tanto la sensibilidad como la especificidad son 80 por ciento.

Así mismo, miremos que la mayoría de los casos observados corresponden a no fraude.

Como es natural, dado que posiblemente la mayoría de transacciones no son fraude.

Si ignoramos la sensibilidad y la especificidad,

¿cómo podríamos mejorar el modelo de predicción sin estimar ni siquiera otro modelo?

La respuesta que muchos intuyen es correcta.

Para todos los puntos en la muestra de test predecimos no fraude.

De esta forma se genera la siguiente matriz de confusión.

Note que en este caso la sensibilidad es 0 y la especificidad es del 100 por ciento.

Lo interesante es que el error de predicción es 100 sobre 2.100,

que es 4,76 por ciento.

Esto quiere decir que bajamos el error de

predicción del 20 por ciento a menos del 5 por ciento,

con un método que siempre predice "No".

Esta disminución en el error parece muy buena.

Sin embargo, es importante

considerar si este algoritmo que predice siempre "No" es útil.

Significa que en total se equivocan menos veces que la regresión logística,

pero es incapaz de reconocer casos de fraude,

que es seguramente el principal objetivo del análisis.

Estas situaciones pueden aparecer cuando las clases 1 y 0 son desbalanceadas.

Esto es, la proporción de datos en una clase es mucho más alta que en la otra.

La respuesta sobre si es mejor

la regresión logística o el algoritmo que predice siempre "No",

depende de la aplicación particular que se lleva a cabo.

Sin embargo, notemos que si

el algoritmo clasificador f gorro o es una función discriminante,

la cual predice 1 si sobrepasa cierto límite, como por ejemplo,

si se estima la probabilidad de límite es 0,5,

entonces para mejorar la sensibilidad o la especificidad respectivamente,

es posible cambiar el límite de corte entre las dos clases.

Esto se puede hacer para ajustar el desbalance entre sensibilidad y especificidad;

o por el contrario, para lograr un nivel de error adecuado. Miremos cómo funciona.

El modelo de regresión logística estima la probabilidad de que y pertenezca a la clase 1.

En general, para minimizar el error se predice 1 si f gorro es mayor a 0,5.

Si por ejemplo, se cambia el límite a 0,3,

entonces se predice 1,

cuando f gorro es mayor a 0,3.

Por supuesto, la sensibilidad en este caso aumenta y la especificidad baja.

Teniendo en cuenta este balance entre la sensibilidad y la especificidad,

que puede ser calibrada con el límite de predicción,

es necesario desarrollar métricas que permitan encontrar los mejores modelos

independiente del límite con el cual se decida predecir, 1, 0.

Una métrica adecuada es entonces,

medir el conjunto de valores de sensibilidad y

especificidad que se obtienen para todos los valores posibles de límites de predicción.

Esto genera una curva, que se conoce como la curva ROC.

En el eje vertical ponemos la sensibilidad del modelo y en el horizontal,

para que sea más fácil de leer,

1 menos la especificidad.

A medida que cambia el límite de la predicción o "transform",

la sensibilidad va aumentando y la especificidad va disminuyendo.

De esta forma la curva parametrizada es creciente y une estas dos esquinas.

Como lo que se busca es que tanto la sensibilidad como la especificidad sean altas,

lo que se quiere es que la curva esté lo más arriba y lo más a la izquierda posible.

La curva óptima sería esta.

Un modelo será mejor predictor que otro

si la curva ROC correspondiente está completamente dominada.

Sin embargo, como no es posible asegurar dominancia de un modelo sobre otro,

sino que las curvas se pueden cruzar en algún punto,

debemos buscar una métrica única que permita calibrar no hayos.

Una solución obvia es tomar la integral bajo la curva o el área bajo la curva ROC.

Esta métrica, por sus siglas en inglés,

se conoce como el AUC.

Por supuesto el AUC está entre 0 y 1.

Entre más cerca de 1 esté mejor el modelo.

Notemos que si un AUC es menor a un medio, no tiene mucho sentido.

Dado que en este caso,

si se intercambian las clases predichas 1 por 0 y 0 por 1,

entonces, se obtendría un valor de 1 menos el AUC anterior.

Antes de terminar las métricas de clasificación,

es importante notar que existen muchas otras métricas,

algunas más usadas que otras.

Quisiera mencionar dos de ellas que son de común uso en la práctica.

La primera, es el puntaje F1 score,

la cual corresponde a un tipo de promedio entre sensibilidad y precisión;

es decir, la probabilidad de identificar 1,

dado que es 1 y la probabilidad que sea 1 dado que se predice 1.

Por supuesto, se busca que esta métrica sea alta.

La definición exacta es F1 es 2 por la

precisión por la sensibilidad sobre la precisión más la sensibilidad.

La segunda métrica es Kappa,

que es una métrica que mide la capacidad del modelo de predecir una

clase correctamente con respecto a la predicción aleatoria de cada clase.

Su fórmula exacta se puede ver así.

En este video desarrollamos las métricas usadas en los modelos de clasificación,

las cuales van más allá del error de predicción.

En el próximo video estudiaremos cómo calibrar los

modelos lineales de clasificación más allá de la selección de variables

, como por ejemplo,

estrategias de penalización o transformación de variables.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/Xa3HA/introduccion-a-modelos-no-lineales-para-regresion-y-clasificacion

# Introducción a modelos no lineales para regresión y clasificación

Hola, en este video introduciremos los conceptos

y estrategias fundamentales para estimar modelos no lineales,

tanto para regresión como para clasificación.

Lo primero que debemos considerar es qué significa

en realidad que un modelo sea no lineal.

Supongamos que se tiene una única variable predictora x\_1.

La diferencia más obvia es que los modelos lineales,

la relación entre x\_1 y y sea de manera lineal,

como f de x\_1 es igual Beta 0 más Beta 1 por x\_1.

Sin embargo, en los modelos no lineales esta función no es

lineal y muchas veces ni siquiera puede ser expresada analíticamente.

Esto permite controlar la flexibilidad de las funciones,

sin que necesariamente está sujeta al número de variables a usar,

sino manipulando la flexibilidad de la función misma.

Un ejemplo típico de estos algoritmos es el de k vecinos,

que hemos usado en diferentes ejemplos.

Para entender mejor la gran cantidad de algoritmos que existen,

empezaremos por estudiar modelos no lineales univariados,

es decir, que tienen solamente un predictor.

Esto para evitar confusiones con el número de variables,

con el cual siempre se puede aumentar la flexibilidad.

En el caso univariado,

podemos definir tres estrategias básicas para estimar funciones flexibles.

La primera son los suavizadores de locales.

La segunda son las bases funcionales de la misma variable,

y la tercera son los enfoques dentro

de algoritmos propios para evitar la maldición de la dimensionalidad,

como redes neuronales, árboles y ensamblaje.

Veremos los dos primeros enfoques,

mientras que los que están contenidos en

el tercer grupo los estudiaremos cada uno por aparte.

Empecemos con los suavizadores locales.

En este caso, la idea es que f gorro de x estrella

es un promedio localmente definido alrededor de los puntos más cercanos a x estrella.

Se fundamentan en el supuesto de que la función que mejor predice f óptima

es relativamente suave,

es decir, no presenta saltos abruptos,

con lo cual la mejor información para predecir y estrella es el promedio de los y\_i,

tal que los x\_i correspondientes estén cercanos a x estrella.

La calibración de estos modelos se hace a partir de controlar el tamaño de la vecindad,

con el cual se hacen los promedios locales.

De hecho, k vecinos,

el que ya conocemos,

es un tipo clásico de suavizador local.

El número de vecinos k representa el tamaño del vecindario local.

Aparte de k vecinos,

existe otro tipo de suavizadores lineales llamados o suavizadores por kernels.

La idea no es controlar el número de vecinos,

sino el tamaño de la ventana sobre la que se toma el promedio, es decir,

el radio del intervalo que define el vecindario de

x estrella en el espacio de x. Por ejemplo,

supongamos que tomamos el intervalo definido entre x estrella menos h

y x estrella más h para un valor de h. En este caso,

el estimador de f gorro de x estrella se define como el promedio de los y\_i,

cuyo x\_i respectivos,

se encuentran dentro del intervalo.

En este caso h controla la flexibilidad de la función.

Si h es muy grande, entonces la función es muy plana, es decir, con sesgo.

Si h es muy pequeño,

entonces la función tiene varianza.

La generalización de este último

ejemplo es lo que se conoce como suavización por kernels.

Esta generalización se da en los pesos que se le otorgan a los puntos en el vecindario,

siguiendo el concepto de la suavidad.

Entonces sería conveniente si los puntos más

cercanos reciben más peso en el promedio que los puntos más lejanos,

incluso si todos están dentro del vecindario.

Pensemos en la función que define estos pesos.

En el caso del ejemplo,

esta función es plana sobre el intervalo

x estrella menos h hasta que x estrella más h. De esta forma,

podemos pensar en f gorro como el promedio

ponderado donde los pesos w\_i están definidos por esta función.

Por supuesto, debemos dividir esta suma

ponderada sobre la suma de los pesos correspondientes.

La definición de pesos iguales corresponde al kernel rectangular.

Esto es f gorro de x estrella se puede representar como 1 sobre la suma de los

pesos por la sumatoria de los y\_i por las funciones k de x estrella menos x\_i sobre h,

donde estas funciones k se definen como k de u es un medio

por una función indicadora de cuando u está en el intervalo desde menos 1 a 1.

Notemos que si el valor absoluto de x estrella menos x\_i es menor a h,

entonces, y\_i cuenta para el promedio.

Para generalizar la idea de los kernels,

supongamos ahora un kernel triangular en donde los puntos más cercanos valen más.

En este caso la función de pesos k de u se puede definir como k de u es igual a 1

menos el valor absoluto de u por una función

indicadora de cuándo u está en el intervalo desde menos 1 a 1,

y u es x estrella menos x\_i sobre h,

la cual se representa en la gráfica.

Las diferencias en las funciones estimadas pueden ser sutiles.

Sin embargo, el kernel rectangular puede ser

más agresivo y hacer overfitting más rápido.

En general, es posible usar más tipos de kernel,

siempre y cuando cumplan con estas condiciones.

La primera es que la función debe ser positiva,

es decir, k de u debe ser mayor o igual a 0 para todo.

La segunda es que la integral de k de u debe ser igual a 1.

La tercera es que debe estar centrada en 0.

Es decir, la integral de k de u por u debe ser igual a 0,

y la cuarta es que tenga el segundo momento finito.

Es decir, la integral de k de u por u al cuadrado debe ser finita.

Uno de los kernels más usados es el Kernel Gaussiano,

el cual corresponde a la función de densidad de una variable aleatoria normal estándar.

En este caso, k de u es igual a 1 sobre la raíz cuadrada de 2 por Pi,

por el exponente de menos u al cuadrado sobre 2.

El parámetro de la calibración es,

por supuesto, el tamaño de la vecindad.

Es decir, k para k vecinos y h para el suavizador por kernels.

Esta calibración debe realizarse en una muestra de validación.

Sin embargo, al igual que los modelos lineales,

existe un atajo para estimar el EPE,

sin necesidad de partir la muestra.

Esto es, en el contexto de regresión,

es posible estimar el EPE a través del EPE por Leave-One-Out Cross-Validation.

En general, este atajo es posible para una

familia de modelos que se conocen como suavizadores lineales,

los cuales tienen la característica de que f gorro de x\_i es igual

a la suma de j igual 1 hasta n de h\_i j por y\_ j. Es decir,

la predicción de cada punto de la muestra de entrenamiento es

calculado como una combinación lineal de las observaciones de [inaudible].

Poniéndolo en notación matricial,

el vector de predicciones sobre todos los puntos de la muestra de

train lo podemos llamar f gorro de x vector,

el cual se puede calcular en función del vector de observaciones de

y como f gorro de x es igual a la matriz h por y.

Notemos que esta clase de modelos pertenece a los

de regresión lineal y los de suavización local.

Cuando esto pasa, los elementos de la diagonal de h\_ii tienen la

característica de ser la derivada de f gorro de x\_i con respecto a y\_i.

Es decir, h\_ii es la derivada de f gorro evaluado en x\_i con respecto a y\_i.

Usando esto, se tiene que el

MSE calculado por "leave-one-out cross validation" será igual

a 1 sobre n por la sumatoria de los y\_i menos el f

gorro x\_i sobre 1 menos h\_ii al cuadrado.

Es decir, tienen la misma utilidad en las métricas intramuestrales que ya conocemos.

Ahora estudiaremos el otro tipo de modelos no lineales.

Este corresponde al uso de bases funcionales.

En este caso la idea es sencilla: usar un

modelo lineal donde las variables predictoras son

funciones no lineales de la misma x.

Estas funciones son justamente las bases funcionales.

El ejemplo más sencillo es el modelo polinomial.

En este modelo las bases son: g\_1 de x es igual a x,

g\_2 de x es igual a x al cuadrado,

g\_3 de x es x al cubo hasta g\_q de x es igual a x a la q. Con estas bases se define

el modelo f gorro de x es igual

a Beta\_0 más Beta\_1 por g\_1 de x más Beta\_2 por g\_2 de x,

así hasta Beta\_q por g\_q de x,

donde el parámetro de calibración es q. Si q es igual a 3,

entonces la función estimada tiene dos puntos de inflexión como lo muestra la gráfica.

Otro tipo de bases pueden estar basadas en funciones de senos y

cosenos con frecuencias incrementales para adaptarse a diferentes curvaturas.

Estos dos tipos de bases: polinomial y

cosenos se conocen como bases globales, es decir,

aplican sobre todo el rango de la variable

x. Eso puede ser un inconveniente que hace que en

ocasiones no predigan bien porque no tienen

la capacidad de adaptarse a las variaciones locales.

Por ejemplo, si la función f óptima tiene una forma como lo muestra en la gráfica,

entonces un modelo polinomial con pocas bases tendrá sesgo en la parte ondulada,

mientras que si se usan muchas bases tiene varianza en la parte plana.

Los modelos con bases funcionales que mejor predicen son

aquellos que tienen la capacidad de adaptarse a la información local.

Generalmente para lograr ese objetivo,

las bases están definidas en intervalos más pequeños del rango completo.

Por ejemplo, una base puede ir asociada al segmento 1 del rango,

mientras otra va hacia el segmento 2 y una tercera base puede ser global.

Dentro de los modelos que usan este principio se destacan las bases de Splines,

las cuales definen polinomios sobre fragmentos locales,

los que se unen entre sí para formar una curva suave.

Para entender cómo funcionan miremos el siguiente

ejemplo de cómo estimar una función lineal a trozos.

Supongamos que se hace una partición en 5 y el

rango de x queda dividido en 2 intervalos,

uno de 0 a 5 y otro de 5 a 10.

Ahora, si queremos ajustar una función lineal

a trozos que tenga en cuenta la curvatura que hacen los datos,

se puede estimar una función lineal en cada parte.

Esto se vería así.

Sin embargo, para que la función no tenga un salto en cinco es más fácil usar f de x es

igual a Beta\_0 más Beta\_1 por x más Beta\_2 por una función h de x menos 5,

donde la función h evaluada en un número,

toma el valor del mismo número siempre que sea positivo,

sino toma el valor de 0;

es decir, h de u es u por una función indicadora de cuando u es mayor a 0,

o lo que es lo mismo, es el máximo entre u y 0.

Y se conoce como una función hinge o función de bisagra.

Como es posible ver, la función será lineal entre 0 y 5

con pendiente Beta\_1 y lineal entre 5 y 10 con pendiente Beta\_1 más Beta\_2.

Lo interesante es que cuando x es igual a 5,

la función estimada no presenta saltos y se convierte justamente en un punto de bisagra.

Miremos que si tuviéramos más intervalos,

por ejemplo 3 y 7,

simplemente sería suficiente con agregar bases de tipo hinge para cada partición,

como f de x es igual a Beta\_0 más Beta\_1 por x,

más Beta\_2 por h de x menos 3,

más Beta\_3 por h de x menos 7.

Para que la función sea más suave es mejor ajustar un polinomio en cada intervalo.

Lo más usado es un polinomio cúbico que ubique Splines.

En este caso, se usan igual bases globales más otros de tipo

hinge como f de x es igual a Beta\_0 más Beta\_1 por x,

más Beta\_2 por x al cuadrado,

más Beta\_3 por x al cubo,

más la sumatoria de j igual 1 hasta k de los

parámetros gamma j por h de x menos c\_j al cubo,

donde k es el número de divisiones entre intervalos, o sea,

que hay k más intervalos y c\_j es cada uno de estas divisiones.

Los parámetros Betas y Gammas pueden ser estimados por mínimos cuadrados.

El resultado de esta función es que las uniones entre

los intervalos son imperceptibles en términos de que no solamente no presentan saltos,

sino que en las divisiones la función coincide en la primera y la segunda derivada.

Para calcular una regresión por Splines se deben fijar los intervalos,

entre más intervalos y más pequeño cada uno de ellos,

entonces la función es más flexible.

En la práctica es más fácil usar todos los intervalos de la misma longitud.

Otra opción es usar penalización tipo Ridge o

Lasso sobre los parámetros gamma de las funciones hinge.

Notemos que si por ejemplo,

Lasso saca algunas de las bases tipo hinge es

equivalente a unir dos intervalos consecutivos.

En este video vimos dos estrategias para crear modelos no lineales muy variados,

los suavizadores locales y las bases funcionales.

Por supuesto, la gran mayoría de aplicaciones de

Machine Learning involucran múltiples predictores,

por lo cual debemos generalizar estos métodos.

En el próximo video estudiaremos la maldición de la

dimensionalidad y algunos métodos aditivos.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/pNZEE/introduccion-a-la-maldicion-de-la-dimensionalidad-y-aditividad-como-estrategia

# Introducción a la maldición de la dimensionalidad y aditividad como estrategia para evitarlo

Hola.

En este video explicaremos el problema que ocurre cuando los

modelos no lineales desarrollados se implementan con varios predictores.

Este concepto se conoce como la Maldición de la dimensionalidad.

Así mismo veremos algunos modelos que usan el principio de aditividad

para evitar los efectos de la Maldición de la dimensionalidad.

Empecemos por comprender el concepto de la Maldición de la dimensionalidad en

Machine Learning.

Por supuesto es una expresión alegórica para representar el problema de

hacer algoritmos no lineales con muchos predictores.

Los modelos no lineales que vimos el video pasado tanto suavizadores locales como

bases funcionales, están basados en el concepto de que la información cercana al

punto en el cual se quiere predecir es la más útil para realizar las predicciones.

Estos métodos son fácilmente implementables

cuando se tienen varios predictores.

Por ejemplo, en K-vecinos, es posible seleccionarlas cada observaciones de X

más cercana a X estrella, usando por ejemplo una distancia euclídeana,

igual ocurre en la suavización por Keinels.

También en los modelos en bases funcionales, por ejemplo,

Splines, en lugar de intervalos en una línea

se pueden definir rectángulos en R2 o hipercubos en general para R3.

Sin embargo, al introducir más predictores, incluso si tienen información

relevante para predecir a Y, el desempeño de los modelos empeora.

La razón de esto, es que en espacios X con mayores dimensiones,

las vecindades para obtener información local deben ser mucho más grandes, con lo

cual se crea sesgo o con muchos menos puntos, con lo cual se crea varianza.

Miremos por ejemplo el modelo K-vecinos.

Si tenemos un solo predictor, entonces cuando K es igual a tres,

el promedio para una X estrella se ve así.

Si agregamos un nuevo predictor,

entonces en el espacio R2 los puntos se separan un poco.

Para mantener la misma cantidad de vecinos, las distancias se amplían.

Al cubrir una área mayor,

la función tendrá más sesgo a la hora de predecir un nuevo punto.

En el caso de Keinels, si mantenemos el H constante y miramos las distancias

X estrella menos X sub i en valor absoluto, veremos que en mayores

dimensiones hay menos y menos puntos cada vez con lo cual se aumenta la varianza.

En el caso de bases funcionales, por ejemplo, tipo Splines, el resultado

es que se requieren muchas más variables para representar a las particiones.

Por ejemplo, cuatro particiones en una variable, requieren 16 particiones en dos

dimensiones para mantener el mismo nivel de sesgo.

Eso como you lo sabemos, aumenta la varianza.

Aunque para P igual dos o tres, el problema no parece muy grave, si te creces

significativamente, digamos más de cinco, entonces el efecto será bastante notorio.

Esto es entonces lo que se conoce como la Maldición de la dimensionalidad en

Machine Learning.

Al final de cuentas el resultado es que los métodos

predictivos no lineales en grandes dimensiones,

deben considerar estrategias para evitar el deterioro por este fenómeno.

Todos los algoritmos que estudiaremos de ahora en adelante,

tienen su propia forma de evitar la maldición.

Por ahora empezaremos estudiando cómo usar los suavizadores locales en las bases

funcionales en varias dimensiones, usando el supuesto de aditividad.

Estudiaremos dos tipos de modelos predictivos.

Los modelos adictivos generalizados y el método MARS,

Multivariate Adaptative Regression Splines.

Los modelos aditivos generalizados son una extensión natural

a la no linealidad de los modelos generalizados GLM.

Recordemos que los GLM son modelos que adaptan la teoría de la regresión lineal

para otro tipo de distribuciones en la variable respuesta Y.

Así por ejemplo,

en el caso de clasificación, la regresión logística es un tipo de modelo GLM.

Por simplicidad en la presentación,

consideraremos la regresión normal con Y continua.

Y la regresión logística cuando Y se distribuye Bernoulli con parámetro p de x.

La idea fundamental en los modelos aditivos, es que en lugar de usar una

función lineal, Beta 0, más Beta 1 por X 1 hasta Beta p por X sub p,

se usa una función aditiva como F de X sub 1 hasta X sub p es igual a F1

de X sub 1, más F2 de X sub 2 hasta más Fp de X sub p.

Donde cada función, Fj de X sub j es no lineal.

Esto es en modelos no lineales en cada una de las márgenes o variables, sin embargo,

las estimaciones no se hacen en el espacio Rp,

donde sufriríamos por la dimensionalidad.

Por supuesto al usar un modelo aditivo, se simplifica la función.

Y es un supuesto que limita las formas funcionales que se pueden obtener.

Mirémoslo gráficamente.

Supongamos que esta es la función F1, supongamos que la función F2 es lineal.

Cuando creamos el modelo aditivo, F1 más F2,

entonces la función F1 se desliza por la recta que crea la función F2.

Si bien las funciones aditivas pueden ser bastantes flexibles

y mucho más útiles para predecir que los modelos lineales,

existen muchas funciones que no podrían ser representadas.

Por ejemplo pensemos en esta función, la cual no puede ser separada en margenes.

La practicidad a estos modelos también se debe

a su conveniente algoritmo de estimación, que se conoce como Bad fitting.

La idea es que si Y es igual a F1, más F2, más Fp, más un término del error,

para estimar cada margen, se pasan a restar las demás funciones a Y.

Es decir, Y prima es igual a F1 más el Error.

Donde Y prima es igual a Y menos F2, menos F3 hasta menos Fp.

Es importante notar que al estimar cada margen por aparte

se evitan los efectos de la Maldición de la dimensionalidad.

Otra de las características de los modelos GAM,

es que al ser aditivos ofrecen algunas alternativas para interpretación

en el sentido de que se puedan separar los efectos de cada variable.

Por ejemplo, si se quiere predecir el precio por metro cuadrado de las viviendas

residenciales en función de la edad de la construcción en años X1,

y el nivel de la criminalidad de la zona,

X2, entonces podríamos revisar los efectos de cada variable.

Seguramente el efecto de la edad será decreciente, pero convexo,

mientras el de la criminalidad será creciente concavo.

Otro tipo de modelos que usan el principio de aditividad es el modelo MARS.

En este modelo para cada variable se utilizan bases funcionales tipo Splines,

generalmente lineales, es decir, de grado 1.

Esto quiere decir que cada margen se crea una función lineal en trozos.

Por ejemplo,

en el caso del precio de las viviendas se pueden tener estas dos funciones.

El precio igual a Beta 0, más Beta 1 por un h del 5, menos edad,

más Beta 2 por h de edad menos 5, más Beta 3 por h de 2 menos crimen,

más Beta 4 por h de crimen menos 2.

Sin embargo, MARS usa las propiedades de las bases Splines

para crear algunas interacciones multiplicando las bases.

Así añadiendo a la función anterior,

Beta 5 igual a h de edad menos 5 por h de crimen menos 2.

Esta última variable, que es la multiplicación de dos bases,

tiene la capacidad de cambiar solamente una celda creada por las particiones.

En la gráfica, por ejemplo, solo se afecta esta parte.

De esta forma you no es una función aditiva cuando las interacciones tienen

coeficientes diferentes de ser.

Sin embargo, al usar solo interacciones dobles,

no sufre de la Maldición de la dimensionalidad.

Generalmente para seleccionar las particiones en cada variable,

se usan las n observaciones de las variables X sub j.

Esto puede crear una gran cantidad de bases,

por lo que se requiere un proceso eficiente de selección de variables.

Este procedimiento se hace por un método forward, llegando a hacer un poco de

vertique, al cual se le hace un ajuste backward para encontrar el mejor modelo.

En este video vimos dos estrategias para crear modelos no lineales multivariados

con restricciones de aditividad para evitar los efectos de la Maldición

de la dimensionalidad.

En el próximo video estudiaremos modelos basados en particiones del espacio X y

árboles de decisión

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/2tFVY/introduccion-a-algoritmos-basados-en-particiones-y-arboles-de-decision

# Introducción a algoritmos basados en particiones y árboles de decisión

Hola en este

video veremos los algoritmos de predicción basados en particiones del espacio de

entrada y en particular los árboles para regresión y para clasificación.

Los algoritmos predictivos basados en particiones poseen la idea de partir

del espacio X en múltiples partes excluyentes,

de forma tal que para cada parte se predice una función constante.

En el caso de regresión para todos los X se estrecha en una parte,

entonces se predice un número real y para el problema de clasificación una clase.

El objetivo es encontrar la partición óptima para predecir,

es decir la que minimice el efecto.

Es claro que un modelo de participaciones generales

es difícil de encontrar algorítmicamente, dado que el problema de utilización

combinatorio resultante para determinar las mejores particiones es muy complejo.

No es factible de resolver y eso para tamaños de muestras

relativamente pequeños.

Notemos por ejemplo que la flexibilidad del modelo

depende del número de particiones, pero también de la forma en la cual de hace.

Sin embargo existen simplificaciones que pueden hacer este tipo de algoritmos

fáciles de estimar o de calibrar.

Por ejemplo se pueden restringir las particiones a que sean lineales

o reemplazar las optimizaciones altas por elípticas.

Dentro de estos algoritmos de particiones,

quizás los más famosos son los árboles de decisión.

Estos modelos asumen particiones lineales con estas simplificaciones.

Primero el algoritmos es secuencial o un método Bride para utilizarlo.

En el cual una vez se hace una partición entonces no se puede reversar.

Esto es equivalente a un método Forward en el modelo lineales.

Segundo.

Las particiones además de ser lineales,

son paralelas a las coordenadas definidas por cada variable, en otras palabras

una partición solo se puede generar en un valor particular de una variable.

Pongamos un ejemplo concreto.

Supongamos que se quiere predecir si un estudiante aprueba o no un curso de

maestría en función de las horas de estudio de la semana X1

y el promedio acumulado de sus clases anteriores X2.

De acuerdo a las dos simplificaciones expuestas,

las particiones paralelas a cada variable implican por ejemplo que se parte en dos,

cuando las horas de estudio son mayores a siete y menores a siete.

En la parte de la izquierda se predice que no,

mientras que en la derecha se predice que sí.

Por supuesto la partición que se hace debe ser la que minimice de alguna forma

alguna métrica del error o que maximiza el AUC.

Cuando se hace la segunda partición dado que son secuenciales,

entonces se puede partir cuando el promedio de clases anteriores es decir X2

es mayor o menor a cuatro.

Esto crea cuatro partes.

Supongamos que en estas tres,

la predicción es que sí aprueba y en esta que no aprueba.

Las dos particiones que se mostraron en el ejemplo corresponden a un modelo aditivo,

es decir se podría representar como una función en X1 más una función en X2.

Sin embargo en los modelos de particiones esto es un poco restrictivo.

Por ejemplo supongamos que las observaciones asociadas al ejemplo

son estas, donde en rojo están los puntos con Y igual a no aprueba y en azul

con Y igual a sí aprueba.

Esta partición parece sobrar, pero en el modelo aditivo no es posible de remover.

Para ganar flexibilidad sería mejor si solamente se parte este pedazo,

esto es justamente lo que hacen los árboles de decisión.

Las particiones se hacen de manera secuencial, de forma que

en una iteración una de las partes you formada se parte nuevamente en dos.

Este proceso de partición iterativa en el espacio de X se ve así.

Se puede asociar con una representación gráfica en forma de árbol

o de manera más exacta de árbol invertido, de ahí el nombre de estos modelos.

En lo más alto del árbol se ubica la primera partición,

hacia la izquierda vamos con una de las partes y hacia la derecha con la otra.

Esto se puede escribir como una condición en la variable

en la que se genera la partición.

A la izquierda se genera la rama para la cual la condición es cierta

y a la derecha cuando es falsa.

Cada rama representa cada una de las partes.

En la siguiente partición entonces una de las ramas se bifurca.

Así sucesivamente se va generando la estructura de árbol.

Esta representación en árboles, hacen que el modelo sea relativamente interpretable.

Dado que se puede entender el modelo y el efecto de las variables sobre la

predicción a manera de reglas lógicas.

you sabemos cómo los árboles de decisión simplifican el proceso de particiones,

creando estructuras lógicas interpretables.

Ahora miremos cómo se seleccionan las particiones óptimas en cada iteración.

Primero, entre dos puntos consecutivos de una variable,

no hay diferencia al punto exacto de la partición.

Esto no cambia el desempeño, quiere decir que por cada variable,

tenemos N menos un posibilidades para partir.

En la primera iteración entonces,

se deben revisar N menos 1 por P particiones posibles.

En el caso de regresión es sencillo escoger la primera partición.

Dado que se progrese con un promedio,

entonces se busca es minimizar las sumas de cuadrado del erro o SSE.

Antes de hacer la partición, la función objetivo es, SSE es 0,

es igual a la suma de los Y sub I menos Y barra al cuadrado.

El super índice cero es para representar la primera iteración

o partición del algoritmo en tiempo cero.

Después de partir el SSE queda como SSE1

es igual a la sumatoria de los Y sub I menos

Y barra 1 al cuadrado más la sumatoria de los Y sub I mos Y barra 2 al cuadrado.

Que es el SSE de la primera parte más el SSE1 de la segunda parte.

La relación entre el SSE0 y el SSE1 es que SSE0 es

igual a SSE1 más N1 por Y barra 1 menos,

Y barra al cuadrado más N2 por Y barra 2 menos Y barra al cuadrado.

De esa forma se hace la partición en el valor de la variable para la cual el delta

del SSE es igual a N1 por Y barra 1 menos Y barra al cuadrado

más N2 por Y barra 2 menos Y barra al cuadrado.

Lo que buscamos es que este delta sea lo máximo posible.

Dado que Y barra 2 se puede calcular en función de Y barra y Y barra 1,

entonces solamente se requiere calcular los promedios de los puntos

que se van acumulando como se muestra en la gráfica.

Esto se puede calcular para cada una de las N menos 1 por P posibilidades.

De las cuales se escoja la que tengo el mayor valor de delta del eje SSE.

El procedimiento es actualizado eficientemente

cuando se realiza la siguiente partición.

Dado que la misma descomposición se cumple para cada una de las partes.

Por ejemplo si se parte en dos la segunda parte anterior,

entonces el SSE2 es igual al SSE1 más un

SSE2 de la segunda parte más un SSE3 de una tercera parte.

Con el método secuencial entonces el modelo se calibra

con el número de iteraciones o terminaciones finales del árbol.

Entre más nodos finales, más flexible el modelo.

Para el caso de clasificación,

el procedimiento es muy parecido, sin embargo debemos considerar que el error de

predicción no es la métrica indicada para realizar las particiones secuenciales.

La razón de esto es que no es estrictamente consona

en función de la probabilidad.

Lo que hace que en algunas iteraciones se tengan varias particiones posibles.

La solución para esto es usar otro tipo de métricas que se comportan

como el SSE en términos y hacer la separación en un note.

Dentro de las métricas más usadas están el Gime Index y la función de Entropía

cruzada.

Ambas son medidas de pureza, por ejemplo el Gime Index para una de las partes se

mide con Gime Index es igual a P gorro por 1 menos P gorro, donde P gorro

es la probabilidad estimada de una de las clases dentro de las partes en cuestión.

Notemos que el Gime Index es una medida de pureza o mejor de impureza,

esto es entre mas cercano sea P gorro a 0 o a 1,

entonces la clase es más homogénea y por lo tanto las observaciones que contienen

tienen mejor predicción o menos error.

Cuando miramos el Gime Index como función del P gorro, entonces tiene esta forma.

Por otro lado si miramos el error de predicción tiene esta forma.

Esta diferencia entre las dos curvas es lo que permite que el Gime Index

no permita empates entre los candidatos a las particiones secuenciales.

Otras métricas como la Entropía Cruzada tienen el mismo efecto.

Algunas de las características a favor de los árboles de decisión son.

la primera es que contrario a otros modelos no lineales,

los predictores no informativos no afectan considerablemente el desempeño,

debidos a que no son tenidos en cuenta al momento de hacer particiones.

En este sentido son algoritmos robustos a la maldición de la dimensionalidad.

La segunda es que los árboles no se afectan con datos faltantes.

Porque tienen la capacidad de instruir observaciones para las cuales

algunas variables no se encuentran registradas.

Y la tercera es que al influir predictores categóricos es más eficiente hacer

particiones entre las clases que adicionar variables binarias dumes, sin embargo esta

ventaja se aminora si los predictores categóricos tiene muchas clases.

Aunque los árboles de decisión parecen ser muy buenos predictores,

tienen una característica que hace que su desempeño no sea el mejor.

Esto es especialmente cierto si la mayoría de predictores son variables continuas.

Esta parte negativa tiene que ver justamente con el método

secuencial de encontrar las particiones.

Lo que pasa es que este proceso secuencial aumenta la varianza del estimador F gorro.

Dado que la decisión de donde hacer las particiones

es muy suceptible a la variabilidad de los datos

y una vez realizado una partición no se puede relanzar.

Veámoslo en una caso de regresión con un caso de predictor continuo.

Supongamos que tenemos estas observaciones de Y y de X,

al hacer la primera partición, parece claro que debe estar acá,

dado a que se minimizan las distancias de los promedios a cada parte.

Sin embargo supongamos que se cambian algunos puntos de la muestra,

en este caso la partición puede estar en un valor completamente diferente.

Puede que la función lineal estimada puede ser parecida,

pero el método secuencial you sea particiones

parece que incrementa la variabilidad del algoritmo, esto no quiere decir que en

algunas ocasiones los algoritmos basados en árboles de decisión predigan bien,

pero en general tienen una variabilidad adicional por su método de estimación.

En este video estudiamos los métodos generales para predecir basados en

árboles de decisión o particiones secuenciales del espacio de predictores.

Existen diferentes implementaciones con variaciones alrededor del mismo concepto.

Lo árboles resultan métodos robustos, interpretables,

eficientes con una variabilidad un poco inflada que previene

un mejor desempeño en algunas aplicaciones.

En el próximo video estudiaremos algunas estrategias de ensamblaje de

algoritmos como los árboles de decisión, para obtener mejor desempeño predictivo.

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA] [MÚSICA]

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/joIyY/ensamblaje-de-algoritmos-en-el-modelo-predictivo

# Ensamblaje de algoritmos en el modelo predictivo

Hola, en este video

estudiaremos los algoritmos predictivos basados en estrategias de ensamblaje.

Esto es, combinar varios algoritmos para crear uno con mejor desempeño predictivo.

Los algoritmos de ensamblaje o meta algoritmos

toman varios modelos cuyo desempeño posiblemente no sea óptimo,

como los árboles de decisión,

y los combina con alguna estrategia que hace que se

corrijan los vicios de cada predictor individualmente.

Generalmente, las estrategias

para ensamblar algoritmos están pensadas para disminuir el sesgo o la varianza.

Existen diversas técnicas para ensamblar

: una de ellas está enfocada en la disminución de la varianza,

las cuales llamaremos como técnicas de diversificación.

Por otro lado, otros están más enfocadas a disminuir el sesgo,

a los cuales llamaremos técnicas de aprendizaje lento.

Existen otras técnicas un poco más especulativas

que promedian o combinan algoritmos muy buenos, ya calibrados.

Esas técnicas se conocen como apilamiento o stacking.

Empecemos con las técnicas de diversificación.

Su principio fundamental es que si los algoritmos individuales f gorro 1,

f gorro 2 hasta f gorro b tienen

varianza porque la forma funcional es un poco difícil de controlar,

entonces la estimación se vería beneficiada por no usar

las mismas observaciones para cada uno de ellos de manera individual.

Miremos un ejemplo de regresión con árboles de decisión.

Supongamos que la función f óptima,

es decir, el valor esperado de x tiene esta forma.

Ahora supongamos que tenemos una muestra relativamente pequeña de unos 20 datos.

Usando unas 3 iteraciones o 4 partes

al estimar la función dará algo como esta función constante a trozos de color rojo.

Esta función no parece tener tanta varianza,

pero recordemos que los árboles tienen la variabilidad añadida del método secuenciar.

Supongamos por un momento que tenemos la oportunidad de repetir la muestra.

Esto es, tenemos la posibilidad de tomar otros

20 datos de manera independiente a los de la primera muestra.

Se podría ver algo como esto.

Es muy probable que la primera partición sea en otro sitio muy diferente.

En general, las partes resultantes se parecen, pero no son iguales.

Esta es la variabilidad natural del estimador,

más la que se introduce por el orden en que se hacen las particiones.

Si pudiéramos tomar todavía más muestras,

entonces los estimadores serían diferentes.

En este caso, una muy buena idea sería

promediar todos los estimadores obtenidos para cada muestra.

El resultado se puede ver algo como esto.

Por supuesto, este estimador es mejor porque se acerca mucho más al f óptimo.

Claro, lo que acabamos de describir no es posible de realizar,

porque solo tenemos a nuestra disposición una muestra.

Si partimos, la muestra tampoco sería buena idea

porque el modelo tendría menos datos para estimar.

La idea entonces de la diversificación

es emular el proceso de conseguir diferentes muestras,

de forma tal que el algoritmo trabaje con datos diferentes cada vez.

Como no queremos disminuir el tamaño de muestra,

una estrategia interesante es producir

muestreos con reemplazo de las n observaciones de la muestra.

Este proceso se conoce como "Bootstrapping".

Lo que hace es producir B muestras de tamaño n cada uno,

donde n es el número de datos disponibles para entrenamiento.

Cada muestra está producida por un muestreo con reemplazo.

Pongámoslo así: la muestra original tiene los datos indexados con igual i igual a 1,

2 hasta n. Miremos la primera muestra de "Bootstrap".

La primera posición corresponde a un índice de 1 a n seleccionado aleatoriamente.

Para la segunda se repite el proceso,

un índice escogido aleatoriamente.

Notemos que las muestras difieren entre ellas,

debido a que algunos de los índices quedan repetidos en cada muestra.

Si bien la información original es la misma

y cada muestra no tiene información realmente nueva e independiente,

este proceso puede crear suficiente diversidad en los modelos estimados en cada muestra,

como para que el promedio de ellos disminuya la varianza.

Cuando se hace "Bootstrap" y para cada muestra se estima un

modelo f gorro sub j para j igual 1,

2 hasta B, entonces la

estrategia de promediar todos los modelos se conoce como

"Bagging" por "Bootstrap aggregation".

En este caso f gorro bagging es igual a uno sobre B por la suma desde j igual 1 hasta

B de los f gorro sub j. Aunque pareciera

que el parámetro B es un parámetro de calibración,

debemos considerar que para valores de B muy grandes,

el algoritmo de Baggins se satura.

Dado que las muestras ya no producen diversificación.

En términos de B,

EPE estimado baja hasta alcanzar un llano.

Por lo general, es más recomendable dejar que el algoritmo se

sature con respecto a B y calibrar los estimadores individuales f gorro sub j.

Notemos que el proceso de Bagging funciona mejor con algunos tipos

de modelos para los estimadores individuales de f gorro sub j que para otros.

En general funciona muy bien con árboles de

decisión o algoritmos de particiones genéricos.

La razón de esto es que el promedio,

después de crear diversidad,

desaparece casi por completo la varianza adicional creada a los mecanismos de estimación.

Además, mejora considerablemente la forma funcional

estimada para suavizar los saltos provocados por las particiones.

Por lo general es usado con árboles

de decisión con un número de nodos que puede estar entre dos y cinco.

Por el contrario, métodos como regresión

lineal tendrían muy poca mejora al hacer Bagging.

Miremos por ejemplo, al crear los diferentes muestras que obtienen planos parecidos.

Al promediar es posible que disminuya un poco la varianza,

pero la forma funcional es muy parecida.

Una adaptación especial del método de Baggins se

conoce como "Random Forest" o bosque aleatorio.

La idea es usar el mismo procedimiento de Bootstrap,

con la diferencia de que en cada una de las iteraciones

en f gorro sub j no se usan todos los predictores x\_1, x\_2, x\_p,

sino que se usa un subconjunto de ellos seleccionado de manera aleatoria.

Esto con el ánimo de crear todavía mayor

diversidad que pueda disminuir la varianza al promediar.

En este método aparece un parámetro adicional que se puede calibrar,

que es m o el número de predictores a usar en cada iteración.

Por supuesto, m puede tomar valores desde 1 hasta

p. Algunos expertos prefieren no usar ese parámetro para calibrar,

dado que la curva l p estimado no siempre da convexa,

y en su lugar escogen un valor de m cercano a la raíz cuadrada de p. En general,

por simple que parezca,

de todos los modelos predictivos estudiados hasta ahora,

Random Forest es uno de los que mejor desempeño predictivo tiene,

tanto en regresión como en clasificación.

La otra estrategia, aparte de la diversificación,

es lo que llamaremos aprendizaje lento y su algoritmo más representativo es "Boosting".

La idea es también relativamente sencilla.

Lo ilustraremos en un problema de regresión.

Supongamos que al estimar la función predictora f gorro en estos puntos,

dejamos el modelo sesgado a propósito.

Es decir, la función estimada es muy suave.

Este puede ser el primero de los algoritmos individuales f gorro 1.

Como la función tiene sesgo,

entonces calculemos los residuos o errores de esta predicción

para cada uno de los puntos como r i es igual a y sub i,

menos el f gorro 1 de x sub i. Ahora miremos

estos errores en función de x. Como f gorro 1 tenía sesgo en los residuos,

todavía hay cierta información que puede ser

explicada con x. Para recoger esa información,

estimamos una función usando como variable de salida los errores.

A esta función la llamamos g gorro 2.

Notemos que si esta función se la sumamos a f gorro 1,

entonces se reduce el sesgo.

Sin embargo, para dejar lugar a nuevas iteraciones no la sumamos completa,

sino con un efecto reducido multiplicando por una tasa de aprendizaje Lambda.

De esta forma, la segunda iteración del método Boosting

se calcula como f gorro 2 es igual a f gorro 1 más Lambda por g gorro 2,

y se siguen obteniendo los residuos y sumando las

funciones de forma tal que f gorro k es igual

a f gorro de k menos 1 más

Lambda por g gorro k hasta lograr un número máximo de iteraciones B,

es decir, que k va desde 1 hasta B. Con la tasa de

aprendizaje en Lambda forzamos a que el algoritmo siga manteniendo un poco de sesgo,

el cual va disminuyendo en cada iteración.

La idea de ir ingresando información de manera lenta es permitir que el modelo explore

zonas del espacio de entrada de x que se hubieran podido ignorar con

un modelo que se llama la función de una sola vez.

El algoritmo debe ser calibrado con respecto a B y con respecto a Lambda.

Notemos que si B aumenta,

entonces el algoritmo es cada vez más flexible.

Como última estrategia para hacer modelos de ensamblaje,

hablaremos de "Stacking" o métodos de apilamiento.

La idea fundamental es tomar algoritmos base f gorro 1,

f gorro 2 hasta f gorro B,

los cuales están correctamente calibrados cada uno de ellos.

Por ejemplo, uno de ellos puede ser un modelo GAM,

otro un árbol de decisión y otro modelo de regresión lineal.

Hacer Stacking es promediar estos modelos o también

combinarlos con pasos diferentes de forma tal que los pesos sumen 1.

No se busca diversidad en las observaciones.

Si cada uno de los predictores individuales es muy bueno y está bien calibrado,

entonces pueden predecir de manera muy similar,

por lo que el promedio no debería mejorar notoriamente el desempeño.

En este sentido,

Stacking es un método especulativo en el cual a veces se obtienen mejores resultados.

Cuando funciona es generalmente porque entre los algoritmos se

corrigen algunos vicios individuales o formas funcionales del proceso.

Por ejemplo, si combinamos un método regresión con splines con un método MARS,

en donde en estas partes se puede ver que se mejora la forma funcional.

En este vídeo estudiamos las estrategias para ensamblar algoritmos predictivos.

Vimos el método de Bagging con Random Forest como un caso especial.

Además, Boosting como el ejemplo genérico de algoritmos de aprendizaje lento.

En el próximo vídeo,

estudiaremos cómo gestionar los datos faltantes

y comprenderemos los efectos de las muestras no balanceadas

en los problemas de clasificación.

# https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/P7MN8/gestion-de-datos-faltantes-en-la-implementacion-de-modelos-predictivos

# Gestión de datos faltantes en la implementación de modelos predictivos

[MÚSICA]

[MÚSICA]Hola.

En este video miraremos el fenómeno de los datos faltantes y qué hacer cuando

estos se presentan en nuestra base de datos.

Empecemos definiendo qué son los datos faltantes en las bases de datos que

utilizamos para construir nuestros modelos.

Estos corresponden a observaciones no registradas

de algunas variables para algunos individuos.

Por ejemplo, en el caso de una matriz de datos estructurada donde las filas son

los individuos observados y las columnas son las variables correspondientes,

entonces el problema de datos faltantes corresponde al hecho de que el valor de

algunas celdas no se encuentra disponible.

En la mayoría de aplicaciones los datos faltantes son una realidad inevitable,

y por supuesto, una incomodidad para el analista de datos,

dado que implica al menos una de dos consecuencias.

La primera es la posible pérdida de información o tamaño de muestra reducido

lo cual afecta el desempeño de los modelos.

La segunda es el trabajo adicional para la inspección y el preprocesamiento de datos.

Antes de concluir sobre qué hacer en caso que tengamos datos faltantes

debemos entender el efecto que pueden tener ciertas estrategias

para corregir el fenómeno.

Supongamos para simplificar la anotación

que se quiere hacer un modelo de regresión lineal de Y contra X1, X2 hasta Xp.

Posiblemente, si tenemos datos faltantes en Y

no es mucho lo que podemos hacer si queremos ajustar un modelo predictivo.

Sin embargo, ¿qué pasa si en la matriz de los datos X existen datos faltantes?

Pensemos esto.

Si para un individuo o fila en la matriz X hay una celda vacía

y P menos 1 celdas con observaciones, el modelo no puede usar esta observación.

No tenemos cómo correr el modelo.

Una idea obvia es correr el modelo con n menos 1 datos,

sacando la fila correspondiente al dato faltante.

Es claro que este proceso puede ser ineficiente.

Desperdiciamos P menos 1 datos por uno que no conocíamos.

Al estimar modelos predictivos con menos datos aumenta la varianza.

Si el tamaño de muestra es muy grande posiblemente esto no sea un gran problema.

La pregunta que queda un poco en el aire es si el problema es solamente la pérdida

de una parte de la muestra o hay algo más de fondo que pueda crear problemas en

nuestros modelos.

La respuesta depende de la forma en la cual algunas celdas de la matriz faltan.

Pongamos un ejemplo concreto, supongamos que tenemos que predecir si una persona

sufrirá de insuficiencia cardiaca en los próximos

10 años a partir de tres variables predictoras, X1, el peso en kilogramos,

X2, las horas de actividad física que hace en promedio a la semana,

y X3 La ingesta promedio de calorías diarias.

La matriz de datos contiene cuatro variables incluyendo la respuesta

que podemos codificar como uno o cero.

Existen varias formas un poco idealizadas en las cuales se producen los

datos faltantes.

La primera de ellas es la forma completamente aleatoria.

o MCAR por Missing completely at Random.

En este caso las celdas de la matriz que tienen datos faltantes presentan un

comportamiento completamente aleatorio, es decir, se seleccionan celdas al azar

independiente de la fila, la columna, o el valor de la variable, en este caso,

al remover las filas con celdas faltantes solamente se presenta pérdida de datos,

es decir, aumento en la varianza por una muestra más pequeña.

Una posible opción cuando el mecanismo de datos faltantes es MCAR,

es imputar o reemplazar los datos faltantes de las celdas

con los promedios de las variables correspondientes.

Sin embargo, existen otras formas de producir datos faltantes.

Por ejemplo, es posible que la probabilidad de que una variable sea

faltante pueda depender del valor de las otras variables.

Esto pasaría si por ejemplo, las personas prefieren omitir

la información sobre la ingesta de calorías diarias

si sienten que no corresponden a su peso o a la cantidad de ejercicio que hacen.

En este caso el patrón de ocurrencia se conoce como aleatoria

o MAR por Missing at Random.

Algo interesante en el modelo MAR es que al sacar las filas con las celdas

faltantes ocurren otras cosas además de la disminución obvia de la muestra.

Las otras variables tendrán faltantes que corresponden a ciertos rangos

o valores particulares.

Por ejemplo, al remover las filas con faltantes en la variable ingesta de

calorías podemos estar removiendo a las personas que hacen poco ejercicio.

Esto puede crear sesgo en los modelos o regiones en el espacio X

en las cuales no tenemos información.

La buena noticia es que cuando el mecanismo es MAR,

es el más fácil de imputar.

Esto se debe a que si los datos faltantes dependen del valor de las demás variables,

entonces, es posible realizar algún modelo predictivo de esta variable,

con datos faltantes con respecto a las demás.

Así se podrá conservar la mayor parte de la muestra.

El último mecanismo se conoce como MNAR, por Missing Not at Random.

En este caso el valor faltante de una variable

depende del valor particular de la misma variable.

Por ejemplo, esto pasa cuando las personas no responden la pregunta

sobre la cantidad de ejercicio que realizan porque sienten que es muy poco.

Cuando esta variable tiene un valor muy bajo entonces la probabilidad de

que falte el dato es mucho mas alta.

Remover las filas con datos faltantes tiene un efecto parecido al mecanismo MAR.

Puede crear sesgo y zonas no representadas.

Sin embargo, el mecanismo MNAR, tiene un grave problema, no es imputable.

De hecho,

es muy difícil reconocer que un conjunto de datos tiene faltantes de este tipo.

En realidad estos mecanismos rara vez son puros, por lo general cuando hay datos

faltantes se presentan por una combinación de las tres formas, MCAR, MAR y MNAR.

Lo importante es que al tomar decisiones sobre qué hacer con los datos faltantes

se tengan presente las consecuencias,

por ejemplo, si se decide remover las filas el problema no es solo perder la

información sino crear posibles sesgos en nuestros modelos.

Es importante entonces realizar análisis descriptivos

sobre el comportamiento de los datos faltantes

para determinar la solución al problema sin crear consecuencias peores.

Por ejemplo, en el caso de variables continuas se pueden hacer gráficas de

dispersión de puntos por pares de variables en donde para los datos

faltantes se puede registrar las variables observadas en el eje correspondiente.

En este caso es posible determinar si el mecanismo es MAR

cuando los faltantes dependen del valor de la otra variable.

En esta gráfica por ejemplo,

hay faltantes cuando la otra variable toma valores pequeños.

Las opciones que tenemos con los datos faltantes puede ser entonces resumidas

como, primero remover columnas o variables completas.

Segundo remover filas o individuos para los cuales hay algunas celdas faltantes.

La tercera es la imputación,

la cual la podemos hacer de varias formas, una forma es la imputación con la media de

la variable o la moda en el caso de las variables categóricas.

La segunda,

es la imputación con modelos de regresión en términos de otras variables.

Y la tercera, es la imputación múltiple en donde se generan imputaciones con varios

valores de manera iterativa para aprovechar toda la información.

En este video estudiamos el fenómeno de los datos faltantes y comprendimos que

dependiendo del mecanismo con el cual creemos que se generan, entonces las

consecuencias de un tipo de corrección pueden ser nocivas para el modelo.

Debemos considerar que para tamaños de muestra pequeño o mediano,

remover las observaciones con celdas faltantes puede crear sesgo.

Imputar puede ser una alternativa más recomendable.

En el próximo vídeo estudiaremos el fenómeno de las clases desbalanceadas en

los problemas de clasificación.

https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/BID9u/gestion-de-clases-desbalanceadas-en-la-implementacion-de-modelos-predictivos-de

# Gestión de clases desbalanceadas en la implementación de modelos predictivos de clasificación

[MÚSICA]

[MÚSICA] Hola.

En este video, estudiaremos un fenómeno típico de los modelos de clasificación de

los cuales una de las clases tiene muchos menos datos que la otra.

Esto se conoce como imbalanceo de clases.

El imbalanceo de clases se genera si una de las categorías de la variable de

respuesta en un problema de clasificación

tiene una proporción de los datos mucho menor a la otra.

Es un problema muy común en la práctica en donde es natural

que una clase quede menos representada en los datos observacionales.

Por ejemplo,

en el reconocimiento de fraude en compras electrónicas no presenciales es de esperar

que la mayoría de las observaciones de transacciones sean no fraudulentas.

O por ejemplo, cuando se estima la probabilidad de tener una enfermedad rara

a partir de los síntomas clínicos.

Los casos reales de enfermedad serán muy bajos proporcionalmente.

Estas proporciones de las clases también se conocen como probabilidades apriori.

No es raro encontrar casos en los que una clase tiene menos del 10% de los datos,

es decir, una probabilidad apriori menor a 0.1.

Primero debemos comprender qué implica ajustar un modelo predictivo cuando las

clases tienen este tipo de desproporciones y después se puede decidir si un

caso determinado requiere o no medidas correctivas.

[MÚSICA] [MÚSICA]

Algo que you habíamos visto cuando las clases tienen desbalanceo

es que el error de proyección no resulta una métrica adecuada para evaluar los

modelos predictivos porque al predecir siempre en la clase mayoritaria se podrían

obtener menos errores que entrenando cualquier modelo de clasificación.

Ahora bien, predecir siempre la clase mayoritaria,

si el problema es muy desbalanceado, es realmente incorrecto.

Pensemos bien la situación.

Cuando hacemos un modelo de machine learning para clasificación

se estima de alguna forma una función discriminante

que bien podríamos asumir como la probabilidad de que y sea igual a 1.

Ahora, supongamos que la clase 1 es minoritaria, con tan pocas observaciones,

que en ninguno de los puntos del espacio x, la probabilidad

de que y sea igual a uno, es mayor a la probabilidad de que y sea igual a 0.

Miremos gráficamente las curvas de las probabilidades estimadas

para 1 y 0 respectivamente.

En este caso, el algoritmo nunca predice la clase 1.

Este fenómeno lo podemos estudiar desde dos puntos de vista.

Por un lado, el algoritmo está haciendo lo que se supone que debe hacer, predecir la

clase para la cual la probabilidad de cometer errores es lo más pequeña posible.

Si en la aplicación del algoritmo implicara

que se recibe un dólar cada vez que se predice bien, se pierde un dólar cada vez

que se comete un error de predicción, entonces este procedimiento es óptimo

en términos de la utilidad general a largo plazo en valor esperado.

Es decir, predecir siempre 0 es óptimo si ponderamos de igual manera

los dos tipos de error, 1 siendo predecir 1 cuando y es 0,

y el otro siendo predecir 0 cuando y es igual a 1.

Esto es interesante.

No hay nada malo con el resultado del algoritmo.

Predecir siempre la clase mayoritaria puede ser lo adecuado.

[MÚSICA] [MÚSICA] Sin embargo, pensemos que y igual a 1,

cuando una transacción electrónica es fraude, y 0 cuando no lo es.

Si siempre predecimos que la transacción no es fraude,

entonces pareciera que el algoritmo no hace lo que queremos que haga,

que es identificar los casos de fraude.

Supongamos que en el espacio x,

el valor máximo que toma la probabilidad de que y sea un fraude, es 0.4.

De esta forma, nunca se predice fraude.

Sin embargo, en una muestra de test es muy probable que existan algunos casos de

fraude los cuales no serían identificados.

Si sentimos que esto no está bien,

entonces lo que pasa es que el costo que le damos a los errores es diferente.

Es decir, para nosotros es peor no predecir un caso de fraude,

que predecir fraude cuando en realidad no lo es.

Es muy posible que el contexto real o el problema de negocio hace

que estos errores sean asimétricos.

La decisión sobre si tomar medidas correctivas al momento de predecir

clases desbalanceadas o dejar el problema óptimo para errores simétricos,

depende entonces de la aplicación que se esté realizando.

Esta es una decisión del analista,

no un resultado de un modelo de machine learning.

[MÚSICA] [MÚSICA] Ahora bien, ¿qué pasa si queremos modificar el modelo

para que sea capaz de predecir los fraudes?

Esto implica aceptar que un tipo de error vale más que el otro.

Si es así, entonces la solución es fácil y you la conocemos.

Es lo que entendimos cuand estudiamos la métrica del AUC.

Como la mayoría de algoritmos estiman una función discriminante,

como por ejemplo la probabilidad, entonces lo que hacemos es cambiar el Threshold.

En lugar de predecir fraude cuando la probabilidad estimada sea mayor a 0.5,

podemos predecirlo cuando sea mayor a 0.3 por ejemplo.

Cuando evaluamos el desempeño de este modelo, entonces el error total sube,

pero la sensibilidad aumenta drásticamente mientras que la sensitividad baja un poco.

Esta es la razón de usar el AUC para comparar modelos de clasificación.

El modelo de machine learning estima de manera óptima la probabilidad y el usuario

puede decidir cómo usarla, es decir, qué Threshold usar.

[MÚSICA] [MÚSICA] Alternativamente, al cambiar el Threshold

existen otras formas para solucionar el problema de costo de errores asimétricos.

Es posible seguir usando el límite de probabilidad de 0.5,

pero alterando las condiciones de la muestra.

Esto es modificando las probabilidades apriori de los datos.

Por ejemplo, existen métodos que se conocen como Up sampling o Down sampling,

cuando respectivamente se aumentan los datos de la clase minoritaria

o se disminuyen los de la mayoritaria.

Para aumentar la clase mayoritaria,

se pueden usar mecanismos de simulación para crear aleatoriedad en los datos.

Las dos versiones de aumentar y disminuir se pueden usar en métodos híbridos

que hacen las dos cosas al mismo tiempo.

Dentro de estos métodos por ejemplo se destaca el método SMOTE,

por Synthetic Minority Oversampling TEchnique.

Además de esto, otra alternativa es usar pesos para las observaciones.

Los datos de la clase minoritaria tienen más peso que los de la mayoritaria.

El efecto es el mismo que el de cambiar las probabilidades apriori.

Otra forma equivalente es usar costos para los errores en la función objetivo del

modelo que se optimiza para estimar la función discriminante

también con un efecto similar.

Estas últimas dos formas, sin embargo,

solo pueden ser usadas en algoritmos que permitan incorporar estas condiciones,

pesos en las observaciones, o costos desiguales.

Por ejemplo, en support electric machines es sencillo cambiar el algoritmo de

estimación para incorporar costos asimétricos en los errores.

Por otro lado, los métodos basados en árboles y sus ensamblajes,

pueden incorporar fácilmente los pesos en las observaciones.

[MÚSICA] [MÚSICA] Pensemos esto.

Si cambiar el Threshold tiene los mismos efectos prácticos que cambiar

artificialmente las probabilidades apriori, sea con rebalanceo,

costos o pesos.

Entonces, es mucho más fácil simplemente cambiar el Threshold de clasificación.

Existe alguna razón para usar estos otros métodos y alteración de las probabilidades

apriori, la respuesta es posiblemente sí.

Resulta que estamos asumiendo que las probabilidades estimadas o funciones

discriminantes están bien estimadas cuando existe un desbalanceo de clases notoria.

Sin embargo, lastimosamente esto no siempre es cierto.

Algunos algoritmos tendrán dificultades para estimar

fielmente la probabilidad de fraude en función de x.

Por ejemplo, un árbol de decisión tendría problemas encontrando

particiones usando el gini index, porque la clase

mayoritaria siempre tendría homogeneidad en todas las regiones.

Algunos algoritmos serán más robustos que otros, pero en general, para todos será

difícil estimar esta probabilidad con pocas observaciones que la representen.

En este caso, sí es recomendable usar métodos que cambien las

probabilidades apriori, o las proporciones iniciales de los datos.

En este video estudiamos el fenómeno de las clases desbalanceadas en los

problemas de clasificación binarios.

Entendimos que existen problemas y consideramos que es importante que la

clase con menos observaciones sea más reconocida por los modelos.

Para solucionar el problema, podemos cambiar el Threshold de clasificación

o modificar artificialmente las probabilidades apriori.

Estos últimos métodos son más recomendables para estimar correctamente

las funciones discriminantes.

En el próximo video, veremos una introducción a las redes neuronales.

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

[MÚSICA]

<https://www.coursera.org/learn/introduccion-al-machine-learning/lecture/foQln/introduccion-a-las-redes-neuronales-y-el-aprendizaje-profundo>

# Introducción a las redes neuronales y el aprendizaje profundo

Hola, en este vídeo explicaremos los modelos de

redes neuronales y veremos una breve

introducción al aprendizaje profundo o "deep learning".

Las redes neuronales son una clase de modelos

predictivos y muy versátiles que se adaptan muy

bien a las entradas o inputs y salidas o

outputs que tienen cierta relación espacial o secuencial.

Por ejemplo, cuando los input son imágenes,

cada pixel tiene una relación espacial con los demás que se debe mantener.

Eso crea una topología que puede ser explotadas por los modelos basados en red.

Otro ejemplo es cuando los datos son secuenciales,

por ejemplo, los textos o conjuntos de palabras con un sentido determinado.

El orden y la separación entre los conjuntos de palabras o

letras hacen que la secuencia se deba mantener.

Algunos tipos de modelos de redes neuronales pueden, por ejemplo,

completar la secuencia y predecir el resto del texto que debe ir en un documento.

Las redes neuronales han existido como modelos predictivos a

partir de datos desde la década de 1970 - 1980.

Al principio ganaron mucha popularidad y se convirtieron en la referencia

a los modelos no interpretables que estaban enfocados netamente a la predicción,

en contraste con muchos modelos estadísticos.

A medida que aparecían muchos más modelos como Support Vector Machines o Random Forest,

quedó simplemente como una opción más que a veces era más difícil de implementar.

Sin embargo, durante la última década, desde 2010,

las redes neuronales han resurgido como modelos muy competitivos

bajo el concepto de aprendizaje profundo,

que es simplemente la utilización de redes neuronales con gran número de capas,

como veremos más adelante.

Esto ocurrió de la mano de nuevos algoritmos más eficientes y estructuras de redes

mucho más especializadas para los datos complejos como imágenes,

videos, documentos, etcétera.

Cuando el tamaño de muestra es muy grande y los datos de entrada tienen

características especiales que le dan cierto orden o topología,

las redes neuronales,

en especial el aprendizaje profundo,

suelen ofrecer mejores niveles de desempeño predictivo.

Esto ha popularizado su investigación durante los últimos años.

No podremos comprender todos los detalles del aprendizaje profundo en este video,

pero veremos los conceptos fundamentales de las redes

pre alimentadas o feed-forward networks y en la

formulación para otro tipo de redes con estructuras

diferentes como redes convolucionales y redes secuenciales.

Empecemos por describir las redes neuronales originales que se conocen

como redes neuronales pre alimentadas o feed-forward neural networks.

La estructura es relativamente sencilla y lo ingenioso quizás es el método de estimación.

Pongamos un caso muy sencillo: una unidad con

un solo nodo oculto que se conoce como perceptrón.

Supongamos que tenemos dos inputs x\_1 y x\_2.

La variable de respuesta puede ser categórica binaria,

es decir, un problema de clasificación.

Representemos gráficamente en términos de nodos cada una de estas variables.

Ahora pongamos un nodo oculto que podemos llamar

z. Este no lo podemos calcular como una función lineal de x\_1 y x\_2.

Es decir, z igual Beta\_0 más Beta\_1 por x\_1 más Beta\_2 por x\_2,

donde los parámetros Beta son desconocidos,

tal como en un modelo de regresión lineal.

Ahora z está conectado con la variable y o con una función discriminante para predecir.

Esta función puede ser f de z igual a exponente de z sobre uno más el exponente de z,

que corresponde a la función sigmoide o al inverso de la función logic.

F de z se puede entender como una probabilidad estimada,

por lo que si f de z es mayor a un medio se clasifica como uno.

Veremos que este modelo corresponde justamente a la formulación de regresión logística,

aunque expresado como una red en la que se unen los nodos o variables.

Las redes neuronales se adaptan de igual manera los números de regresión.

En este caso, la función f de z se

puede multiplicar por una constante para disminuir el error.

La generalización de estas redes es no tener un solo nodo oculto, sino varios.

En este caso se tienen los inputs x\_1 y x\_2,

los cuales se combinan como en regresión lineal para cada nodo intermedio

z\_1 y z\_2 hasta z\_h donde h es el número de nodos ocultos intermedios.

De esta forma z\_j es igual a Beta\_0j más

Beta\_1j por x\_1 más Beta\_2j por x\_2 para j igual 1,

2 hasta h. Después el procedimiento es el mismo con la salida de cada z\_j se

crean las funciones no lineales A\_j igual a la función sigmoide evaluada en z\_j,

es decir, el exponente de z\_j sobre uno más el exponente de z\_j.

Por último, el estimador final se forma como f gorro de x\_1,

x\_2 es igual a w\_0 más w\_1 por A\_1 más w\_2 por A\_2 hasta w\_h por A\_h.

Esta operación puede crear formas no lineales bastante flexibles.

Miremos un ejemplo de regresión con un predictor x,

en este caso el predictor es f gorro de x es igual a w\_0 más la suma de j igual 1 hasta

h de los w\_j por

Beta\_0j más Beta\_1j por x. Miremos bien la forma de esta función.

Cada a\_j que es igual a la función de sigmoide

evaluada en z\_j representa una función así donde el

parámetro Beta\_0j determina el punto medio de cada curva

sigmoide y el parámetro Beta\_1j determina la amplitud de la misma.

Se crean h curvas de esta forma.

Ahora cuando las combinamos linealmente con los pesos wj,

entonces se pueden obtener formas muy flexibles.

Entre más grande h, más flexible es la función.

En esta formulación, el número total de parámetros a estimar es 1 más h por 1 más p,

el cual puede crecer muy rápido con el número de nodos intermedios o

el número de variables predictoras.

El número de parámetros y la simetría entre las unidades ocultas,

que son intercambiables entre ellas,

hace que el problema de optimización sea complejo y difícil de plantear.

Sin embargo, resulta que existe

un método relativamente eficiente para estimar esos parámetros,

que se conoce como propagación hacia atrás o back propagation.

La idea fundamental es que la función que se optimiza,

que se conoce como pérdida, es sencilla.

De hecho, son las mismas que conocemos como mínimos cuadrados

para la regresión y entropía cruzada para clasificación,

que equivale a la misma métrica usada en regresión logística.

Por ejemplo, en caso de regresión,

la optimización se realiza con respecto a los parámetros Betas y

w y se resuelve por métodos numéricos de descenso.

En específico se usa un método conocido como gradiente descendiente.

Si tenemos la función de pérdida en términos de los Betas y los doble us,

que llamaremos genéricamente Theta igual a los Betas y los doble us;

entonces, buscamos el mínimo,

iterativamente, avanzando la búsqueda de esta forma.

Theta en la iteración k más 1

es igual a Theta en la iteración k menos

Gamma por el gradiente de la función de pérdida evaluada en Theta k;

donde esta función,

los evalúan Theta k corresponde a la

función de pérdida evaluada en el valor actual de las variables.

Este método asegura que la función de pérdida evaluada en Theta k más

1 es menor o igual a la función de pérdida evaluada en Theta k. Es decir,

la función va disminuyendo.

En convergencia, el valor de Theta estrella es el que

minimiza la función de pérdida o por lo menos es un mínimo local.

Miremos gráficamente cómo funciona.

Supongamos que en la iteración actual,

Theta k tiene este valor,

entonces el gradiente es positivo

y tiene una magnitud igual a la pendiente de la línea tangente.

De esta forma, para encontrar el mínimo debemos movernos en dirección negativa,

o sea, contrario en gradiente.

Si el valor de Theta k está en este lado,

entonces el gradiente es negativo,

y Theta k más 1 debe ser mayor para que la función baje.

Resulta que en la estructura de redes neuronales,

el cálculo del gradiente es relativamente fácil de obtener.

Incluso es sencillo de adaptar cuando se tienen redes multicapa,

es decir, con varias capas de nodos ocultos.

En este caso la estructura es similar a la que ya conocemos.

La diferencia es que las salidas de los vectores z j de la primera

capa se convierten en los inputs de los nodos de la segunda capa y así sucesivamente.

Llamemos a j l con el superíndice l para representar el número de la capa oculta,

y j igual 1 2 hasta h de l,

donde h de l es el número de nodos en la capa de l. El cómputo genérico del modelo

de red multicapa es a j en la capa l más 1

es igual a la función sigmoide de Beta 0 j l más 1 más la suma desde i igual 1 hasta h

l de Beta y j l más 1 por a i en la capa l. Notemos que desde la segunda capa,

las no linealidades se tornan más profundas,

dado que son funciones no lineales de funciones no lineales de las variables originales.

Esto es lo que hace que las redes neuronales sean bastante flexibles.

Lo que conocemos como "Deep Learning" es el mismo concepto de las redes neuronales,

pero en aplicaciones con muchas capas

de nodos ocultos y posiblemente muchos nodos por capas.

Estas tipologías de redes complicadas fueron posibles de resolver gracias

a los avances computacionales y estrategias

algorítmicas para estimar los parámetros involucrados.

Por ejemplo, el uso de gradiente descendiente estocástico en el

cual no se usan todos los puntos en cada iteración para calcular el gradiente,

sino un subconjunto de ellos.

Por cierto, la función sigmoide se conoce como una función

de activación y es la que añade no linealidad.

Se pueden usar otros tipos de funciones de activación.

De hecho, existe una función que resulta más

práctica para reducir la carga computacional,

que es la función RELU por "Rectified Linear Unit",

la cual se define como RELU de x es el máximo entre 0 y x.

Por la gran cantidad de parámetros involucrados

en la estimación de las redes neuronales profundas,

es necesario utilizar métodos de regularización para

controlar el "overfitting" y la maldición de la dimensionalidad.

Por supuesto, entre más capas y más nodos por capas tenga la red,

la función es más compleja y propensa a hacer "overfitting".

Sin embargo, es difícil de calibrar la topología de la red directamente.

Además del número de capas, existen otras estrategias.

Uno de los métodos más tradicionales es la famosa parada temprana o "early stopping".

Esto corresponde a la idea de no estimar los

parámetros hasta encontrar el mínimo de la función de pérdida,

sino parar antes de convergencia.

Aunque suena un poco extraño,

este método puede funcionar bien y tiene evidentes ventajas computacionales.

Otra opción es usar penalización sobre los parámetros estimados.

Las más usadas son la norma dos,

tipo "Ridge", y la norma 1, tipo "Lasso".

Contrario al "early stopping",

este método añade cierta complejidad al algoritmo,

dado que el gradiente ahora es un poco más elaborado.

Sin embargo, el desempeño suele ser más efectivo.

Una tercera opción, muy usada en los últimos años para las redes muy complejas,

es la estrategia de abandono o "drop out".

Esta estrategia toma un subconjunto aleatorio, por ejemplo,

el 70 por ciento de los nodos de cada capa en cada iteración del gradiente descendiente.

Esto permite que los nodos no tenidos en

cuenta no se especialicen demasiado produciendo "overfitting".

Igual que en "early stopping",

además de producir regularización,

es una estrategia conveniente para reducir el costo computacional.

En redes muy profundas,

se convierte en una muy buena opción.

Como ya lo habíamos mencionado,

existen tipologías de redes particulares que cambian

un poco la topología de crear nodos hacia adelante que vimos,

para explotar estructuras de datos diferentes.

Tal es el caso de las redes convolucionales y de las redes recurrentes.

Las redes convulocionales emergieron como

buenos algoritmos para reconocimiento y clasificación de imágenes.

Aprovechan la distribución espacial natural de los inputs,

en este caso píxeles en la imagen para crear

topología de redes con las mismas relaciones en dos dimensiones.

Las transformaciones en cada iteración usan filtros

convolucionales que son transformaciones en arreglos matriciales.

Al hacer esto, las matrices resultantes de cada

capa suelen especializarse en partes específicas de la imagen original.

Por otro lado, están las redes neuronales

recurrentes que son especializadas en estructuras de datos secuenciales.

Esto es cuando en un input determinado,

el orden en que ocurren los valores de las variables es relevante.

Por ejemplo, el orden de las palabras cuando se quiere interpretar el contenido de un

documento o los sonidos que componen el reconocimiento de canciones o de voz.

La característica es entonces,

que los inputs son secuencias de datos.

Dentro de este tipo de redes sobresalen las llamadas redes de memoria corta y

larga o "Long short-term memory", LSTM.

En este vídeo estudiamos los conceptos

fundamentales de las redes neuronales y del aprendizaje profundo.

Vimos que las redes aparecen como modelos poco diferentes de lo que habíamos visto,

pero que pueden obtener un desempeño predictivo formidable, en especial,

cuando se tienen grandes cantidades de datos y los

inputs tienen características topológicas que se pueden aprovechar.

En el próximo vídeo estudiaremos otro tipo de algoritmos conocidos

como máquinas de soporte vectorial.

SESIONES SINCRONICAS – NOTAS

24-01-2024

