

IArtificial.net

Jose Martinez Heras

<https://www.iartificial.net>

Impreso en 16/09/2023

Table of Contents

[Machine Learning e Inteligencia Artificial](#)
[¿Cómo aprende la Inteligencia Artificial?](#)
[¿Clasificación o Regresión?](#)
[Las 7 Fases del Proceso de Machine Learning](#)
[Error Cuadrático Medio para Regresión](#)
[Generalización en Machine Learning](#)
[Análisis de Errores en Machine Learning](#)
[15 Librerías de Python para Machine Learning](#)
[Inteligencia Artificial aplicada a meneame.net](#)
[Regresión Lineal: teoría y ejemplos en Python](#)
[EU Datathon 2019 – certamen de datos abiertos](#)
[Contraste de Hipótesis 1 – ¿cómo no aceptar lo falso?](#)
[Regresión Polinómica en Python con scikit-learn](#)
[Feliz San Valentín – Amor en 20 minutos](#)
[Análisis Descriptivo, Predictivo y Prescriptivo de datos](#)
[Redes neuronales desde cero \(I\) – Introducción](#)
[Gradiente Descendiente para aprendizaje automático](#)
[Regularización Lasso L1, Ridge L2 y ElasticNet](#)
[Mejora tu Inglés con estas frases elegidas con IA](#)
[Redes Neuronales Generativas Adversarias \(GANs\)](#)
[Empresas con experiencia en Inteligencia Artificial](#)
[Máquinas de Vectores de Soporte \(SVM\).](#)
[Regresión Logística para Clasificación](#)
[¿Cómo usar Regresión Logística en Python?](#)
[Árboles de Decisión con ejemplos en Python](#)
[Datos Fundamentales de Empresas Cotizadas en Bolsa](#)
[Aguathon: mi solución al primer Hackathon del Agua](#)
[Ensembles: voting, bagging, boosting, stacking](#)
[Random Forest \(Bosque Aleatorio\): combinando árboles](#)
[Avances en la generación de caras con GANs](#)
[Basura Espacial: competición con machine learning](#)
[Precision, Recall, F1, Accuracy en clasificación](#)
[Clustering \(Agrupamiento\), K-Means con ejemplos en python](#)
[Google Dataset Search – descubre conjuntos de datos](#)
[Feliz San Valentín menéame](#)
[La Maldición de la Dimensión en Machine Learning](#)
[Redes neuronales desde cero \(II\): algo de matemáticas](#)
[Correlación, Covarianza e IBEX-35](#)
[10 Blogs de Inteligencia Artificial y Machine Learning](#)
[Antenas de Espacio Profundo & Inteligencia Artificial](#)
[Detección de anomalías en espacio](#)
[TorchServe para servir modelos de PyTorch](#)
[Algoritmos Genéticos y Memoria Visual](#)
[Segmentación de Imágenes con Redes Convolucionales](#)
[Machine Learning en Operaciones Espaciales](#)
[Colaboradores](#)
[Sobre mí](#)

Machine Learning e Inteligencia Artificial

Por Jose Martinez Heras
03/12/2018

El Machine Learning (ML), también conocido como aprendizaje automático, era hasta hace poco uno de los campos de la inteligencia artificial. Ahora, la mayoría de la gente, cuando dice «Inteligencia Artificial» en realidad se está refiriendo al Machine Learning. Veamos la diferencia.

Inteligencia Artificial

De acuerdo con la wikipedia, la [Inteligencia Artificial](#) (IA), es la inteligencia exhibida por máquinas. De esta forma, consideramos a una identidad inteligente si imita las funciones cognitivas que los humanos asociamos con otras mentes humanas, como por ejemplo *aprender* y *resolver problemas*.

La Inteligencia Artificial abarca muchos campos, incluyendo:

- Resolución de problemas, razonamiento
- Representación del conocimiento
- Hacer planes
- **Aprendizaje automático (Machine Learning)**
- Procesamiento de Lenguaje Natural
- Percepción
- Movimiento y Manipulación

El campo que más rápido se está desarrollando en los últimos años es el del aprendizaje automático, también conocido como Machine Learning. Y de hecho, es el principal responsable de la popularidad de la Inteligencia Artificial.

Entonces ... ¿qué es el Machine Learning?

Machine Learning

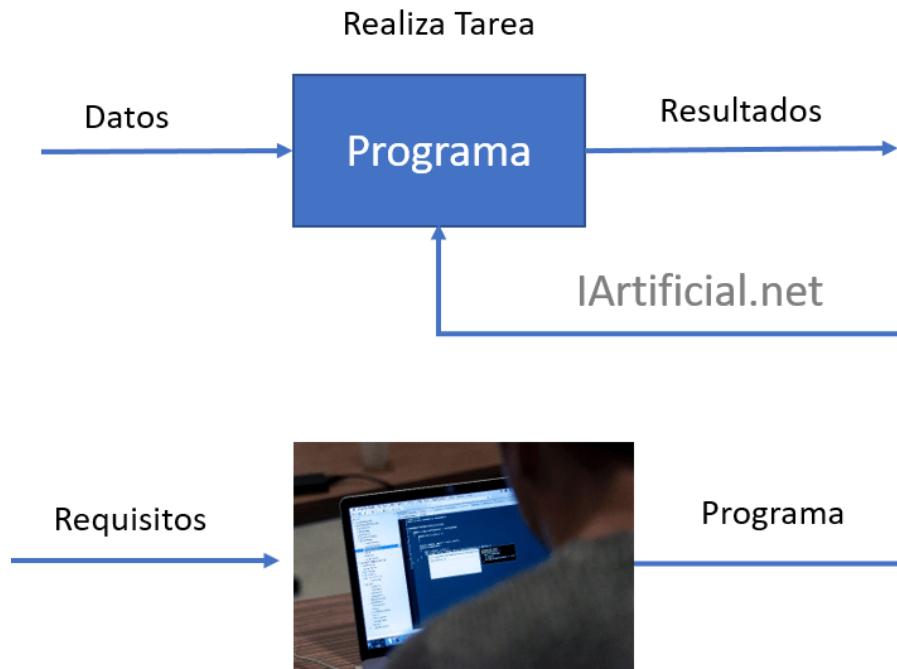
«La ciencia (y el arte) de programar ordenadores para que puedan aprender de los datos»

Aurélien Géron, 2017

Hay otras definiciones, por supuesto, pero ésta es lo suficientemente simple y corta para que todos podemos comprenderla. Además me gusta que incluya «arte» porque esa es también mi experiencia.

En la programación clásica, el objetivo es realizar una tarea. Para realizarla, tenemos los siguientes elementos (leyendo el siguiente gráfico de abajo hacia arriba):

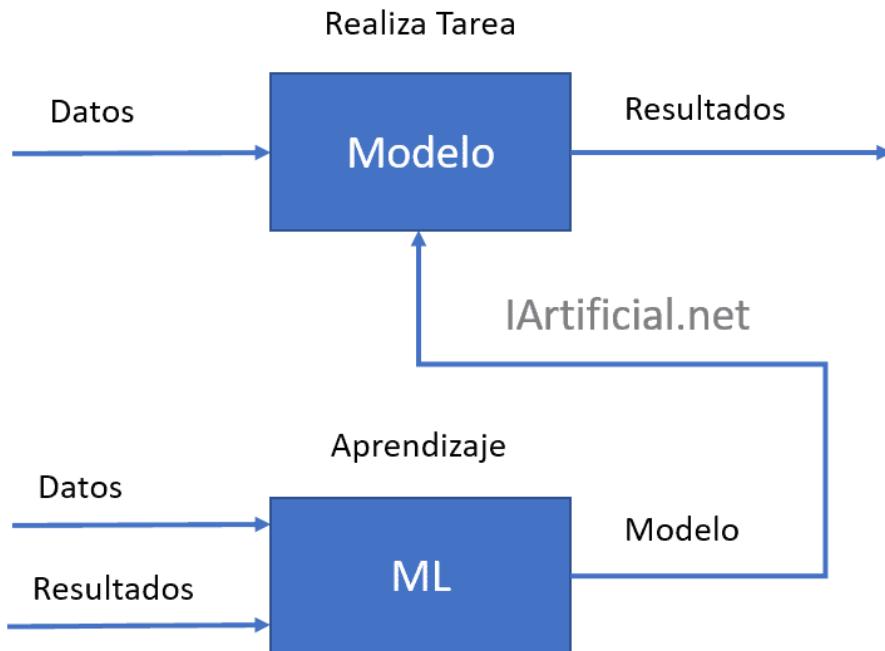
- **Requisitos:** lo que el programa tiene que hacer
- **Desarrollador:** partiendo de los requisitos, escribe un programa
- **Programa:** software que realiza la tarea
- **Datos:** que el programa necesita como entrada
- **Resultados:** que produce el programa a partir de los datos



Un programa, escrito por un programador a partir de requisitos, realiza una tarea

Cuando usamos machine learning, el objetivo es también el de realizar una tarea. Para realizarla, tenemos los siguientes elementos (leyendo el siguiente gráfico de abajo hacia arriba):

- **Datos históricos:** necesarios para aprender un modelo
- **Resultados históricos:** que sirve para que la técnica de aprendizaje automático aprenda qué resultados van con cada dato
- **Aprendizaje:** que ya veremos que se trata de encontrar los mejores parámetros para el modelo
- **Modelo:** por ejemplo, regresión lineal o una red neuronal
- **Datos:** que el modelo necesita como entrada
- **Resultados:** que produce el modelo a partir de los datos.



Un modelo de machine learning realiza una tarea. El modelo ha sido aprendido a partir de datos y resultados históricos

En el caso que usemos un modelo de aprendizaje de datos, no necesitamos a ningún programador. Esto es posible porque hubo programadores que desarrollaron estos algoritmos de machine learning. Por supuesto, los siguen mejorando, lo que hace que cada vez funcionen mejor. Pero si dejases de seguir mejorándolos, todavía podríamos obtener modelos a partir de datos y resultados históricos automáticamente.

Esto no quiere decir que ya no necesitamos desarrolladores de software. Los modelos de machine learning sólo sirven para realizar algunas tareas. Los desarrolladores, en cambio, pueden crear software para realizar cualquier tipo de tareas.

¿Qué tareas se pueden resolver con Machine Learning?

Estos son algunos ejemplos de lo que el aprendizaje automático puede hacer:

- **Filtro de spam:** ahora recibimos muchísimo menos correos electrónicos de spam que hace años
- **Recomendaciones personalizadas:** como las que recibimos en Amazon o Netflix
- **Publicidad:** predicción de cuáles son los anuncios que nos van a interesar más
- **Mercado inmobiliario:** predicción de cuál va a ser el precio de venta de un inmueble
- **Reconocimiento de imágenes:** reconocer qué contiene una imagen (¿es un coche?, ¿un perro?, ¿un gato?)
- **Percepción:** saber qué objetos hay en una imagen, dónde están y cómo se mueven (necesario para los coches autónomos)
- **Procesado del habla:** por ejemplo Siri, Alexa o el asistente de Google
- **Traducción automática:** para traducir entre idiomas como en Google Translate
- **Noticias:** predecir qué artículos tendrán un mayor número de visitas
- **Juegos:** crear jugadores artificiales que jueguen mejor que nosotros (por ejemplo, Google AlphaZero AI)

¿Cuándo merece la pena usar Machine Learning?

No siempre es una buena idea usar machine learning para resolver un problema. De hecho, mi consejo es que intentes resolver problemas sin usar machine learning! Entonces ... ¿por qué se está volviendo tan popular? La respuesta es que el uso de Machine Learning permite resolver problemas que, de otra forma, no serían resolubles.

Para que merezca la pena enfocar la resolución de un problema con Machine Learning, se deben cumplir estas 3 condiciones:

1. Debe haber un patrón entre los datos de entrada y los resultados (no se pueden predecir los números de la lotería, por ejemplo)
2. La cantidad de datos disponibles debe ser suficiente para encontrar este patrón (al fin y al cabo, el modelo de machine learning se crea a partir de los datos)
3. Debe ser difícil formular una expresión matemática que explique este patrón (si fuera fácil, utilizaríamos la fórmula)

Recursos

- [\[vídeo\]](#) introductorio a la Inteligencia Artificial y al Aprendizaje Automático (en inglés)
- Descubre cómo aprende la Inteligencia Artificial

¿Cómo aprende la Inteligencia Artificial?

Por Jose Martinez Heras
07/12/2018

Una de las características más importante de la Inteligencia Artificial, es su capacidad de aprender automáticamente. En este artículo vamos a explicar cómo aprende la Inteligencia Artificial y los tipos de aprendizaje automático que existen: supervisado, no-supervisado, semi-supervisado y por refuerzo.

Para hacerlo más interesante, vamos a empezar con un ejemplo. Imaginaos que tenemos que escribir un programa para detectar si un correo electrónico es spam o no. Podríamos intentar eliminar los correos que tuviesen algunas palabras. Las palabras las pondríamos en un fichero para que fuese más fácil editarla, claro está. Pero habría dos problemas:

- Sería casi imposible que se nos ocurran todas las palabras (o combinaciones de palabras) que poner en el fichero. Así que seguiríamos recibiendo spam.
- Probablemente algunos correos serían clasificados como spam sin realmente serlo. Así que perderíamos mensajes.

Podríamos mejorar nuestro fichero, añadir otro fichero diciendo que otras combinaciones de palabras están autorizadas, etc. Pero requeriría de mucho trabajo por nuestra parte, y la gente que envía correos basura, acabaría encontrando la forma de saltarse nuestras reglas.

Usando técnicas de Inteligencia Artificial, y en particular usando el Machine Learning podemos ahorrarnos este trabajo. Le podemos dar a la IA una lista de correos deseados y otra de correos basura. Esta es todo la información que la Inteligencia Artificial necesita para aprender la diferencia. No sólo nos ahorrará el trabajo de hacerlo a nosotros; además los resultados serán mejores de lo que nosotros podríamos haber hecho.



Spam o Correo no deseado

La forma de aprender de la inteligencia artificial depende del **tipo de aprendizaje automático** (machine learning) que estemos usando. Podemos distinguir 4 tipos de machine learning según la supervisión que necesiten: supervisado, no supervisado, semi-supervisado y por refuerzo. Veamos la diferencia.

Aprendizaje Supervisado

El aprendizaje supervisado, recibe el nombre de «supervisado» porque necesita que siempre le enseñemos la respuesta correcta. Por ejemplo, para resolver el problema de clasificar correos

electrónicos como spam o no spam, necesita que le demos ejemplos históricos correctamente clasificados. En otras palabras, para cada caso de ejemplo que le demos, necesita saber si el correo era spam o no.

Sabiendo cuál era la respuesta correcta, la Inteligencia Artificial **aprende de sus propios errores**. Muchos algoritmos supervisados empiezan dando respuestas aleatorias y después van mejorando a medida que aprenden de sus errores.

El aprendizaje supervisado es muy útil cuando queremos que la Inteligencia Artificial realice una tarea en el futuro. Esta característica hace a este tipo de aprendizaje automático muy atractivo para muchos negocios. En el entorno [empresarial](#), el aprendizaje supervisado también se conoce como «[análisis predictivo](#)».

Proceso del aprendizaje supervisado

1. Recopilar datos históricos, incluyendo la respuesta correcta
2. Construir un modelo de machine learning con esos datos
3. Evaluar el modelo, para entender qué rendimiento podemos esperar de él
4. Usar este modelo con datos nuevos. Por ejemplo, cuando llegue un correo electrónico nuevo

Este proceso está un poco simplificado. En otro artículo lo extenderé, pero da una idea de la forma de pensar acerca del aprendizaje supervisado.

Ejemplos de aprendizaje supervisado

- Clasificar correos electrónicos en genuinos o spam
- Predecir por cuánto dinero se va a vender una propiedad inmobiliaria
- Calcular la probabilidad de que hagas click en un anuncio
- Decidir qué hotel enseñarte primero en una web de hoteles
- Estimar si el cliente de un banco será capaz de pagar un crédito
- Recomendar si es mejor comprar ahora tu vuelo o es mejor esperar
- El traductor de Google
- Detección de objetos en una imagen
- Reconocimiento y síntesis de voz
- Estimar qué edad tienes a partir de una foto
- Predecir qué clientes van a cancelar su contrato con su compañía telefónica
- Estimar cuántas visitas va a tener un artículo en la web
- Generación de música por ordenador

Aprendizaje No Supervisado

El aprendizaje no supervisado, no necesita supervisión. Esto quiere decir que no necesita que le digamos cuál es la respuesta correcta. También significa que no podemos calcular el error de sus resultados. Cuando usamos aprendizaje no supervisado, la Inteligencia Artificial aprende de los datos mismos, por así decirlo.

El aprendizaje no supervisado es útil cuando queremos entender mejor nuestros datos históricos. Puede tener implicaciones en el futuro pero no son tan obvias. La principal ventaja de cara al futuro es que nosotros, los humanos, entendemos mejor nuestros datos. Con este nuevo conocimiento revelado por la IA, podemos tomar mejores decisiones. Veamos un ejemplo.



Algunos supermercados usan aprendizaje no supervisado para entender mejor sus datos.

Los supermercados generan muchísimos datos. Entre ellos, están los datos de lo que cada cliente compra. Aunque no tengan forma de saber quién eres, sí saben lo que un cliente anónimo ha metido en el carro de la compra. Cuando le damos a la IA los datos del carro de la compra, no le decimos lo que tiene que hacer con ellos. En este sentido no hay ninguna «solución correcta». Sólo le pedimos que analice los datos y nos diga si hay algo interesante en ellos.

Usando aprendizaje no supervisado, hace tiempo se descubrió que muchos de los carros de la compra que tenían cerveza, también tenían pañales. Esta relación sobre datos históricos nos permite idea una estrategia: ¿y si ponemos nuestra marca de pañales al lado de la cerveza? El resultado fue que muchos clientes compraron los pañales de la marca el supermercado sólo por la comodidad de tenerlos tan a mano.

Proceso del aprendizaje no supervisado

1. Recopilar datos históricos
2. Instruir a la Inteligencia Artificial a que busque patrones, grupos, etc.
3. Utilizar la información nueva para tomar mejores decisiones en el futuro

Ejemplos de aprendizaje no supervisado

- Ayudar a elegir qué medidas del cuerpo hay que usar para las tallas (XS, S, M, L, XL, XXL, etc.)
- Comprimir imágenes (con pérdida de calidad)
- Sistemas de recomendación. Por ejemplo, los que usa Netflix o Amazon

Aprendizaje Semi-supervisado

El aprendizaje semi-supervisado está entre el aprendizaje supervisado y el aprendizaje no supervisado:

- utiliza datos etiquetados, como en el aprendizaje supervisado
- y también datos no etiquetados, como en el aprendizaje no supervisado

Conseguir datos etiquetados, en algunos casos, es bastante difícil. Por ejemplo, si queremos construir nuestro sistema de detección de spam, alguien tiene que decidir cuáles mensajes son

spam y cuáles no. Y ese alguien somos nosotros, los humanos. Necesitamos hacer este trabajo manual para que la Inteligencia Artificial pueda aprender qué es lo que tiene que hacer.

Usando una estrategia de aprendizaje semi-supervisado, podríamos etiquetar manualmente algunos correos electrónicos, dejar que la Inteligencia Artificial aprenda y empezar a usarlo. No va a funcionar tan bien como si hubiésemos etiquetado más correos. Pero por lo menos, podríamos empezar a utilizar el sistema anti-spam antes. Seguramente tendríamos correos mal clasificados, pero conforme los fuésemos etiquetando correctamente iría aprendiendo mejor.

Cuando esto se hace a una escala mayor, no resulta tan pesado. Por ejemplo, si usas el correo de Google, cada vez que marcas un correo como spam, estás ayudando a todos los demás usuarios de gmail. Google usa [crowdsourcing \(colaboración distribuida\)](#) para que funcione muy bien con muy poco trabajo individual ... aunque sea mucho trabajo si tenemos en cuenta a todas las personas que contribuyen.

Proceso del aprendizaje semi-supervisado

1. Recopilar los pocos datos históricos históricos que tengan resultados disponibles
2. Recopilar los datos históricos sin resultados
3. Evaluar la posibilidad de etiquetar manualmente más datos históricos
4. Instruir a la Inteligencia Artificial a que aprenda utilizando aprendizaje supervisado los datos históricos para los que tenemos resultados
5. Usar el modelo de machine learning aprendido para etiquetar automáticamente el resto de los datos
6. Usar otro modelo de machine learning supervisado con los datos etiquetados inicialmente y los datos etiquetados automáticamente

Ejemplos de aprendizaje semi-supervisado

- Detección de correos basura
- Detección automática de anomalías
- Reconocimiento facial

Aprendizaje por Refuerzo

El aprendizaje por refuerzo es una especie de aprendizaje supervisado ... sin llegar a serlo. En el aprendizaje supervisado, para cada dato que ofrecemos como ejemplo, también decimos cuál fue solución correcta. En el aprendizaje por refuerzo, no podemos dar la solución correcta hasta saber lo que la Inteligencia Artificial va a hacer. Normalmente hará muchas cosas, y sólo sabremos cómo de bien o de mal lo hizo después de un tiempo.



La Inteligencia Artificial aprende a ganar al ajedrez con aprendizaje por refuerzo

Como esto es un poco abstracto, vamos a poner un ejemplo. A lo mejor has visto en las noticias que [AlphaZero](#) gana a los mejores jugadores al ajedrez y al Go. Si queremos que la IA aprenda a ganar a humanos, o a otros programas de ordenador, no podemos decirle cuál es la solución correcta. Al decírselo, sólo podría ser tan bueno como nosotros o como los otros programas de ordenador. Si queremos que juegue mejor que nosotros, tenemos que enseñarle las reglas del juego y dejarle aprender por sí mismo. Sólo así podrá encontrar soluciones creativas, poco comunes y geniales.

El aprendizaje por refuerzo es el tipo de aprendizaje más difícil. La razón es que la IA no puede saber cómo de buena es la acción que acaba de realizar; en otras palabras, no puede calcular cuál ha sido su error. Tiene que esperar al final de la partida para saber si ha ganado, empatado o perdido. Esta es la única información que tiene. En el aprendizaje por refuerzo, la Inteligencia Artificial aprende de sus éxitos y de sus fracasos (no de sus errores).

Nota: utilizo los términos «machine learning» y «aprendizaje automático» indistintamente.

Recursos

- [\[vídeo\]](#) introductorio a la Inteligencia Artificial y al Aprendizaje Automático (en inglés)

¿Clasificación o Regresión?

Por Jose Martinez Heras
15/12/2018

Cuando usamos aprendizaje automático, podemos realizar tareas de clasificación o de regresión. La diferencia está en el tipo de resultado que queremos que la técnica de machine learning produzca. Veamos la diferencia.

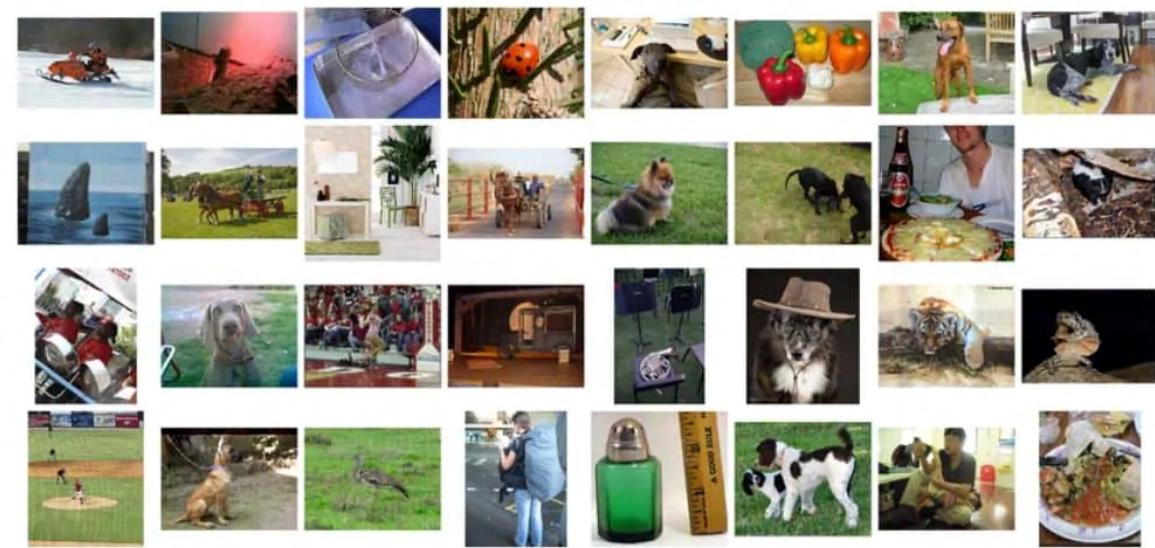
Clasificación

Cuando usamos clasificación, **el resultado es una clase, entre un número limitado de clases**. Con clases nos referimos a categorías arbitrarias según el tipo de problema.

Por ejemplo, si queremos detectar si un correo es spam o no, sólo hay 2 clases. Y el algoritmo de machine learning de clasificación, tras darle un correo electrónico, tiene que elegir a qué clase pertenece: spam o no-spam. Hay muchos más ejemplos, por supuesto:

- ¿comprará el cliente este producto? [sí, no]
- ¿tipo de tumor? [maligno, benigno]
- ¿subirá el índice bursátil? IBEX mañana [sí, no]
- ¿es este comportamiento una anomalía? [sí, no]
- ¿nos devolverá este cliente un crédito? [sí, no]
- ¿qué deporte estás haciendo? tal y como lo detectan los relojes inteligentes [caminar, correr, bicicleta, nadar]
- ¿obtendrá una historia un número alto de visitas en un agregador de noticias? [sí, no]

Otro ejemplo famoso en el mundo del Machine Learning, es el concurso de clasificación de imágenes [ImageNet](#). En este concurso, el objetivo es reconocer qué hay en una imagen. Hay 1000 posibilidades. Entre las 1000 clases posibles están: perro, gato, coche, avión, globo, manzana, etc.



ImageNet – Concurso de Machine Learning usando Clasificación. Dí que hay en la imagen entre 1000 posibles clases

Clasificación con probabilidades

Muchos algoritmos de machine learning dan los resultados de clasificación con probabilidades. Es decir, nos pueden decir que un correo es spam con una probabilidad del 89%. O que una imagen tiene un 67% de probabilidades ser un perro, un 18% de ser un gato, un 9% de ser una oveja, etc.

Normalmente, en el caso que usemos probabilidades, se tiende a elegir la clase con probabilidad más alta como resultado del proceso de clasificación. Otra posibilidad que está a nuestro alcance es fijar un mínimo de probabilidad antes de estar dispuestos a dar un resultado. Por ejemplo, si no estamos seguros con una probabilidad 80% o mayor, diremos que no estamos seguros, en vez de decir que es un perro. Esta estrategia puede ser útil cuando el coste de equivocarnos es alto, comparado con el beneficio de obtener la respuesta correcta.

Técnicas de Machine Learning para Clasificación

Hay varias técnicas de machine learning que podemos usar en problemas de clasificación. Podemos destacar:

- regresión logística (logistic regression)
- máquinas de vectores de soporte (support vector machines)
- [árboles de decisión](#) (decision trees)
- [bosques aleatorios](#) (random forests)
- [redes neuronales](#) y [aprendizaje profundo](#) (deep learning)

Regresión

Cuando usamos regresión, el **resultado es un número**. Es decir, el resultado de la técnica de machine learning que estemos usando será un valor numérico, dentro de un conjunto infinito de posibles resultados.

Aquí van algunos ejemplos de regresión:

- Predecir por cuánto se va a vender una propiedad inmobiliaria
- Predecir cuánto tiempo va a permanecer un empleado en una [empresa](#)
- Estimar cuánto tiempo va a tardar un vehículo en llegar a su destino
- Estimar cuántos productos se van a vender

Técnicas de Machine Learning para Regresión

Hay varias técnicas de machine learning que podemos usar en problemas de Regresión. Podemos destacar:

- regresión lineal y [regresión no lineal](#)
- máquinas de vectores de soporte (support vector machines)
- [árboles de decisión](#) (decision trees)
- [bosques aleatorios](#) (random forests)
- [redes neuronales](#) y [aprendizaje profundo](#) (deep learning)

Aclaraciones

Aunque hay algunas técnicas que son específicas de clasificación y otras de regresión, la mayoría de las técnicas funcionan con ambos.

Un motivo de confusión frecuente es la técnica de *regresión logística*. Su nombre podría inducirnos a pensar que puede usarse en problemas de regresión. Sin embargo, la regresión logística, sólo funciona para problemas de clasificación. La explicación completa está [aquí](#).

Recursos

- [vídeo](#) introductorio a la Inteligencia Artificial y al Aprendizaje Automático (en inglés)

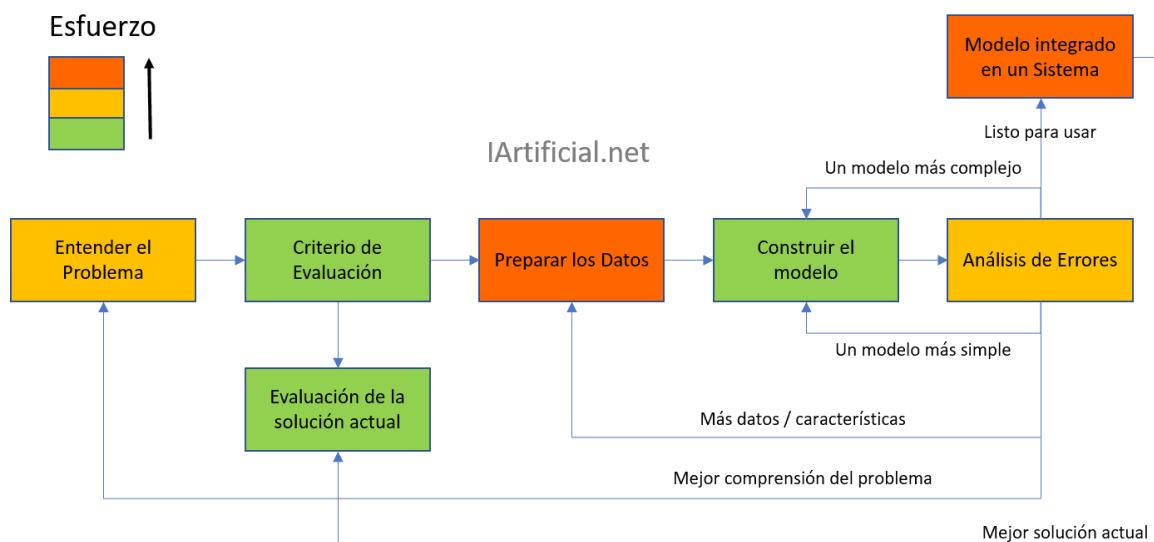
Las 7 Fases del Proceso de Machine Learning

Por Jose Martinez Heras
21/12/2018

A la hora de usar Machine Learning, es conveniente seguir un proceso si queremos obtener buenos resultados. Al seguir un proceso, podemos hacer un mejor uso de nuestro tiempo. Además tendremos una orientación sobre qué es lo que debemos hacer en caso de que nuestros resultados no sean tan buenos como esperábamos. En este artículo vamos a hablar sobre las fases del proceso de machine learning y cómo están relacionadas entre ellas.

Resumen de las 7 fases del proceso de machine learning

El diagrama muestra las distintas fases del proceso de machine learning y cómo interactúan entre ellas. Están basadas en el estándar [CRISP-DM](#), con la excepción de que me he permitido incluir la fase de «evaluación de la solución actual».



Las 7 Fases del Proceso de Machine Learning

Fase 1: Entender el Problema

Es muy importante entender el problema que tenemos que resolver. A lo mejor no me crees si te digo que he visto casos en los que se encontró una solución perfecta para un problema diferente del que se tenía. Entender el problema normalmente lleva bastante tiempo, sobre todo si el problema proviene de un sector en el que tienes pocos conocimientos. Le he asignado un esfuerzo relativo medio a esta fase porque la idea es colaborar con las personas que saben mucho acerca del problema.

Mi consejo para tener un mejor entendimiento del problema es preguntar ¿por qué? ... y cuando te respondan, pregunta otra vez ¿por qué? ... y así hasta que te quedes satisfecho. En muchos casos, sabiendo el porqué de las cosas puede ayudarte a entender rápidamente la forma de pensar en esa industria. Además te va a resultar más fácil imaginarte las respuestas a otras preguntas que seguro que tendrás cuando estés trabajando en el problema.

Entender los datos

Tan importante como entender el problema es entender los datos que tenemos disponibles. Es común hacer un [análisis exploratorio de datos](#) para familiarizarnos con ellos. En el análisis exploratorio se suelen hacer [gráficos](#), correlaciones y estadísticas descriptivas para comprender

mejor qué historia nos están contando los datos. Además ayuda a estimar si los datos que tenemos son suficientes, y relevantes, para construir un modelo.

Fase 2: Definir un Criterio de Evaluación

Imagínate que obtenemos un modelo de machine learning perfecto ... ¿cómo lo sabríamos? O peor aún, ¿cómo sabríamos que un modelo da resultados de poca calidad? Antes de empezar siquiera a pensar en utilizar un modelo de aprendizaje automático, tenemos que definir cómo vamos a evaluarlo.

El criterio de evaluación se trata normalmente de una medida de error. Típicamente se usa el [error cuadrático medio](#) para problemas de regresión y la [entropía cruzada](#) para problemas de clasificación. Para problemas clasificación con 2 clases (que son muy comunes), podemos utilizar otras medidas tales como la precisión y exhaustividad.

Le he asignado un esfuerzo relativo bajo a esta fase porque hay ya varios criterios de evaluación estándar que funcionan para muchos problemas. Además son tan generales que funcionan en casi todos las industrias y sectores.

Fase 3: Evaluación de la solución actual

Probablemente el problema que queremos resolver con Machine Learning, ya se esté resolviendo de otra forma. Seguramente, la motivación de usar aprendizaje automático para resolver este problema sea la de obtener mejores resultados. Otra motivación común es la de obtener resultados similares de forma automática, reemplazando posiblemente un trabajo manual aburrido.

Si medimos el rendimiento de la solución actual (con el criterio de evaluación elegido), podemos compararlo con el rendimiento del modelo de machine learning. Así podremos saber si merece la pena usar el modelo de machine learning o si nos quedamos con la solución actual.

Si no hay ninguna solución actual, podemos definir una solución simple que sea muy fácil de implementar. Por ejemplo, si queremos predecir el precio de una vivienda con machine learning, podríamos compararlo con una solución simple (por ejemplo, valor mediano del metro cuadrado por barrio). Sólo así, cuando tengamos un modelo de machine learning terminado, podremos decir si es suficientemente bueno, si necesitamos mejorarlo, o si no merece la pena.

Si al final resulta que la solución actual o una solución simple es similar a la solución que nos da el machine learning, probablemente sea mejor usar la solución simple. Funcionará casi siempre mejor y será más robusta.

Fase 4: Preparar los datos

La preparación de datos es una de las fases del machine learning que supone un mayor esfuerzo. Los principales desafíos a los que nos vamos a encontrar en esta fase son los siguientes:

Datos Incompletos

Es bastante normal que no tengamos todos los datos que nos gustaría tener. Por ejemplo, si queremos predecir qué clientes tiene más probabilidad de comprar un producto y los datos que tenemos provienen de una encuesta por internet. Habrá muchas personas que no hayan llenado todos los campos. Pero la vida es así, y tener datos incompletos es mejor que no tener datos en absoluto. Así que ¿cómo podemos lidiar con datos incompletos?

- Eliminándolos: una opción fácil es sólo quedarnos con los datos completos. Esto puede ser una opción si se trata de una minoría. Pero casi nunca es la mejor opción porque habría muchos datos (encuestas) que estaríamos obligados a desechar.
- Imputarlos con un valor razonable: cuando falte un valor, pondremos automáticamente un valor que tenga sentido. Por ejemplo, si alguien no ha puesto su edad en una encuesta, podríamos usar la edad media de los encuestados que sí han especificado su edad.

- Imputarlos con un modelo de aprendizaje automático: si queremos ser más sofisticados, podemos construir un modelo de machine learning que prediga cuál es el valor que nos falta aprendiendo de los casos en los que sí tenemos datos.
- No hacer nada y usar alguna técnica de machine learning que pueda manejar datos incompletos.

Combinar datos de varias fuentes

Casi siempre tenemos que combinar datos de diferentes fuentes. Algunos datos pueden venir de una base de datos, otros de una hoja de cálculo, de ficheros, etc. Tenemos que combinar los datos de forma que los algoritmos de machine learning puedan considerar toda la información.

Darle el formato adecuado a los datos

Si queremos usar librerías de machine learning que ya están disponibles, tenemos que darle formato a nuestros datos. En general, estas librerías esperan que los datos tenga forma de [matriz](#) o de [tensor](#). Un tensor es una generalización de una matriz. Si la matriz tiene 2 dimensiones, el tensor tiene un número «n» de dimensiones.

Calcular características relevantes (features)

Los algoritmos de machine learning funcionan mucho mejor si le ofrecemos características relevantes en vez de los datos puros. Por poner un ejemplo, a nosotros nos resulta mucho más fácil saber la temperatura en grados Celsius que saber cuánto se han dilatado tantos miligramos de mercurio en un termómetro tradicional. De la misma forma, es muy útil transformar los datos para hacer la tarea de aprendizaje más fácil.

La fase de calcular características relevantes («features»en inglés) requiere un esfuerzo mayor. Hay que pensar en qué características van a ser más relevantes para solucionar el problema y probarlo. Algunas veces la característica que hemos creado no funciona tan bien como hubiéramos pensado... pero otra muchas veces sí. Aquí la experiencia es un grado. Cuantos más problemas de machine learning resolvamos, no volveremos mejores creando características.

Normalización de datos

En muchos casos, es útil normalizar los datos para hacerle más fácil a la técnica de machine learning el aprendizaje. Por normalizar nos referimos a poner a todos los datos en una escala similar. Hay varias formas de normalizar los datos, que ya veremos en otro artículo.

Fase 5: Construir el modelo

La fase de construir un modelo de machine learning, una vez que tengamos los datos preparados, requiere sorprendentemente poco esfuerzo. Esto es así porque ya existen varias [librerías de machine learning](#) disponibles. Muchas de ellas son gratuitas y de código abierto.

Durante esta fase, tenemos que elegir que tipo de técnica de machine learning queremos usar. El algoritmo de machine learning aprenderá automáticamente a obtener los resultados adecuados con los datos históricos que hemos preparado. Eso sí, tendrá un error.

Fase 6: Análisis de Errores

La fase del análisis de errores requiere un esfuerzo relativo medio. Analizar errores es importante para entender qué es lo que tenemos que hacer para mejorar los resultados de machine learning. En particular las opciones serán:

- usar un modelo más complejo
- usar un modelo más simple
- darnos cuenta de que necesitamos más datos y / o más características

- desarrollar una mejor comprensión del problema y entender mejor cuál es el siguiente paso a dar

En la fase de análisis de errores intentaremos asegurarnos que nuestro modelo es capaz de generalizar. La generalización es la capacidad que tienen los modelos de machine learning de producir buenos resultados cuando usan datos nuevos.

En general, no es difícil conseguir resultados aceptables usando este proceso. Sin embargo, si queremos obtener resultados realmente buenos, deberemos iterar sobre las fases anteriores varias veces. Con cada iteración, nuestro entendimiento del problema y de los datos será cada vez mayor. Esto hará que podamos diseñar mejores características relevantes y reducir el error de generalización. Un mayor entendimiento también nos ofrecerá la posibilidad de elegir con más criterio la técnica de machine learning que más se ajuste al problema.

Casi siempre, tener más datos ayuda. En la práctica, más datos y un modelo simple tiende a funcionar mejor que un modelo complejo con menos datos.

Puedes consultar el artículo Análisis de Errores en Machine Learning para profundizar más en este tema.

Fase 7: Modelo integrado en un Sistema

Una vez que estemos satisfechos con el error, tenemos que compararlo con el error de la solución actual. Si es lo suficientemente mejor, integraremos el modelo del machine learning en nuestro sistema.

La fase de integrar un modelo de machine learning en un sistema requiere un esfuerzo relativo mayor. Esto es así porque necesitamos:

- poder repetir de forma automática las fases de preparación de datos
- lo que requiere que el modelo de machine learning se comunique con otras partes del sistema
- y que los resultados del modelo se usen en el sistema
- además, debemos monitorizar automáticamente los errores del modelo
- y avisar si los errores del modelo crecen con el tiempo
- para re-construir el modelo de machine learning con nuevos datos, ya sea manual o automáticamente

Una parte considerable del esfuerzo va a construir *interfaces* de datos. Estos interfaces son necesario para que el modelo pueda obtener datos automáticamente y para que el sistema pueda usar su predicción de forma automática. Aunque es un esfuerzo considerable, es esencial. Para que el machine learning y la inteligencia artificial sean útiles, en la mayoría de los casos deben integrarse en un sistema mayor. Así que la cuestión no es tanto *¿qué puede hacer el machine learning?* sino *¿qué puedo hacer con el machine learning?*

Ejemplo de modelo integrado

El [sistema de traducción de Google](#) no sería muy útil si Google sólo hubiese construido un modelo de machine learning. Esto estaría muy bien como estudio académico pero no tendría valor comercial. El verdadero valor está en que el modelo está integrado en un sistema web en la nube. Y este sistema permite a millones de personas usar este modelo de machine learning para traducir frases

Recursos

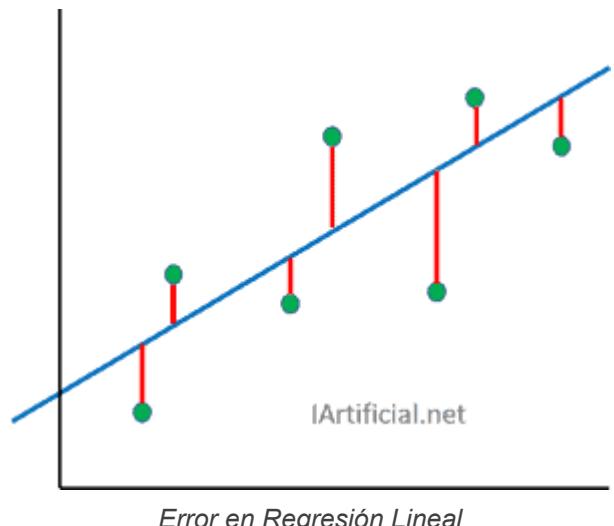
- [vídeo](#) introductorio a la Inteligencia Artificial y al Aprendizaje Automático (en inglés)
- [Empresas de Inteligencia Artificial](#) que pueden ayudarte a seguir este proceso en tu negocio

Error Cuadrático Medio para Regresión

Por Jose Martinez Heras
28/12/2018

El Error Cuadrático Medio es el [criterio de evaluación](#) más usado para problemas de regresión. Se usa sobre todo cuando usamos aprendizaje automático supervisado. Para cada dato histórico podremos indicar el resultado correcto. Vamos a ver como se calcula.

Cálculo del Error Cuadrático Medio



Vamos a calcular el error cuadrático medio con un ejemplo. En la figura vemos que estamos usando una regresión lineal (en azul) para estimar los datos que tenemos (los puntos verdes). El modelo lineal tiene un error (en rojo) que podemos definir con la siguiente fórmula:

$$\text{error cuadrático} = (\text{real} - \text{estimado})^2$$

El valor estimado es el valor que nos da el modelo. En este caso, la línea azul.

Calculamos el error al cuadrado, en lugar del error simple, para que el error siempre sea positivo. De esta forma sabemos que el error perfecto es 0. Si no elevásemos el error al cuadrado, unas veces el error sería positivo y otras negativo. Otra posibilidad sería usar el valor absoluto, en lugar de elevarlo al cuadrado.

cuadrado. Sin embargo, si usamos el valor absoluto, obtendremos una función no-derivable. Y como ya veremos en otro artículo, tener una función derivable hace posible el uso de algoritmos de optimización muy efectivos (por ejemplo, el gradiente descendente).

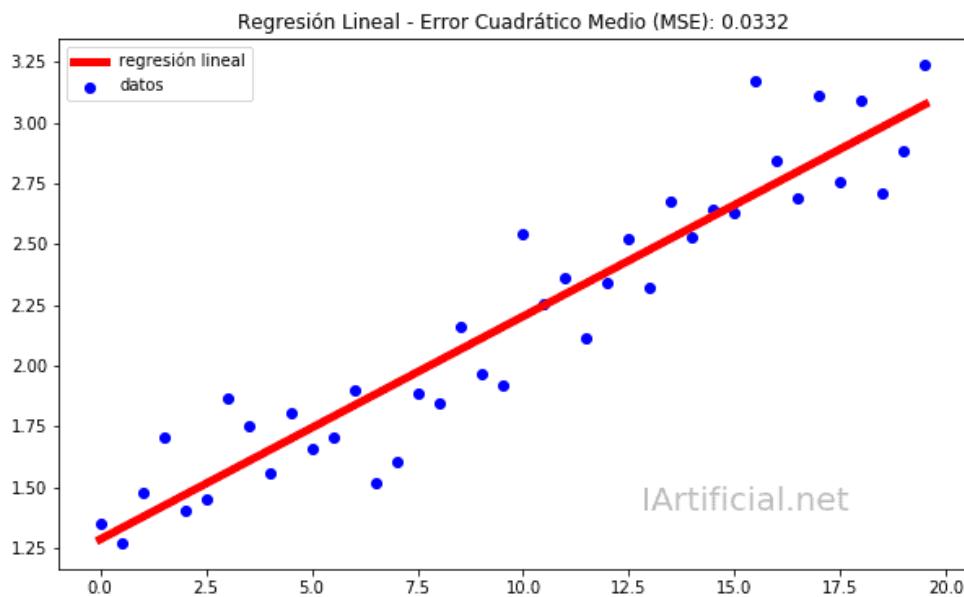
Ahora que sabemos cómo calcular el error en cada punto, podemos calcular cual es el error medio. Para ello, sumamos todos los errores y los dividimos entre el número total de puntos. Si llamamos M al número total de puntos nos queda la fórmula del Error Cuadrático Medio (MSE, por sus siglas en inglés, [Mean Squared Error](#)):

$$MSE = \frac{1}{M} \sum_{i=1}^M (\text{real}_i - \text{estimado}_i)^2$$

Ejemplo del MSE en una Regresión Lineal

Vamos a ver cómo funciona el MSE en un ejemplo. Como técnica de machine learning, vamos a usar una de las más simples, la regresión lineal.

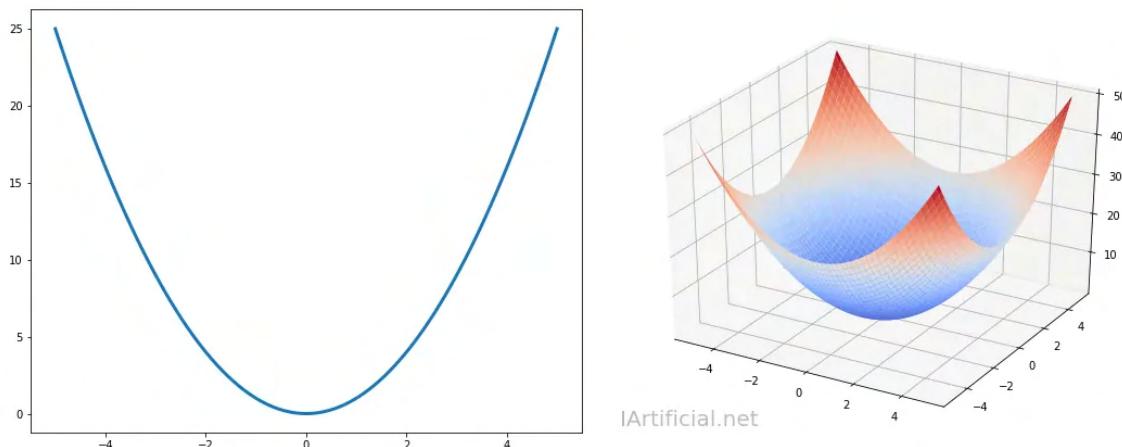
En el gráfico podemos ver que hemos usado una regresión lineal para calcular la línea (en rojo). Esta línea obtiene el menor MSE posible para los datos que hemos usado (en azul). El MSE (por sus siglas en inglés) es de 0.0332.



Ejemplo de Error Cuadrático Medio (MSE en inglés)

La forma del error cuadrático medio

Podemos apreciar la forma del MSE en la siguiente gráfica. A la izquierda, vemos el MSE para una variable (en 2D), y a la derecha el MSE para dos variables (en 3D). Es fácil ver que el MSE amplifica los errores mayores ya que calcula el cuadrado del error.



Raíz Cuadrada del Error Cuadrático Medio

El error cuadrático medio no es del todo intuitivo porque nos da el error medio al cuadrado. Así que si nuestro modelo para predecir el valor de bienes inmuebles tiene un error cuadrático medio de 1.000.000 de euros nos daría la impresión de que tiene mucho error. En realidad, el error sería más / menos 1.000 euros en media, porque la raíz cuadrada de 1.000.000 es 1.000. La fórmula sería:

$$RMSE = \sqrt{\frac{1}{M} \sum_{i=1}^M (real_i - estimado_i)^2}$$

Os pongo su nombre también en inglés para que os suene cuando lo uséis en las librerías de machine learning: RMSE (Root Mean Squared Error). Así, en el ejemplo anterior, el RMSE sería 0.1822 (para un MSE de 0.0332).

¿Por qué no usamos la Raíz Cuadrada del Error Cuadrático Medio?

Entonces, si el RMSE es más intuitivo que el MSE, ¿por qué no usamos el RMSE todo el tiempo? Hay dos razones:

1. Es más costoso computacionalmente. Muchos algoritmos supervisados necesitan calcular el error en cada iteración para aprender de sus errores; así que cuanto más rápido mejor.
2. La forma del error cuadrático medio hace que cierto tipos de algoritmos de optimización (tales como el gradiente descendente), encuentre la mejor solución más rápido.

Así que, normalmente, se suele usar el MSE durante el proceso de aprendizaje y su raíz cuadrada al final, para dar una estimación en términos intuitivos de la calidad de la predicción.

Cálculo del Error Cuadrático Medio en python

Vamos a ver dos formas de calcularlo en python

Usando scikit-learn

Lo más fácil es usar la librería de python scikit-learn. El modulo `sklearn.metrics` implementa la función `mean_squared_error`. Esta función necesita dos argumentos:

- `y_true`: los valores correctos
- `y_pred`: nuestra predicción

Por ejemplo:

```
1. from sklearn.metrics import mean_squared_error
2. y_true = [3, -0.5, 2, 7]
3. y_pred = [2.5, 0.0, 2, 8]
4. mse = mean_squared_error(y_true, y_pred)
5. mse
6. # resultado: 0.375
```

Usando sólo NumPy

```
1. import numpy as np
2. y_true = [3, -0.5, 2, 7]
3. y_pred = [2.5, 0.0, 2, 8]
4. mse = (np.square(y_true - y_pred)).mean()
5. mse
6. # resultado: 0.375
```

El Error Cuadrático Medio es una métrica de **regresión**. El artículo **Precision, Recall, F1, Accuracy en clasificación** explica las métricas de **clasificación**.

Resumen

El error cuadrático medio es el criterio de evaluación más usado para problemas de aprendizaje supervisado de regresión. En este artículo hemos visto cómo se calcula, la posibilidad de usar la

raíz cuadrada para obtener un resultado más intuitivo y cómo calcularlo fácilmente con código python.

Recursos

- Función para el cálculo el [mean squared error](#) en python en scikit-learn
- Cómo se usa el error cuadrático medio y su derivada en el aprendizaje automático con gradiente descendiente
- Métricas para problemas de aprendizaje supervisado de clasificación tales como precision, recall, F1, accuracy
- Otras librerías de python para machine learning que te pueden servir

Generalización en Machine Learning

Por Jose Martinez Heras
04/01/2019

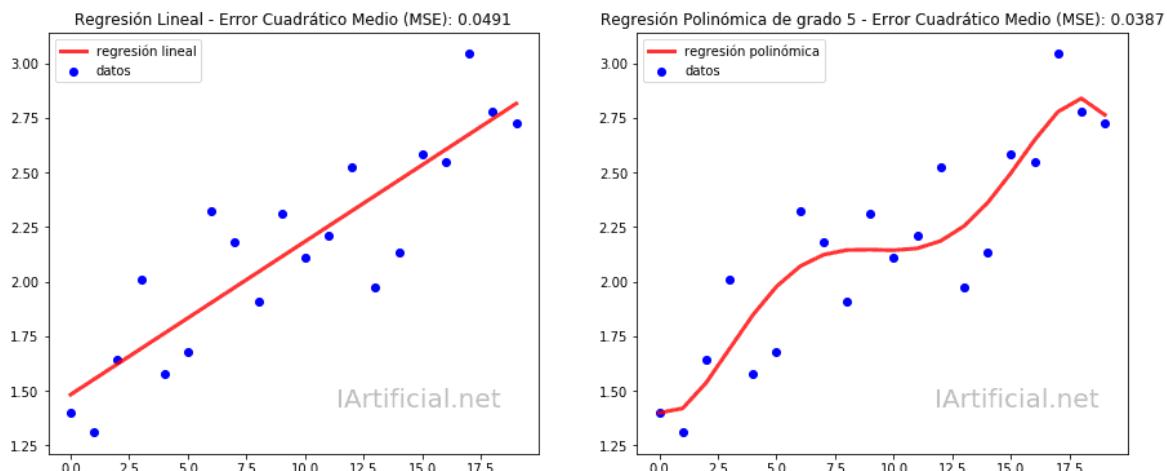
En Machine Learning, la **Generalización** es la capacidad de obtener buenos resultados con datos nuevos. En este artículo vamos a hablar de por qué es tan importante la generalización y cómo medirla.

Intuición gráfica de la Generalización en Machine Learning

Para explicar el concepto de generalización, vamos a intentar usar un ejemplo gráfico y usar el [error cuadrático medio](#). Una vez que tengamos la intuición gráfica, será más fácil describir matemáticamente cómo podemos medir la generalización.

He generado unos datos siguiendo una línea y les he añadido un poco de ruido. A continuación, he usado 2 modelos de regresión:

- El de la izquierda es un modelo de regresión lineal. Tiene un Error Cuadrático Medio de 0.0491
- El de la derecha es un modelo de regresión polinómica de grado 5. Tiene un Error Cuadrático Medio de 0.0387

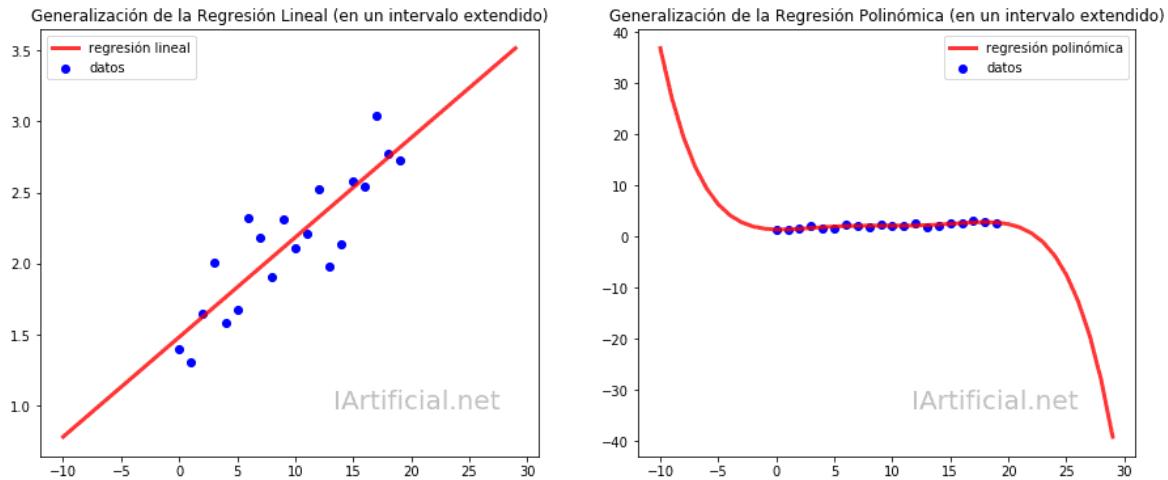


Regresión Lineal y Polinómica: la regresión polinómica tiene un menor error que la regresión lineal

Así que como el error de la regresión polinómica es menor que el error de la regresión lineal ... deberíamos usar la regresión polinómica, ¿no? Intuitivamente esto no parece lo más correcto.

Para entenderlo intuitivamente, vamos a ver cómo se comportan la regresión lineal y la regresión polinómica en un intervalo extendido. Es decir, vamos a visualizar las predicciones de estos dos modelos cuando tenemos datos nuevos.

Como veis, la regresión polinómica ¡es un desastre! Toma valores muy extremos para valores nuevos.



La regresión lineal generaliza mucho mejor que la regresión polinómica de grado 5

Espero que veais claro que construir un modelo de machine learning que se ajuste muy bien a los datos, no siempre garantiza que el modelo sea útil. Por eso buscamos un modelo que generalice bien. Con generalizar nos referimos a que el modelo funcione bien, no sólo con los datos con los que ha aprendido, sino también con los datos que obtendremos en el futuro.

¿Cómo podemos medir la Generalización de un modelo?

Hemos visto que si hacemos gráficos de los datos, podemos entender cuándo un modelo no está generalizando correctamente. Esto es fácil porque, en el ejemplo anterior, sólo teníamos una dimensión. Tenemos los siguientes casos:

- Cuando tenemos una dimensión (x), podemos hacer un gráfico en 2 dimensiones y ver cómo se comporta el resultado (y)
- Cuando tenemos 2 dimensiones, podemos hacer un gráfico en 3 dimensiones.
- Pero cuando tenemos 3 o más dimensiones ... ya no podemos visualizar cómo se comporta el modelo.

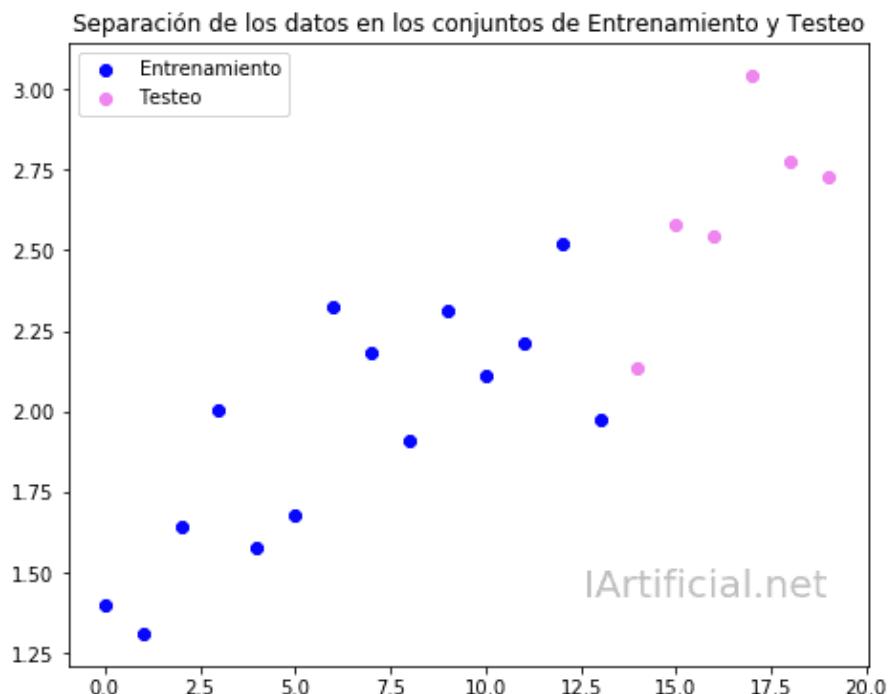
En machine learning, es muy común tener más de 3 dimensiones. Por ejemplo, si queremos predecir si un cliente de un banco aceptará una oferta de un plazo fijo en una llamada promocional, podemos considerar: la edad (1), qué trabajo tiene (2), el balance de su cuenta (3), qué otros productos tiene ya (4), la ciudad (5), cuánto tiempo lleva siendo cliente del banco (6), estado civil (7), etc. Como veis, es muy muy fácil tener más 2 dimensiones.

Necesitamos una forma de medir la generalización que no necesite hacer gráficos. Hay varias formas de hacerlo. Vamos a explicar las dos técnicas más populares para medir la generalización:

- Con conjuntos de entrenamiento y de testeo.
- Con validación cruzada

Conjuntos de Entrenamiento y Testeo

Una forma de medir la generalización es la de dividir los datos que tengamos en dos conjuntos: el conjunto de entrenamiento y el conjunto de testeo. En el gráfico, vemos los datos del conjunto de entrenamiento en azul, y en rosa los de testeo.



Separación de datos en los conjuntos de Entrenamiento y Testeo

Conjunto de Entrenamiento

El conjunto de entrenamiento contiene los datos que el modelo de machine learning usará para aprender. El conjunto de entrenamiento es mayor que el conjunto de testeo, en el sentido de que contiene más datos.

Conjunto de Testeo

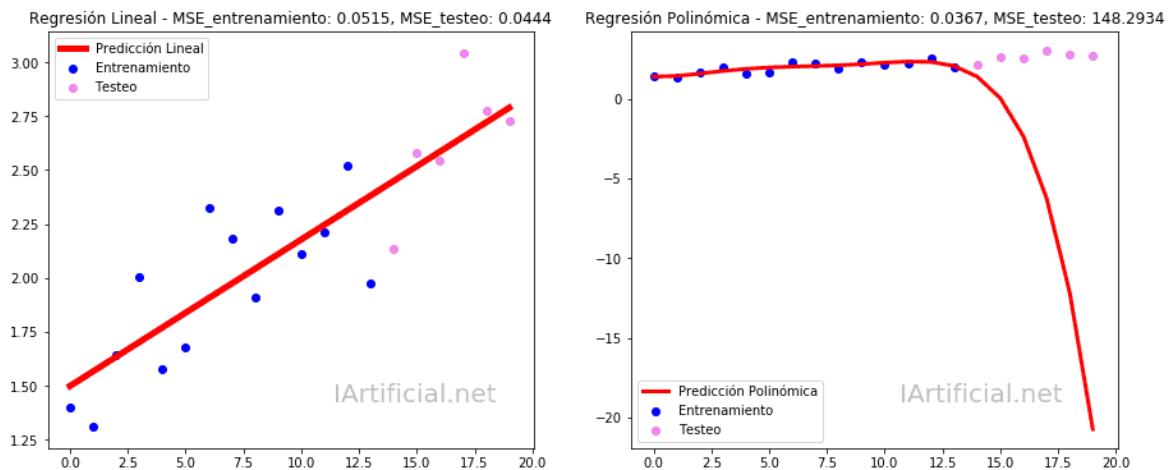
El conjunto de testeo **no se usa en el aprendizaje**. Usaremos el conjunto de testeo para saber cómo se comporta el modelo con datos nuevos.

No son nuevos realmente, porque ya teníamos estos datos. Pero son «nuevos» para el modelo porque nos hemos asegurado de ocultárselos durante el aprendizaje.

Debemos evitar que el modelo use estos datos en el aprendizaje. Cuando esto sucede accidentalmente, recibe el nombre de «fuga de datos». El efecto que tiene es que podemos pensar que nuestro modelo es mejor de lo que en realidad es.

Medición de la capacidad de generalización con conjuntos de entrenamiento y testeo

Para medir la generalización, medimos el error del modelo en el conjunto de testeo. Es decir, usamos los datos de entrenamiento para que el modelo aprenda y luego le pedimos al modelo que nos de los resultados correspondientes a los datos de testeo. Como sabemos los resultados que esperamos de los datos de testeo, podemos medir el error de predicción. Para medir el error podemos usar el [error cuadrático medio](#) (MSE por sus siglas en inglés), u otra medida apropiada al problema.



Medimos la Generalización con el Error en el Conjunto de Testeo

Así podemos ver que en el caso de la regresión lineal, tanto el error de entrenamiento (0.0514) como el de testeo (0.0444), son pequeños y similares. Sin embargo, en el caso de la regresión polinómica de grado 5, el error de testeo (148.2934) es muy grande en comparación al error de entrenamiento (0.0367).

Podemos estimar el error de generalización midiendo el error en el conjunto de testeo.

Midiendo el error en el conjunto de testeo, podemos estimar cuál es el rendimiento del modelo en datos nuevos. Esto nos permite experimentar con varios modelos y elegir el que mejor generaliza. Para esto ya veremos que es mejor usar 3 conjuntos de datos: entrenamiento, *validación* y testeo. Hablaremos del conjunto de validación en otro artículo.

Validación cruzada

La validación cruzada estima el error de generalización de forma más robusta. Cuando usamos los conjuntos de entrenamiento y de testeo, hay una parte aleatoria que puede influir los resultados. Dependiendo cómo dividamos los datos en estos dos conjuntos tendremos una estimación diferente del error de generalización. En unos casos pensaremos que el modelo generaliza mejor y en otros peor.

Para combatir este problema, la validación cruzada propone crear los conjuntos de entrenamiento y testeo varias veces, cada vez con una separación diferente. De esta forma obtendremos estimaciones diferentes. La media aritmética de todos los errores de testeo se considera la estimación del error de generalización.

Es una estimación más robusta. El inconveniente es que es computacionalmente más costoso. La solución de compromiso es usar validación cruzada pero limitar el número de veces que la hacemos. Típicamente es 5 veces.

Recursos

- [\[vídeo\]](#) introductorio a la Inteligencia Artificial y al Aprendizaje Automático (en inglés)

Análisis de Errores en Machine Learning

Por Jose Martinez Heras
11/01/2019

El análisis de errores es una de las fases del proceso de machine learning más importantes. El análisis de errores nos va a permitir saber qué hacer para mejorar el rendimiento de un modelo de machine learning.

Para analizar errores, nos vamos a concentrar en los errores de entrenamiento y los errores de generalización. En particular:

- Nos aseguraremos que el modelo de machine learning sea capaz de aprender. Para ello procuraremos que el error de entrenamiento sea bajo.
- Intentaremos que el modelo de aprendizaje automático sea capaz de generalizar. Para ello buscaremos que el error de generalización sea bajo.

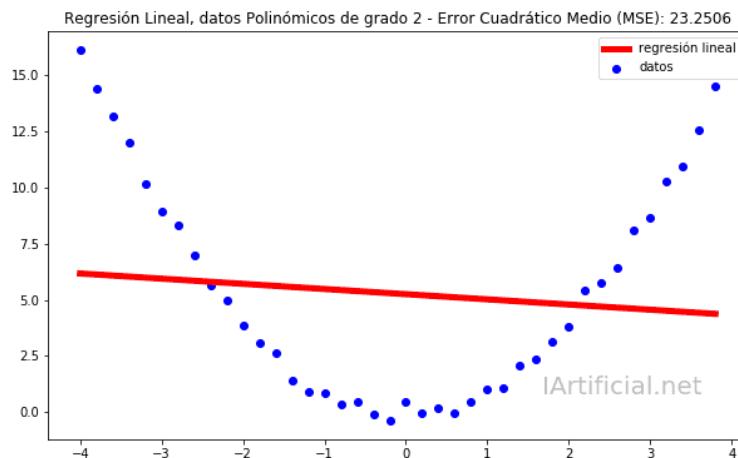
Analizando los Errores de Entrenamiento

Si el error de entrenamiento es alto, el modelo de machine learning tiene problemas para aprender. Esto se denomina *underfitting* en inglés. En español se podría llamar *infraajuste*.

Causas del *underfitting* (*infraajuste*)

Modelo demasiado simple

En el gráfico a continuación podemos ver un ejemplo de *underfitting*. Los datos siguen una función polinómica de grado 2 (una parábola). Si intentamos usar un modelo de regresión lineal, veremos que el modelo no se ajusta bien a los datos. En otras palabras, el modelo es tan simple, que no es capaz de aprender la relación entre los datos de entrada y los de salida.



El alto error de entrenamiento indica que el modelo de machine learning es demasiado simple. La regresión lineal no tiene capacidad para aprender los datos de una parábola.

Datos incompletos

Otro motivo para el *underfitting* (*infraajuste*) puede ser la falta de datos. Por ejemplo, si los resultados dependen de dos variables $y = a + b$. Si intentamos aprender sólo con uno de ellos (por

ejemplo, predecir «y» sabiendo sólo «a»), habrá muchos casos en los que el rendimiento será malo.

Un ejemplo puede ser marketing. Si intentamos predecir quién comprará una crema «anti-arrugas» y olvidamos incluir la edad del cliente en la entrada del modelo.

Atributos relevantes (features) insuficientes

En ocasiones, tenemos todos los datos necesarios para un buen rendimiento del modelo de machine learning. Sin embargo, necesitamos transformar los datos para que el modelo pueda entender mejor la relación entre la entrada y la salida. En el caso de que tengamos pocos datos, la creación de atributos relevantes, es todavía más importante.

Por ejemplo, en un termómetro analógico, el dato sería cuánto se ha expandido tantos miligramos de mercurio. El atributo relevante sería cuántos grados Celsius marca el termómetro.

Al crear atributos relevantes estamos ayudando al modelo con nuestro conocimiento del problema.

Datos aleatorios, caóticos o con demasiado ruido

Una de las condiciones para que merezca la pena usar Machine Learning, es que debe haber un patrón entre los datos de entrada y los resultados. Así que no podemos usar machine learning para predecir datos aleatorios. Si lo intentamos, obtendremos un error de entrenamiento muy alto.

Si los datos son caóticos, también va a ser muy difícil obtener un error de entrenamiento bajo. Los [sistemas caóticos](#) son muy sensibles a las condiciones iniciales. Tanto es así que aunque el sistema sea determinista, pequeños cambios en las condiciones iniciales hacen que su comportamiento futuro sea tan diferente que sea impredecible. Por ejemplo, la bolsa de valores tiene un comportamiento caótico. A efectos prácticos de machine learning, podemos considerar los datos de sistemas caóticos como aleatorios.

Si los datos tienen mucho ruido las técnicas de machine learning van a tener problemas para aprender. Podemos intentar reducir el ruido, o aumentar la señal respecto al ruido. Si no lo conseguimos, el algoritmo de aprendizaje automático «pensará» que los datos son aleatorios.

Datos insuficientes

Otra de las condiciones para que merezca la pena usar Machine Learning, es que la cantidad de datos debe ser suficiente. Si hay pocos datos, es difícil encontrar un patrón entre los datos de entrada y los resultados.

Buen error de entrenamiento

Cuando tenemos un buen error de entrenamiento, podemos decir que el modelo ha sido capaz de aprender la relación entre los datos de entrada y los resultados.

Para poder saber si el modelo ha aprendido correctamente tendremos que analizar el error de generalización.

Analizando los Errores de Generalización

En general, no merece la pena analizar los errores de generalización, hasta tener un buen error de entrenamiento

Si sólo tuviésemos en cuenta el error de entrenamiento no podríamos saber si el modelo ha aprendido «de memoria» o «entendiéndolo». Cuando digo «entendiéndolo» no quiero decir que el modelo «entienda» de la misma forma que nosotros los humanos entendemos. Quiero decir que en lugar de aprenderlo «de memoria», el modelo ha aprendido el patrón entre los datos de entrada y los resultados.

Decimos que el modelo de machine learning ha aprendido «de memoria» cuando:

- el error de entrenamiento es bajo
- el error de generalización es alto

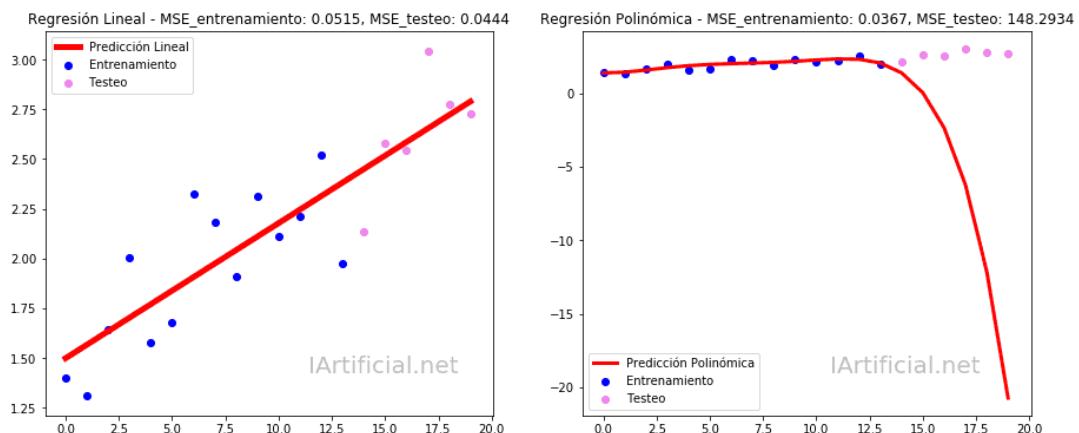
El nombre técnico para «aprendizaje de memoria» es *overfitting* en inglés. En español podríamos llamarlo *sobreajuste*.

Causas del *overfitting* (sobreajuste)

Modelo demasiado complejo

Si el modelo es demasiado complejo, podrá aprender muchos de los datos de memoria. Para evitarlo, podemos usar un modelo más simple.

El gráfico muestra los mismos datos usando dos modelos diferentes: una regresión lineal y una regresión polinómica de grado 5. La regresión polinómica es un modelo muy complejo para el problema que tenemos. Por eso, el error de entrenamiento es bajo y el error de generalización es tan alto. Por otra parte, la regresión lineal tiene un error bajo tanto de aprendizaje como de generalización.



Al medir los errores de generalización podemos distinguir si el modelo ha aprendido los datos «de memoria» o encontrando un patrón entre los datos de entrada y los resultados

Regularización

Otra opción a la hora de hacer un modelo más simple, es usar un modelo complejo y usar regularización. La regularización, en machine learning, es el proceso de limitar, a propósito, la capacidad del modelo. La regularización es una forma de conseguir hacer más simple un modelo complejo.

Demasiados pocos datos

Si la técnica de aprendizaje automático ha aprendido con pocos datos, es más fácil que haya aprendido qué resultado va con cada dato de entrada de memoria. Tener más datos, hace más difícil que se pase el filtro de tener un error de aprendizaje bajo.

Los datos nuevos proceden de otra distribución

También puede ocurrir, que estemos intentando medir la generalización con datos que venga de otra distribución.

Por ejemplo, supongamos que estamos haciendo un modelo para estimar la probabilidad de que un cliente cancele un contrato. Para estimar el error de generalización, podemos dividir los datos

en los conjuntos de entrenamiento y testeo. Para tener una mejor idea del rendimiento real, deberemos asegurarnos de que están bien distribuidos.

Una mala distribución sería construir el modelo con datos de los clientes de Francia, Alemania, Bélgica y Holanda; y usar los clientes de España para estimar la generalización. Una mejor partición sería elegir un 80% de los clientes de todos los países, y estimar la generalización con el 20% restante.

Buen error de generalización

Cuando tenemos un buen error de generalización, podemos decir que el modelo de machine learning ha aprendido el patrón que hay entre los datos de entrada y los resultados. También podemos confiar en que el modelo va a funcionar bien con datos nuevos. Al menos, siempre que los datos nuevos vengan de una distribución similar.

Conjunto de validación

Hasta ahora hemos medido el error de entrenamiento y el error de generalización con los conjuntos de entrenamiento y de testeo. Hemos usado el error de generalización para saber si el modelo estaba aprendiendo de memoria o realmente encontrando el patrón entre los datos de entrada y los resultados. Analizando los datos de generalización, podemos decidir que es mejor usar un modelo más complejo, o más simple (ya sea regularizando un modelo complejo o usando uno más simple).

El problema es que estamos haciendo una pequeña «trampa». Si alguien nos pregunta, ¿cuál es el error de generalización del modelo de aprendizaje automático? ... realmente no lo sabemos. No lo sabemos, porque hemos usado el conjunto de testeo para decidir con qué modelo nos quedamos. En otras palabras, nos hemos quedado con el modelo que menor error de generalización tenía en el conjunto de testeo. Es decir, si diéramos una estimación del error, la estimación sería muy optimista.

Nueva partición del conjunto de datos

Para solventar este problema, en vez de dividir los datos solo en los conjuntos de entrenamiento y testeo, vamos a dividirlos en tres conjuntos:

- Entrenamiento: el modelo de machine learning aprenderá exclusivamente usando los datos de entrenamiento
- Validación: usaremos los datos de validación para elegir qué modelo funciona mejor con los datos, para configurar los hiperparámetros (regularización), etc.
- Testeo: usaremos el conjunto de testeo al final del todo, una vez que el modelo y la regularización estén elegidos. El conjunto de testeo nos permitirá estimar el error de generalización de cara a un uso real.

¿Cómo analizar errores en Machine Learning?

Aquí tienes una guía rápida que resume cómo analizar los errores en el proceso de Machine Learning y qué hacer en cada caso.

Asegúrate que el error de entrenamiento es bajo

Si el error de entrenamiento es alto, significa que el modelo no puede aprender la relación entre los datos de entrada y los resultados.

Si el error de entrenamiento es alto, mejora la calidad de los datos

Intenta conseguir más datos y añade atributos relevantes (features). Comprueba que los datos nos sean aleatorios.

Si el error de entrenamiento sigue siendo alto, usa un modelo de machine learning más complejo.

Puede ser que el patrón entre los datos de entrada y los resultados, sólo pueda aprenderse con un modelo más complejo

Comprueba que el modelo sea capaz de generalizar

Para que el modelo de machine learning sea útil, debe funcionar bien con datos nuevos. Usamos el conjunto de validación para estimar la capacidad de generalización.

Si el error de generalización es alto prueba con un modelo más simple

Un error de entrenamiento bajo y un error de generalización alto indica que el modelo ha aprendido los datos «de memoria». Usa un modelo más simple para forzar a la inteligencia artificial a «entender» lo que está haciendo.

Usa el conjunto de testeo para estimar el error del modelo de machine learning

Para estimar el error de generalización en los pasos 4 y 5, hemos usado el conjunto de validación. Reservamos el conjunto de testeo para estimar de una forma más realista cómo se comportará el modelo de machine learning aprendido con datos nuevo.

Recursos

- [\[vídeo\]](#) introductorio a la Inteligencia Artificial y al Aprendizaje Automático (en inglés)

15 Librerías de Python para Machine Learning

Por Jose Martinez Heras
18/01/2019

Estas son las 15 mejores librerías de [python](#) para Machine Learning. Temas: Visualización, Cálculo Numérico, Análisis de Datos, Aprendizaje Automático, Deep Learning, Inteligencia Artificial Explicable, Procesamiento del Lenguaje Natural y mucho más.

Todas las librerías de python que vamos a ver son gratuitas.

Librerías de Python para Visualización

Una de las fases del proceso de Machine Learning más importantes es [entender el problema](#) que vamos a resolver. Una forma que tenemos de mejorar nuestra comprensión del problema es [entender mejor los datos](#). La visualización de datos nos ayuda a entender mejor tanto los datos y como el problema.

Así mismo, la visualización de datos será también muy útil para comprender los resultados y analizar los errores. Aunque hay muchas librerías en python para la visualización de datos, nos vamos a concentrar en: matplotlib, seaborn y bokeh por el momento.

Matplotlib



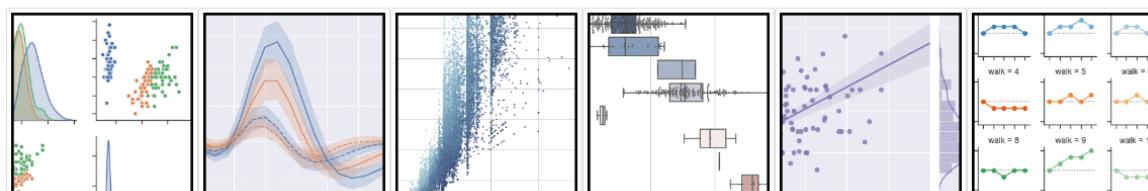
[Matplotlib](#) es la librería gráfica de python estándar y la más conocida. Puedes usar matplotlib para generar gráficos de calidad necesaria para publicarlas tanto en papel como digitalmente.

Con matplotlib puedes crear muchos tipos de gráficos: series temporales, histogramas, espectros de potencia, diagramas de barras, diagramas de errores, etc.

Si quieres ver de lo que matplotlib es capaz, mira sus [gráficos de ejemplo](#) y su [galería de gráficos](#).

Seaborn

seaborn: statistical data visualization

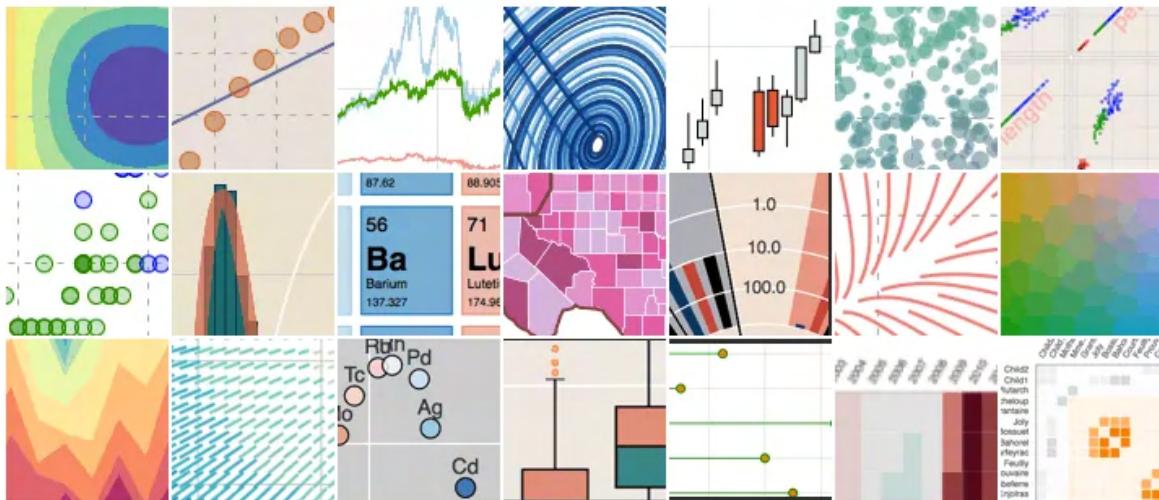


[Seaborn](#) es una librería gráfica basada en matplotlib, especializada en la visualización de datos estadísticos. Se caracteriza por ofrecer un interfaz de alto nivel para crear gráficos estadísticos visualmente atractivos e informativos.

Seaborn considera la visualización como un aspecto fundamental a la hora de explorar y entender los datos. Se integra muy bien con la librería de manipulación de datos [pandas](#).

Si quieres ver de lo que seaborn es capaz, mira su [galería de ejemplos](#). También ofrecen un [tutorial](#) en inglés.

Bokeh



[Bokeh](#) es una librería para visualizar datos de forma interactiva en un navegador web. Con bokeh podemos crear gráficos versátiles, elegantes e interactivos. Los desarrolladores de bokeh buscan un buen rendimiento con gran cantidad de datos, incluso con datos que vayan llegando en tiempo real.

Si quieres ver de lo que bokeh es capaz, mira su [galería de ejemplos](#). También puedes consultar su [manual de usuario](#) en inglés.

Librerías de Python para Cálculo Numéricico y Análisis de Datos

Otra de las fases del proceso de Machine Learning que más tiempo consumen es la [preparación de datos](#) y el [cálculo de atributos relevantes o características \(features\)](#). NumPy, SciPy y Pandas son las librerías de python ideales para análisis de datos y computación numérica.

Seguramente también nos enfrentaremos a problemas que no requieren el uso de aprendizaje automático sino sólo el análisis de datos. Por supuesto, también podemos usar estas librerías en estos casos.

NumPy



[NumPy](#) proporciona una estructura de datos universal que posibilita el análisis de datos y el intercambio de datos entre distintos algoritmos. Las estructuras de datos que implementa son vectores multidimensionales y matrices con capacidad para gran cantidad de datos.

Además, esta librería proporciona funciones matemáticas de alto nivel que operan en estas estructuras de datos. Para aprender más, sigue el [tutorial de NumPy](#) (en inglés).

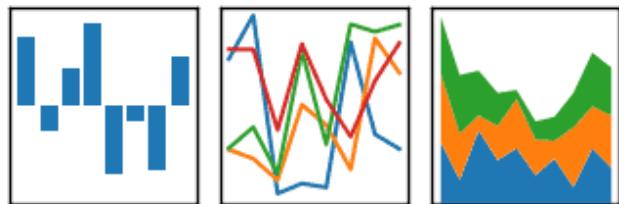
SciPy

[SciPy](#) proporciona rutinas numéricas eficientes fáciles de usar y opera en las mismas estructuras de datos proporcionadas por NumPy. Por ejemplo, con SciPy puedes realizar: integración numérica, optimización, interpolación, transformadas de Fourier, álgebra lineal, estadística, etc. Para aprender más, sigue el [tutorial de SciPy](#) (en inglés).

Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



[Pandas](#) es una de las librerías de python más útiles para los científicos de datos. Las estructuras de datos principales en pandas son *Series* para datos en una dimensión y *DataFrame* para datos en dos dimensiones.

Estas son las estructuras de datos más usadas en muchos campos tales como finanzas, estadística, ciencias sociales y muchas áreas de ingeniería. Pandas destaca por lo fácil y flexible que hace la manipulación de datos y el análisis de datos.

Para aprender más, puedes mirar la [documentación de pandas](#) (en inglés).

Numba



[Numba](#) traduce funciones escritas en python to código máquina optimizado a la hora de ejecutarse. Lo consigue usando el estándar industrial [LLVM](#) como librería de compilación. Los algoritmos numéricos compilados con Numba pueden alcanzar velocidades de ejecución tan altas como las de C o FORTRAN.

Así que si te interesa optimizar la velocidad de tu código, no tienes por qué compilar código por separado, ni tan siquiera necesitas tener el compilador de C/C++ instalado. Sólo aplica uno de los decoradores de Numba a tu función de python y Numba hará el resto.

Para saber más, puedes consultar la [documentación de Numba](#) (en inglés)

Librerías de Python para Machine Learning

Cuando vimos las fases del proceso de Machine Learning, vimos que [construir el modelo de Machine Learning](#) consumía relativamente poco tiempo. Esto es así porque ya existen librerías de aprendizaje automático. Hay varias, Scikit-Learn es la más utilizada.

scikit-learn



[scikit-learn](#) es una librería de python para Machine Learning y Análisis de Datos. Está basada en NumPy, SciPy y Matplotlib. La ventajas principales de scikit-learn son su facilidad de uso y la gran cantidad de técnicas de aprendizaje automático que implementa.

Con scikit-learn podemos realizar aprendizaje supervisado y no supervisado. Podemos usarlo para resolver problemas tanto de clasificación y como de regresión.

Es muy fácil de usar porque tiene una interfaz simple y muy consistente. El interfaz es muy fácil de aprender. Te das cuenta que el interfaz es consistente cuando puedes cambiar de técnica de machine learning cambiando sólo una línea de código.

Otro punto a favor de scikit-learn es que los valores de los hiper-parámetros tienen unos valores por defecto adecuados para la mayoría de los casos.

Estas son algunas de las técnicas de aprendizaje automático que podemos usar con scikit-learn:

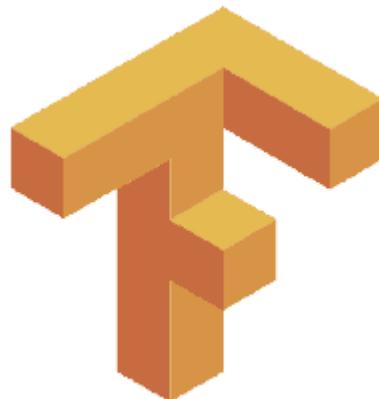
- regresión lineal y polinómica
- regresión logística
- máquinas de vectores de soporte
- árboles de decisión
- bosques aleatorios (random forests)
- agrupamiento (clustering)
- modelos basados en instancias
- clasificadores bayesianos
- reducción de dimensionalidad
- detección de anomalías
- etc.

Para aprender más, puedes mirar la [documentación de scikit-learn](#) (en inglés).

Librerías de Python para Deep Learning

Aunque el Deep Learning se engloba dentro del Machine Learning, he puesto las librerías de python para aprendizaje profundo al mismo nivel. La razón es que últimamente, el deep learning es el mayor responsable de la reciente popularidad del machine learning.

TensorFlow



[TensorFlow](#) es una librería de python, desarrollada por Google, para realizar cálculos numéricos mediante diagramas de flujo de datos. Esto puede chocar un poco al principio, porque en vez de codificar un programa, codificaremos un grafo. Los nodos de este grafo serán operaciones matemáticas y las aristas representan los tensores (matrices de datos multidimensionales).

Con esta computación basada en grafos, TensorFlow puede usarse para deep learning y otras aplicaciones de cálculo científico.

Si te estás preguntando por qué necesitamos diseñar un grafo en vez de un programa, es por la flexibilidad de ejecución que TensorFlow permite. Por ejemplo, el grafo que representa la red neuronal profunda y sus datos, se podrá ejecutar en una o varias CPU o GPU en un PC, en un servidor o en un móvil.

Para aprender más, puedes mirar el [tutorial de TensorFlow](#) (en inglés).

Keras



[Keras](#) es un interfaz de alto nivel para trabajar con redes neuronales. El interfaz de Keras es mucho más fácil de usar que el de TensorFlow. Esta facilidad de uso es su principal característica.

Con Keras es muy fácil comprobar si nuestras ideas tendrán buenos resultados rápidamente. Keras utiliza otras librerías de deep learning (TensorFlow, CNTK o Theano) de forma transparente para hacer el trabajo que le digamos.

Para aprender más, puedes mirar la [documentación de Keras](#) (en inglés).

PyTorch



[PyTorch](#) es una librería de python, desarrollada por Facebook, que permite el cálculo numérico eficiente en CPU y GPUs.

Puedes pensar en PyTorch como una librería que te da las capacidades de NumPy en una GPU. En otras palabras, si tu tarjeta gráfica tiene un procesador gráfico (por ejemplo, una NVIDIA

moderna), tu código se puede ejecutar unas ¡10 – 20 veces más rápido!

El aprendizaje profundo (deep learning) usa cálculos matriciales y de derivadas masivos y paralelizables en GPUs. Por eso, PyTorch también se especializa en deep learning.

Para aprender más, puedes mirar los [tutoriales de PyTorch](#) y su [documentación](#) (ambos en inglés).

Librerías de Python para IA explicable

SHAP



SHAP

[SHAP](#) es una librería para realizar Inteligencia Artificial Explicable (XAI por sus siglas in inglés eXplainable Artificial Intelligence). Utiliza cálculos del campo de la teoría de juegos para averiguar qué variables tienen más influencia en las predicciones de las técnicas de machine learning.

SHAP permite entender cómo se toman las decisiones en modelos de caja negra (random forest o redes neuronales). Puedes obtener explicaciones tanto para predicciones individuales como de forma global. Su API es bastante fácil de usar.

Librerías de Python para Procesamiento de Lenguaje Natural

Algunas de las librerías que hemos visto se pueden usar también para algunas de las fases del [procesamiento del lenguaje natural](#). Por ejemplo, scikit-learn puede usarse para calcular frecuencias normalizadas de los términos que aparecen en documentos.

Las librerías de deep learning y scikit-learn también permiten construir modelos de machine learning con datos de texto, una vez estos se hayan convertido a un formato estándar.

En este apartado, vamos a ver las librerías que están principalmente dedicadas al procesamiento del lenguaje natural.

NLTK: Natural Language Toolkit

[NLTK](#) es una de las librerías más antiguas en python para procesamiento de lenguaje natural. Sigue siendo muy útil para tareas de preprocesado de texto tales como la tokenización, lematización, exclusión de palabras irrelevantes, etc. NLTK también se usa mucho como herramienta de estudio y enseñanza de procesamiento del lenguaje.

Para aprender más, puedes leer [el libro de NLTK](#) (en inglés).

gensim



[gensim](#) es una librería para el procesamiento de lenguaje natural creada por Radim Řehůřek. El punto fuerte de Gensim es el modelado de temas. Es decir, puede identificar automáticamente de que tratan un conjunto de documentos.

Además, Gensim es útil para construir o importar representaciones de vectores distribuidas tales como [word2vec](#). También podemos usar Gensim para analizar la similaridad entre documentos, lo que es muy útil cuando realizamos búsquedas.

Para aprender más, mira los [tutoriales de Gensim](#) (en inglés).

spaCy



[spaCy](#) es la librería de procesamiento natural más rápida que existe. Está diseñada para usarse en aplicaciones reales y extraer información relevante. spaCy también es muy útil para preparar texto para otras tareas de aprendizaje automático. Por ejemplo, podemos preparar los datos para usarlos con TensorFlow, PyTorch, scikit-learn, Gensim, etc.

Con spaCy también vamos a poder construir modelos lingüísticos estadísticos sofisticados para muchos de los problemas de procesamiento de lenguaje natural.

Para saber más, mira la [documentación de spaCy](#) (en inglés).

Jupyter Notebook



[Jupyter](#) Notebook es una aplicación web para crear documentos que contienen código, ecuaciones, visualizaciones y texto. Puedes usar Jupyter notebooks para limpiar datos, transformarlos, realizar simulaciones numéricas, modelos estadísticos, visualizaciones de datos, machine learning y mucho más.

A efectos prácticos es como una consola interactiva de python en un navegador que permite la ejecución de código python, visualización de datos y gráficos, y documentar lo que estés haciendo.

Jupyter no es en realidad una librería de python. Sin embargo, ya que estamos viendo cuáles son las herramientas que más usa un científico de datos, la lista no estaría ni mucho menos completa sin Jupyter. Utilizo Jupyter constantemente para probar ideas y construir prototipos simples. No lo recomiendo cuando el código sea más complejo o cuando queramos crear librerías con nuestro trabajo para reutilizarlo en otros proyectos.

Anaconda



[Anaconda](#) es una distribución de python para Cálculo Numérico, Análisis de Datos y Machine Learning. Contiene las librerías más usadas por los científicos de datos. Además hace muy fácil la instalación de otras librerías que puedas necesitar.

Con Anaconda también es posible crear varios entornos de trabajo si estás trabajando en varios proyectos . Esto puede ser útil, por ejemplo, si uno de los proyectos necesita python 3 y el otro python 2. O si estás trabajando en un proyecto que necesita unas librerías específicas o que tengan una versión específica.

A no ser que tengas que trabajar con aplicaciones antiguas, te recomiendo que utilices la distribución de Anaconda con Python 3. Ve a la sección de [descargas](#) para conseguirla. Para más información, puedes seguir la [documentación de Anaconda](#).

Resumen

Hemos visto las mejores librerías de python para:

- visualización
- cálculo numérico
- análisis de datos
- manipulación de datos
- machine learning
- deep learning
- inteligencia artificial explicable
- procesamiento de lenguaje natural.

Jupyter Notebook no es una librería, pero un entorno web que va a facilitarnos mucho la vida. Con Jupyter podemos probar nuestras ideas y ver los resultados de forma muy intuitiva, a la vez que lo documentamos.

Finalmente, la forma más fácil de instalar todas estas librerías es instalar Anaconda. Anaconda va a instalar muchas de estas librerías. El resto, podrás instalarlas manualmente cuando las necesites.

Esta lista de librerías no tiene por objetivo ser completa, si no sólo indicar cuáles librerías de Machine Learning son más útiles ... por lo menos para mí. Si tu librería favorita no está en la lista, te invito a que la pongas en los comentarios.

Inteligencia Artificial aplicada a meneame.net

Por Jose Martinez Heras
27/01/2019



En este artículo, vamos a aplicar Inteligencia Artificial a todas las noticias de portada de meneame.net en 2018. Empezaremos realizando un análisis estadístico y visualización de datos. Después usaremos Procesamiento del Lenguaje Natural y Aprendizaje Automático.

Quisiera agradecer a [Alfonso Martínez Heras](#) su colaboración en este proyecto. Alfonso se ha encargado de crear un *web scrapper* para obtener las historias de portada de meneame.net automáticamente.

¿Qué es meneame.net?

A lo mejor nunca has visto Menéame y te estás preguntando qué es. Aquí tienes una definición:

[Menéame](#) es un sitio donde compartir enlaces que creas interesantes.
Menéame es un sitio social: funciona gracias a las aportaciones de los usuarios (como tú)

[Documentación de Menéame](#)

The screenshot shows the menéame homepage with two main news items and a sidebar:

- Top Story:** "Cómo saludar a tus vecinos sin hacerles creer que eres un asesino" by [elbertito12 a elmundotoday.com](#). It has 65 meneos and 593 clics. The text discusses how to greet neighbors without sounding like a killer. It includes a small profile picture of a man and a woman.
- Second Story:** "Psicosis en un pub al sacar un policía de paisano su arma y apuntar a los clientes" by [joya a elconfidencial.com](#). It has 75 meneos and 731 clics. The text describes a police officer's outburst in a pub. It includes a small video thumbnail showing a street scene.
- Sidebar (EN SUBS):** A list of 10 items:
 - 10 [GILIPOLLECES](#) Muere el inventor del autocorrector a los 71 años
 - 14 [ARTÍCULOS](#) Strike sin explicación
 - 16 [RETUIT](#) Macron lo clava sobre el Brexit.[FR][EN]
 - 104 [ARTÍCULOS](#) Censurame y la incitación al odio como excusa
 - 9 [MOVILIDAD](#) Dacia lanzará un coche eléctrico "asombrosamente barato"
 - 31 [GILIPOLLECES](#) La edad media que viene

En menéame, los usuarios registrados comparten enlaces. Otros usuarios registrados o visitantes votan (menean) estos enlaces para hacerlos más visibles. Puedes acceder a las historias que hay tras estos enlaces haciendo click en el título. Para acceder a la sección de comentarios, donde los usuarios registrados comentan la historia, puedes hacer click en el número de meneos.

Datos básicos

Si nos fijamos todas las noticias que se han publicado en la portada de meneame en 2018, podemos extraer los siguientes datos:

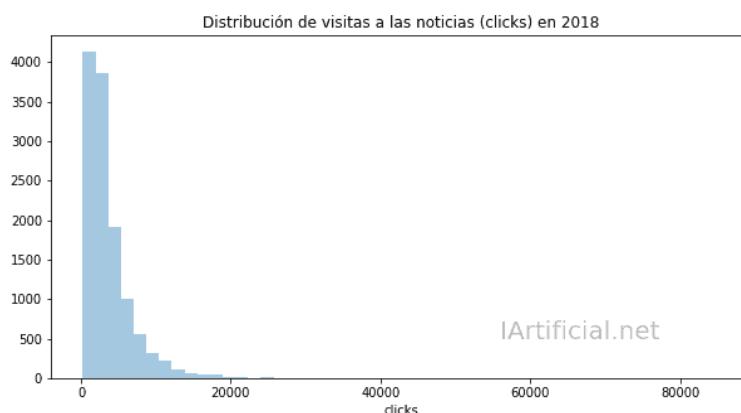
- **12.332** noticias
- **45.254.081** visitas a la noticias (clicks)
- **5.764.374** meneos
- **1.103.313** comentarios
- **1.746** autores.

Análisis Estadístico de todas las noticias en portada de Menéame en 2018

Distribuciones

Una forma de entender mejor los datos es visualizando las distribuciones de datos. En esta sección vamos a visualizar la distribución de clicks, de meneos y la distribución conjunta de clicks y meneos.

Distribución de visitas a las noticias (clicks)

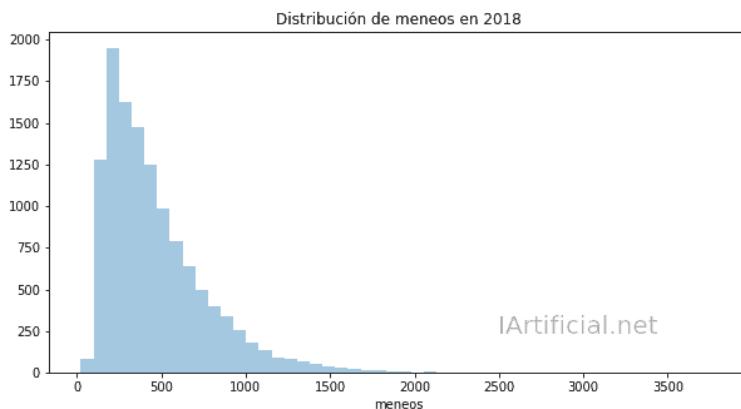


Distribución del número de clicks (visitas) para las noticias de portada de meneame.net en 2018.

Como podemos ver, el número de visitas a las noticias de menéame, sigue una distribución exponencial. Podemos observar, tal y como cabe esperar de una distribución exponencial, que:

- La mayoría de las noticias reciben un número relativamente pequeño de visitas
- Hay muy pocas noticias que reciban un número alto de visitas

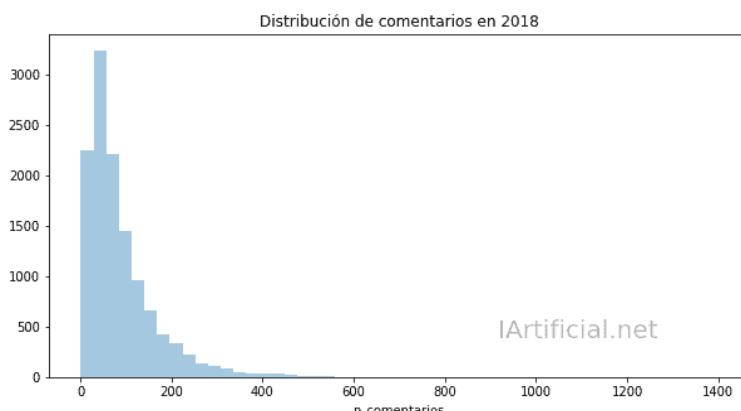
Distribución de meneos



Distribución del número de meneos para las noticias de portada de meneame.net en 2018

La distribución de meneos también sigue una distribución exponencial. Hay una minoría de noticias con un número alto de meneos.

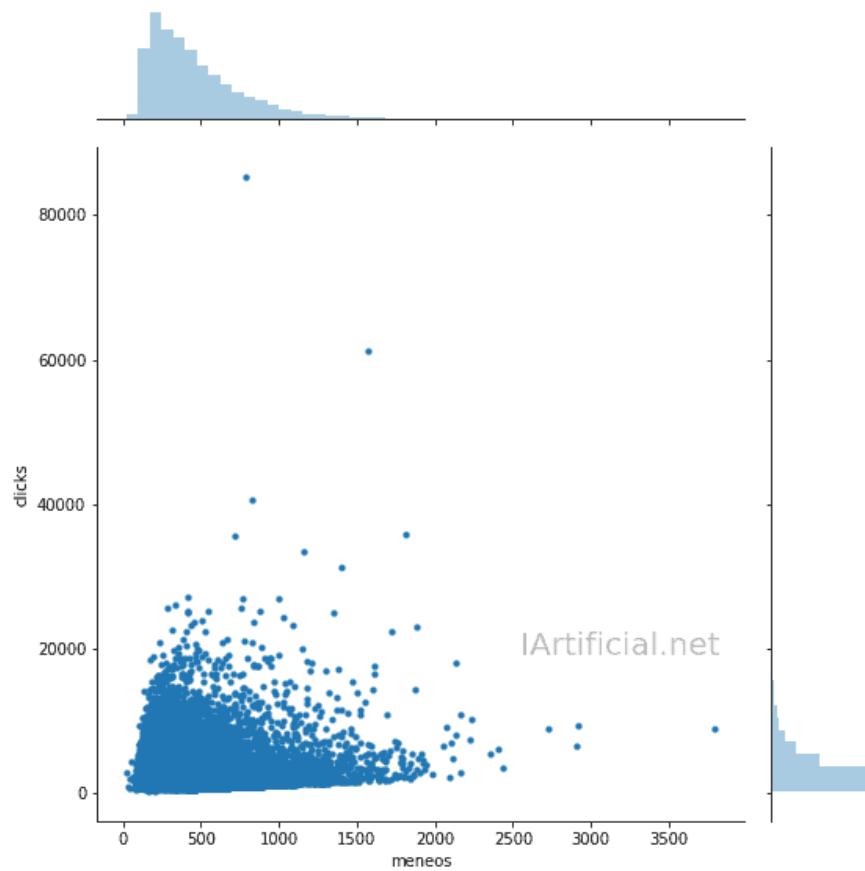
Distribución de comentarios



Distribución del número de comentarios para las noticias de portada de meneame.net en 2018.

La distribución del número del número de comentarios es también exponencial. Hay una minoría de noticias con un número alto de comentarios.

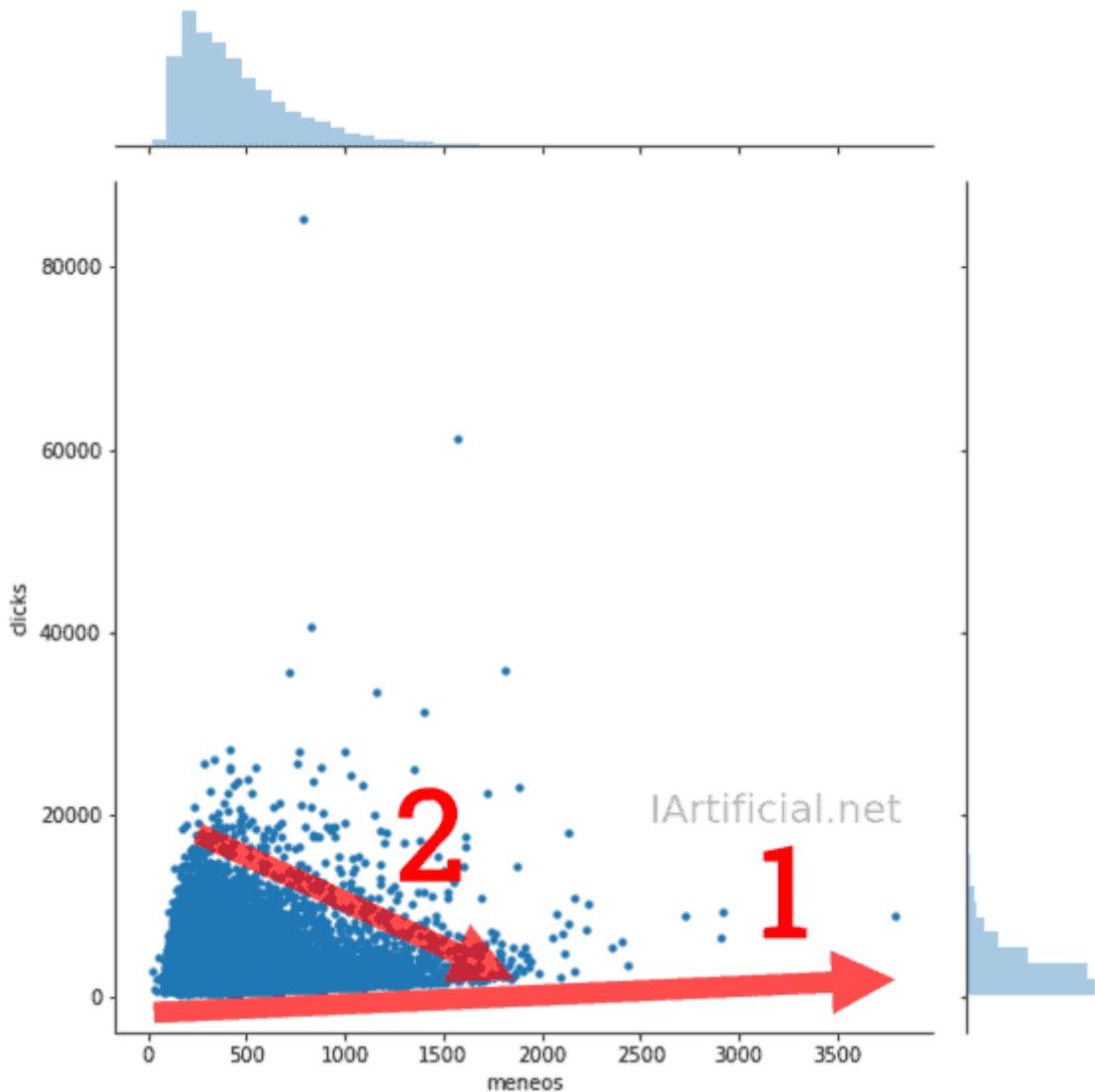
Distribución de clicks y meneos



Distribución entre el número de meneos y el número de clicks para las noticias de portada de meneame.net en 2018. Cada punto representa una noticia.

En este gráfico quería representar cómo los meneos afectan al número de clicks. Hay dos cosas que me han sorprendido de este gráfico y que he destacado en el siguiente:

1. Hay un suelo en el número de clicks que va creciendo con el número de meneos.
Investigando un poco, ya sé por qué ocurre esto. Menéame implementa un sistema que asegura que una noticia no puede tener más meneos que visitas. Dicho de otra forma, menéame requiere un cierto número de visitas (clicks) antes de aceptar otro meneo.
2. Lo más sorprendente para mí es que parece que el número de clicks decrece con el número de meneos! **Cuantos más meneos tiene una noticia, menos visitas recibe** (salvo excepciones, claro está). Me esperaba que las noticias más visitadas serían aquellas con más meneos. Aunque también hay una explicación para esto... Lo vemos en un momento.



Análisis de la distribución entre el número de meneos y el número de clicks para las noticias de portada de meneame.net en 2018. 1: hay un suelo en el número de clicks que va creciendo con el número de meneos 2: cuantos más meneos tiene una noticia, menos visitas recibe.

Los top 11

Vamos a ver unas estadísticas simples. En casi todos los análisis estadísticos, siempre se habla de los *top 10*. A mi siempre me queda el gusanillo de saber si había tanta diferencia entre el 10 y el 11. Siempre me da un poco de pena que el 11, estando muchas veces tan cerca del 10, no salga nunca. Así que vamos a hablar de los *top 11*.

Las 11 noticias más visitadas

Número de visitas para cada noticia:

- 85.211 – [No, Arabia Saudí NO ha decapitado a la activista Esra al-Ghamgam](#)
- 61.162 – [Cosas que Franco no hizo por mucho que se repitan...](#)
- 40.622 – [Factura de la luz del año 2001 vs 2017](#)
- 35.778 – [Contemos chistes sobre gitanos](#)
- 35.582 – [La mentira del twerking y el balón de oro](#)
- 33.442 – [Sentencia de La Manada ¿Cómo puede absolvérseles de violación con semejantes Hechos Probados?](#)

- 31.218 – [Al primer ministro eslovaco se le cae la coca durante una entrevista en la televisión](#)
- 27.118 – [La Policía Nacional avisa: Si ves estos testigos en tu puerta, en la del vecino o caídos en el suelo, llámanos #091](#)
- 26.949 – [Por qué la izquierda no gana las elecciones](#)
- 26.874 – [El truco ganador del Campeonato Mundial de Magia te dejará seis minutos con la boca abierta](#)
- 26.034 – [Estas son las empresas que más facturan en cada provincia española](#)

Las 11 noticias con más meneos

Número de meneos para cada noticia:

- 3.791 – [Senadores del PP, ovacionándose y en pie, tras conseguir paralizar con sus votos la subida de las pensiones](#)
- 2.922 – [Hola: soy Diputada y necesito ayuda](#)
- 2.904 – [Fallece Stephen Hawking](#)
- 2.722 – [Cristina Cifuentes obtuvo su título de máster en una universidad pública con notas falsificadas](#)
- 2.436 – [Cristina Cifuentes se matriculó en su máster tres meses después de que empezaran las clases](#)
- 2.406 – [La indignación de Pablo Iglesias ante la decisión del Supremo sobre el impuesto hipotecario](#)
- 2.355 – [Sánchez, presidente: la moción de censura tumba a Rajoy](#)
- 2.231 – [Pedro J. Ramírez aporta 21 pruebas que implican a Rajoy en la caja B del PP](#)
- 2.220 – [Jueces de mierda y católicos de mierda](#)
- 2.169 – [El Supremo concluye por 15 votos a 13 que es el cliente el que debe pagar el impuesto hipotecario](#)
- 2.168 – [Lo de Rajoy es mucho peor que lo de Cifuentes](#)

Las 11 noticias más comentadas

Número de comentarios para cada noticia:

- 1.397 – [Ciudadanos pide que conocer las lenguas cooficiales no sea obligatorio para trabajar en la administración autonómica](#)
- 1.013 – [Iglesias y Montero anuncian una consulta en Podemos sobre la compra de su casa: si pierden, dimitirán](#)
- 913 – [La Justicia alemana deja en libertad a Puigdemont y descarta delito de rebelión](#)
- 886 – [El detenido por la muerte de Laura Luelmo confiesa el asesinato](#)
- 788 – [Paso en firme en materia de violencia contra la mujer: desaparece el abuso; todo delito sexual será agresión o violación](#)
- 764 – [Torrent pospone la investidura hasta que se pueda celebrar «sin injerencias»](#)
- 755 – [Sentencia de La Manada: ¿Cómo puede absolvérseles de violación con semejantes Hechos Probados?](#)
- 744 – [La policía alemana detiene a Puigdemont cuando entraba desde Dinamarca](#)
- 724 – [La Guardia Civil detiene a una líder de los CDR por terrorismo y rebelión](#)
- 714 – [Dos transexuales ganan a mujeres en competición de atletismo \[eng\]](#)
- 699 – [El actor Willy Toledo, detenido en Madrid](#)

Los 11 usuarios que más veces han llegado a portada

Número de noticias en portada por usuario:

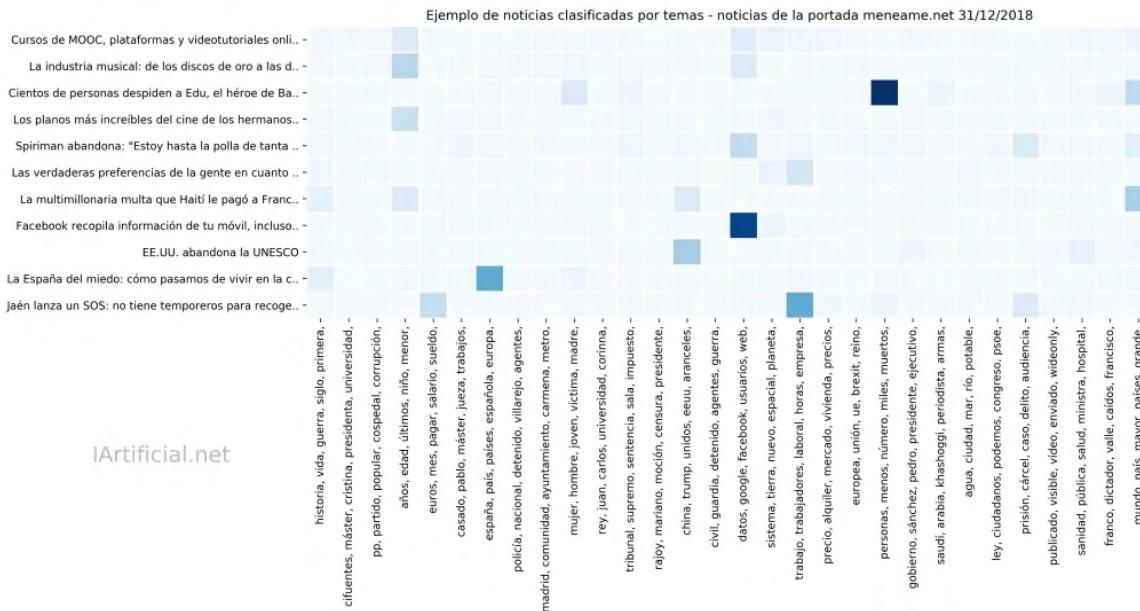
- 495 – [ccguy](#)
- 240 – [Ripio](#)
- 201 – [Ratoncolorao](#)
- 198 – [Danichaguito](#)
- 190 – [_550559_](#)
- 174 – [Joya](#)
- 169 – [Wurmspiralmaschine](#)

- 162 – [Cubillina](#)
- 152 – [rataxuelle](#)
- 151 – [Quinqui](#)
- 147 – [Meneador_Compulsivo](#)

Después de este análisis estadístico simple, vamos a utilizar técnicas de Inteligencia Artificial para hacer un análisis más concienzudo. En particular, vamos a usar técnicas de Machine Learning, que es el área de la Inteligencia Artificial que permite aprender de los datos.

Inteligencia Artificial para el Modelado de Temas

El Modelado de Temas es una técnica de [aprendizaje automático no-supervisado](#). Con esta técnica asumimos que cada historia en menéame se compone de una combinación de varios temas. Veamos un ejemplo con las últimas noticias del 2018 (si haces click en la imagen, deberías verla ampliada).



Los temas que la Inteligencia Artificial ha encontrado automáticamente son:

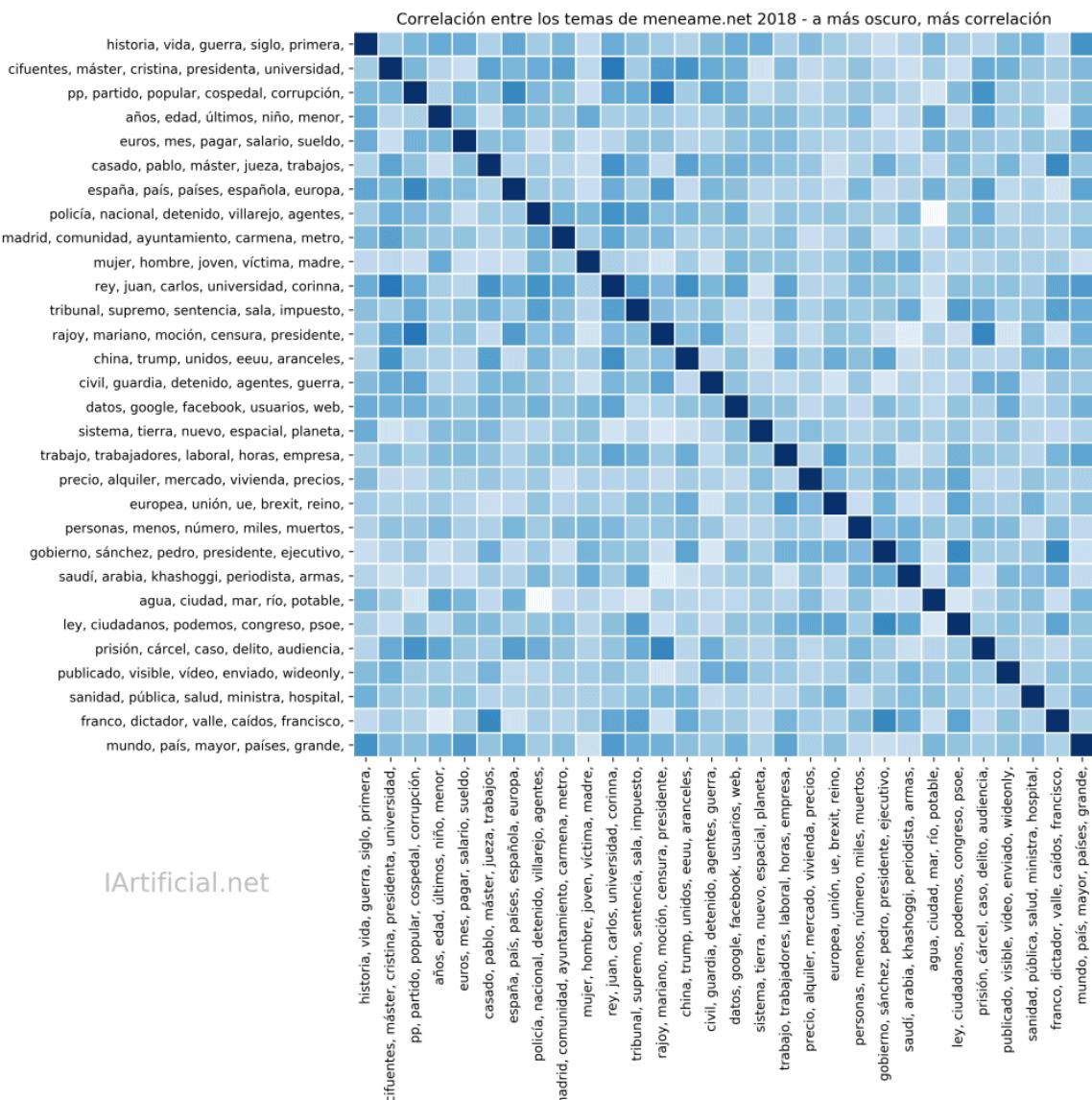
1. historia, vida, guerra, siglo, primera,
2. cifuentes, máster, cristina, presidenta, universidad,
3. pp, partido, popular, cospedal, corrupción,
4. años, edad, últimos, niño, menor,
5. euros, mes, pagar, salario, sueldo,
6. casado, pablo, máster, jueza, trabajos,
7. españa, país, países, española, europa,
8. policía, nacional, detenido, villarejo, agentes,
9. madrid, comunidad, ayuntamiento, carmen, metro,
10. mujer, hombre, joven, víctima, madre,
11. rey, juan, carlos, universidad, corinna,
12. tribunal, supremo, sentencia, sala, impuesto,
13. rajoy, mariano, moción, censura, presidente,
14. china, trump, unidos, eeuu, aranceles,
15. civil, guardia, detenido, agentes, guerra,
16. datos, google, facebook, usuarios, web,
17. sistema, tierra, nuevo, espacial, planeta,
18. trabajo, trabajadores, laboral, horas, empresa,
19. precio, alquiler, mercado, vivienda, precios,
20. europea, unión, ue, brexit, reino,
21. personas, menos, número, miles, muertos,

22. gobierno, sánchez, pedro, presidente, ejecutivo,
23. saudí, arabia, khashoggi, periodista, armas,
24. agua, ciudad, mar, río, potable,
25. ley, ciudadanos, podemos, congreso, psOE,
26. prisión, cárcel, caso, delito, audiencia,
27. publicado, visible, vídeo, enviado, wideonly,
28. sanidad, pública, salud, ministra, hospital,
29. franco, dictador, valle, caídos, francisco,
30. mundo, país, mayor, países, grande,

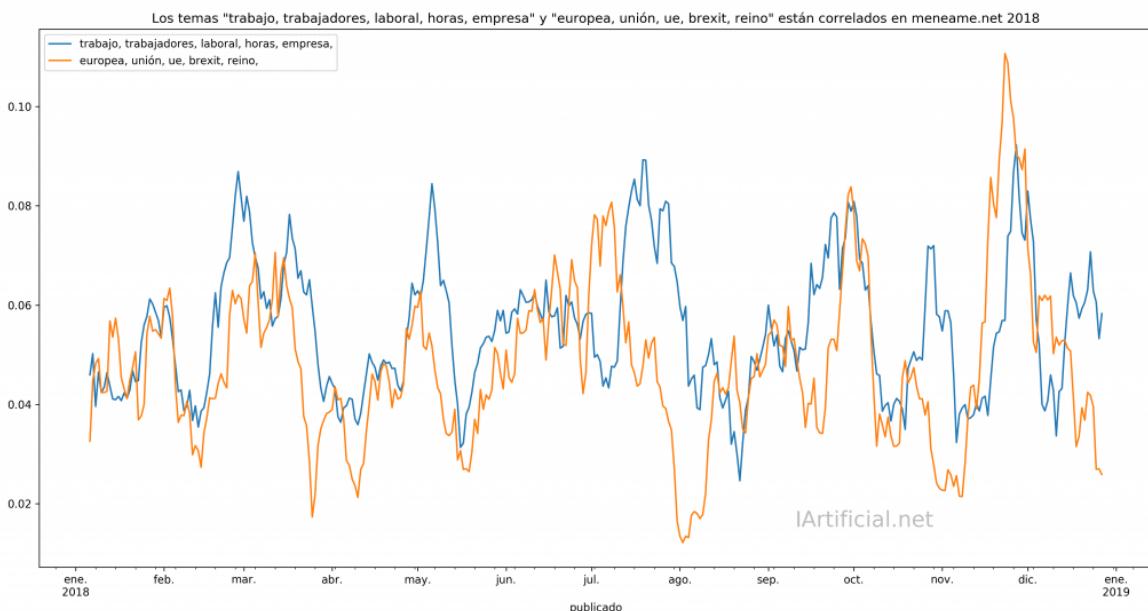
Seguramente te estás preguntando que dónde están los temas que faltan. La verdad es que no están todos los temas, ni mucho menos. Pero tampoco era lo que pretendíamos. El modelado de temas es útil para entender cuáles son los temas de los que más se habla en menéame. Además nos va a permitir hacer otro tipo de análisis como son la correlación entre temas y la predicción de clicks y meneos.

Correlación entre temas

Al analizar la correlación entre temas, podemos averiguar qué temas ocurren a la vez en la mayoría de los casos. En la siguiente figura, puedes ver la correlación entre los temas de menéame en 2018 (si haces click en la imagen, deberías verla ampliada).



Por ejemplo, si te das cuenta, los temas «*trabajo, trabajadores, laboral, horas, empresa,*» y «*europea, unión, ue, brexit, reino,*» están bastante correlados. Así que esperamos que, en la mayoría de los casos, cuando haya más artículos sobre uno de los temas, habrá más artículos sobre el otro tema. Con un gráfico quedará más claro (recuerda que haciendo click puedes ampliarlo).



La Inteligencia Artificial averigua qué afecta al número de meneos, al de clicks y al de comentarios

La Inteligencia Artificial nos va a ayudar a averiguarlo! En particular, los [modelos supervisados](#) de machine learning.

Enfocando el problema como un problema de clasificación

En vez de predecir cuántos meneos / clicks / comentarios tendrá una historia, vamos a intentar predecir si una noticia tendrá un número alto o bajo de meneos / clicks / comentarios. Esto hace el problema más fácil. Por ejemplo, sería muy difícil predecir cuántos meneos tendrían los *top 11 meneos*. Es mucho más fácil predecir que los top 11 tendrán un número alto de meneos. De esta forma, transformamos un problema de regresión en uno de clasificación.

Para saber si un número de meneos / clicks / comentarios es alto o bajo, nos vamos a fijar en todas las historias del 2018. Las historias cuyo número de meneos / clicks / comentarios sean mayores que su valor mediano, estarán en la categoría de «alto»; las otras en el «bajo». Dicho de otra forma, ordenamos las historias por el número de meneos de menos a más. La mitad más alta tiene se clasifica como «alta» y la mitad más baja como «baja». También hacemos lo mismo para el número de clicks.

De esta forma, consideramos que:

- Una historia con **más de 385 meneos**, tiene un **número alto de meneos**
- Una historia con **más de 2.655 clicks**, tiene un **número alto de clicks**
- Una historia con **más de 63 comentarios**, tiene un **número alto de comentarios**

Modelos de Machine Learning

He construido varios modelos de aprendizaje automático supervisados para estudiar qué influye en el número de meneos:

1. para entender qué temas tienen más / menos meneos, clicks, comentarios
2. para entender qué palabras tienen más / menos meneos, clicks, comentarios

Hay varios tipos de modelos que podría haber utilizado. Como el objetivo de este proyecto es entender las influencias, he usado modelos de Inteligencia Artificial explicables. En particular he experimentado con árboles de decisión y regresión logística. La regresión logística parece dar mejores resultados. Así que usaremos este modelo para el resto del artículo.

¿Qué influye en el número de meneos?

Vamos a ver como los temas y las palabras de la historia afectan al número de meneos.

¿Qué temas son los mejores y peores para tener más meneos?

El modelo de de inteligencia artificial para predecir meneos a partir de temas, tiene un F1 de 0.72 en entrenamiento y un F1 de 0.70 en testeo.

Nota: Uso las historias publicadas entre enero y noviembre para construir los modelos de IA. Los datos del mes de diciembre 2018 lo uso para medir la generalización del modelo.

[F1](#) es una medida de rendimiento de modelos de machine learning para problemas de clasificación binaria. El valor de F1 va del 0 al 1, siendo 1 lo mejor. F1 combina las medidas de [precisión](#) y [exhaustividad](#).

Consideramos que una historia tiene un número de meneos alto si tiene más de 385 meneos (que es el valor mediano de número de meneos)

Los temas que obtienen *más meneos*

Estos son los temas que favorecen un **número alto de meneos**. Están ordenados de mayor a menor influencia (cuanto más arriba en la lista, más meneos):

1. pp, partido, popular, cospedal, corrupción,
2. cifuentes, máster, cristina, presidenta, universidad,
3. prisión, cárcel, caso, delito, audiencia,
4. sanidad, pública, salud, ministra, hospital,
5. trabajo, trabajadores, laboral, horas, empresa,
6. casado, pablo, máster, jueza, trabajos,
7. tribunal, supremo, sentencia, sala, impuesto,
8. rajoy, mariano, moción, censura, presidente,
9. euros, mes, pagar, salario, sueldo,
10. gobierno, sánchez, pedro, presidente, ejecutivo,
11. ley, ciudadanos, podemos, congreso, psOE,
12. rey, juan, carlos, universidad, corinna,
13. madrid, comunidad, ayuntamiento, carmena, metro,
14. saudí, arabia, khashoggi, periodista, armas,
15. españa, país, países, española, europa,
16. franco, dictador, valle, caídos, francisco,
17. precio, alquiler, mercado, vivienda, precios,
18. europea, unión, ue, brexit, reino,
19. civil, guardia, detenido, agentes, guerra,
20. policía, nacional, detenido, villarejo, agentes,
21. mujer, hombre, joven, víctima, madre,
22. china, trump, unidos, eeuu, aranceles,

Los temas que obtienen *menos meneos*

Estos son los temas que favorecen un **número bajo de meneos**. Están ordenados de mayor a menor influencia (cuanto más arriba en la lista, menos meneos):

1. historia, vida, guerra, siglo, primera,

2. sistema, tierra, nuevo, espacial, planeta,
3. agua, ciudad, mar, río, potable,
4. mundo, país, mayor, países, grande,
5. publicado, visible, vídeo, enviado, wideonly,
6. datos, google, facebook, usuarios, web,

¿Qué palabras son las mejores y peores para tener más meneos?

El modelo para predecir meneos a partir de palabras, tiene un F1 de 0.89 en entrenamiento y un F1 de 0.75 en testeo.

Las 11 palabras que obtienen *más meneos*

1. pp
2. cífuente
3. rajoy
4. máster
5. gobierno
6. euros
7. franco
8. casado
9. españa
10. denuncia
11. ciudadanos

Las 11 palabras que obtienen *menos meneos*

1. historia
2. siglo
3. guerra
4. arte
5. espacial
6. tecnología
7. época
8. imágenes
9. ciudad
10. mapa
11. kilómetros

¿Qué influye en el número de clicks?

Vamos a ver como los temas y las palabras de la historia afectan al número de visitas (clicks). Consideramos que una historia tiene un número de clicks alto si tiene más de 2.655 visitas (que es el valor mediano de número de clicks)

¿Qué temas son los mejores y peores para tener más clicks?

El modelo para predecir clicks a partir de temas, tiene un F1 de 0.65 en entrenamiento y un F1 de 0.65 en testeo.

Los temas que obtienen *más clicks*

Estos son los temas que favorecen un **número alto de visitas (clicks)**. Están ordenados de mayor a menor influencia (cuanto más arriba en la lista, más clicks):

1. historia, vida, guerra, siglo, primera,
2. mujer, hombre, joven, víctima, madre,
3. mundo, país, mayor, países, grande,
4. publicado, visible, vídeo, enviado, wideonly,

Los temas que obtienen *menos clicks*

Estos son los temas que favorecen un **número bajo de visitas (clicks)**. Están ordenados de mayor a menor influencia (cuanto más arriba en la lista, menos clicks):

1. prisión, cárcel, caso, delito, audiencia,
2. sanidad, pública, salud, ministra, hospital,
3. gobierno, sánchez, pedro, presidente, ejecutivo,
4. tribunal, supremo, sentencia, sala, impuesto,
5. ley, ciudadanos, podemos, congreso, psOE,
6. sistema, tierra, nuevo, espacial, planeta,
7. europea, unión, ue, brexit, reino,
8. pp, partido, popular, cospedal, corrupción,
9. china, trump, unidos, eeuu, aranceles,
10. madrid, comunidad, ayuntamiento, carmenA, metro,
11. euros, mes, pagar, salario, sueldo,
12. franco, dictador, valle, caídos, francisco,
13. civil, guardia, detenido, agentes, guerra,
14. trabajo, trabajadores, laboral, horas, empresa,
15. policía, nacional, detenido, villarejo, agentes,
16. datos, google, facebook, usuarios, web,
17. rey, juan, carlos, universidad, corinna,
18. cíFuenteS, máster, cristina, presidenta, universidad,
19. saudí, arabía, khashoggi, periodista, armas,
20. personas, menos, número, miles, muertos,
21. casado, pablo, máster, jueza, trabajos,

¿Qué palabras son las mejores y peores para tener más clicks?

El modelo para predecir clicks a partir de palabras, tiene un F1 de 0.87 en entrenamiento y un F1 de 0.68 en testeо.

Las 11 palabras que obtienen *más clicks*

1. vídeo
2. aquí
3. imágenes
4. imagen
5. foto
6. fotos
7. mentiras
8. peor
9. coche
10. viñeta
11. problema

Las 11 palabras que obtienen *menos clicks*

1. gobierno
2. ministerio
3. muere
4. justicia
5. tribunal
6. investigación
7. estudio
8. investigadores
9. ley
10. ing
11. ministra

¿Qué influye en el número de comentarios?

Vamos a ver como los temas y las palabras de la historia afectan al número de comentarios en menéame. Consideramos que una historia tiene un número de comentarios alto si tiene más de 63 comentarios (que es el valor mediano de número de comentarios)

¿Qué temas son los mejores y peores para tener más comentarios?

El modelo para predecir el número de comentarios a partir de temas, tiene un F1 de 0.64 en entrenamiento y un F1 de 0.67 en testeo.

Los temas que obtienen *más comentarios*

Estos son los temas que favorecen un **número alto de comentarios** en menéame. Están ordenados de mayor a menor influencia (cuanto más arriba en la lista, más comentarios):

1. mujer, hombre, joven, víctima, madre,
2. ley, ciudadanos, podemos, congreso, psOE,
3. precio, alquiler, mercado, vivienda, precios,
4. gobierno, sánchez, pedro, presidente, ejecutivo,
5. españa, país, países, española, europa,
6. china, trump, unidos, eeuu, aranceles,
7. trabajo, trabajadores, laboral, horas, empresa,
8. casado, pablo, máster, jueza, trabajos,
9. prisión, cárcel, caso, delito, audiencia,
10. personas, menos, número, miles, muertos,
11. euros, mes, pagar, salario, sueldo,
12. europea, unión, ue, brexit, reino,
13. franco, dictador, valle, caídos, francisco,
14. madrid, comunidad, ayuntamiento, carmen, metro,

Los temas que obtienen *menos comentarios*

Estos son los temas que favorecen un **número bajo de comentarios** en menéame. Están ordenados de mayor a menor influencia (cuanto más arriba en la lista, menos comentarios):

1. historia, vida, guerra, siglo, primera,
2. sistema, tierra, nuevo, espacial, planeta,
3. agua, ciudad, mar, río, potable,
4. pp, partido, popular, cospedal, corrupción,
5. mundo, país, mayor, países, grande,
6. rey, juan, carlos, universidad, corinna,

¿Qué palabras son las mejores y peores para tener más comentarios?

El modelo para predecir comentarios a partir de palabras, tiene un F1 de 0.87 en entrenamiento y un F1 de 0.72 en testeo.

Las 11 palabras que obtienen *más comentarios*

1. mujer
2. puigdemont
3. ciudadanos
4. podemos
5. mujeres
6. precio
7. joven
8. gobierno
9. país
10. violencia

11. sánchez

Las 11 palabras que obtienen *menos comentarios*

1. historia
2. siglo
3. investigadores
4. obras
5. espacial
6. arte
7. obra
8. corrupción
9. ii
10. documental
11. época

Análisis de los resultados de la Inteligencia Artificial

En la tabla podemos ver el rendimiento [F1](#) de cada modelo ya sea para la predicción de meneos, clicks o comentarios, usando temas o palabras. Se muestran los valores F1 para entrenamiento / testeo, para considerar la generalización.

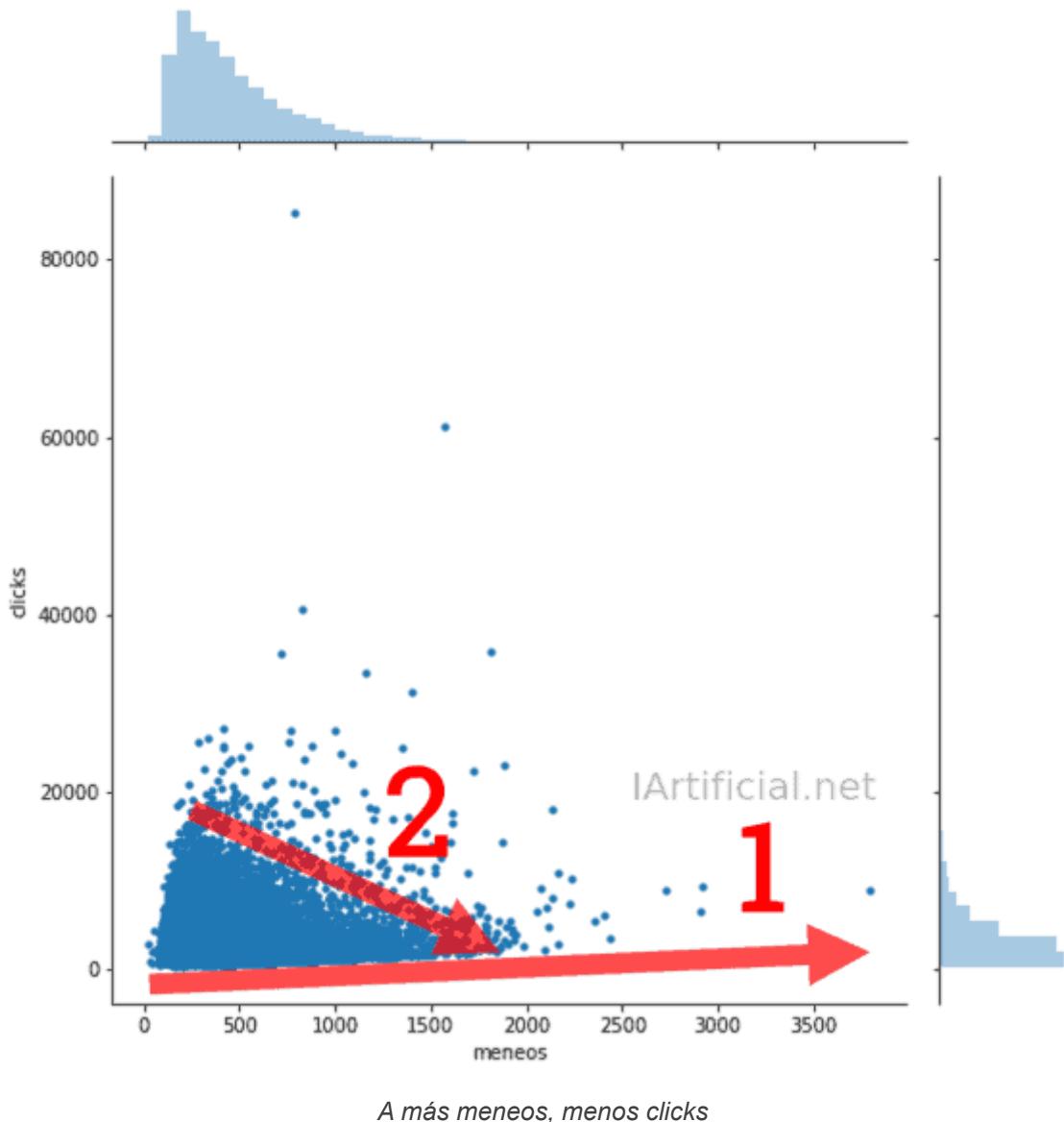
F1 (1.0 es lo máximo)	Temas	Palabras
Predicción de meneos	0.72 / 0.70	0.89 / 0.75
Predicción de visitas (clicks)	0.65 / 0.65	0.87 / 0.68
Predicción de comentarios	0.64 / 0.67	0.87 / 0.72

Según los resultados, parece que las **palabras funcionan mejor que los temas** tanto para predecir meneos como visitas como comentarios. Los temas funcionan algo peor pero son más fáciles de interpretar.

También podemos concluir que **los meneos son más fáciles de predecir que los clicks**. Esto lo sabemos porque el rendimiento del modelo de predicción de meneos generaliza mejor que el de predicción de clicks.

¿Por qué a más meneos, menos clicks?

Con respecto a la pregunta que nos hacíamos al principio, que corresponde a «2» en el siguiente gráfico ... ¿por qué a más meneos, menos clicks?



Para contestar a la pregunta de por qué las historias que más meneos tienen reciben menos clicks, tenemos que mirar esta tabla. La tabla compara los 11 temas con **más meneos** con los 11 temas con **menos clicks**. Me he permitido ~~tachar~~ aquellos temas que simultáneamente están en ambas categorías.

Temas para más meneos	Temas para menos clicks
pp, partido, popular, cospedal	prisión, cárcel, caso, delito
cifuentes, máster, cristina, presidenta	sanidad, pública, salud, ministra
prisión, cárcel, caso, delito	gobierno, sánchez, pedro, presidente
sanidad, pública, salud, ministra	tribunal, supremo, sentencia, sala
trabajo, trabajadores, laboral, horas	ley, ciudadanos, podemos, congreso
casado, pablo, máster, jueza	sistema, tierra, nuevo, espacial

tribunal, supremo, sentencia, sala	europea, unión, ue, brexit
rajoy, mariano, moción, censura	pp, partido, popular, cospedal
euros, mes, pagar, salario	china, trump, unidos, eeuu
gobierno, sánchez, pedro, presidente	madrid, comunidad, ayuntamiento, carmen
ley, ciudadanos, podemos, congreso	euros, mes, pagar, salario, sueldo

Si te das cuenta, muchos de los temas que más influencian el número de meneos, también son los que influencian que hayan pocas visitas (clicks), según los modelos de inteligencia artificial.

¿Pero por qué los temas que tienen más meneos son los que menos clicks reciben?

Explicación de por qué el número de visitas decrece con el número de meneos

La inteligencia artificial no nos da una respuesta clara en este punto. Así que echar mano de la inteligencia humana asistida por la inteligencia artificial.

La inteligencia artificial nos ha dicho que muchos de los temas de los que cabe esperar muchos meneos, también cabe esperar pocos clicks. Y ahí se queda.

Así que he estado mirando las noticias para varios temas clave y palabras clave para ofreceros mi interpretación humana.

Interpretación humana

Muchas noticias de actualidad (política, la muerte de alguien, la decisión de un tribunal, etc.) están **autocontenidoas**.

Con **autocontenidoas** quiero decir que mirando la información que hay en la portada menéame, es decir, el título y el resumen, es suficiente para estar informado.

Otras historias, en cambio, realmente necesitan que hagamos click. Sin hacer click no seremos capaces de saber qué ha pasado. Esto es evidente en palabras tales como: vídeo, imágenes, imagen, foto, fotos, viñeta, etc. Aunque también para noticias de historia, cultura, etc.

Dicho esto, encuentro muy útil que muchas historias estén autocontenidoas. Así podemos informarnos rápidamente de lo que está ocurriendo. Las historias no-autocontenidoas nos permiten profundizar más en los temas que nos interesen.

Ejemplo de noticias **autocontenidoas** y **no-autocontenidoas**

228
meneos
¿Por qué cayó el Imperio romano? Una respuesta inesperada (y fascinante)

por cani90 a blogs.elconfidencial.com 15/01 14:20 publicado: 17/01 01:50

12794 clics
 76 comentarios
121
107
3
K 313
cultura

550
meneos
El juez archiva la causa contra Dani Mateo por sonarse los mocos con la bandera

por Livingstone85 a elespanol.com 16/01 20:52 publicado: 17/01 01:40

594 clics
 89 comentarios
197
353
2
K 253
actualidad

Por ejemplo, en la figura anterior vemos 2 historias:

1. La primera historia **no está autocontenida**. Si queremos saber «por qué cayó el imperio romano», necesariamente tendremos que visitar (hacer click) en la noticia. Tiene **pocos meneos y muchos clicks**.
2. La segunda historia **está autocontenida**. Para saber lo que pasó con la causa de Dani Mateo no es imprescindible visitar la noticia. Tiene **muchos meneos y pocos clicks**.

¿Cuántos clicks, meneos y comentarios tendrá este artículo según la Inteligencia Artificial?

Después de este análisis, podríamos preguntarnos ¿cuántos clicks y meneos tendrá este artículo?

Lo primero de todo, es que para que tenga alguna oportunidad, debería llegar a portada de meneame.net. Como a la Inteligencia Artificial le hemos dado sólo los datos de artículos que ya estaban en portada, seguramente la predicción será más optimista de lo normal. Así que no debemos fiarnos ciegamente.

Además, la predicción está basada en el título y el resumen que se use para enviar la historia. Imaginando que tuviésemos lo siguiente:

Título: Una Inteligencia Artificial analiza la portada de menéame de 2018

Resumen de la historia: Una Inteligencia Artificial analiza todas las historias de portada de menéame en 2018. Averigua por qué las historias que más meneos tienen son las que menos visitas (clicks) reciben. También descubre cuáles son los temas principales del año 2018. Además del análisis de inteligencia artificial hay un análisis estadístico de la distribución de meneos, clicks, meneos y clicks, y los top 11 de menéame en 2018.

Si has seguido este artículo, verás que al transformar el problema de regresión en uno de clasificación, no puedo decir cuántos clicks y meneos tendrá este artículo. Lo que sí puedo hacer es predecir la probabilidad de estar en la parte alta de meneos y clicks. Para ello, voy a usar el modelo basado en palabras, ya que daba mejores resultados.

Predicción de probabilidades

Asumiendo que este artículo llegue a la portada de menéame, estas son las probabilidades que nos da la inteligencia artificial:

- **21%** de que este artículo tenga un número alto (más de 385) de meneos
- **72%** de que este artículo reciba un número alto (más de 2.655) de clicks
- **42%** de que este artículo tenga un número alto (más de 63) de comentarios

La rama de Machine Learning de la Inteligencia Artificial da resultados probabilísticos. En general, es muy difícil estar seguro de algo. Así que estos resultados son lo mejor que puedo ofrecer.

Resumen

En este artículo hemos visto los resultados del análisis de todos los artículos que han salido en la portada de menéame en 2018.

Hemos comenzado con un análisis estadístico para ver las distribuciones de clicks, meneos, comentarios y clicks / meneos. Analizando estas distribuciones nos hemos dado cuenta que cuantos más meneos tiene una historia, menos clicks recibe en la mayoría de los casos.

También hemos hecho la lista de los top 11 para meneos, clicks, comentarios y los top 11 usuarios que más veces han llegado a portada.

Hemos usado Inteligencia Artificial para descubrir cuáles son los temas de los que más se ha hablado en menéame en 2018 y cómo están relacionados entre ellos.

La Inteligencia Artificial (IA) también nos ha ayudado a descubrir qué temas y qué palabras influyen en el número de meneos, de clicks y de comentarios. La IA ha puesto de manifiesto que los temas que favorecen un mayor número de meneos, también son los que producen un menor número de clicks.

Aquí nos ha hecho falta tirar de Inteligencia Humana (IH). Me he aventurado a postular que la razón por la que muchas de las noticias muy meneadas tienen pocos clicks, es porque están *autocontenidas*.

Finalmente, me he atrevido a predecir que esta historia, que estás leyendo ahora mismo, recibirá pocos meneos pero muchos clicks ... en el hipotético caso que llegue a la portada de menéame.

Gracias

Gracias por leer este artículo, espero que te haya resultado interesante. Si tienes un momento, me gustaría pedirte tu ayuda. Aquí tienes para elegir:

- ¿Conoces a alguien que le interese la Inteligencia Artificial? Recoméndale este blog.
- [Menea este artículo en menéame](#) ... a ver si conseguimos pasar de los 385 meneos (que según la predicción, va a ser bastante difícil)
- Deja un comentario
- Suscríbete

Regresión Lineal: teoría y ejemplos en Python

Por Jose Martinez Heras
31/01/2019

La regresión lineal es una de las técnicas más usadas en Machine Learning. Su fortaleza estriba en su simplicidad e interpretabilidad. La regresión polinómica, como ya veremos, es una extensión de la regresión lineal.

Regresión Lineal – Teoría

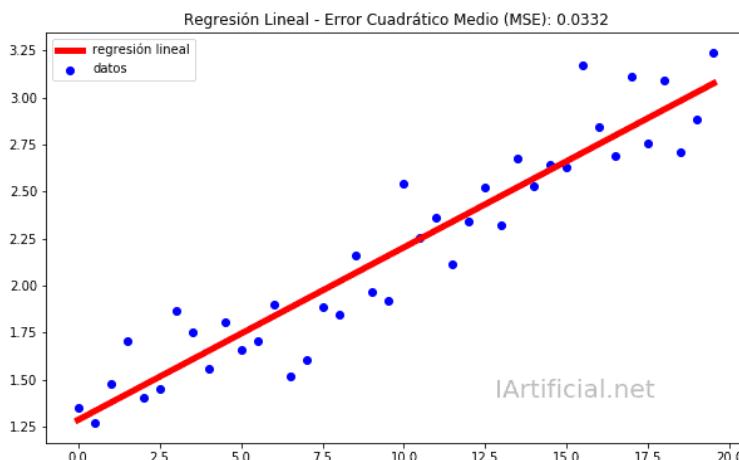
La regresión lineal es una técnica paramétrica de machine learning. Con «paramétrica» queremos decir que incluso antes de mirar a los datos, ya sabemos cuántos parámetros (o coeficientes) vamos a necesitar.

En el caso que estemos usando una sola variable, x , sabemos que una línea necesita 2 parámetros. La fórmula para la regresión lineal con una sola variable x es:

$$y = wx + b$$

El aprendizaje consiste en encontrar cuáles son los mejores parámetros (coeficientes) para los datos que tengamos. Los mejores coeficientes serán los que minimicen alguna medida de error. Para la regresión lineal usaremos el error cuadrático medio.

Ejemplo de Regresión Lineal



Hemos usado una regresión lineal para encontrar los parámetros de la línea que minimiza el error de los datos que tenemos. El proceso de aprendizaje consiste en estimar los parámetros w y b . Así nos queda que para estos datos, los mejores valores son:

$$\begin{aligned} w &= 0.0918 \\ b &= 1.2859 \end{aligned}$$

así que nos queda:

$$y = 0.0918x + 1.2859$$

Podemos usar este modelo de regresión lineal para estimar cuáles serán los resultados para otros valores de x . Por ejemplo, si queremos saber el resultado para $x = 5$, usaremos el modelo anterior

y veremos que el resultado es 1.7449:

$$y = 0.0918 \cdot 5 + 1.2859 = 1.7449$$

Este es un ejemplo muy simple. En realidad, los problemas de machine learning tienen muchas más variables. Sin embargo, he escogido este ejemplo porque es muy fácil de visualizar, explicar y entender. Espero que la intuición de este ejemplo sirva para entender lo que está pasando cuando haya más variables.

Notación

Antes de explicar el método de los mínimos cuadrados para resolver regresiones lineales, tenemos que expandir la notación. Debemos tener en cuenta que normalmente, tendremos muchas variables.

Con una variable, la ecuación para la regresión lineal es:

$$y = wx + b$$

Por conveniencia, vamos a reescribir la ecuación anterior:

$$y = b + wx$$

Cuando tengamos un dato con N variables, llamaremos al dato X . También tenemos que expandir los parámetros W para que cada parámetro vaya con una variable:

$$\begin{aligned} X &= [x_0, x_1, x_2, \dots, x_N] \\ W &= [w_0, w_1, w_2, \dots, w_N] \end{aligned}$$

Si hacemos que

$$x_0 = 1, w_0 = b$$

nos queda una expresión equivalente a la original. Podemos ver que:

$$y = b + wx = w_0x_0 + w_1x_1$$

Para el caso general, la ecuación lineal quedaría:

$$y = WX$$

usando el producto matricial. Si tienes la [multiplicación de matrices](#) un poco oxidada, la versión intuitiva sería:

$$y = w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_Nx_N$$

Formato de los datos

Para que la técnica de regresión lineal pueda aprender de nuestros datos, tenemos que proporcionar los resultados y en forma de vector de M elementos, y los datos de entrada X en forma de matriz. El convenio para la matriz X es el siguiente:

- M filas: cada fila es un dato (por ejemplo, un inmueble, si queremos predecir su valor de venta)
- N columnas: cada columna es un atributo relevante (por ejemplo, cuántas habitaciones tiene, metros cuadrados, etc).

Aprendizaje: El método de los mínimos cuadrados

El proceso de aprendizaje consiste en averiguar qué parámetros W minimizan el error cuadrático medio entre los resultados reales y los estimados.

El método de los mínimos cuadrados proporciona una solución analítica. Es decir, nos da una fórmula para obtener la mejor estimación posible de W para los datos de entrada y resultados que hemos proporcionado. La fórmula es la siguiente:

$$\hat{W} = (X^T X)^{-1} X^T y$$

En la práctica hay librerías numéricas que calculan automáticamente la mejor estimación de W por nosotros. Ya veremos algún ejemplo práctico del cálculo de regresión lineal.

De momento, sólo quería indicar que dependiendo de la cantidad de datos y atributos, puede ser una operación costosa computacionalmente hablando. Fíjate que hay que transponer matrices, multiplicar matrices e invertir matrices. Todo ello muy costoso computacionalmente para grandes cantidades de datos.

Otras formas de resolver el problema de la regresión lineal

El método de los mínimos cuadrados no es la única forma de estimar los mejores parámetros W . También podemos utilizar métodos de optimización numérica tales como el gradiente descendiente.

El gradiente descendiente va a servir no sólo para resolver regresiones lineales y polinómicas sino que es también fundamental para el aprendizaje automático de redes neuronales y aprendizaje profundo.

Ejemplo del uso del método de los mínimos cuadrados



Carl Friedrich Gauss

Carl Friedrich Gauss es famoso, entre otras muchas contribuciones, por la distribución Gaussiana o el método de los mínimos cuadrados para resolver regresiones lineales.

Cuentan que la primera aplicación del método de los mínimos cuadrados fue la determinación de la posición de Ceres. Ceres es un asteroide que se «perdió» a los 40 días de descubrirse. Realmente no se perdió, sino que al acercarse a la claridad del Sol, dejó de verse.

Varios científicos y astrónomos intentaron localizar Ceres. La única información que tenían eran los datos de su observación durante 40 días. Gauss fue el único capaz de predecir dónde se encontraría el asteroide Ceres cuando abandonó la parte del firmamento tan iluminada por el Sol. Para ello, Gauss usó el método de los mínimos cuadrados.

A finales de 1801 los astrónomos encontraron [el asteroide Ceres exactamente donde Gauss predijo que estaría](#).

Regresión Lineal en Python

Para hacer una regresión lineal en python, vamos a usar [scikit-learn](#), que es una librería de python para aprendizaje automático. En particular, la clase [LinearRegression](#) implementa la funcionalidad descrita en la parte teórica de este artículo. Vamos a explicarlo con un ejemplo.

Datos de ejemplo

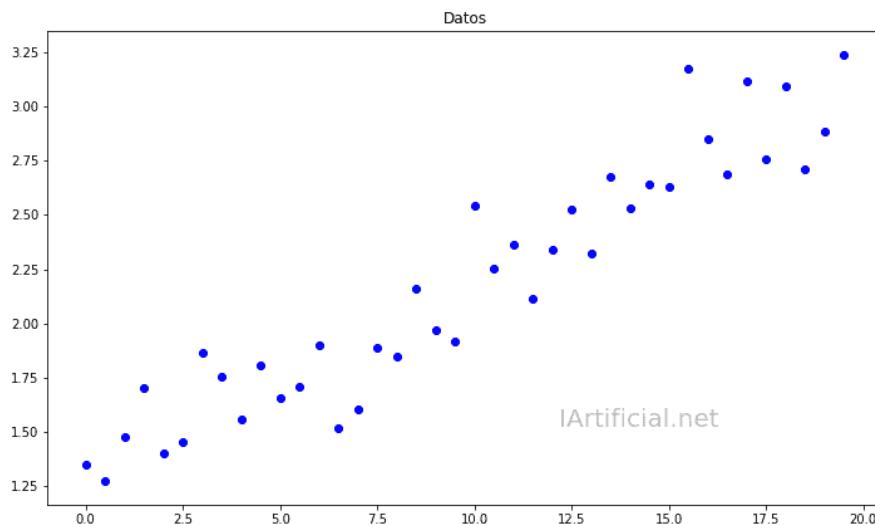
Primero vamos a generar unos datos que siguen una línea, y le añadimos ruido gaussiano. Para ello usaremos la librería de python [NumPy](#). La fórmula que he usado para generar los datos es:

$$y = 0.1x + 1.25 + N(0, 0.2)$$

El código en python quedaría así:

```
1. import numpy as np #Librería numérica
2. import matplotlib.pyplot as plt # Para crear gráficos con matplotlib
3. %matplotlib inline # Si quieres hacer estos gráficos dentro de un jupyter
notebook
4.
5. from sklearn.linear_model import LinearRegression #Regresión Lineal con
scikit-learn
6.
7. def f(x): # función f(x) = 0.1*x + 1.25 + 0.2*Ruido_Gaussiano
8.     np.random.seed(42) # para poder reproducirlo
9.     y = 0.1*x + 1.25 + 0.2*np.random.randn(x.shape[0])
10.    return y
11.
12. x = np.arange(0, 20, 0.5) # generamos valores x de 0 a 20 en intervalos de
0.5
13. y = f(x) # calculamos y a partir de la función que hemos generado
14.
15. # hacemos un gráfico de los datos que hemos generado
16. plt.scatter(x,y,label='data', color='blue')
17. plt.title('Datos');
```

Este código genera los datos que se ven en la siguiente imagen:



Datos de ejemplo para hacer una regresión lineal: $y = 0.1*x + 1.25 + N(0, 0.2)$

Entrenando un modelo de Regresión Lineal en python

Para entrenar el modelo, simplemente tendremos que hacer uso de scikit-learn. El método ***fit*** se encarga de ajustar los parámetros de regresión lineal a los datos.

```

1. # Importamos la clase de Regresión Lineal de scikit-learn
2. from sklearn.linear_model import LinearRegression
3.
4. regresion_lineal = LinearRegression() # creamos una instancia de
   LinearRegression
5.
6. # instruimos a la regresión lineal que aprenda de los datos (x,y)
7. regresion_lineal.fit(x.reshape(-1,1), y)
8.
9. # vemos los parámetros que ha estimado la regresión lineal
10. print('w = ' + str(regresion_lineal.coef_) + ', b = ' +
      str(regresion_lineal.intercept_))
11.
12. # resultado: w = [0.09183522], b = 1.2858792525736682

```

Como vemos, la regresión lineal casi ha averiguado cómo hemos generado los datos:

- Estima 0.092 en lugar de 0.1 para w
- Estima 1.286 en vez de 1.25 para b

Este pequeño error es normal debido a la cantidad de ruido gaussiano que hemos introducido y al hecho de que hay muy pocos datos.

Prediciendo con Regresión Lineal en python

Una vez que tenemos entrenado el modelo de regresión lineal, podemos hacer predicciones usando el método ***predict*** de la clase *LinearRegression*. Por ejemplo, si quisieramos saber qué valor de y corresponde para $x=5$ usamos este código.

```

1. # vamos a predecir y = regresion_lineal(5)
2. nuevo_x = np.array([5])
3. prediccion = regresion_lineal.predict(nuevo_x.reshape(-1,1))
4.

```

```

5.     print(prediccion)
6.
7.     # resultado: [1.7449]

```

Así vemos, que la estimación de la regresión lineal del modelo que acabamos de entrenar para $x = 5$ es $y = 1.7449$.

Evaluando la calidad de la regresión lineal

Vamos a evaluar la calidad del modelo aprendido usando solamente los datos de entrenamiento. Recuerda que en un problema real, hay que evaluar también la capacidad de generalización del modelo. Podemos evaluar la calidad del modelo midiendo el error cuadrático medio y el coeficiente de determinación R^2 .

Error Cuadrático Medio

```

1.     # importamos el cálculo del error cuadrático medio (MSE)
2.     from sklearn.metrics import mean_squared_error
3.
4.     # Predecimos los valores y para los datos usados en el entrenamiento
5.     prediccion_entrenamiento = regresion_lineal.predict(x.reshape(-1, 1))
6.
7.     # Calculamos el Error Cuadrático Medio (MSE = Mean Squared Error)
8.     mse = mean_squared_error(y_true = y, y_pred = prediccion_entrenamiento)
9.
10.    # La raíz cuadrada del MSE es el RMSE
11.    rmse = np.sqrt(mse)
12.
13.    print('Error Cuadrático Medio (MSE) = ' + str(mse))
14.    print('Raíz del Error Cuadrático Medio (RMSE) = ' + str(rmse))

```

Nos da el siguiente resultado:

Error Cuadrático Medio (MSE) = 0.033
Raíz del Error Cuadrático Medio (RMSE) = 0.182

Coeficiente de determinación R²

El coeficiente de determinación R^2 determina la calidad del modelo para replicar los resultados, y la proporción de variación de los resultados que puede explicarse por el modelo

Fuente: [wikipedia](#)

El rango de R^2 está entre 0 y 1, siendo 1 lo mejor. Para medir el coeficiente de determinación R^2 de la regresión lineal usaremos el método **score**.

```

1.     # calculamos el coeficiente de determinación R2
2.     r2 = regresion_lineal.score(x.reshape(-1, 1), y)
3.
4.     print('Coeficiente de Determinación R2 = ' + str(r2))

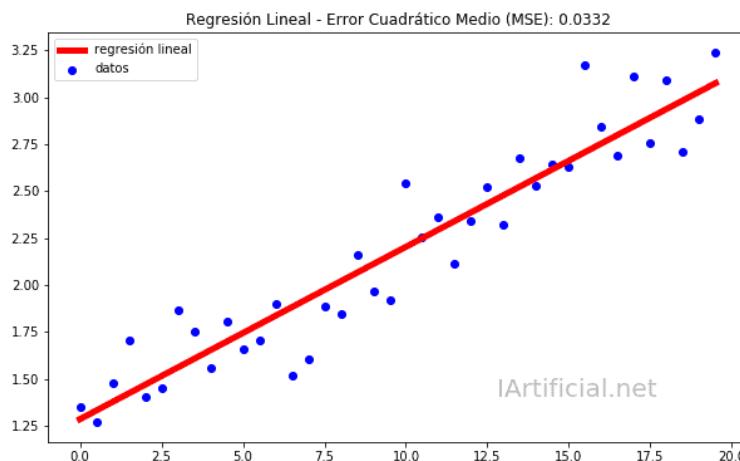
```

Nos da el siguiente resultado, que es bastante bueno:

Coeficiente de Determinación R2 = 0.894359363768311

Visualización

Siempre que podemos, es bueno visualizar los resultados, para saber cómo se está comportando el modelo. Este ejemplo que estamos haciendo es muy simple, y por tanto, muy fácil de visualizar. La línea que el modelo ha aprendido siguiendo el método de los mínimos cuadrados aparece en el siguiente gráfico en rojo.



Resumen

En este artículo hemos hablado de la regresión lineal. Esta es la lista de los puntos que hemos cubierto:

- matemáticas de regresión lineal con una variable
- extensión de la notación para varias variables
- el método de los mínimos cuadrados que Gauss nos proporcionó
- cómo entrenar una regresión lineal en python
- cómo hacer predicciones con una regresión lineal ya entrenada
- evaluación con el error cuadrático medio, el coeficiente de determinación R^2 y visualizando datos

Recursos

- [vídeo](#) donde explico la regresión lineal y polinómica (en inglés)
- Error Cuadrático Medio
- [Coeficiente de Determinación \$R^2\$](#)
- [LinearRegression](#): Documentación de la Regresión Lineal en la librería scikit-learn

EU Datathon 2019 – certamen de datos abiertos

Por Jose Martinez Heras
01/02/2019



EU Datathon 2019

EU Datathon 2019 es un certamen que tiene como objetivo fomentar el uso de los datos abiertos que proporciona la Unión Europea. El certamen permite a los concursantes mostrar sus habilidades en la utilización de los datos y también sus ideas innovadoras. Además, les ofrece la oportunidad de establecer contactos y encontrar apoyo para seguir desarrollando sus ideas después del certamen.

Se invita a los participantes a desarrollar aplicaciones y visualizaciones interactivas que ofrezcan nuevos servicios o información práctica de carácter público a los ciudadanos, las administraciones y las [empresas](#). Deberán utilizar al menos uno de los conjuntos de datos facilitados por las instituciones y agencias de la UE o por los socios internacionales del certamen, pudiendo vincular dichos datos con otros conjuntos de datos.

Los equipos participantes podrán competir con sus apps en uno o dos retos temáticos:

- **Ideas innovadoras a través de los datos abiertos de la UE**
- **Generación de nueva información en economía y finanzas**
- **Lucha contra el cambio climático**

Premios

Los participantes en la edición 2019 tendrán la posibilidad de ganar uno de los **premios en efectivo** otorgados en cada reto:

- 1^{er} premio: 15.000 euros
- 2^º premio: 7.000 euros
- 3^{er} premio: 3.000 euros.

El certamen culminará en la conferencia final que se celebrará en Bruselas el 13 de junio de 2019, en la cual los finalistas presentarán sus proyectos ante un jurado de expertos y profesionales de los datos abiertos.

Patrocinadores

Estos son los patrocinadores del certamen EU Datathon 2019:

- Agencia Europea de Medio Ambiente
- Banco Central Europeo

- Banco Europeo de Inversiones
- Centro Común de Investigación
- Dirección General de Asuntos Económicos y Financieros de la Comisión Europea
- Dirección General de Energía de la Comisión Europea
- Dirección General de Presupuestos de la Comisión Europea
- Eurostat
- Fondo Europeo de Inversiones
- Organización de las Naciones Unidas para la Agricultura y la Alimentación.

¿Vas a participar?

Si te animas a participar, tienes más información en la web del [EU Datathon 2019](#). Aunque no lo dice explícitamente, es una oportunidad de usar Machine Learning con estos datos. ¡Suerte!

Contraste de Hipótesis 1 – ¿cómo no aceptar lo falso?

Por José David Villanueva García
06/02/2019

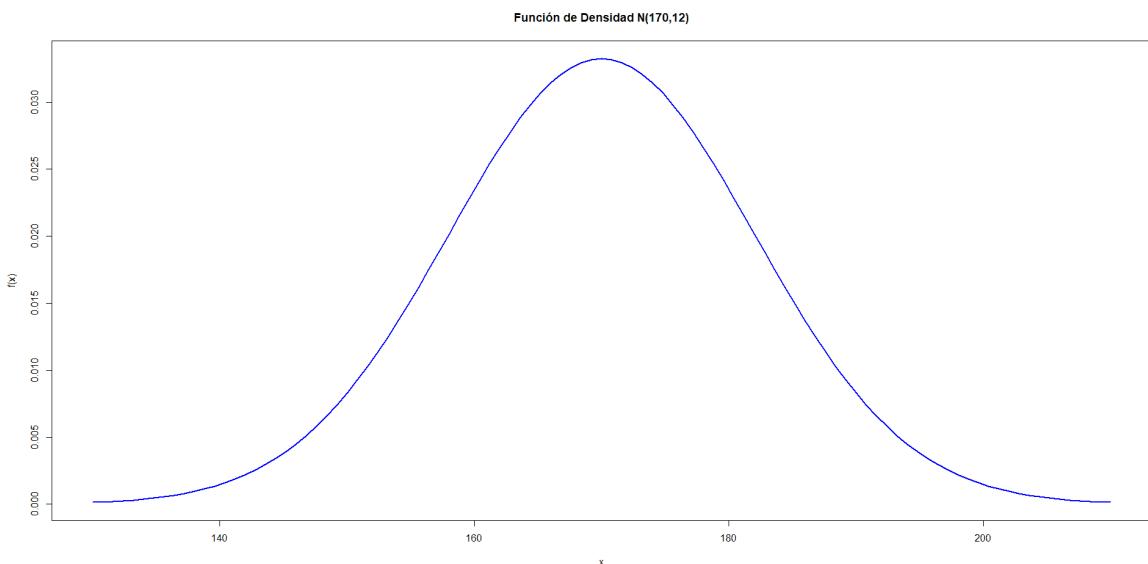
El contraste de hipótesis es una de las técnicas estadísticas más conocidas para juzgar si una determinada propiedad de una población es compatible con lo que podemos observar obteniendo una muestra de esta población.

Un poco de estadística básica

Vayamos poco a poco. Para comprender la base de la técnica del contraste de hipótesis, necesitamos repasar apenas unos cuantos términos sencillos de estadística.

- **Población estadística:** es el conjunto completo que vamos a estudiar. Por ejemplo, todos los hombres de España; o todos los mamíferos del mundo; o todos los estudiantes menores de 18 años.
- **Muestra:** es un subconjunto de la población. Una muestra de la población «todos los hombres de España» sería un conjunto determinado de los hombres de España, por ejemplo, 100 hombres, o 1000 hombres. Por supuesto, estas muestras se eligen de una determinada forma para que sea lo suficientemente «buena» (representativa) para toda la población.
- **Media:** supongamos ahora que queremos estudiar una propiedad sobre la población «todos los hombres de España», por ejemplo, la edad. Si cogemos una muestra de 100 hombres, la media muestral es la suma de las edades de los 100 hombres, dividida entre 100 (o el número de hombres en la muestra). Por lo tanto, la media de la población (denotada por μ) es la suma de las edades de todos los hombres de España, dividido entre el número de hombres en España. Esto es muy difícil de calcular, el simple hecho de preguntar a toda una población sobre una o más propiedades de ella es prácticamente imposible. Por ello, existen técnicas para estimar la media de la población
- **Desviación estándar:** al igual que la media, tenemos la desviación estándar poblacional (denotada por σ) y la muestral. Mide la variación de los datos respecto a la media. Una desviación estándar baja significa que los datos están agrupados cerca de la media, mientras que una desviación estándar alta indica que los datos se extienden sobre un rango de valores más amplio.
- **Distribución normal:** esta distribución permite modelar numerosos fenómenos naturales y sociales. Tiene forma de campana y usándola podemos calcular la probabilidad de un suceso ocurra dentro de un determinado rango de valores. La distribución normal se da en términos de dos parámetros: uno de ellos es la media y el otro puede ser la varianza o la desviación estándar. La varianza es el cuadrado de la desviación estándar.

La siguiente figura muestra el gráfico de la función de densidad de una distribución normal de media $\mu=170$ y desviación estándar $\sigma=12$, hecha con el programa R.



En todo caso, indicaremos cuándo la distribución está dada por la desviación estándar, $N(\mu, \sigma)$, o por la varianza, $N(\mu, \sigma^2)$.

En estos [Apuntes de Estadística](#) podéis encontrar un recurso muy bueno para repasar desde cero los conceptos básicos pero muy importantes sobre inferencia estadística.

¿Qué es un contraste de hipótesis?

Introducción

Seguro que alguna vez te has levantado a trabajar y has sentido un fuerte dolor de cabeza, dolor en la garganta y malestar en general. Sospechas que puedes tener fiebre. Tienes que comprobarlo. Para ello usarás un termómetro, y no tendrás fiebre si la temperatura que marca es menor o igual a 37 grados. Por lo tanto, si marca más de 37 grados, tendrás fiebre. Esto es un contraste de hipótesis. Basándonos en ciertos hechos, formulamos una hipótesis inicial y, dependiendo de cierto test, **la rechazamos o no** (quédate con esta última frase).

Las matemáticas del contraste de hipótesis

Existen dos tipos de contraste de hipótesis: **contrastos paramétricos y contrastes no paramétricos**. Los primeros son aquellos en los que la hipótesis concierne a parámetros poblacionales, como la media o la varianza. Los segundos son los que afectan a cualidades de la propia distribución, como la homogeneidad o la independencia. Nos centraremos en este post en los **contrastos paramétricos**.

Ya en términos estadísticos, los hechos en los que nos basaremos serán ciertas muestras que extraeremos de una población, la hipótesis que planteamos será la **hipótesis nula** (generalmente denotada por **H_0**) , y la hipótesis contraria, la **hipótesis alternativa** (generalmente denotada por **H_1**). La hipótesis nula es la que suponemos que es cierta. La hipótesis alternativa sustituye a la hipótesis nula cuando ésta es rechazada.

En nuestro ejemplo anterior, asumiendo como cierto **que no tenemos fiebre**, tendríamos que:

$$H_0 : \text{temperatura} \leq 37 \text{ grados}$$

$$H_1 : \text{temperatura} > 37 \text{ grados}$$

Ahora, haciendo un determinado test, **rechazaremos o no la hipótesis nula**. Como ves, es la segunda vez que repito esta última frase; y es que es muy importante, ya que lo que hacemos con la hipótesis nula es algo parecido a lo que se hace en un juicio, donde la hipótesis nula sería «el acusado es no culpable». Si tenemos suficientes pruebas, descartamos la hipótesis nula y nos

quedamos con la alternativa, es decir, declaramos al acusado culpable. Pero si no tenemos suficientes pruebas, no podemos descartar la hipótesis nula, es decir, no podemos declarar al acusado culpable. Pero eso no significa que la hipótesis nula sea correcta (que el acusado sea inocente), simplemente que no la podemos descartar.

Tipos de errores

Por supuesto, este método nos asegura que si rechazamos la hipótesis nula H_0 , la probabilidad de que efectivamente H_0 sea errónea es muy alta. Por lo tanto, tiene sentido fijar un número que nos indique la probabilidad de que, siendo H_0 verdadera, sea rechazada, es decir, cometer un error de tipo I. A este número le llamaremos **nivel de significación**, y lo denotaremos por α .

Es decir, fijando nuestra hipótesis nula como lo que creemos cierto, lo que nos interesa es que H_0 sea cierta, minimizando cometer un error de tipo I y rechazarla, por lo que generalmente escogeremos un α muy pequeño. En esta tabla resumimos los tipos de errores que podemos cometer.

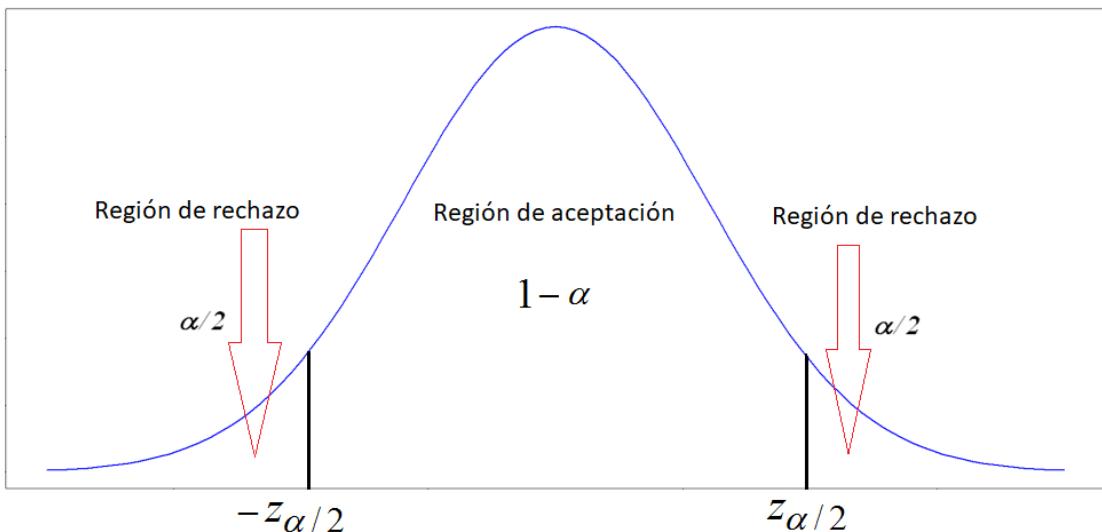
Tipo de errores	H_0 Verdadera	H_0 Falsa
Aceptar H_0	No error	Error tipo II
Rechazar H_0	Error tipo I	No error

Describamos los pasos necesarios para hacer un contraste de hipótesis desde el principio hasta el final.

1. Identificar el parámetro que vamos a estudiar, puede ser la media, la desviación típica, la proporción, etc., de una cierta población.
2. Formular la hipótesis nula y la hipótesis alternativa.
3. Fijar un valor para el nivel de significación α .
4. Elegir una muestra de tamaño n de nuestra población y obtener el valor del estadístico para esta muestra en concreto.
5. Determinar la región de aceptación y la región de rechazo.
6. Decidir si rechazamos o no la hipótesis nula.
7. Interpretar los resultados obtenidos.

Algunos de estos conceptos, como «el valor del estadístico» o «la región de aceptación y rechazo», no los hemos detallado todavía, pero los veremos pronto.

Una vez realizados los pasos del 1 al 5 , y asumiendo que la población sigue una distribución normal, esta es la pinta que tendrán los gráficos obtenidos, dependiendo de la condición que tengamos en la hipótesis nula.



Tipos de contraste de hipótesis

- Si la hipótesis nula se formula en términos «de igual» y la alternativa «de distinto», tendremos un **contraste de hipótesis bilateral**.

$$\begin{aligned} H_0 : \mu &= \mu_0 \\ H_1 : \mu &\neq \mu_0 \end{aligned}$$

- Si la hipótesis nula se formula en términos «de mayor o igual» y la alternativa en términos «de menor», tendremos un **contraste de hipótesis unilateral izquierdo**.

$$\begin{aligned} H_0 : \mu &\geq \mu_0 \\ H_1 : \mu &< \mu_0 \end{aligned}$$

- Si la hipótesis nula se formula en términos «de menor o igual» y la alternativa en términos «de mayor», tendremos un **contraste de hipótesis unilateral derecho**.

$$\begin{aligned} H_0 : \mu &\leq \mu_0 \\ H_1 : \mu &> \mu_0 \end{aligned}$$

Dependiendo del tipo de contraste, las zonas de rechazo y no rechazo cambiarán.

Ya estamos en condiciones de abordar técnicamente los contrastes de hipótesis. Veamos primero un caso poco probable en la realidad, pero muy ilustrativo para seguir avanzando.

Contraste sobre la media de una distribución normal de desviación estándar conocida.

El título suena complicado, pero no lo es en absoluto. Significa que tenemos una población que sigue una distribución normal de media μ desconocida y desviación estándar σ conocida, $N(\mu, \sigma^2)$, y lo que vamos a hacer es un **contraste de hipótesis** sobre la media. Para ello, veamos un ejemplo bastante ajustado a un ejemplo real.

Supongamos nos contrata un equipo de baloncesto interesado en incorporar nuevos jugadores a la plantilla. Sabemos que la altura del jugador es una característica muy importante en este deporte. La directiva te proporciona un estudio hecho hace 30 años en la ciudad sobre la altura de los jugadores de todos los equipos de la región. En este estudio puedes ver que la media de la altura hace 30 años era $\mu=170$ centímetros, que la varianza (el cuadrado de la desviación estándar) era $\sigma^2=500$ centímetros y que la población (todos los jugadores de todos los equipos de

la región) sigue una distribución normal $N(\mu, \sigma^2) = N(150, 500)$. Lo primero que queremos averiguar es si la media de la altura de los jugadores sigue siendo la misma después de 30 años, considerando la misma desviación estándar.

Sigamos los pasos establecidos anteriormente para este estudio.

1 Identificar el parámetro que vamos a estudiar

En este caso, el parámetro es la media μ .

2 Formular la hipótesis nula y la hipótesis alternativa

En nuestro estudio, queremos saber si la media de la altura ha cambiado en los últimos 30 años o no, es decir

$$\begin{aligned} H_0 : \mu &= 170 \\ H_1 : \mu &\neq 170 \end{aligned}$$

3 Fijar un valor para el nivel de significación α

Si rechazamos la hipótesis nula en favor de la alternativa, que el error de equivocarnos sea pequeño, por ejemplo fijamos $\alpha=0.1$ (10%).

4 Elegir una muestra de tamaño n de nuestra población y obtener el valor del estadístico para esta muestra en concreto

Supongamos que finamos $n=10$, y tomamos una muestra de este tamaño de la distribución normal con la que estamos trabajando, $N(\mu, \sigma^2) = N(150, 500)$, obteniendo los siguientes resultados:

176; 174; 152; 141; 192; 189; 190; 194; 191; 174

La media muestral (estadístico) es por tanto el resultado de sumar todos los valores y luego dividir entre 10, es decir,

$$\bar{x} = 177.3$$

5 Determinar la región de aceptación y la región de rechazo

Este es el paso en el que aún no hemos profundizado y lo haremos ahora. La media muestral obtenida en el paso anterior, proviene de una distribución normal, por lo que ella misma también sigue una distribución normal de la forma

$$N\left(\mu, \frac{500}{n}\right)$$

Procedemos entonces a **tipificar** la media muestral. Esto consiste en transformar la distribución de la media muestral en una distribución normal de media 0 y varianza 1, mediante la siguiente fórmula:

$$Z = \frac{\bar{x} - \mu}{\sqrt{\frac{\sigma^2}{n}}}$$

Ahora, nuestro estadístico Z (llamado **estadístico experimental**) sigue una distribución normal $N(0,1)$. Con el nivel de significación $\alpha=0.1$ fijado anteriormente, consultamos la tabla de la distribución $N(0,1)$ para los valores

$$Z_{1-\frac{\alpha}{2}}, Z_{\frac{\alpha}{2}}$$

que por la simetría de la distribución normal, es lo mismo que hallar en la tabla los valores

$$-Z_{\frac{\alpha}{2}}, Z_{\frac{\alpha}{2}}$$

Por lo tanto, nuestras regiones de aceptación y de rechazo son

- Región crítica o de rechazo:

$$C_r = (-\infty, -Z_{\frac{\alpha}{2}}) \cup (Z_{\frac{\alpha}{2}}, \infty)$$

- Región de aceptación:

$$C_a = (-Z_{\frac{\alpha}{2}}, Z_{\frac{\alpha}{2}})$$

Consultamos la tabla de la distribución normal para los valores anteriores, y obtenemos que nuestra región de aceptación para $\alpha=0.1$ es

$$(-1.64, 1.64)$$

Veamos el valor del estadístico experimental sustituyendo los valores en su fórmula. Tenemos que

$$Z = \frac{\bar{x} - \mu}{\sqrt{\frac{\sigma^2}{n}}} = \frac{\bar{x} - 170}{\sqrt{\frac{500}{n}}} = 1.032$$

El valor obtenido para el estadístico experimental **Z está dentro de la región de aceptación**, es decir,

$$Z = 1.032 \in (-1.64, 1.64)$$

6 Decidir si rechazamos o no la hipótesis nula

Como nuestro estadístico experimental está dentro de la región de aceptación, **se concluye que no se puede rechazar la hipótesis nula**. Es decir, no tenemos pruebas suficientes para concluir que la media de la altura de los jugadores de baloncesto de los equipos de la región haya cambiado después de pasados 30 años. Además, afirmamos esto con 90% de probabilidad de no equivocarnos.

7 Interpretar los resultados obtenidos

En el paso anterior, nuestro modelo ha decidido que no podemos rechazar la hipótesis nula, pero tenemos que tener en cuenta algunos factores muy importantes a la hora de entregar nuestro informe al cliente.

1.- Hemos asumido que la varianza de la población no ha cambiado. Esto es mucho asumir si no conocemos la media poblacional. Una forma de afinar nuestro análisis es suponer que tampoco conocemos la varianza poblacional. Todo es igual, pero tenemos un problema al tipificar la media muestral

$$Z = \frac{\bar{x} - \mu}{\sqrt{\frac{\sigma^2}{n}}}$$

Simplemente no podemos calcularlo, ya que no conocemos el parámetro σ^2 . No obstante, podemos estimarla con la **cuasivarianza**. En este caso, cuando tipificamos, el estadístico experimental obtenido ya no sigue una distribución normal sino una **distribución t de Student**, pero eso es otra historia que podemos contar en una segunda parte si estás interesados. Tampoco es muy complicado, simplemente tendremos que consultar las tablas para esta distribución en lugar de consultar las tablas de la distribución normal.

2.- Hemos supuesto también que la población sigue una distribución normal, lo cual habría que verificar utilizando también herramientas estadísticas.

Aplicaciones del Contraste de Hipótesis

El contraste de hipótesis es una herramienta estadística muy potente y es ampliamente utilizada en muchas ramas de la ciencia y de la sociología. Mostramos aquí algunos de los ejemplos más comunes:

Usos en medicina

En concreto, se utiliza ampliamente para **ensayos clínicos de nuevos medicamentos**. Entre ellos, se utiliza para comprobar si varios tratamientos son igualmente efectivos en términos de una variable cualitativa, es decir, cuando los valores que toma la variable son cualidades o categorías, o nombres. Por ejemplo: la variable sexo (Hombre, Mujer), la variable tener o no una determinada patología, etc.

Usos en ecología

Una de las cuestiones más importantes que se plantean los ecólogos es el análisis de las diferencias entre dos poblaciones. En este caso, plantear como hipótesis nula si hay diferencia entre dos poblaciones concretas carece de sentido, ya que no existen en la naturaleza dos seres vivos idénticos. Por lo tanto, lo que nos interesa en este caso es valorar **cómo de grande es la diferencia entre ambas poblaciones**.

Usos en sociología

Este es uno de los campos donde también el contraste de hipótesis es bastante utilizado. En política, un sociólogo puede pronosticar que en una región determinada el nivel de abstención será, digamos, de un 30%. Elige una muestra aleatoria de, pongamos, 500 individuos con derecho a votar y ya puede determinar con un nivel de significación de, por ejemplo, un 0.5%, si puede admitir el pronóstico inicial.

Resumen

Sabemos que no todos los problemas a los que se enfrenta un científico o un ingeniero se refieren a estimar únicamente un parámetro de la población, sino que generalmente hay que formular un proceso de decisión que está basado en datos y puede producir una conclusión acerca de algún sistema o población. Esto es justamente lo que hemos hecho en este post, se hace una conjetaura sobre el sistema o población que queremos estudiar y el procedimiento del contraste de hipótesis conduce a su aceptación o rechazo, basándonos en hipótesis estadísticas. La potencia de estos modelos es enorme, y es usado para tomar decisiones en asuntos tan sensibles como la medicina.

En siguientes posts, veremos cómo varían las distribuciones y el método dependiendo del conocimiento que tengamos sobre la población y, lo que es muy importante, **cómo obtener muestras representativas de estas poblaciones**.

Acerca del autor

Este es un post invitado por [José David Villanueva García](#). José David es Ingeniero Técnico en Informática de Sistema por la Universidad Rey Juan Carlos, Graduado en Matemáticas por la UNED, y Máster en Matemáticas Avanzadas por la UNED ([Máster Thesis](#)).

Actualmente trabaja como ingeniero en Darmstadt, Alemania, en diferentes proyectos para la ESA (European Space Agency) y EUMETSAT (European Organisation for the Exploitation of Meteorological Satellites).

Regresión Polinómica en Python con scikit-learn

Por Jose Martinez Heras
10/02/2019

En algunas ocasiones nos encontraremos con datos que siguen una función polinómica. En estos casos, el mejor modelo que podemos usar es la regresión polinómica. Este artículo explica la teoría detrás de la regresión polinómica y cómo usarla en python.

Regresión Polinómica – Teoría

La regresión polinómica es, en realidad, una regresión lineal. El truco está en:

1. Calcular atributos polinómicos
2. Usar la regresión lineal que ya hemos visto.

Vamos a verlo con fórmulas, porque creo que va a ser más fácil de entender. Antes poníamos en X todos nuestros atributos y usábamos el método de los mínimos cuadrados para obtener los mejores parámetros W . Antes X era:

$$X = [x_0, x_1, x_2, x_3, \dots, x_N]$$

Podemos ahora expandir los atributos calculando sus valores polinómicos:

$$X_{\text{polinómica}} = [x_0, x_1, x_1^2, x_2, x_2^2, \dots, x_N, x_N^2]$$

En este ejemplo, simplemente he elevado al cuadrado cada atributo ... lo que puede ser suficiente en algunos casos. Dependiendo del grado del polinomio y si deseamos multiplicar un atributo por otro, podemos expandir X mucho más. Afortunadamente, existen librerías de machine learning que hacen este proceso de expansión automáticamente siguiendo nuestras instrucciones.

¿Cómo hacer una Regresión Polinómica en Python?

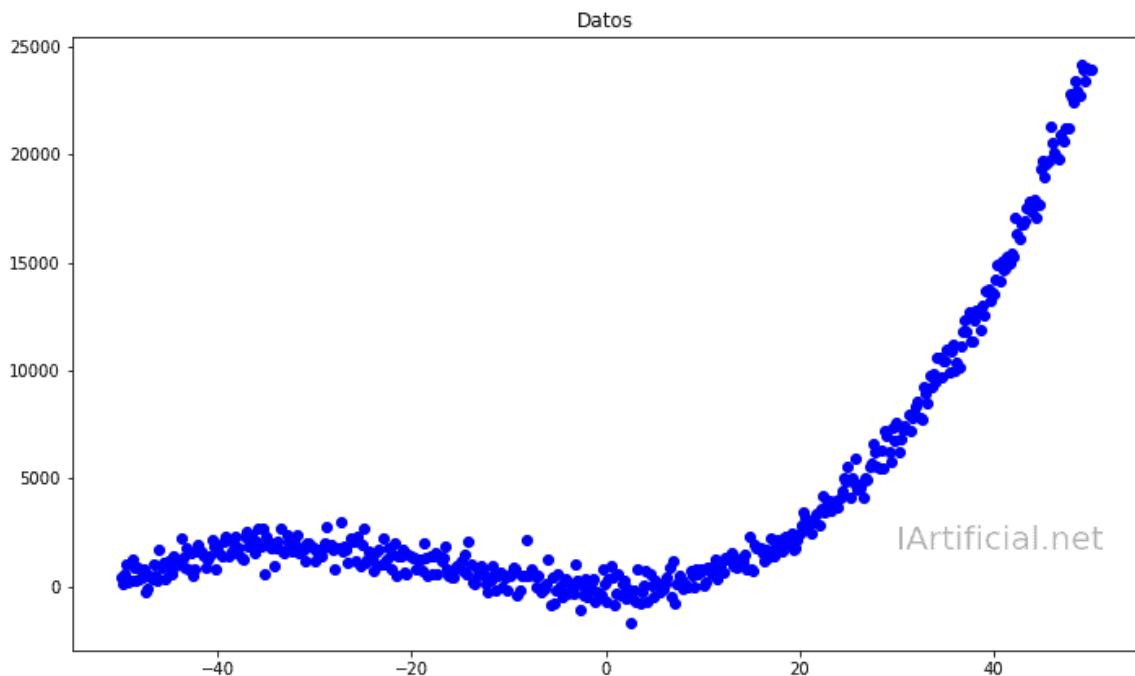
Para hacer la regresión polinómica, usaremos la librería [scikit-learn](#). Necesitaremos la clase [PolynomialFeatures](#) para que calcule los atributos polinómicos. Después usaremos la clase [LinearRegression](#) para hacer la regresión lineal con los datos polinómicos.

Datos de ejemplo

Como datos de ejemplo, he generado una función polinómica de grado 3 a la que le he añadido ruido Gaussiano. La función que he usado es:

$$y = -100 - 5x + 5x^2 + 0.1x^3 + N(0, 500)$$

La siguiente gráfica muestra los datos generados en esta fórmula



Construyendo un modelo de Regresión Polinómica en Python

Para entrenar un modelo de regresión polinómica, vamos a usar una regresión lineal con atributos polinómicos.

```

1. # Importamos la clase de Regresión Lineal de scikit-learn
2. from sklearn.linear_model import LinearRegression
3.
4. # para generar características polinómicas
5. from sklearn.preprocessing import PolynomialFeatures
6.
7. pf = PolynomialFeatures(degree = 3)      # usaremos polinomios de grado 3
8. X = pf.fit_transform(x.reshape(-1,1))    # transformamos la entrada en
polinómica
9.
10. regresion_lineal = LinearRegression() # creamos una instancia de
LinearRegression
11.
12. # instruimos a la regresión lineal que aprenda de los datos (ahora
polinómicos) (X,y)
13. regresion_lineal.fit(X, y)
14.
15. # vemos los parámetros que ha estimado la regresión lineal
16. print('w = ' + str(regresion_lineal.coef_) + ', b = ' +
str(regresion_lineal.intercept_))
17.
18. # resultado: w = [0 -4.54 4.95 0.1], b = -57.52

```

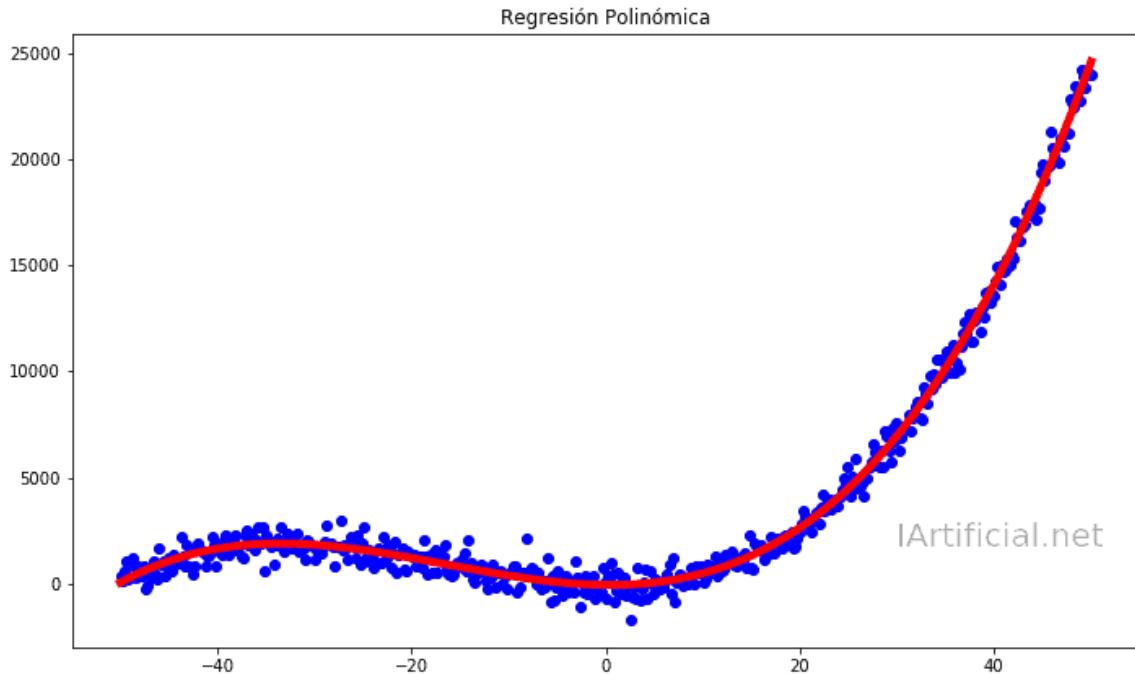
El resultado de la estimación es:

$$\hat{y} = -57.52 - 4.54x + 4.95x^2 + 0.1x^3$$

En vez de

$$y = -100 - 5x + 5x^2 + 0.1x^3 + N(0, 500)$$

Podemos ver que tiene algún error en la estimación de los parámetros, pero es normal para los pocos datos que le hemos dado. Sólo ha aprendido con 500 muestras en el intervalo [-50, 50] ... y hay muchos polinomios compatibles con estos datos.



Evaluación del modelo

El código para evaluar el modelo en entrenamiento es el siguiente:

```

1.  from sklearn.metrics import mean_squared_error # importamos el cálculo del
   error cuadrático medio (MSE)
2.
3.  # Predecimos los valores y para los datos usados en el entrenamiento
4.  prediccion_entrenamiento = regresion_lineal.predict(X)
5.
6.  # Calculamos el Error Cuadrático Medio (MSE = Mean Squared Error)
7.  mse = mean_squared_error(y_true = y, y_pred = prediccion_entrenamiento)
8.
9.  # La raíz cuadrada del MSE es el RMSE
10. rmse = np.sqrt(mse)
11.
12. print('Error Cuadrático Medio (MSE) = ' + str(mse))
13. print('Raíz del Error Cuadrático Medio (RMSE) = ' + str(rmse))
14.
15. # calculamos el coeficiente de determinación R2
16. r2 = regresion_lineal.score(X, y)
17. print('Coeficiente de Determinación R2 = ' + str(r2))

```

Y los resultados que nos da son los siguientes:

Error Cuadrático Medio (MSE) = 238896.642324

Raíz del Error Cuadrático Medio (RMSE) = 488.770541588

Coeficiente de Determinación R² = 0.993254587646

Como podemos ver, las medidas de evaluación del modelo son muy buenas:

- Un R² casi perfecto (lo máximo es 1)
- La Raíz del Error Cuadrático Medio ([RMSE](#)), es aproximadamente igual al ruido Gaussiano que introducimos en la función para generar los datos.

Resumen

Esta es la lista de los puntos que hemos cubierto:

- la regresión polinómica (también conocida como regresión polinomial) no es más que una regresión lineal con atributos polinómicos
- cómo hacer una regresión polinómica en python
- evaluación con el error cuadrático medio, el coeficiente de determinación R² y visualizando datos

Recursos

- [\[vídeo\]](#) donde explico la regresión lineal y polinómica (en inglés)
- Regresión Lineal
- Error Cuadrático Medio
- [Raíz Cuadrada del Error Cuadrático Medio](#)
- [Coeficiente de Determinación R²](#)
- [LinearRegression](#): Documentación de la Regresión Lineal en la librería scikit-learn
- [PolynomialFeatures](#): Documentación del preprocessado para crear atributos polinómicos en la librería scikit-learn

Feliz San Valentín – Amor en 20 minutos

Por Jose Martinez Heras
14/02/2019

¡Feliz San Valentín! Vamos a celebrarlo haciendo un análisis de todas las noticias relacionadas con el **amor** en el periódico [20 minutos](#). Para el análisis, usaremos técnicas de Procesamiento del Lenguaje Natural y Visualización de datos.

Los artículos de 20 minutos sobre amor, enamorar, enamorado/a ...

Para este análisis, vamos a usar los artículos de 20 minutos que tengan que ver con **amor, enamorar y enamorado/a**. Afortunadamente, 20 minutos ofrece la posibilidad de buscar por palabras clave. Así que se nos ha quedado [esta búsqueda](#).

Quisiera agradecer a [Alfonso Martínez Heras](#) su colaboración en este proyecto. Alfonso se ha encargado de crear un *web scrapper* para obtener estos artículos automáticamente.

Regalo de San Valentín para 20 minutos

Este es del regalo de San Valentín de [IArtificial](#) para el periódico [20 minutos](#). Es un corazón hecho con las palabras de los artículos relacionados con el amor. El tamaño de las palabras corresponde a la frecuencia de su uso. La posición y color son aleatorios. ¡Espero que os guste!



Los 14 artículos más interesantes

Vamos descubrir, automáticamente, cuáles son los 14 artículos más interesantes. Normalmente son los 10 más interesantes ... pero como San Valentín es el 14 de Febrero, me decanto por 14 !

Para encontrar los artículos más interesantes he usado técnicas de procesamiento del lenguaje natural [no-supervisado](#) (modelo bolsa de palabras, tf-idf, similitud coseno, modelado de temas, etc...). En otros posts más técnicos explicaré cómo funcionan. De momento, nos podemos quedar con que interpreto «*más interesante*» como «*más diferente* en comparación con el resto de artículos».

Estos son los 14 artículos más interesantes (diferentes):

1. [¿Cree usted en el amor verdadero? | El blog de Lilih Blue](#)
2. [Desmontando mitos machistas: "El amor puede con todo" | El blog ...](#)
3. [Unos científicos norteamericanos descubren que el amor verdadero ...](#)
4. [Siete historias de amor nacidas en los campos de concentración nazis](#)
5. [Haz estas 36 preguntas para enamorar a alguien](#)
6. [Lista: Mejores canciones de amor en español](#)
7. [El mágico y maravilloso amor entre amigas | El blog de Lilih Blue](#)
8. [La Trapecista Autómata estrena en Cuarta Pared «Tres canciones de ...](#)
9. [Vídeo: Paula Echevarría y Miguel Torres: un año de amor](#)
10. [Demuestran que existe esa fina línea que separa el amor del odio](#)
11. [Rosalía: «Lo he sacrificado todo por amor a la música»](#)
12. [Crítica de «La gran enfermedad del amor»: Aire fresco para la ...](#)
13. [Medalla del amor | Ya está el listo que todo lo sabe](#)
14. [Mireia y Gonzalo, una historia de amor sordo](#)

¡Feliz San Valentín!

Espero que te haya gustado este artículo dedicado al día de San Valentín. Deseo que te vaya muy bien en el amor ... y si no, por lo menos que te vaya bien en el juego! Lo dicho, feliz San Valentín y hasta el siguiente post.

Análisis Descriptivo, Predictivo y Prescriptivo de datos

Por Jose Martinez Heras
21/02/2019

Hay tres tipos de análisis de datos que podemos realizar: descriptivo, predictivo y prescriptivo. Veamos en qué consiste cada uno de ellos y cómo combinarlos.

Análisis Descriptivo

El análisis descriptivo se ocupa de estudiar el pasado. Como el nombre indica, el análisis descriptivo se usa para **describir lo que ha pasado**. Hay varias formas de describir el pasado:

- Usando **estadísticas** fáciles de entender, en plan: mínimo, máximo, media, mediana, cuartiles, desviación típica, los 10 mejores / peores. La misma información se puede dar agrupada. Por ejemplo, estadísticas de las ventas de una multinacional agrupadas por país.
- Usando **gráficos** que resuman los datos. Hay varios tipos de gráficos que pueden ser útiles. Por ejemplo, evolución de ventas, beneficios y costes como serie temporal. O también de distribución de datos; por ejemplo, histogramas con la edad de los clientes.
- Las **tablas** también son un aspecto fundamental del análisis descriptivo. Si habéis visto el balance de una empresa a final de año, sabes de qué estoy hablando.

 See what was trending in 2018 - Spain ↗

General	Personalidades	¿Cómo...?
1 Mundial	1 Rosalía	1 ¿Cómo hacer la declaración de la renta?
2 Cristina Cifuentes	2 Avicii	2 ¿Cómo reclamar irpf maternidad?
3 Fortnite	3 Stan Lee	3 ¿Cómo funciona Tinder?
4 Bohemian Rhapsody	4 Stephen Hawking	
5 Cómeme el donut	5 Freddie Mercury	

Ejemplo de análisis descriptivo de las tendencias de las búsquedas en Google en 2018

Análisis Predictivo

El análisis predictivo se refiere al uso de [aprendizaje automático supervisado](#). A efectos prácticos, normalmente consta de estos pasos:

1. Decidir **qué es lo que queremos predecir** en el futuro. Por ejemplo, predecir la probabilidad de que un cliente haga click en un anuncio de Internet.
2. Decidir **en qué datos van a estar basadas estas predicciones** (atributos relevantes). Por ejemplo, historial de compras, productos que ha buscado, productos que normalmente se compran juntos, etc.
3. Darle a la Inteligencia Artificial **datos históricos**. Debemos incluir tanto los atributos relevantes como el resultado que obtuvimos.
4. Como estamos interesados en predecir, tenemos que **estar seguros que la IA no haya aprendido los datos de memoria**. Es importante estar seguros que «ha entendido» los datos. Esto quiere decir que ha extraído un modelo que refleja la relación entre los datos de entrada y los resultados a predecir. Para estar seguros tendremos que medir la generalización del modelo y realizar un análisis de errores.

5. Una vez que tenemos un modelo de IA del que nos podemos fiar, sólo nos queda usarlo para **predecir**. Siguiendo el ejemplo anterior, podemos calcular la probabilidad de que un usuario haga click en varios anuncios. Después podemos ordenar los anuncios, de mayor a menor probabilidad, y mostrar los más probables.

Si quieres profundizar en las fases del proceso del análisis predictivo, puedes consultar este artículo.

Como resumen, quédate con que en el análisis predictivo, lo más importante es la calidad de la predicción.

Análisis Prescriptivo

Con el análisis prescriptivo intentamos cambiar el futuro. Es decir, vamos a usar Inteligencia Artificial para descubrir qué tenemos que hacer para obtener los resultados que queremos.

Una forma de realizar análisis prescriptivo es usando modelos de inteligencia artificial explicables. Por «explicables» quiero decir que podemos entender cómo el modelo de machine learning está realizando su predicción. Actualmente los modelos de machine learning que mejor resultados dan para predecir, son difíciles de interpretar. Por ejemplo, esto le pasa a las redes neuronales profundas (*deep learning*) y a los bosques aleatorios (*random forests*).

Por ejemplo, para realizar el análisis de las noticias de portada de menéame, usé regresión logística. La regresión logística es un modelo fácilmente interpretable. Así que podemos inspeccionar lo que ha aprendido y cómo los datos de entrada influyen en los resultados.

Si conseguimos encontrar qué causa los resultados que queremos ... entonces ya sabremos lo que tenemos que hacer. Aunque con cuidado.

Tenemos que tener cuidado cuando inspeccionamos los modelos de machine learning. Aunque un modelo sea muy bueno prediciendo, podría haber aprendido los datos de memoria. Primero hay que asegurarse de que el modelo generaliza bien.

También ayuda mucho la Inteligencia Humana (IH). Nosotros podemos en muchos casos validar lo que nos está diciendo la IA, especialmente si tenemos una mente abierta y sentido común.

Experimentos aleatorios (*randomized tests*)

Después de usar una IA explicable, asegurarnos que generaliza bien y usar nuestra inteligencia humana ... el último paso es realizar experimentos aleatorios (*randomized tests*). Estos experimentos se usan mucho en farmacia. Son el estándar para demostrar la efectividad de un medicamento. También se usan mucho para la optimización de sitios web y marketing. En este contexto se les llama [tests A/B](#).

Ejemplo de test aleatorio

Por ejemplo, creamos un modelo de machine learning para predecir cuánto tiempo un usuario va a visitar un sitio web. Usamos un modelo de machine learning explicable y nos damos cuenta que uno de los factores principales es el tamaño de letra de la página que visitan.

Podríamos concluir que lo que tenemos que hacer es aumentar el tamaño de letra de todos los artículos. Pero hay otros factores que no estamos considerando. Por ejemplo, a lo mejor los distintos tamaños de letra corresponden a las preferencias de los autores de los artículos. Así que, en estos casos, los visitantes del sitio web tendrían preferencia por el autor, en lugar de por el tamaño de letra.

Para estar seguros tendríamos que realizar un experimento aleatorio. Conforme vayan llegando usuarios, aleatoriamente les daríamos una de dos versiones: una en la que todas las páginas tienen una fuente mayor, y otra en la que la fuente es normal. Al medir los resultados de este experimento podríamos saber realmente si un tamaño de letra mayor **es una causa** del tiempo en el sitio web.

Resumen

En este artículo hemos visto las diferencias entre los análisis descriptivo, predictivo y prescriptivo. Una resumen rápido sería:

- **Análisis Descriptivo:** describe lo que ha pasado con estadísticas, gráficos, tablas e informes.
- **Análisis Predictivo:** realiza predicciones que van a ser útiles en el futuro. La calidad de la predicción es lo más importante
- **Análisis Prescriptivo:** ayuda a entender qué tenemos que hacer para obtener los resultados que queramos en el futuro

Redes neuronales desde cero (I) – Introducción

Por José David Villanueva García
28/02/2019

En este primer post de una serie de tres, hablaremos de una de las ramas más importantes del Machine Learning y la Inteligencia Artificial, **las redes neuronales**. Abordaremos este tema desde cero, empezando por la historia de las redes neuronales, sus conceptos básicos, nos adentraremos en las matemáticas que están involucradas en ellas e implementaremos un ejemplo de Redes Neuronales desde cero para reconocer cierto tipo de patrones en imágenes.

Introducción

Las redes neuronales (*neural networks*) se enmarcan dentro del campo de la Inteligencia Artificial. En primer lugar, decir que hace unos 25 años prácticamente nadie sabía lo que era Internet ni su significado, como podéis comprobar en este vídeo del programa [Today Show](#) de 1994. Las redes neuronales se inventaron en los años 60, ¿qué ha ocurrido entonces?

Un poco de historia

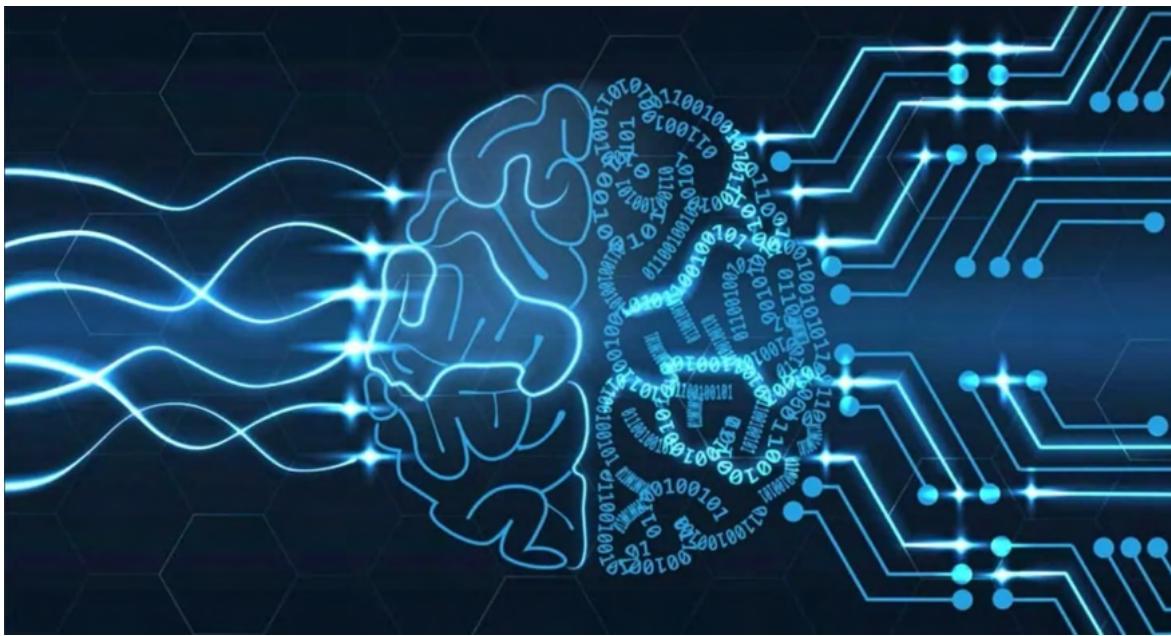
En 1994, los ordenadores tenían una capacidad muy limitada . Este es el principal motivo por el cual las redes neuronales no fueron usadas en esa época, a pesar de ser inventadas décadas antes. Los ordenadores no tenían todavía la capacidad de cálculo. Tampoco la capacidad de tratamiento y almacenamiento de datos para trabajar con redes neuronales. Sin embargo, los años 70 y 80 están caracterizados por películas futuristas con robots y máquinas que toman el control.

En **Terminator o Blade Runner**, donde se ve claramente que el concepto de máquina inteligente estaba ya latente en aquella época. Lamentablemente, en las siguientes décadas, este interés se fue apagando. Aunque había una amplia investigación en las universidades sobre la Inteligencia Artificial. De hecho, se llegó a pensar que el mundo de las redes neuronales llegó a su fin. Era una bonita teoría pero sin posible aplicación práctica.

Sin embargo, el crecimiento en todo los sentidos de los ordenadores ha sido exponencial desde aquella época.

Aprendizaje Profundo o Deep Learning

Seguramente habéis visto muchas veces una imagen como la siguiente:



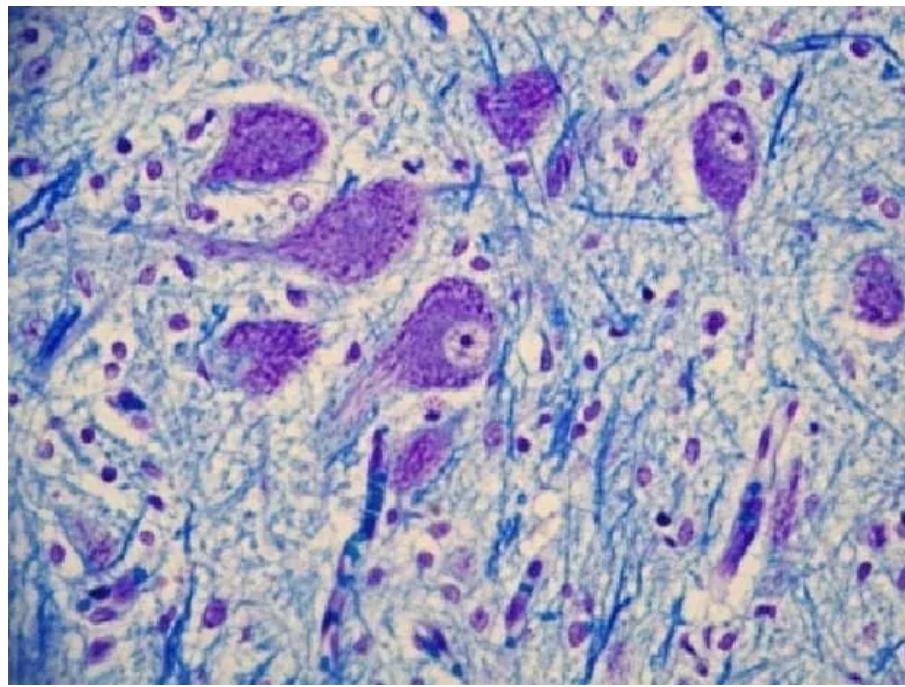
Deep Learning

con unas flechas de entrada a la izquierda, cierto cerebro las procesa y a partir de ese momento son transformadas en datos digitales:

El padre del concepto de Deep Learning fue el británico [Geoffrey Hinton](#). Investigó sobre el Deep Learning en los años 80, actualmente trabaja en Google, y muchas cosas de las que hablaremos provienen de la nomenclatura de Geoffrey Hinton. La idea principal que hay detrás del **concepto de Deep Learning es observar el cerebro humano e inspirarse en él para intentar reproducir de forma informática su comportamiento**. Por lo tanto, si lo que intentamos es imitar el cerebro humano, tenemos que llevar ciertos elementos de la neurociencia al ordenador. Veamos cómo hacemos esto.

La Neurona humana

En la siguiente imagen podemos observar neuronas humanas en el microscopio.



Neuronas del cerebro humano

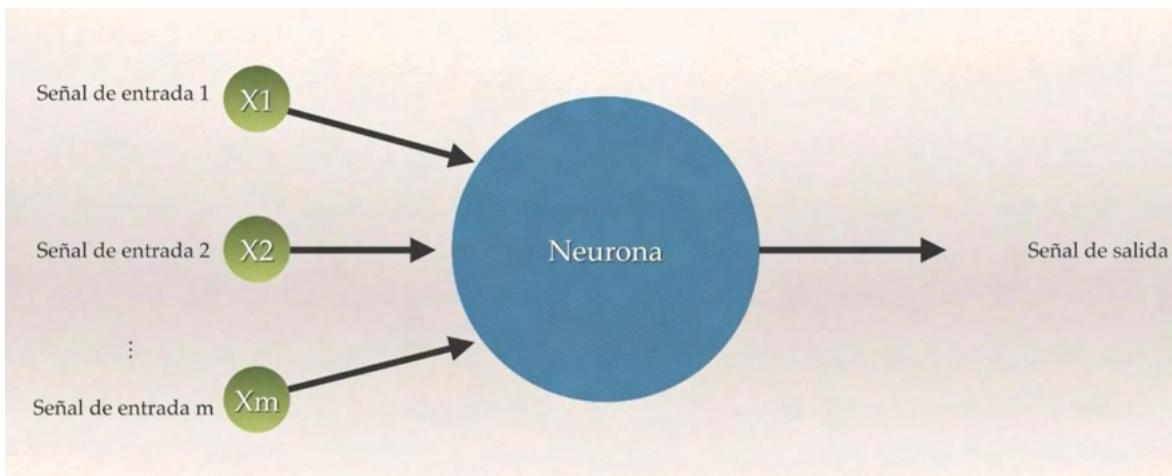
En ella podemos observar una especie de masa o cuerpo más o menos redondo. Además, vemos un núcleo o centro, y una serie de ramificaciones que se van extendiendo. En el cerebro, más o menos, tenemos unas cien mil millones de neuronas. Por lo tanto, lo que tratamos de hacer es trasladar el comportamiento del cerebro humano a una máquina, teniendo en cuenta que el conocimiento que tenemos del cerebro humano es todavía muy limitado.

Redes neuronales

Veamos entonces cómo pasamos de las neuronas que hay en el cerebro humano a formar una red neuronal en una máquina que simule su comportamiento.

La Neurona Artificial

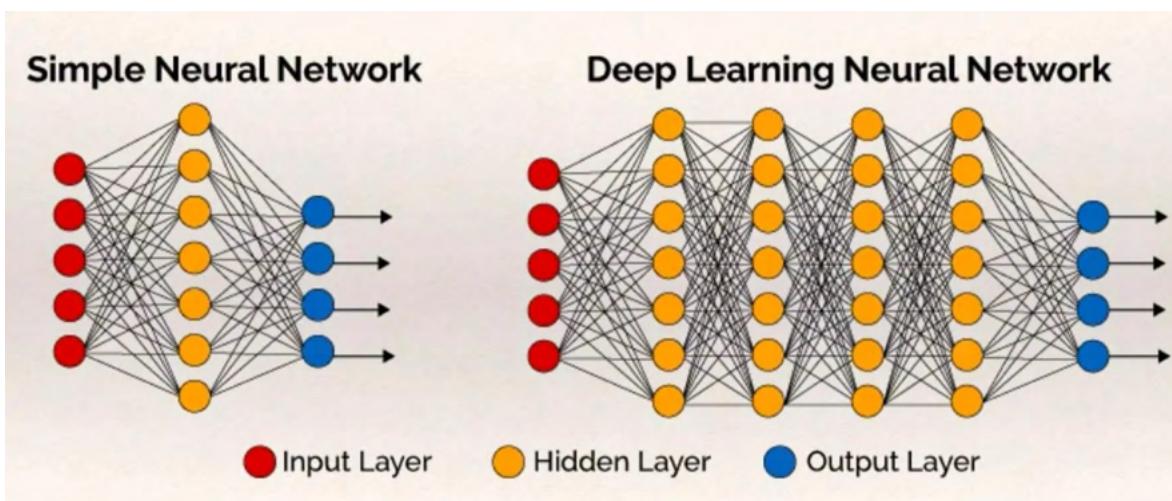
Las neuronas artificiales se modelan de tal forma que imiten el comportamiento de una neurona cerebral. Tendrán unas ramificaciones y un núcleo o nodo. Habrá ramificaciones de entrada al nodo, que serán las entradas de la neurona, procedentes de otras neuronas. Esta información se procesará en un nodo, y se generará una información de salida que se trasmisirán por las ramificaciones de salida a otras neuronas. Podemos pensar en las conexiones entre neuronas artificiales como en las **sinapsis** de las neuronas del cerebro. La imagen de una típica neurona artificial es la siguiente.



Neurona artifical

La red neuronal

En la siguiente imagen podemos ver algunos modelos de redes neuronales.



Red Neuronal simple y Red Neuronal profunda.

Vemos que las ramificaciones de salida de algunas neuronas son las ramificaciones de entrada de otras, y así sucesivamente. Pero vemos un par de diferencias entre las capas. Las neuronas de color rojo no tienen ramificaciones de entrada. Son información que vamos a dar a las neuronas «desde el exterior», o «estímulo inicial». También vemos que las neuronas de color azul tienen ramificaciones de salida que no están conectadas a otras neuronas. Son la información de salida de la red neuronal o «estímulo final». Dependiendo del número de capas ocultas (amarillas) podemos hablar de una red neuronal simple o profunda.

Valores de salida y pesos de las redes neuronales

Los valores de salida pueden ser continuos, como el precio de una determinada compra. También pueden ser binarios, como por ejemplo si una persona va a padecer una enfermedad o no. O pueden ser una categoría, como la marca de coche que más se venderá el próximo año. En este contexto, el caso binario es un caso particular donde tenemos dos categorías.

Para poder llevar a cabo el procesado, cada sinapsis tendrá asignado un valor o **peso**. Equivalen a la «fuerza» de la señal que se transmite por cada sinapsis. El ajuste de estos pesos para tener una red neuronal que haga lo que queremos es fundamental. Hablaremos de ello ampliamente cuando hablamos del **entrenamiento de la red neuronal**. Veremos métodos para ajustar estos

pesos, como el **descenso de gradiente** y el **backpropagation**. Veamos entonces cómo se procesa la información dentro de cada neurona.

Procesado de información de una neurona artificial

Las señales de entrada que vemos en la imagen anterior son **variables independientes**, parámetros de entradas que serán procesados por el cuerpo de la neurona artificial para finalmente propagar el resultado a una señal de salida. Por lo tanto, la flechas de transmisión de las entradas hacia el núcleo de la neurona harían el papel de la sinapsis de las neuronas cerebrales. Tendremos entonces una capa de neuronas de entrada, una capa de neuronas ocultas (en la imagen una) y una capa de neuronas de salida (una o más).

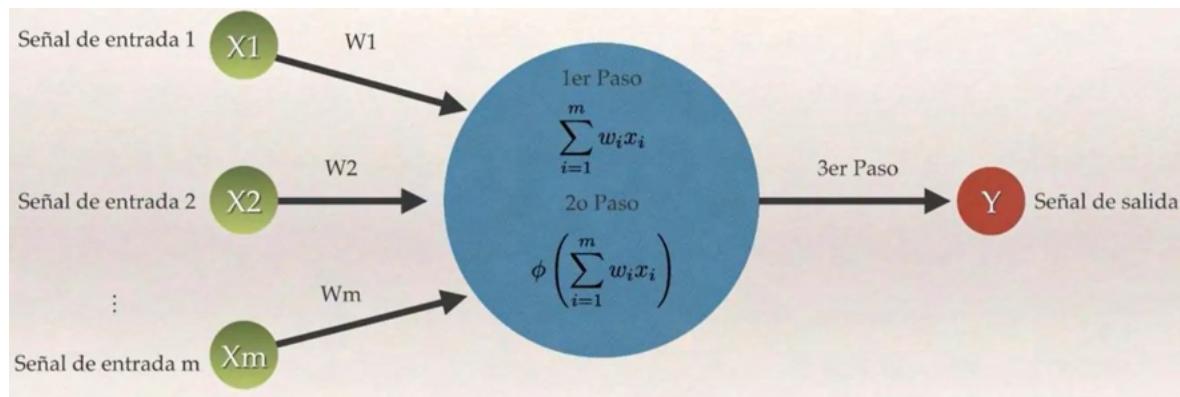
Todas las variables independientes de la capa de entrada, **pertenecen a una única observación**, una única muestra, «un sólo registro de la base de datos».

En el núcleo de la neurona es donde se procesan las señales de entrada y los pesos. Una de las formas es multiplicar cada señal de entrada por su peso y sumarlo todo, es decir, hacer una **combinación lineal** de los pesos de las conexiones y las entradas

$$w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_Nx_m$$

como vemos en el primer paso de la siguiente imagen. Posteriormente, se aplica una determinada función de activación al resultado del primer paso, y el resultado final se propaga a la salida, tercer paso de la imagen.

$$y = \phi(w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_Nx_m)$$



Procesado de información en una neurona

Veamos las cuatro funciones de activación más típicas que suelen usarse.

Funciones de activación

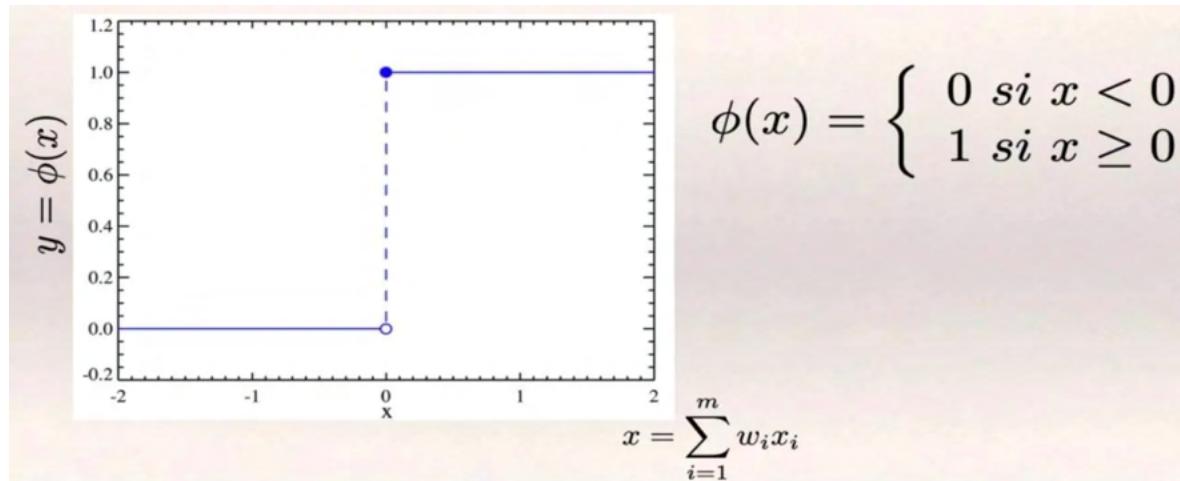
Primero se calcula la combinación lineal de los pesos y las entradas. Las funciones de activación no son más que la manera de trasmisir esta información por las conexiones de salida. Puede que nos interese transmitir esa información sin modificar, por lo que usaríamos sin más la **función identidad**. En otros casos, quizás no queramos que se transmita información alguna. Es decir, transmitir un cero en las salidas de la neurona.

En general, las funciones de activación se utilizan para dar una «no linealidad» al modelo y que la red sea capaz de resolver problemas más complejos. Si todas las funciones de activación fueran lineales, la red resultante sería equivalente a una red sin capas ocultas.

Vamos a ver las cuatro familias de funciones de activación más usadas en redes neuronales.

Función escalón (threshold)

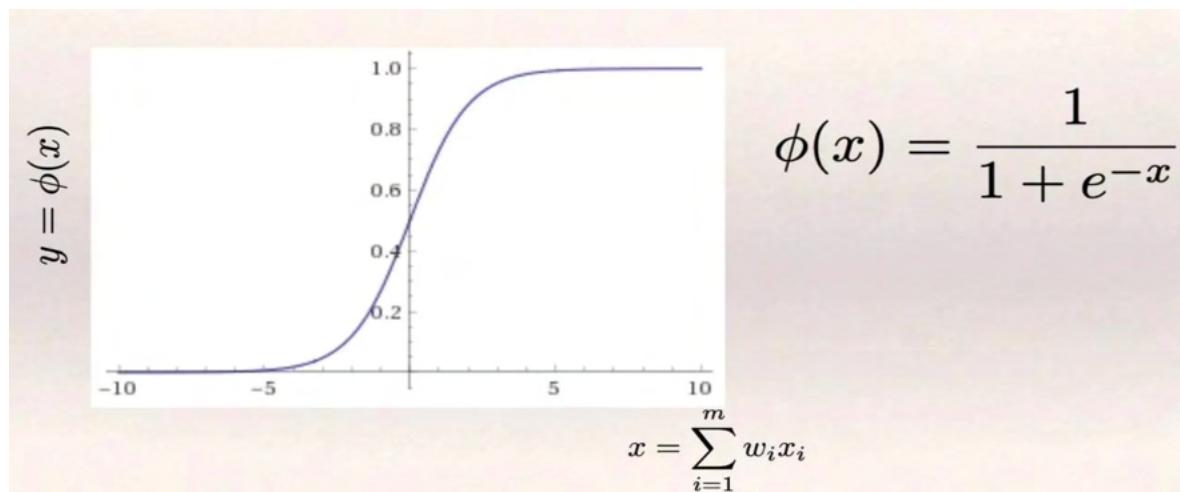
Como vemos en la figura de abajo, en el eje x está representado el valor ya ponderado de las entradas y sus respectivos pesos, y en el eje y tenemos el valor de la función escalón. Esta función propaga un 0 si el valor de x es negativo, o un 1 si es positivo. No hay casos intermedios, y sirve para clasificar de forma muy estricta. Es la función más rígida.



Función escalón

Función sigmoide

Como en la función escalón, existe una división entre los valores negativos y positivos de x, pero la función sigmoide no es tan estricta, el cambio se hace de manera suave. Se usa en la regresión logística, una de las técnicas más usadas en Machine Learning. Como vemos en la figura de abajo, la función no tiene aristas, es más suave (en términos matemáticos, es derivable). Cuanto más positivo es el valor de x más nos acercamos a 1 y por el contrario, cuanto más negativo es x más nos acercamos a 0.

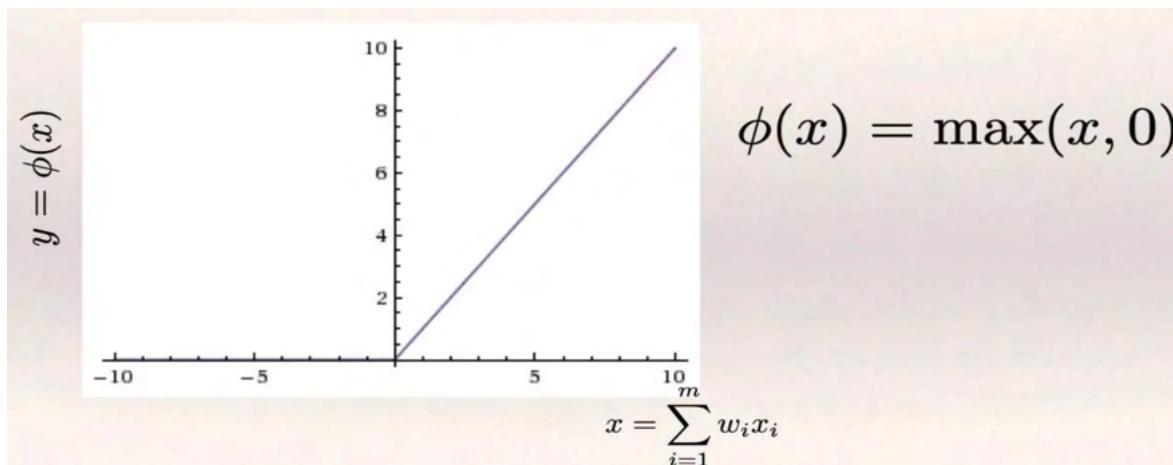


Función sigmoide

Esta función es muy útil en la capa final de salida al final de la red neuronal, no solo para clasificar con valores categóricos, sino también para intentar predecir las probabilidades de pertenencia a cada categoría, donde sabemos que la probabilidad de un suceso imposible es 0 y la de un suceso seguro es 1.

Función rectificadora (ReLU)

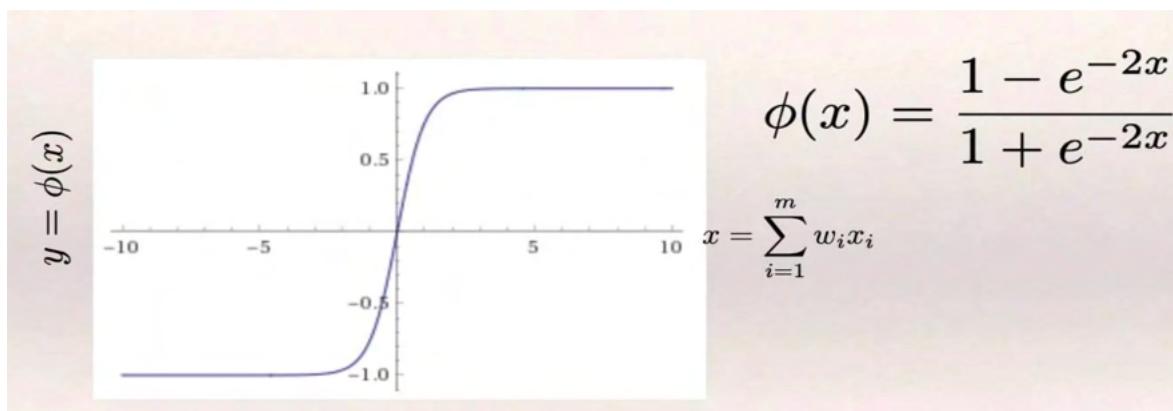
Volvemos a tener en esta función una arista. Vemos en la figura de abajo que su valor es 0 para cualquier valor negativo de x . Si x es positivo la función retorna el propio valor de x . Por lo tanto, se obvian todas aquellas entradas que una vez ponderadas tienen un valor negativo y nos quedamos con el valor exacto de las positivas. Vemos también que, al contrario de las anteriores, los valores no se restringen a valores entre 0 y 1.



Función rectificadora

Función tangente hiperbólica

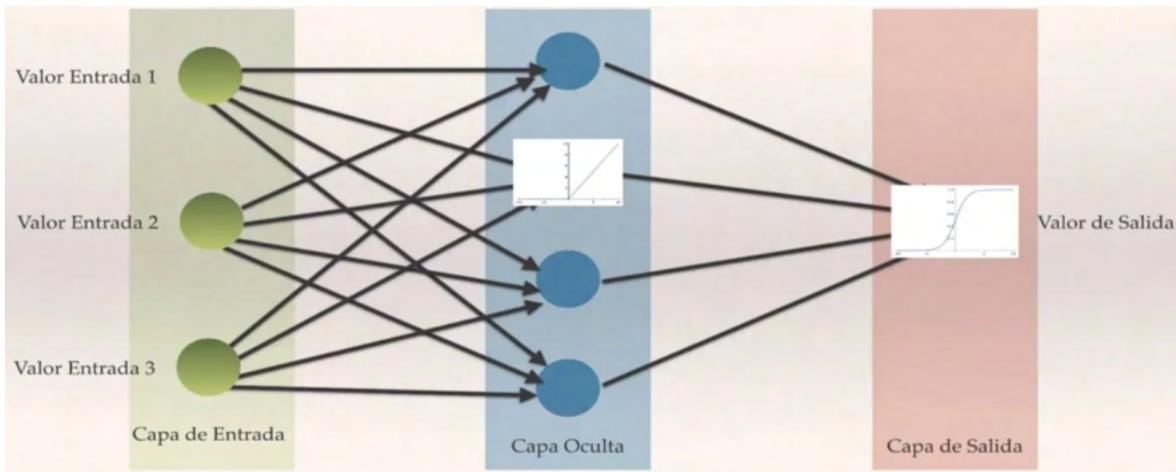
Esta función es muy parecida a la función sigmoidea, pero en este caso, como vemos en la figura de abajo, no empieza en 0 y termina en 1, sino que empieza en -1 y termina en 1.



Función tangente hiperbólica

Configuración de la red neuronal

Ya tenemos los ingredientes básicos para configurar una red neuronal. Una capa de neuronas de entrada, donde pasaremos información a la red a modo de estímulos, una capa de neuronas oculta, que se encargará del procesamiento de la información proporcionada por la capa de entrada, y una capa de salida que procesará la información proporcionada por la capa oculta para obtener el resultado final de la red neuronal para una entrada específica. Una posible configuración podría ser como la mostrada en la figura de abajo.



Ejemplo de configuración de una red neuronal

Decir que la ponderación de las señales de salida la hemos hecho con una combinación lineal, en la capa oculta usamos una función rectificadora y en la de salida usamos una función sigmodea.

Podéis usar en vuestras redes neuronales cualquier otro tipo de función que mejor se ajuste al problema que hay que resolver, esto es sólo un ejemplo.

La elección de una función de activación u otra dependerá de varios factores. Uno de ellos es el rango de valores que queremos en la salida de la neurona. Si no hay restricciones en el rango, utilizaremos la función identidad que propagará en la salida el valor de entrada. Si la restricción es únicamente que el valor de salida debe ser positivo, sin límite de cantidad, elegiremos una función rectificadora. Quizás nos interese una salida binaria 1 y 0 para clasificar, por lo que elegimos, por ejemplo, la función escalón. Puede que usemos un algoritmo de aprendizaje que hace uso de derivadas. Entonces utilizaremos funciones de activación donde la derivada esté definida en todo el intervalo. Por ejemplo la sigmoidal o la tangente hiperbólica, teniendo en cuenta que son más costosas en términos computacionales.

La función de costes

Durante el entrenamiento, tenemos los valores de entrada y sabemos cuáles son sus respectivas salidas. Lo que queremos es ajustar los pesos para que la red neuronal aprenda, y que los valores de salida de la red se acerquen lo más posible a los valores reales conocidos. Veamos con un pequeño ejemplo el proceso completo.

Supongamos que queremos construir una red neuronal que, una vez entrenada, prediga el precio de una casa. Para ello, recolectamos los siguientes datos sobre miles de casas:

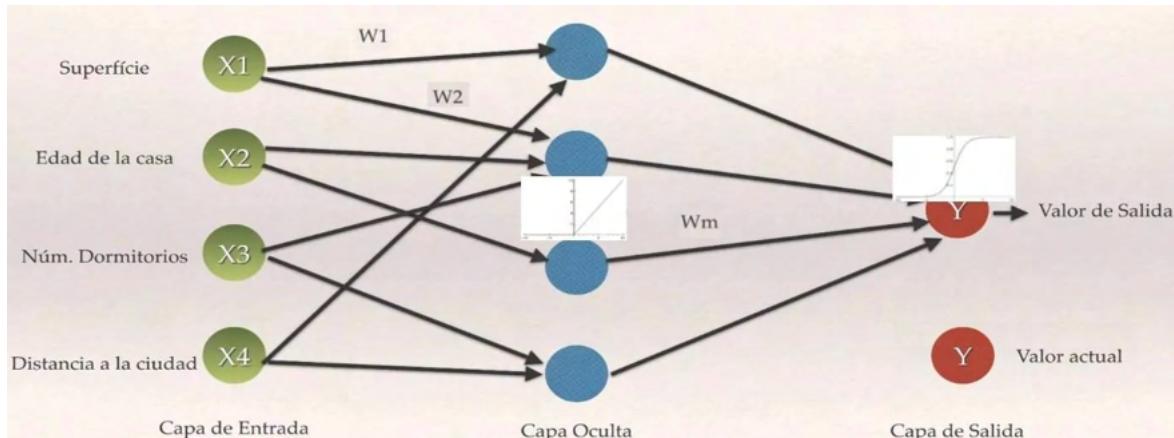
- Superficie
- Edad de la casa
- Número de dormitorios
- Distancia a la ciudad

Tendremos miles de registros en una base de datos, con información sobre las casas y su **valor actual**. Se inicializarán los pesos de las diferentes conexiones entre neuronas de alguna forma (ya veremos cómo). Empezaremos a introducir en la red los registros correspondientes a cada casa y, para cada uno de ellos, obtendremos un **valor de salida** que generalmente no serán iguales a los valores reales del precio de las casas.

La función de coste es una función que mide, en cierto modo, la «diferencia» entre el **valor de salida** y el **valor actual**. En este ejemplo usaremos el [error cuadrático medio](#). Por supuesto, puede ser una función más compleja.

Alimentamos en diferentes iteraciones a la red neuronal con los mismos **datos de entrenamiento** que tenemos, y en cada iteración intentamos **minimizar la función de coste**. Es decir que la

diferencia entre los valores de salida y los reales sean cada vez menores. Como los datos de entrada son los que son, sólo podemos conseguir esto **ajustando los pesos de las conexiones entre neuronas** al final de cada iteración.



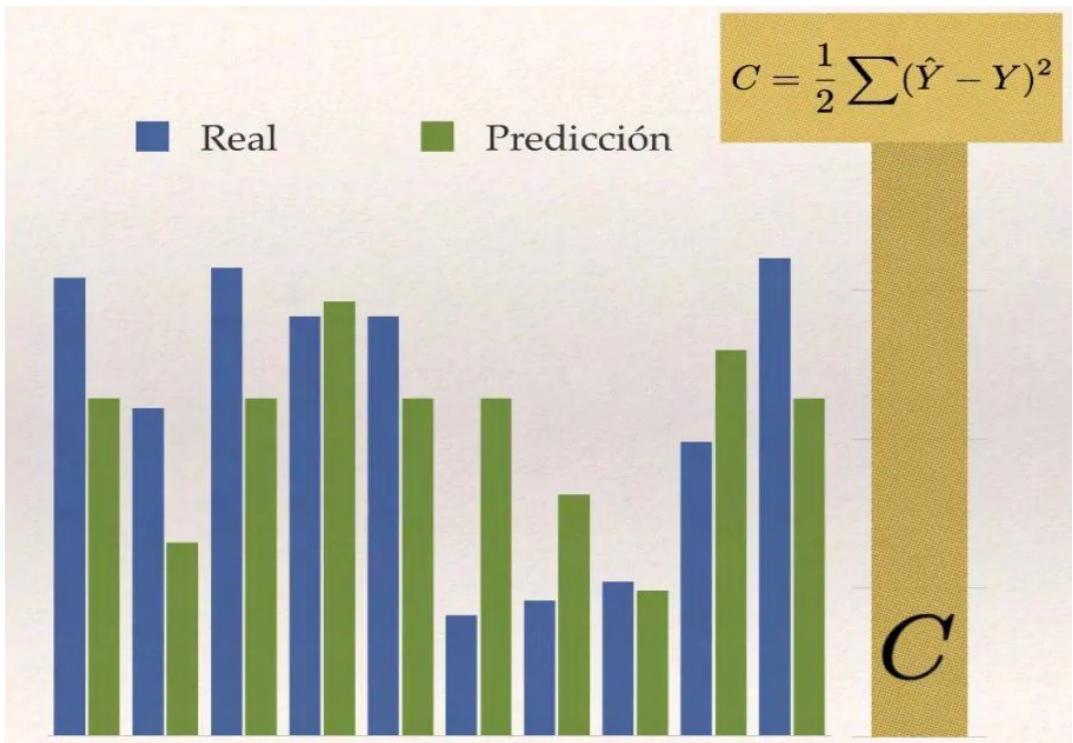
Ejemplo red neuronal

Valor real – predicción

Resumiendo, al final de cada iteración, para cada piso se obtendrá un valor de salida (predicción) de la red neuronal. Este valor diferirá en una cantidad con respecto al valor real que medirá la función de costes.

Ahora, actualizaremos los pesos de las conexiones neuronales y retroalimentaremos la red neuronal con los datos iniciales de los pisos. Se repite el proceso de nuevo y se van ajustando los pesos de forma que la función de costes sea cada vez menor.

Una manera de calcular la función de costes es como la que vemos en la figura de abajo. Calculamos el cuadrado de la diferencia entre el valor de salida y el real (evitando valores negativos) de cada casa. Luego sumamos el valor para todas las casas y dividirlo entre 2 (abordaremos las matemáticas en siguientes post).



Valores para la función de costes

Una vez finalizado el proceso, tendremos una red neuronal entrenada lista para predecir el precio de una casa suministrando los datos de esta. En general la red hará predicciones excelentes para datos que hemos usado para entrenarla. Pero para evaluar la calidad de predicción de la red neuronal es necesario usar datos que no hayan sido usados para optimizar los pesos de las conexiones de la red.

Resumen

Ya tenemos las nociones básicas de lo que es una red neuronal. Con toda la información que tenemos sobre un determinado problema, en nuestro caso precios de pisos, construimos una red neuronal que aprenda a partir de esos datos. Finalmente, la red será capaz de predecir correctamente cuando preguntemos por datos que la red nunca ha visto.

En este post hemos visto algunos conceptos matemáticos sencillos, como las funciones de activación. En el siguiente, nos adentraremos de verdad en las matemáticas que hay detrás de las redes neuronales. Las funciones de activación de las neuronas que hemos visto son muy importantes. Su correcta elección nos proporcionará una red neuronal que haga buenas predicciones. Básicamente, las funciones de activación que hemos visto devuelven, o bien un valor entre cero y uno, o bien el mismo valor de entrada si este es positivo (rectificador).

En el siguiente post veremos cómo se ajustan los pesos de las conexiones de la red neuronal. Para empezar a trabajar con ejemplos de redes neuronales, no es necesario conocer las matemáticas en las que se basan. Pero hay ventajas en conocer las matemáticas que hay detrás de cualquier algoritmo de machine learning. Cuando los resultados no sean lo suficientemente buenos, podremos obtener ideas de cómo mejorar el rendimiento. Si no tenemos esta base matemática, no tendremos más remedio que conformarnos con el resultado que tenemos.

Veremos que el problema del ajuste de pesos «por fuerza bruta», es decir, probando todas las posibilidades y quedándonos con la mejor, es en la práctica imposible ya que **tardaríamos millones de años de computación en encontrar la mejor solución**. Sin embargo, las matemáticas nos permiten encontrar esta solución en un tiempo razonablemente corto.

Actualización: la segunda parte ya está disponible: Redes neuronales desde cero (II): algo de matemáticas

Créditos

Las imágenes de este artículo han sido reproducidas, con permiso, del [Curso completo de Inteligencia Artificial con Python](#).

Acerca del autor

Este es un post invitado por [José David Villanueva García](#). José David es Ingeniero Técnico en Informática de Sistema por la Universidad Rey Juan Carlos, Graduado en Matemáticas por la UNED, y Máster en Matemáticas Avanzadas por la UNED ([Máster Thesis](#)).

Actualmente trabaja como ingeniero en Darmstadt, Alemania, en diferentes proyectos para la ESA (European Space Agency) y EUMETSAT (European Organisation for the Exploitation of Meteorological Satellites).

Gradiente Descendiente para aprendizaje automático

Por Jose Martinez Heras
06/03/2019

El gradiente descendiente es la base de aprendizaje en muchas técnicas de machine learning. Por ejemplo, es fundamental en deep learning para entrenar redes neuronales. También es necesario para la regresión logística. Y en muchos casos, al hacer regresión lineal o polinómica es mejor usar el método del gradiente descendiente que el de los [mínimos cuadrados](#).

Repasemos el Error Cuadrático Medio

Como vimos, el error cuadrático medio es una forma de medir el error de un modelo de aprendizaje automático para problemas de regresión. La fórmula intuitiva es:

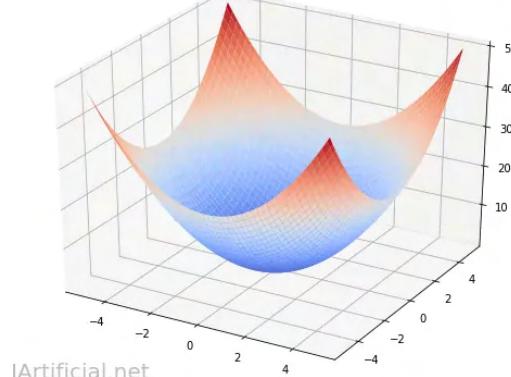
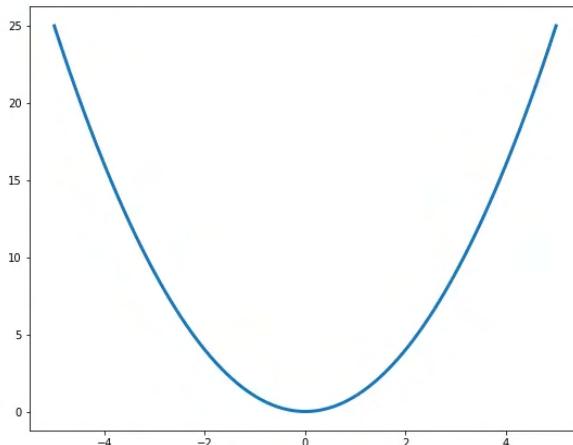
$$\text{error cuadrático} = (\text{real} - \text{estimado})^2$$

y matemáticamente podemos expresarlo así:

$$MSE = \frac{1}{M} \sum_{i=1}^M (\text{real}_i - \text{estimado}_i)^2$$

MSE significa *Mean Squared Error* – Error Cuadrático Medio en inglés.

Cuando estamos resolviendo problemas con una sola variable, el error cuadrático medio tiene forma de parábola. Con dos variables, de bol o de cuenco. Y con tres o más variables ... es difícil de saber qué forma tiene porque necesitaríamos 4 dimensiones para visualizarlo.



Función de Coste

La función de coste J es la función de lo que queremos optimizar. En el caso más simple, es igual al error que queremos minimizar.

$$J = MSE$$

Ya veremos que no siempre es tan simple. En algunos casos nos interesaría introducir términos de regularización, pero eso lo dejaremos para otro artículo.

¿Qué es el Gradiente?

Creo que la mejor forma de explicar lo que es el [gradiente](#), es diciendo que es una generalización de la derivada. Seguramente ya sabéis lo que es una derivada. Aquí tenéis un pequeño recordatorio.

En matemáticas, la **derivada** de una función mide la rapidez con la que cambia el valor de dicha función matemática, según cambie el valor de su variable independiente. La derivada de una función es un concepto local, es decir, se calcula como el límite de la rapidez de cambio media de la función en cierto intervalo, cuando el intervalo considerado para la variable independiente se torna cada vez más pequeño. Por ello se habla del valor de la derivada de una función *en un punto dado*.

Fuente: [wikipedia](#)

El gradiente es el conjunto de todas las derivadas parciales de una función. En el caso de machine learning, estamos interesados en el gradiente de la función de coste.

Método del Gradiente Descendiente

En muchas de las técnicas de aprendizaje automático, el «aprendizaje» consiste en encontrar qué parámetros W minimizan la función de coste. Esto es así para la regresión lineal y polinómica, la regresión logística, el deep learning, etc. El gradiente descendiente es un método de optimización numérica para estimar los mejores coeficientes.

Ejemplo intuitivo

¿Te acuerdas cuando eras pequeño y en el colegio había que hacer alguna cuenta? Y en vez de hacerla como te habían enseñado la hacías por tanteo ... o como decimos en España «por la cuenta de la vieja». Pues justo así funciona el gradiente descendiente. Sorprendente! Sabías lo que era aunque todavía no supieras como se llamaba.

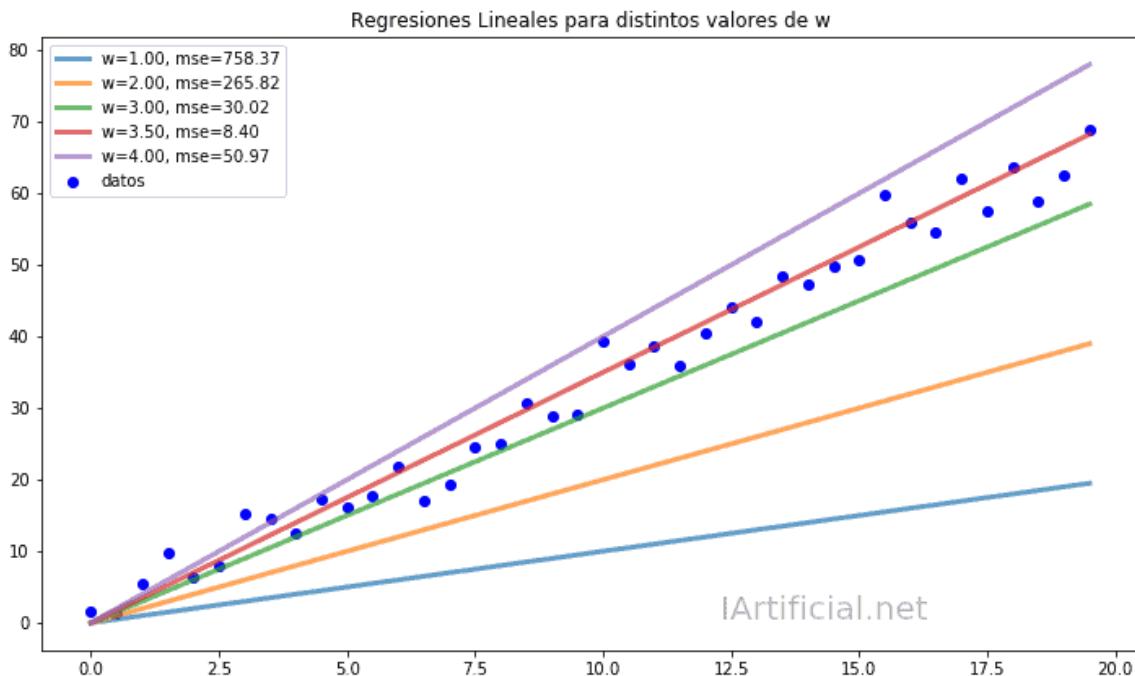
Pongamos por ejemplo que tenemos una función f que es una línea pero no sabemos cuáles son sus parámetros. Tenemos los datos para la x , tenemos los datos para la y ... pero no sabemos los parámetros de la función f en la expresión

$$y = f(x) = wx + b$$

En este ejemplo, para hacerlo más simple, vamos a suponer que $b=0$.

Haciendo el gradiente descendiente por tanteo

Podemos ir probando. Por ejemplo, para $w = 1$, tendríamos un error cuadrático medio (mse) de 758.37. Podemos probar con otro w para ver si aumentamos w o lo disminuimos. Si probamos con $w = 2$, el mse sería 265.82. Parece que aumentar el valor para w es una buena idea. Si lo seguimos aumentando a $w = 3$, el mse es de 30.02. ¡Vamos por el buen camino! vamos ahora a aumentarlo hasta $w = 4$... para 4, el mse es 50.97. Claramente nos hemos pasado, podemos echar para atrás y probar $w = 3.5$ que nos da un mse de 8.40.



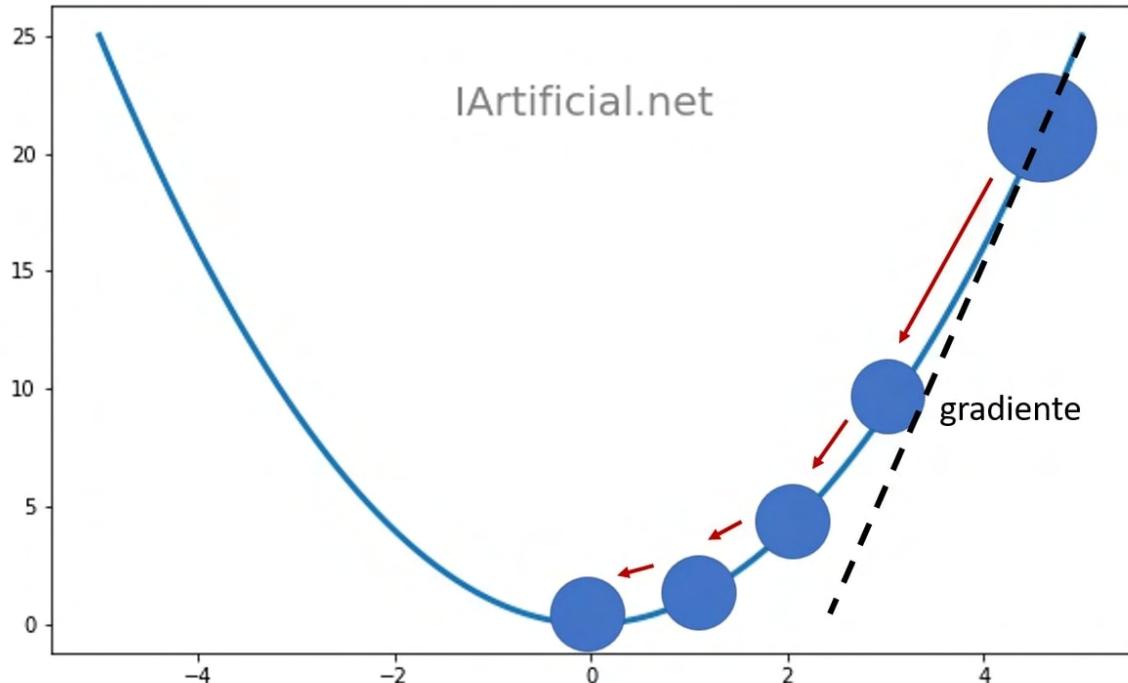
En esta lista resumimos los intentos que hemos hecho hasta que dar con la solución. ¿Que cómo sabemos que $w = 3.50$ es la solución? Porque es la fórmula que he usado para generar los datos de este ejemplo. Ya veremos cómo hacerlo cuando no sepamos la solución correcta.

1. $w = 1.00, \text{mse} = 758.37$
2. $w = 2.00, \text{mse} = 265.82$
3. $w = 3.00, \text{mse} = 30.02$
4. $w = 3.50, \text{mse} = 8.40$
5. $w = 4.00, \text{mse} = 50.97$

Las matemáticas del Gradiente Descendiente

El método del gradiente descendiente nos permite automatizar de forma más eficiente el ir probando coeficientes de los modelos de machine learning. Se puede usar tanto con modelos simples (como el que hemos visto en el ejemplo) como en modelos complejos (por ejemplo, redes neuronales con muchas variables).

Lo que vamos buscando es encontrar el conjunto de parámetros que minimizan la función de coste (error cuadrático medio). Como vemos en el gráfico, el gradiente representa la pendiente en el punto que nos encontramos de la función de coste. Cuanto más «empinada» sea la pendiente, más queremos ir en sentido contrario (para abajo), para minimizar el error.



El gradiente lo podemos calcular con derivadas parciales:

$$\frac{\partial \text{MSE}}{\partial W}$$

Así que podemos actualizar los coeficientes W usando esta regla:

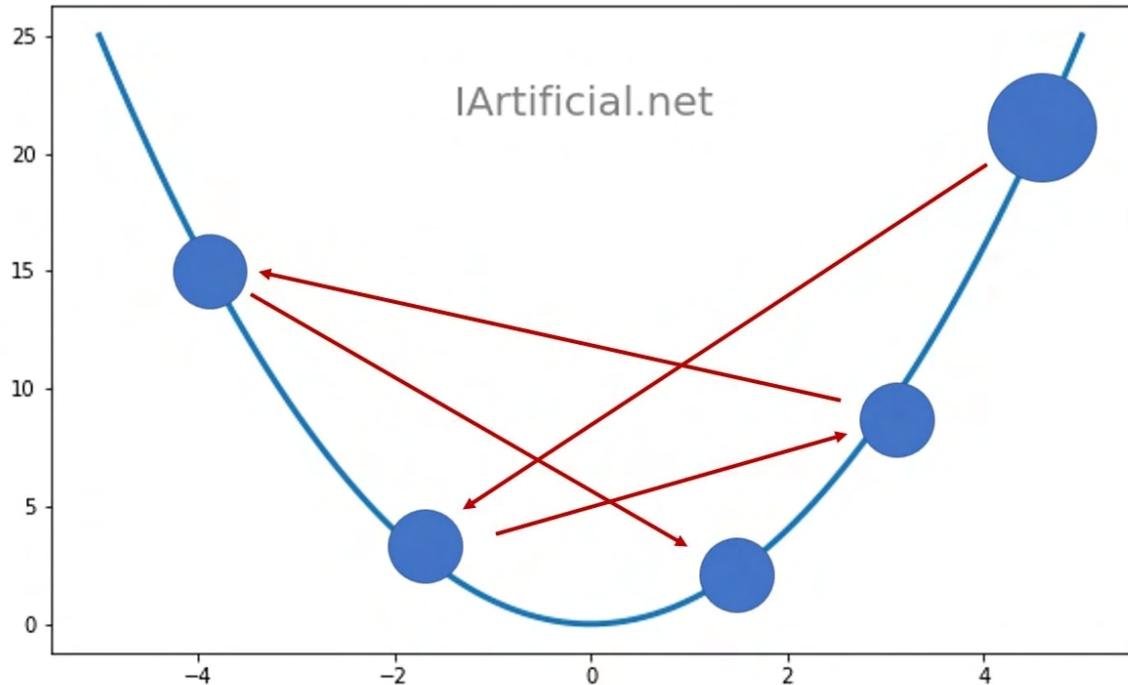
$$W \leftarrow W - \alpha \frac{\partial \text{MSE}}{\partial W}$$

Podemos ver varias cosas interesantes:

- Usamos ‘-’ (resta), para ir en la dirección opuesta al gradiente
- La actualización de W es proporcional al gradiente (a mayor pendiente [más empinado], mayor es el cambio)
- α , también llamado ‘ratio de aprendizaje’, controla el tamaño de la actualización

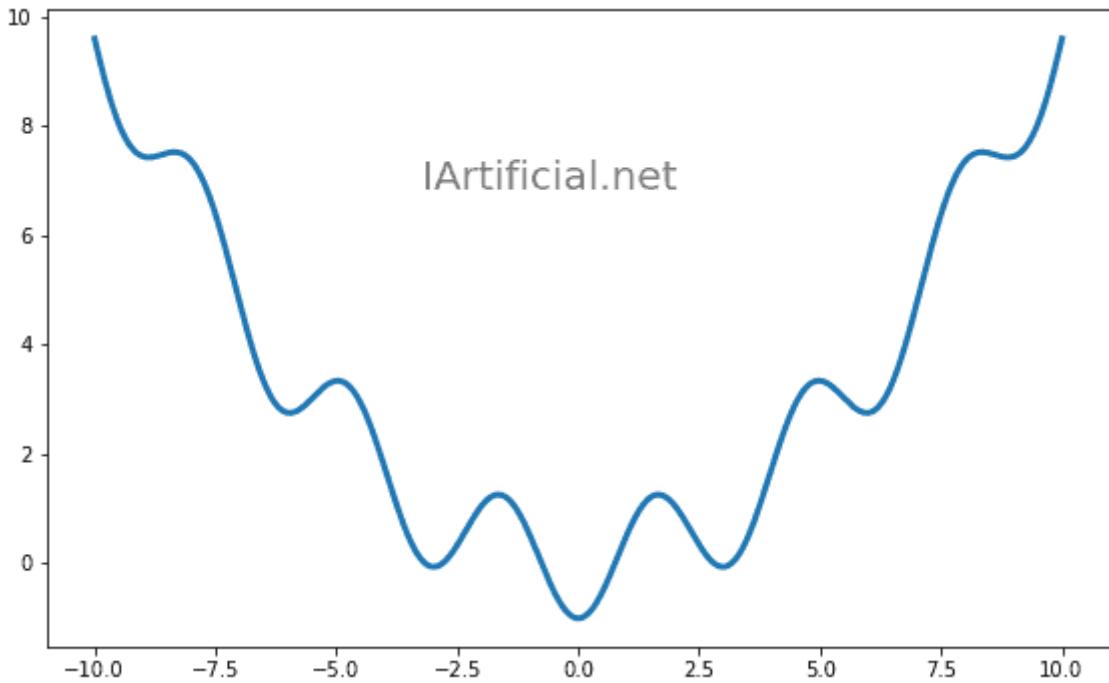
La importancia de elegir un buen ratio de aprendizaje

El ratio de aprendizaje (*learning rate* en inglés) es necesario para asegurarnos que no vamos dando tumbos. Me explico: si el ratio de aprendizaje es demasiado grande, los cambios en W serán también muy grandes y será difícil encontrar los coeficientes que minimicen la función de coste.



Por otra parte, si el ratio de aprendizaje es demasiado pequeño, el gradiente descendiente tardará mucho en encontrar la solución adecuada.

¿Y si el gradiente descendiente se queda atrapado en un mínimo local?



Cuando usamos el gradiente descendiente nos arriesgamos a caer en un mínimo local. Cuando caemos en un mínimo local, distintos valores de W' cercanos a W tendrán un valor de la función de coste mayor que en el punto W . El gradiente descendente nos devolverá pues a W una y otra vez. Así que creeremos que hemos encontrado la mejor solución. Sin embargo, el mínimo global bien pudiera estar en otra parte.

Durante un tiempo, esto trajo de cabeza a muchos científicos de datos en la era del *data mining*. Se usaron varias estrategias para mitigar el efecto de caer en un mínimo local. Una estrategia muy popular por su simplicidad era la de empezar el gradiente descendente en distintos puntos al azar y elegir la mejor solución.

En la práctica, no debemos preocuparnos

Actualmente se ha comprobado que, en la práctica, el riesgo de caer en un mínimo local es muy bajo. Esto se debe que ahora los problemas que resolvemos con machine learning tienen muchas variables. Esto resulta en un gran número de dimensiones si lo interpretamos geométricamente. Así que lo que antes era un mínimo local ahora se denomina [punto de silla](#) (*saddle point* en inglés).

Un **punto de silla** o **punto de ensilladura** es el punto sobre una superficie en el que la pendiente es cero pero no se trata de un extremo local (máximo o mínimo). Es el punto sobre una superficie en el que la elevación es máxima en una dirección y mínima en la dirección perpendicular.

Fuente: [wikipedia](#)

En los puntos de silla, el gradiente disminuye hasta cero en algunas dimensiones, pero sigue habiendo gradiente relevante en otras dimensiones.

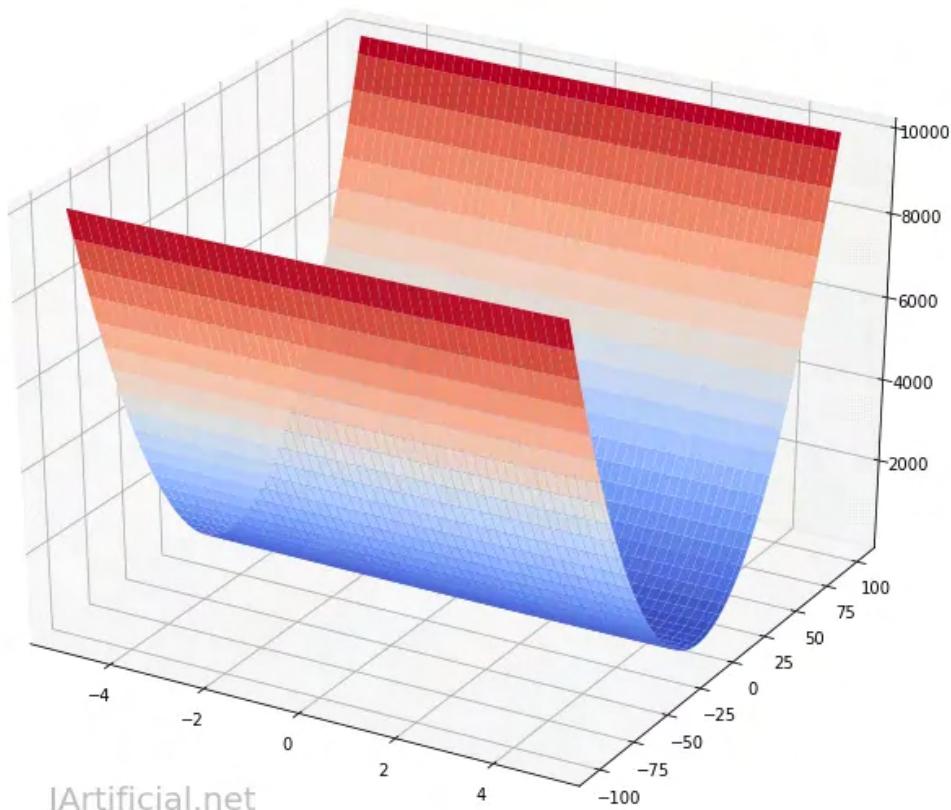
Información adicional

Hemos visto el uso del gradiente descendente para un problema de regresión donde la función de coste es el error cuadrático medio. El gradiente descendente también puede usarse para problemas de clasificación. En estos casos, la función de coste suele ser la entropía cruzada. También podemos usar otras funciones de coste o incluso diseñar las nuestras.

La versión del gradiente descendente que hemos visto es la más básica y fácil de entender. Hay varias mejoras que se han ido realizando a lo largo de los años que ya veremos en otros artículos. Si quieres profundizar puedes leer acerca del gradiente descendente estocástico, momentum, AdaGrad, RMSProp, Adam, etc. Algunas de estas mejoras hace que el elegir el ratio de aprendizaje adecuado no sea tan relevante ya que lo van adaptando sobre la marcha.

La importancia de escalar los datos al usar el Gradiente Descendiente

El gradiente descendiente funciona mucho mejor cuando los datos tienen una escala similar. Esto se debe a la forma geométrica de la función de coste. Cuando todas las dimensiones tiene una escala similar, podemos esperar la forma de cuenco multidimensional. Sin embargo, si distintas dimensiones tienen una escala diferente, la forma geométrica será muy diferente. Esto hace que el gradiente descendiente tarde mucho en converger.



Como vemos en la siguiente figura, el atributo que está entre [-100, 100] tiene mucha más influencia en la función de coste que el atributo que está el rango [-5, 5]. La consecuencia es que la derivada parcial de algunos atributos es mucho mayor que la derivada parcial de los otros. Esta diferencia provoca que se sobre-optimalicen algunos coeficientes en detrimento de otros. Al final, posiblemente, todos los coeficientes acabarán optimizados; eso sí, a costa de un número muy superior de iteraciones.

Este es un caso muy frecuente. La mayoría de los problemas que queremos resolver tienen atributos relevantes en escalas diferentes. Por ejemplo, si queremos estimar el valor de una propiedad inmobiliaria, los atributos «número de dormitorios» y «metros cuadrados» estarán en dos escalas muy diferentes.

Hay varias estrategias para escalar los datos. Las más comunes son:

- Estandarizado de datos: transformación de los datos para que cada atributo tenga una media de 0 y una desviación típica de 1
- Escala mínimo máximo: después del escalado, cada atributo estará en el rango [0, 1]
- Escala logarítmica: transformación de los datos aplicando el operador logaritmo

¿Gradiente Descendiente o Mínimos Cuadrados para la Regresión Lineal?

Cuando estemos resolviendo una regresión lineal o polinómica, ¿nos interesa más el método del gradiente descendiente o el método de los mínimos cuadrados? La respuesta es: *depende*.

Con el método de los mínimos cuadrados obtenemos una solución exacta. Está matemáticamente demostrado que la estimación de los coeficientes W es la que minimiza el error cuadrático medio. Con el método del gradiente descendiente obtenemos una solución aproximada y no hay

garantías. Aunque normalmente encontramos una solución que está muy cerca del mínimo teórico siempre que usamos un ratio de aprendizaje adecuado y un número suficiente de iteraciones.

Entonces, ¿por qué vamos a usar un método aproximado cuando podemos usar uno exacto? La respuesta es que el método del gradiente descendiente tiene otras ventajas con respecto al método de los mínimos cuadrados.

El [método de los mínimos cuadrados](#) es más complejo computacionalmente. Si repasamos la fórmula podemos ver que tiene una transposición de matrices, varias multiplicaciones de matrices y una inversión de matrices.

$$\hat{W} = (X^T X)^{-1} X^T y$$

Las operaciones de transponer, multiplicar e invertir matrices son atómicas. Es decir, las podemos hacer o no, pero no hay forma de transponer un poco, o invertir una matriz un 20%.

El método del gradiente descendiente sí que permite optimizar «un poco». Así que podemos usarlo para resolver problemas del tipo «tienes 2 minuto para estimar W ». Además, el gradiente descendiente puede encontrar soluciones casi óptimas en mucho menos tiempo que el método de los mínimos cuadrados para problemas con muchos datos. Cuando hablamos de muchos datos nos referimos tanto al número de muestras como al número de atributos.

Resumen

En este post hemos hablamos del gradiente descendiente. El método del gradiente descendiente es muy usado para entrenar redes neuronales y también en aprendizaje profundo (*deep learning*). En particular, nos interesa utilizar el gradiente descendiente para estimar los parámetros W que minimizan la función de coste. También hemos visto por qué es tan importante escalar los datos antes de utilizar este método de optimización numérica. Por último, hemos visto que para un conjunto de datos muy amplio, el gradiente descendiente nos puede interesar más que el uso del método de los mínimos cuadrados.

Recursos

- [\[vídeo\]](#) donde explico cómo funciona el gradiente descendiente para resolver regresiones lineales (en inglés)

Regularización Lasso L1, Ridge L2 y ElasticNet

Por Jose Martinez Heras
14/03/2019

En muchas técnicas de aprendizaje automático, el aprendizaje consiste en encontrar los coeficientes que minimizan una función de coste. La regularización consiste en añadir una penalización a la función de coste. Esta penalización produce modelos más simples que generalizan mejor. En este artículo vamos a hablar de las regularizaciones más usadas en machine learning: Lasso (también conocida como L1), Ridge (conocida también como L2) y ElasticNet que combina tanto Lasso como Ridge.

¿Cómo funciona la regularización?

Cuando vimos el gradiente descendente, usamos el error cuadrático medio como función de coste J .

$$J = \text{MSE}$$

Cuando usamos regularización, añadimos un término que penaliza la complejidad del modelo. En el caso del MSE, tenemos:

$$J = \text{MSE} + \alpha \cdot C$$

C es la medida de complejidad del modelo. Dependiendo de cómo midamos la complejidad, tendremos distintos tipos de regularización. El hiperparámetro α indica cómo de importante es para nosotros que el modelo sea simple en relación a cómo de importante es su rendimiento.

¿Por qué funciona la regularización?

Cuando usamos regularización minimizamos la complejidad del modelo a la vez que minimizamos la función de coste. Esto resulta en modelos más simples que tienden a generalizar mejor. Los modelos que son excesivamente complejos tienden a [sobreajustar](#). Es decir, a encontrar una solución que funciona muy bien para los datos de entrenamiento pero muy mal para datos nuevos. Nos interesan los modelos que además de aprender bien, también funcionen bien con datos nuevos.

Regularización Lasso (L1)

En la regularización Lasso, también llamada L1, la complejidad C se mide como la media del valor absoluto de los coeficientes del modelo. Esto se puede aplicar a regresiones lineales, polinómicas, regresión logística, redes neuronales, máquinas de vectores de soporte, etc. Matemáticamente quedaría:

$$C = \frac{1}{N} \sum_{j=1}^N |w_i|$$

Para el caso del error cuadrático medio, este es el desarrollo completo para Lasso (L1):

$$J = \frac{1}{M} \sum_{i=1}^M (\text{real}_i - \text{estimado}_i)^2 + \alpha \frac{1}{N} \sum_{j=1}^N |w_i|$$

¿Cuándo es efectiva Lasso (L1)?

Lasso nos va a servir de ayuda cuando sospechemos que varios de los atributos de entrada (features) sean irrelevantes. Al usar Lasso, estamos fomentando que la solución sea poco densa. Es decir, favorecemos que algunos de los coeficientes acaben valiendo 0. Esto puede ser útil para descubrir cuáles de los atributos de entrada son relevantes y, en general, para obtener un modelo que generalice mejor. Lasso nos puede ayudar, en este sentido, a hacer la selección de atributos de entrada. Lasso funciona mejor cuando los atributos no están muy correlados entre ellos.

Regularización Ridge (L2)

En la regularización Ridge, también llamada L2, la complejidad C se mide como la media del cuadrado de los coeficientes del modelo. Al igual que ocurría en Lasso, la regularización Ridge se puede aplicar a varias técnicas de aprendizaje automático. Matemáticamente quedaría:

$$C = \frac{1}{2N} \sum_{j=1}^N w_i^2$$

Para el caso del error cuadrático medio, este es el desarrollo completo para Lasso (L1):

$$J = \frac{1}{M} \sum_{i=1}^M (real_i - estimado_i)^2 + \alpha \frac{1}{2N} \sum_{j=1}^N w_i^2$$

¿Cuándo es efectiva Ridge (L2)?

Ridge nos va a servir de ayuda cuando sospechemos que varios de los atributos de entrada (features) estén correlados entre ellos. Ridge hace que los coeficientes acaben siendo más pequeños. Esta disminución de los coeficientes minimiza el efecto de la correlación entre los atributos de entrada y hace que el modelo generalice mejor. Ridge funciona mejor cuando la mayoría de los atributos son relevantes.

Regularización ElasticNet (L1 y L2)

ElasticNet combina las regularizaciones L1 y L2. Con el parámetro r podemos indicar que importancia relativa tienen Lasso y Ridge respectivamente. Matemáticamente:

$$C = r \cdot \text{Lasso} + (1 - r) \cdot \text{Ridge}$$

Si lo desarrollamos, para el caso del error cuadrático medio tenemos:

$$J = \frac{1}{M} \sum_{i=1}^M (real_i - estimado_i)^2 + r \cdot \alpha \frac{1}{N} \sum_{j=1}^N |w_i| + (1 - r) \cdot \alpha \frac{1}{2N} \sum_{j=1}^N w_i^2$$

¿Cuándo es efectiva ElasticNet?

Usaremos ElasticNet cuando tengamos un gran número de atributos. Algunos de ellos serán irrelevantes y otros estarán correlados entre ellos.

Resumen

En este artículo hemos explicado qué es la regularización y cómo podemos usarla para mejorar la capacidad de generalización de nuestros modelos de machine learning. En general, la regularización nos va a ayudar a reducir el [sobreajuste](#) de los modelos de aprendizaje automático.

Recursos

- [\[vídeo\]](#) donde explico la regularización Lasso L1, Ridge L2 y ElasticNet (en inglés)
- [Lasso](#): para hacer regresión lineal usando regularización L1
- [LassoCV](#): para encontrar el mejor α para Lasso
- [Ridge](#): para hacer regresión lineal usando regularización L2
- [RidgeCV](#): para encontrar el mejor α para Ridge
- [ElasticNet](#): para hacer regresión lineal usando simultáneamente las regularizaciones L1 y L2
- [ElasticNetCV](#): para encontrar los mejores α y r para ElasticNet
- [scikit-learn](#) ofrece el hiperparámetro 'penalty' en muchos de sus modelos. Puedes usar 'l1' o 'l2' en 'penalty' para elegir la regularización que más te interese.

Mejora tu Inglés con estas frases elegidas con IA

Por Jose Martinez Heras
20/03/2019

En este artículo exploramos la posibilidad de usar Machine Learning para mejorar nuestro inglés. La premisa es que si nos aprendemos las estructuras más comunes en inglés, mejoraremos notablemente nuestra soltura con el inglés.

Encontrando la estructura sintáctica de una frase en inglés

Para encontrar las estructuras en un lenguaje, vamos a utilizar la librería de python spaCy. [spaCy](#) nos va a permitir hacer un análisis de las oraciones. En otras palabras, nos va a permitir descubrir qué función realiza cada palabra en una frase. spaCy soporta el análisis de varios idiomas: inglés, alemán, español, portugués, francés, italiano, holandés, etc.

Podemos pedirle a spaCy que nos de la estructura para la frase «**Learn English with artificial intelligence at iartificial.net**». Este es el resultado que obtenemos:

Palabra	TAG	POS	DEP
Learn	VB	VERB	ROOT
English	NNP	PROPN	dobj
with	IN	ADP	prep
artificial	JJ	ADJ	amod
intelligence	NN	NOUN	pobj
at	IN	ADP	prep
www.iartificial.net	ADD	X	pobj

Este es el significado:

- TAG es la etiqueta (por ejemplo, si es un verbo modal)
- POS es la abreviatura de 'Part Of Speech' que significa 'parte de la oración' (por ejemplo, el verbo)
- DEP es la dependencia sintáctica (por ejemplo, el complemento directo)

Para saber el significado de todas estructuras podemos consultar la [especificación de anotaciones en spaCy](#).

Si concatenamos TAG_POS_DEP nos queda la siguiente equivalencia sintáctica:

- Learn English with artificial intelligence at iartificial.net
- VB_VERB_ROOT NNP_PROPN_dobj IN_ADP_prep JJ_ADJ_amod NN_NOUN_pobj
IN_ADP_prep ADD_X_pobj

Es decir,

- 'Learn' es el verbo principal
- 'English' es un nombre que va haciendo de complemento directo
- 'with' es un modificador preposicional que está actuando como conjunción, subordinación o preposición
- 'artificial' es un adjetivo que modifica al nombre
- 'intelligence' es un nombre que hace de complemento indirecto

- 'at' es una preposición de conjunción, subordinación o preposición
- 'iartificial.net' es algo desconocido que hace la función de complemento indirecto

Las estructuras sintácticas más comunes en inglés

Introducción a los n-grams

Para encontrar cuáles son las estructuras más comunes, vamos a utilizar n-grams en el análisis sintáctico. Los n-grams son los distintos tipos de agrupamientos posibles, de n en n. En nuestro caso, vamos a usar n=3.

Así por ejemplo, para nuestra frase «Learn English with artificial intelligence at iartificial.net» tendremos el análisis sintáctico: VB_VERB_ROOT NNP_PROPN_dobj IN_ADG_prep JJ_ADJ_amod NN_NOUN_pobj IN_ADG_prep ADD_X_pobj

Si consideramos todos los 3-grams tendremos los siguientes:

- VB_VERB_ROOT NNP_PROPN_dobj IN_ADG_prep (Learn English with)
- NNP_PROPN_dobj IN_ADG_prep JJ_ADJ_amod (English with artificial)
- IN_ADG_prep JJ_ADJ_amod NN_NOUN_pobj (with artificial intelligence)
- JJ_ADJ_amod NN_NOUN_pobj IN_ADG_prep (artificial intelligence at)
- NN_NOUN_pobj IN_ADG_prep ADD_X_pobj (intelligence at iartificial.net)

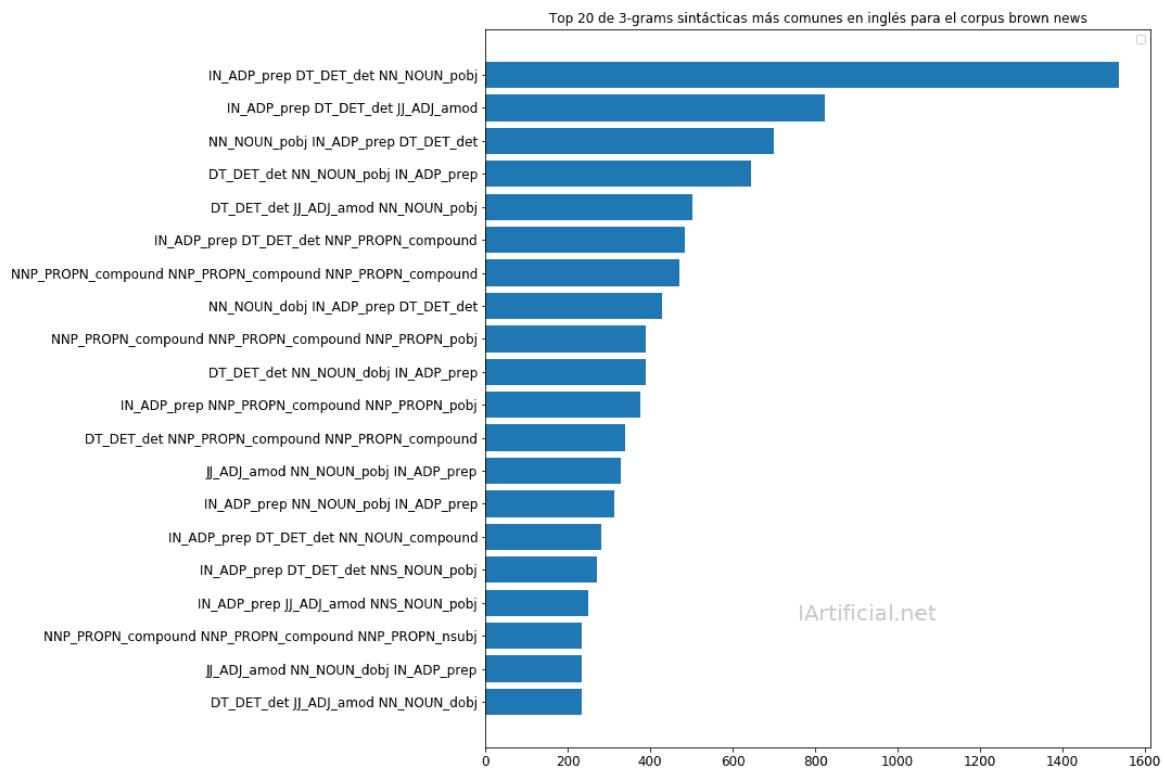
El corpus brown de NLTK

Para encontrar las estructuras sintácticas más comunes en inglés, necesitamos muchos textos en inglés. Los textos que vamos a usar para este análisis vienen del [corpus brown de la librería NLTK](#). Este *corpus* contiene textos en 15 categorías: 'adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction'.

Haremos el análisis con los textos de la categoría 'news' (noticias). Para que te hagas una idea, estas son las tres primeras frases del *corpus brown* para la categoría *news*:

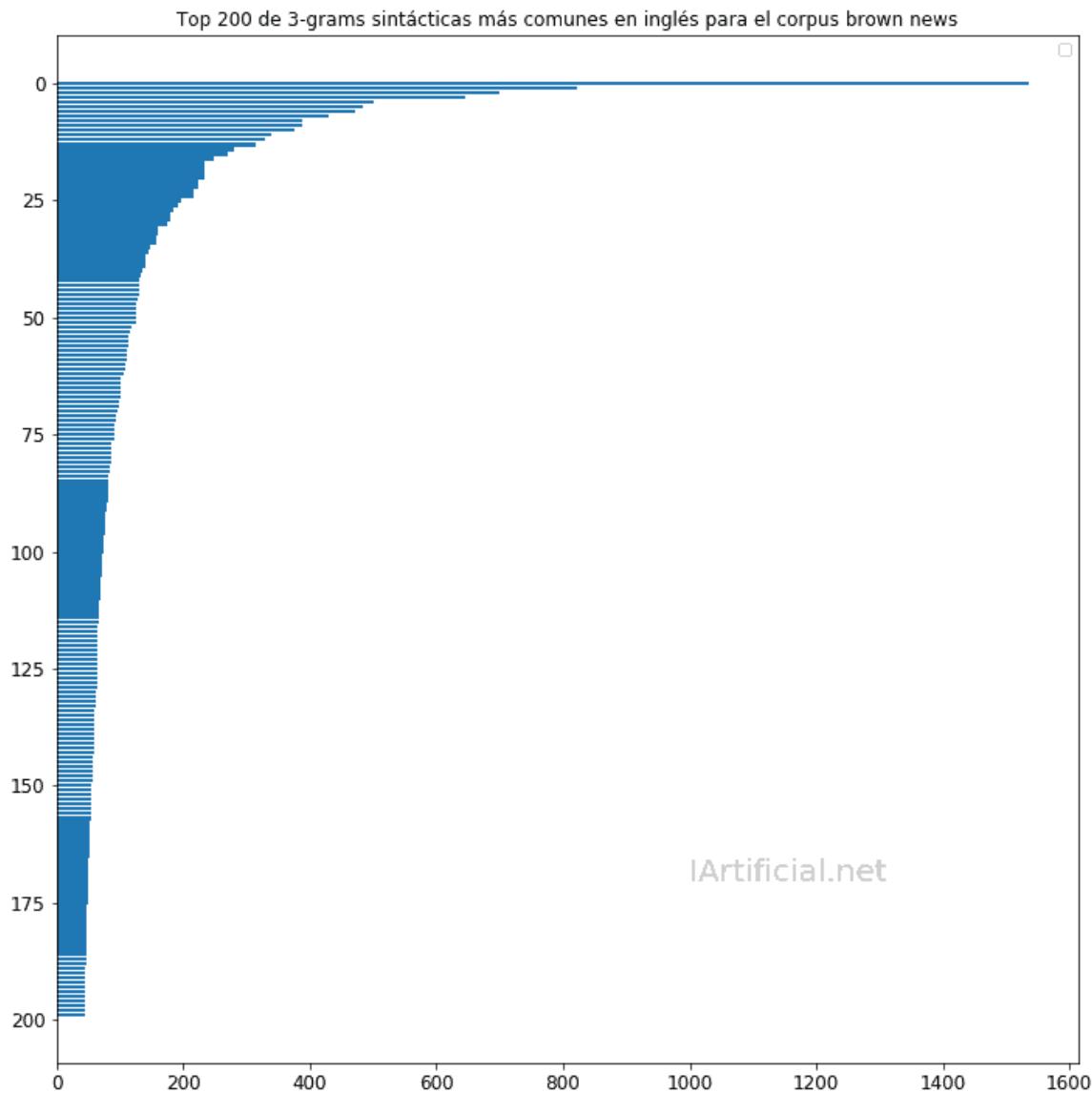
1. «The Fulton County Grand Jury said Friday an investigation of Atlanta's recent primary election produced « no evidence » that any irregularities took place .»
2. «The jury further said in term-end presentations that the City Executive Committee , which had over-all charge of the election , « deserves the praise and thanks of the City of Atlanta » for the manner in which the election was conducted .»
3. «The September-October term jury had been charged by Fulton Superior Court Judge Durwood Pye to investigate reports of possible « irregularities » in the hard-fought primary which was won by Mayor-nominate Ivan Allen Jr. .»

Las estructuras sintácticas más comunes en *news brown corpus*



El siguiente gráfico muestra el número de veces que aparecen los 3-grams sintácticos en la sección de noticias inglesas del corpus brown de NLTK. En este gráfico sólo se muestran los top 20.

Como te habrás dado cuenta, las estructuras más comunes aparecen muchas, pero que muchas más veces que las que no son tan comunes. Sigue una [ley potencial](#) que es muy común en la forma que usamos los lenguajes. Esto es mucho más evidente si en vez de visualizar los top 20, visualizamos los top 200 n-grams.

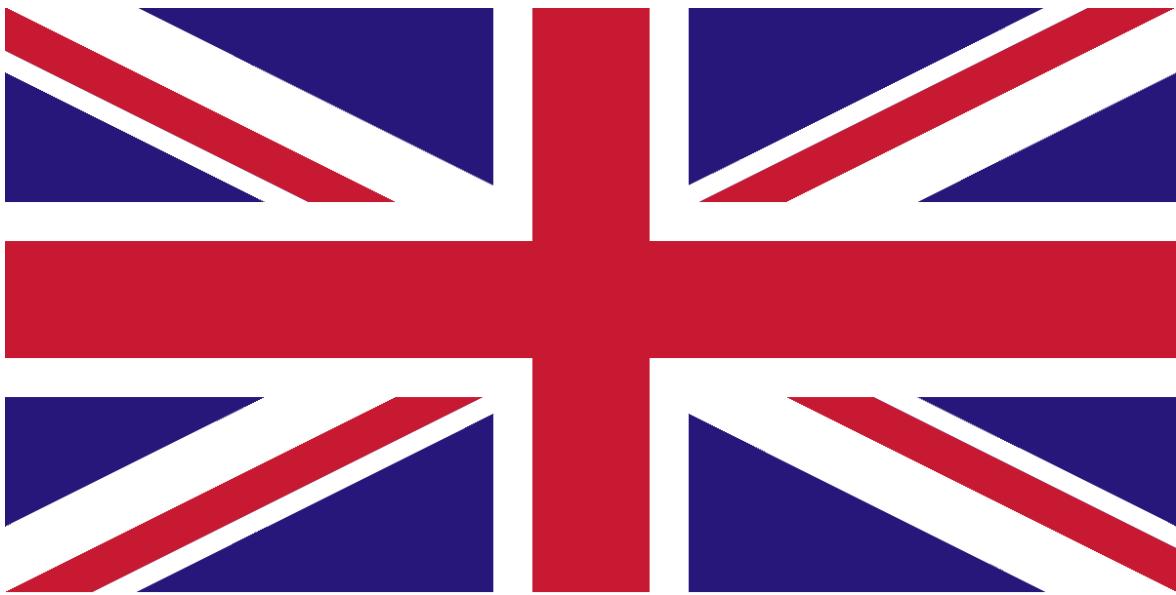


Encontrando las mejores frases ... usando palabras comunes

Si nuestro objetivo es familiarizarnos con las estructuras sintácticas más comunes, podríamos elegir al azar frases que las contuviesen. El problema de esta estrategia, es que correríamos el riesgo de que usen palabras muy raras (poco frecuentes). Por eso, además de escoger frases que tengan palabras más o menos conocidas. Sin entrar mucho en detalles, tengo en cuenta la media de los logaritmos de la frecuencia de cada palabra en una frase. De esta forma obtenemos las frases que tienen palabras conocidas y que tienen estructuras sintácticas comunes.

Las frases de inglés con las que te debes familiarizar

Estas son frases cortas, con estructuras sintácticas frecuentes y con palabras comunes del *corpus brown* por categorías. He usado la categoría *news* pero el mecanismo sería análogo para cualquier otra categoría. Las frases se han obtenido usando los 50 3-grams más comunes. Observarás que hay menos de 50 frases porque algunas de ellas combinan varios de los 3-grams más comunes.



1. Hatfield also is scheduled to hold a public United Nations Day reception in the state capitol on Tuesday .
2. Registrations of new cars in Dallas County cracked the 3,000 mark in March for the first time this year .
3. Since Election Day , Vice President Richard Nixon had virtually retired — by his own wish — from public view .
4. J. A. W. Iglehart , chairman of the Oriole board of directors , and Public Relations Director Jack Dunn .
5. Mr. Kennedy was convinced that insistence on the demand would make international agreements , or even negotiations , impossible .
6. The committee for the annual Central City fashion show has been announced by Mrs. D. W. Moore , chairman .
7. Mr. Kennedy told Moscow he would give his answer by May 20 after consultation with the Allies .
8. Stein said he needed the money , Leavitt said , to « meet the payroll » at National Maintenance company .
9. One house was without power for about half an hour , a Narragansett Electric Co. spokesman said .
10. Legislators who last year opposed placing aged-care under the social security system criticized the President's plan .
11. Several police cars , loaded with armed officers , raced alongside , blazing away at the tires of the big jet .
12. Within a year , without reducing wages , Underwood's production costs were cut one third , prices were slashed .
13. One of these men is former Fire Chief John A. Laughlin , he said .
14. Halleck said the voluntary care plan enacted last year should be given a fair trial first .
15. Asked Mrs. Grace O. Peck , representative from Multnomah County , of the commission chairman , Joseph E. Harvey Jr. .
16. Barbara Borland of Tigard took top senior individual home economics honors with a demonstration called filbert hats .
17. — A Houston teacher , now serving in the Legislature , proposed Thursday a law reducing the time spent learning « educational methods » .
18. Mrs. Clayton Nairne , whose daughter , was among the court maids , chose a deep greenish blue lace gown .
19. The Continental League never got off the ground , but after two years it forced the existing majors to expand .
20. Former Vice-President Richard M. Nixon in Detroit called for a firmer and tougher policy toward the Soviet Union .
21. And the election of President Kennedy has attracted new attention to the ethical climate of his home state .

22. « Los Angeles has said they would send the children to their homes in case of disaster » , he said .
23. The Senate also voted \$5.2 billion to finance the government's health , welfare , and labor activities .
24. Mr. and Mrs. Anderson were entertained at dinner on Sunday by Mr. and Mrs. Frank Coulson , of Fairless Hills .
25. however , the first belief stood for entire revision with a new third point added to the list .
26. Sam Caldwell , State Highway Department public relations director , resigned Tuesday to work for Lt. Gov. Garland Byrd's campaign .
27. He is a native of New Orleans and attended Allen Elementary school , Fortier High school and Soule business college .
28. Taking precedence over all other legislation on Capitol Hill last week was the military strength of the nation .
29. The first one , two years ago , changed the routine of their home life .
30. Assisting as chairmen of various committees are Mrs. Alvin Blum , Mrs. Leonard Malmud , Mrs. Edward Fernberger , Mrs. Robert Cushman .

Resumen

En este artículo hemos obtenido un listado mínimo de frases que maximiza el número de frases con estructuras sintácticas comunes. Las estructuras sintácticas las hemos obtenido con spaCy, con técnicas de procesado del lenguaje natural. Al requerir que el vocabulario sea común y que las frases sean cortas, nos hacemos la vida más fácil. Si te familiarizas con estas frases, su estructura y su significado, tu nivel de inglés mejorará mucho ... al menos estadísticamente hablando!

Recursos

- Librerías de Python para Machine Learning
- [NLTK](#): Natural Language Tool Kit para el corpus en inglés
- [spaCy](#): para el análisis sintáctico de una frase con técnicas de procesado del lenguaje natural

Redes Neuronales Generativas Adversarias (GANs)

Por Jose Martinez Heras
28/03/2019

Las Redes Neuronales Generativas Adversarias son una forma nueva de usar deep learning para generar imágenes que parecen reales. También pueden generar otro tipo de datos tales como música. En este artículo vamos a ver qué son los modelos generativos, cómo funcionan y algunos ejemplos recientes.

Las Redes Neuronales Generativas Adversarias también se denominan GANs por sus siglas en inglés (Generative Adversarial Networks). También lo he visto traducido al español como Redes Antagónicas.

¿Cómo funcionan las Redes Generativas Adversarias?

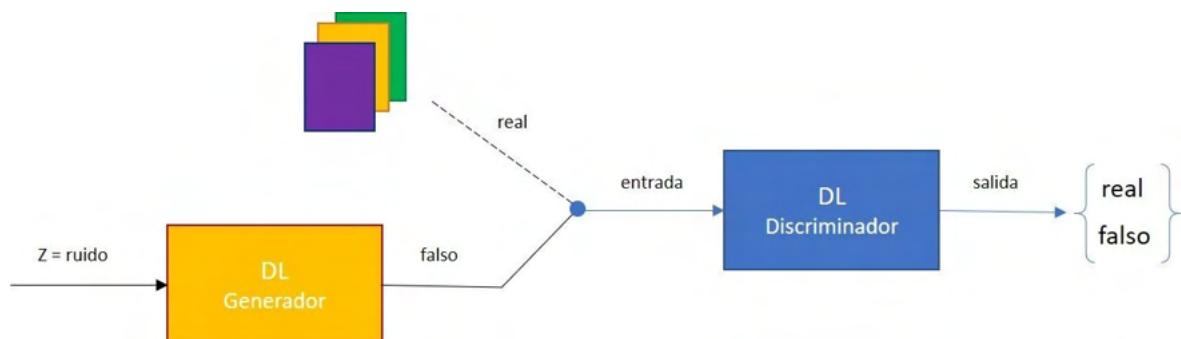
Los modelos generativos están basados en redes neuronales profundas. En los diagramas vamos a utilizar la notación «DL» para indicar Deep Learning, que significa red neuronal profunda. Las redes neuronales reciben datos de entrada y produce resultados como salida.



Las redes neuronales reciben datos de entrada y produce resultados como salida. «DL» significa Deep Learning.

Los modelos generativos usan 2 redes neuronales profundas. Estas dos redes son adversarias, es decir, «juegan» un juego de suma cero donde lo que una red gana, la otra pierde. Me explico. Voy a explicarlo con el ejemplo de la generación de fotos. Imagínate que queremos generar fotos de caras de personas.

- La tarea del *Discriminador* será decir si una cara es auténtica o falsa
- La tarea del *Generador* será la de crear fotos de caras que parezcan auténticas



Para entrenar esta arquitectura de redes neuronales necesitamos tener un conjunto de datos reales. En el ejemplo de las caras de personas, deberemos tener muchas fotos de personas reales. De esta forma, cuando le demos una foto real al discriminador, sabremos que la respuesta correcta es «real». Y cuando le demos una foto creada por el generador, la respuesta correcta es «falso».

Al principio, es muy fácil para el discriminador acertar cuando dice cuáles fotos son reales. Esto es así porque al principio, las fotos creadas por la red neuronal generadora parecen aleatorias. Sin embargo, a medida que el generador mejora, la tarea de la red neuronal discriminadora se vuelve más difícil. Así pues, tanto el generador como el discriminador van mejorando simultáneamente.

El generador crea caras distintas. Las caras son distintas porque la entrada es distinta. Como entrada utilizamos un vector aleatorio (que llamamos Z). Usando otro vector aleatorio, tendremos otra cara ... o lo que quiera que estemos generando.

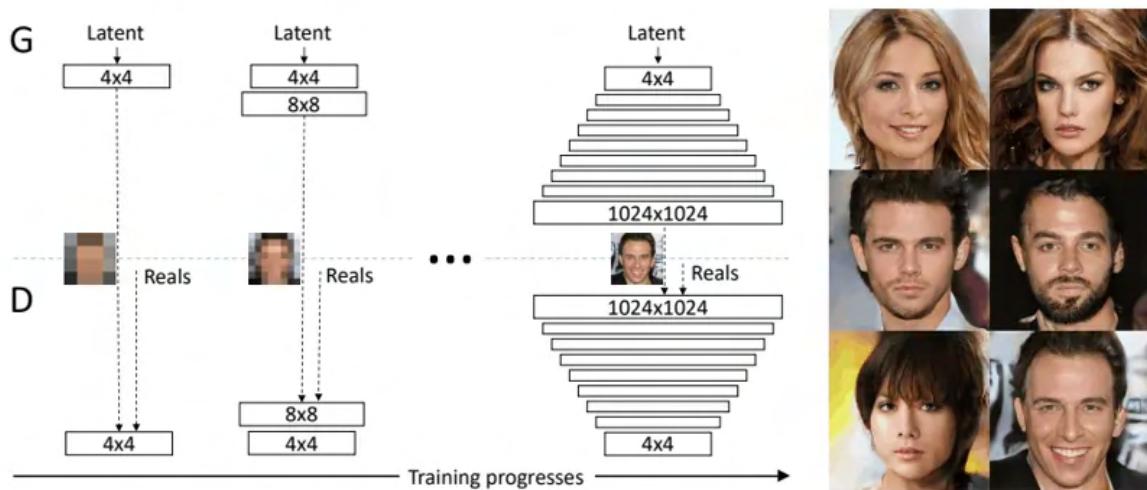
Ejemplo: generando dormitorios

Uno de los primeros ejemplos del uso de modelos generativos es el de la generación de imágenes de dormitorios. Por aquellos entonces (año 2016), las imágenes que se podían generar con GANs tenían poca resolución. Si queréis leer más, la referencia del artículo está [aquí](#).



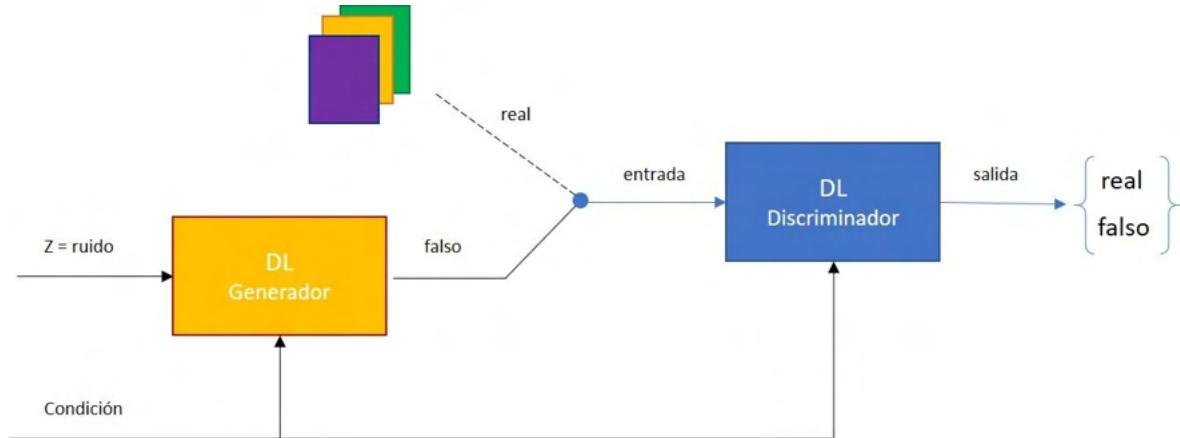
Mejorando la calidad de los modelos generativos

Para mejorar la calidad de los modelos generativos, se usan redes neuronales generativas adversarias **progresivas**. La clave está en progresivamente ir aumentando la resolución de las imágenes. Al principio, tenemos una arquitectura generador / discriminador que funcionan con imágenes de 4×4 píxeles. Cuando los resultados son lo suficientemente buenos, vamos a por 8×8 , luego a 16×16 ... y así sucesivamente hasta la calidad de necesitemos. Puedes leer más acerca de los GANs progresivos [aquí](#).



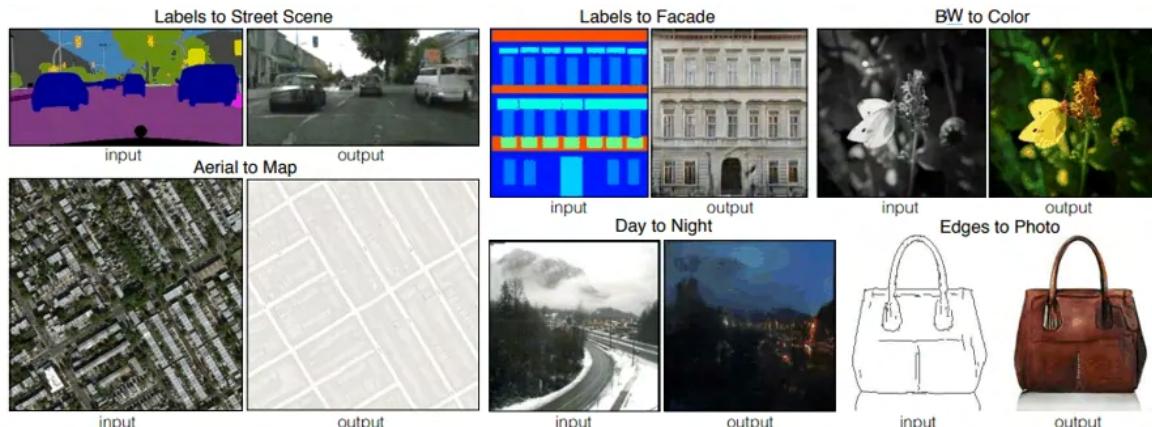
Redes Generativas Adversarias Condicionadas

También podemos influenciar el resultado de las imágenes que el generador crea. Para ello, le damos la misma condición tanto al generador como al discriminador. Por ejemplo, le podemos pedir al generador que cree la foto de un bolso a partir de un boceto de un bolso. Esto necesita que tengamos ejemplos de imágenes reales que también vayan bien con estos bocetos.



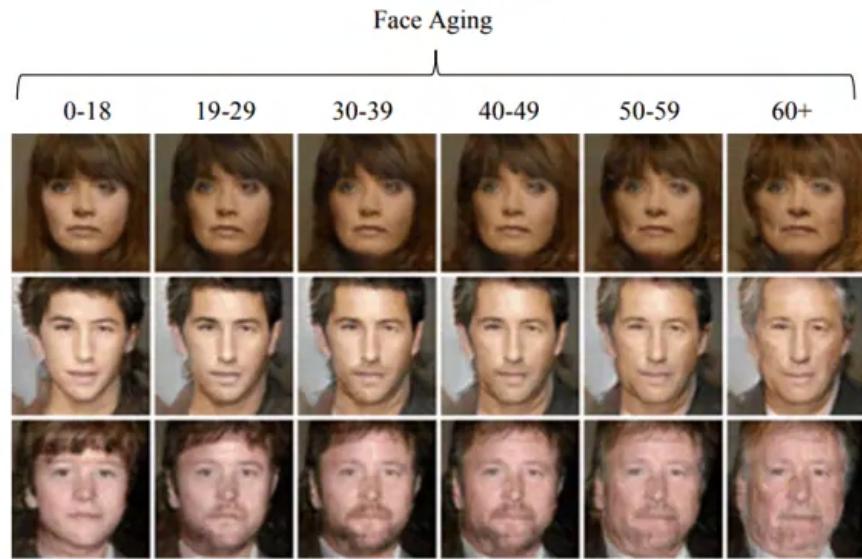
Fotos a partir de bocetos

Por ejemplo, la siguiente imagen, sacada de [este trabajo](#), muestra cómo se generan, a partir de bocetos, fotos de carreteras, de edificios y de bolsos. Además se genera un mapa a partir de la imagen de satélite, se convierte una foto de día a de noche y se genera una foto en color a partir de una en blanco y negro.



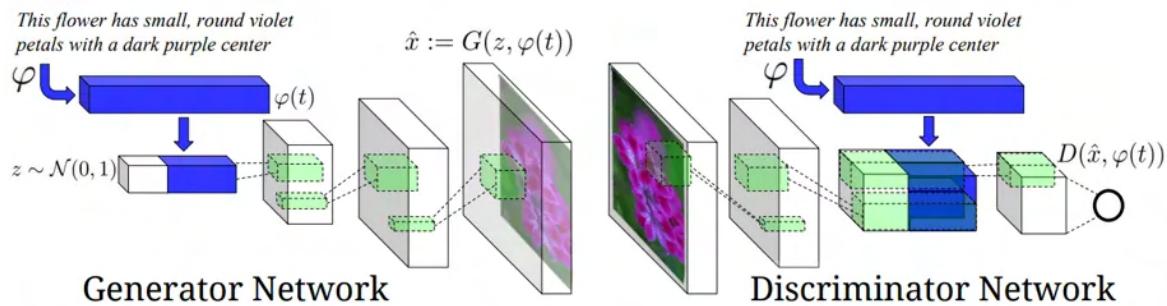
10 Years Challenge

Para los que habéis participado en el **10 years challenge**, a lo mejor habéis ayudado con vuestras fotos a mejorar este [otro trabajo](#). En la siguiente imagen puedes ver cómo el modelo generativo crea diferentes versiones de caras *condicionadas* a la edad que queramos.



Generación de imágenes a partir de texto

También es posible condicionar a las redes generativas adversarias para que generen imágenes a partir de un texto. Así podremos escribir la descripción de una imagen y el generador creará una imagen compatible con esa descripción. Puedes obtener más información [aquí](#).



Lo último en redes neuronales generativas adversarias

thispersondoesnotexist.com

thispersondoesnotexist.com es una web creada por nvidia que crea caras en alta definición de personas que no existen. Aquí tienes un ejemplo de 3 caras que he obtenido. Te animo a que lo pruebes tú también.



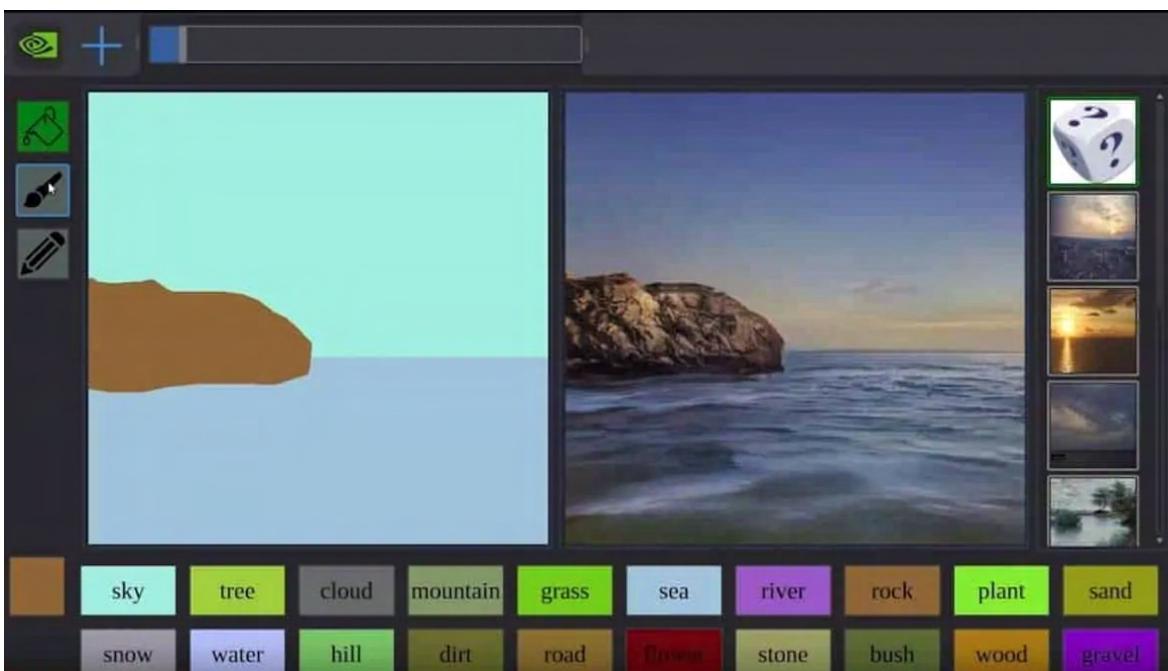
Style GAN

Style GAN permite cambiar estilos tales como la pose, color de pelo, texturas, etc. Pueden elegir cómo lo cambian porque usan una forma muy novedosa de controlar el ruido que usa el generador como entrada. Puedes mirar cómo funciona en detalle en [este artículo](#).



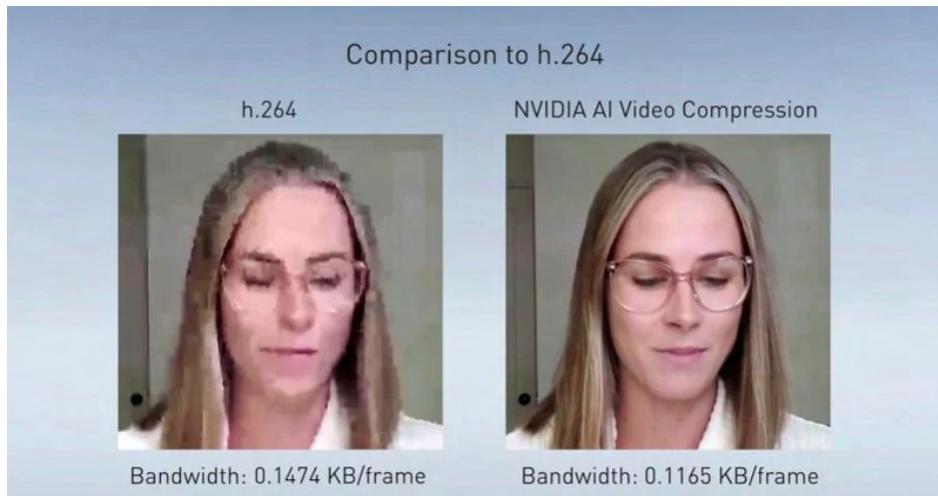
Generación de paisajes realistas a partir de bocetos

nvidia nos sorprende una vez más con un prototipo basado en redes neuronales generativas adversarias. Con este [prototipo](#), cualquiera se puede convertir en un pintor profesional, ya que permite generar escenas realistas a partir de bocetos muy simples. Haz click en la imagen para ver el vídeo.



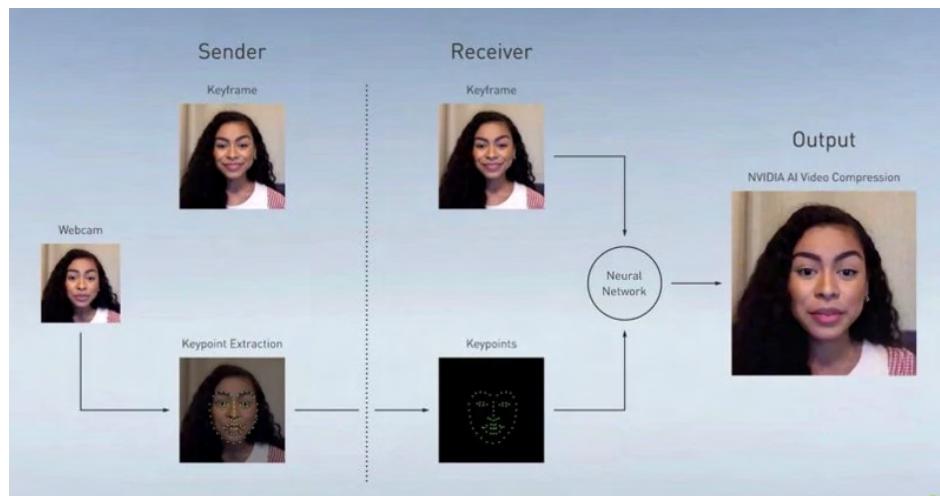
Compresión de caras para video conferencias

Nvidia ha desarrollado una [técnica nueva de compresión de vídeo](#) que permite ahorrar ancho de banda al mismo tiempo que mejora la calidad de imagen en videoconferencias.



Funciona tan bien porque sólo envía alguna que otra foto de la persona que aparece en la imagen. Los cambios faciales al hablar, sonreir, moverse, etc. no se transmiten como imágenes (frames) sino que se transmiten como puntos de referencia (*landmarks*) de la cara.

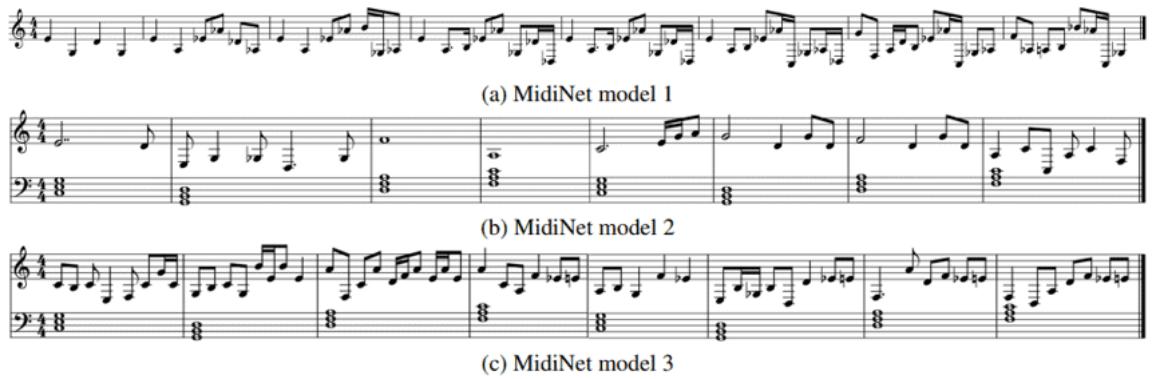
Cuando reciben estos puntos de referencia, una red neuronal generativa adversaria los aplica a la imagen de la foto original. El resultado es muy convincente, mejorando la calidad de las videoconferencias y reduciendo los requisitos de ancho de banda.



Todavía no he tenido oportunidad de probarlo, ya nos contaréis si alguno lo ha probado ya.

No sólo para imágenes

Hasta ahora, todos los ejemplos que hemos visto de modelos generativos estaban enfocados a imágenes. Pero no tiene por qué ser así. También pueden aplicarse a otros tipos de datos tales como las series temporales o incluso la música. En [este trabajo](#), han usado modelos generativos para generar partituras.



Resumen

Las redes neuronales generativas adversarias son una forma nueva de combinar dos redes neuronales. El discriminador ayuda al generador a crear cada vez mejores imágenes. Tanto el generador como el discriminador van mejorando simultáneamente hasta que llega un punto que para nosotros, los humanos, las imágenes creadas por el generador nos parecen tan realistas que no podemos distinguirlas de imágenes reales.

Los modelos generativos son muy nuevos en el mundo del aprendizaje profundo. Sin embargo, la velocidad de la investigación en este campo está siendo asombrosa y sus aplicaciones son fascinantes.

Empresas con experiencia en Inteligencia Artificial

Por Jose Martinez Heras
03/04/2019

En IArtificial.net queremos ayudar a tu empresa. Nos gustaría daros más visibilidad para que los clientes os encuentren más fácilmente. Para ello, sólo tenemos que enseñarle al mundo lo buenos que sois.

¿Qué servicios o productos ofrece la empresa?

Dinos qué es lo que un cliente puede comprar en vuestra empresa. ¿Qué beneficio va a conseguir el cliente cuando haga negocios con vosotros? ¿Cuál es vuestro modelo de negocio? ¿Cómo pueden los clientes comprar vuestros productos / servicios? ¿Tenéis una página web que queráis compartir?

Demuestra de lo que sois capaces

Estamos convencidos de que vuestra empresa es seria y es capaz de ofrecer lo que promete. Sin embargo, como sabéis, la inteligencia artificial está de moda y algunas empresas se apunta a la moda más como marketing que como otra cosa.

Para demostrarlo, lo mejor es poner algunos ejemplos de proyectos y actividades recientes con un impacto positivo en vuestros clientes. Si te fijas, os estamos ayudando a que le enseñéis al mundo qué clase de impacto obtienen vuestros clientes. Estamos convencidos de que si construimos una lista de empresas que ofrecen un impacto real con Inteligencia Artificial, no van a faltarlos los clientes.

Rellena un formulario con esta información

Para organizar mejor el trabajo de recopilar información sobre empresas de habla hispana especializadas en inteligencia artificial, hemos creado un formulario. Los datos de vuestra empresa aparecerán en la página [Empresas de Inteligencia Artificial](#). El link al formulario para agregar tu empresa se encuentra también en esa página.

Por conveniencia, también incluimos aquí un enlace al [formulario](#).

Y después de llenar el formulario ...

Añadiremos tu empresa al listado de empresas con experiencia en Inteligencia Artificial de habla hispana. El proceso no es automático porque nos gusta repasar los detalles que habéis incluido y verificarlos. Ten paciencia y tu empresa aparecerá pronto en la [lista](#).

Máquinas de Vectores de Soporte (SVM)

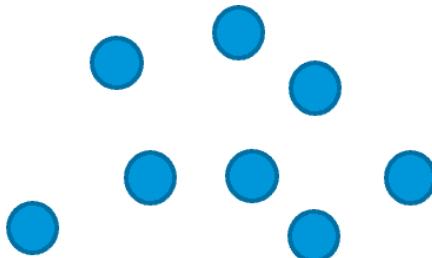
Por Jose Martinez Heras
12/04/2019

En este artículo hablamos de las máquinas de vectores de soporte. También son conocidas con el acrónimo SVM por sus siglas en inglés (Support Vector Machines). Se pueden usar tanto para regresión como para clasificación.

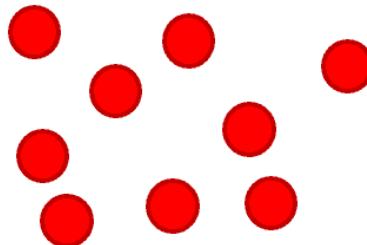
Conceptualmente, los SVM son más fáciles de explicar para problemas de clasificación. En este artículo vamos a dar una explicación intuitiva. Las máquinas de vectores de soporte surgen para encontrar la manera óptima de clasificar.

Datos de ejemplo

Vamos a usar estos datos de ejemplo. Suponemos que los puntos azules corresponden a la clase «azul» y los puntos rojos a la clase «rojo». Ahora intentaremos dibujar una línea que separe los puntos azules de los rojos. De esta forma, cuando haya un punto nuevo, podemos decir qué color va a tener, dependiendo del lado de la línea en el que se encuentre.



IArtificial.net

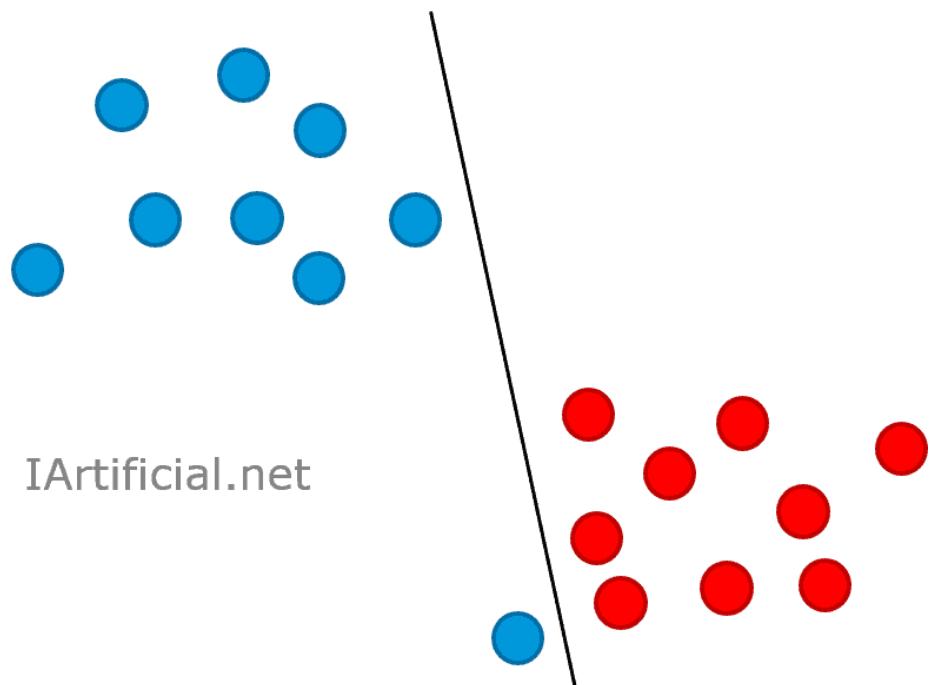


Formas equivocadas de clasificar

A continuación veremos algunos ejemplos de formas equivocadas de clasificar.

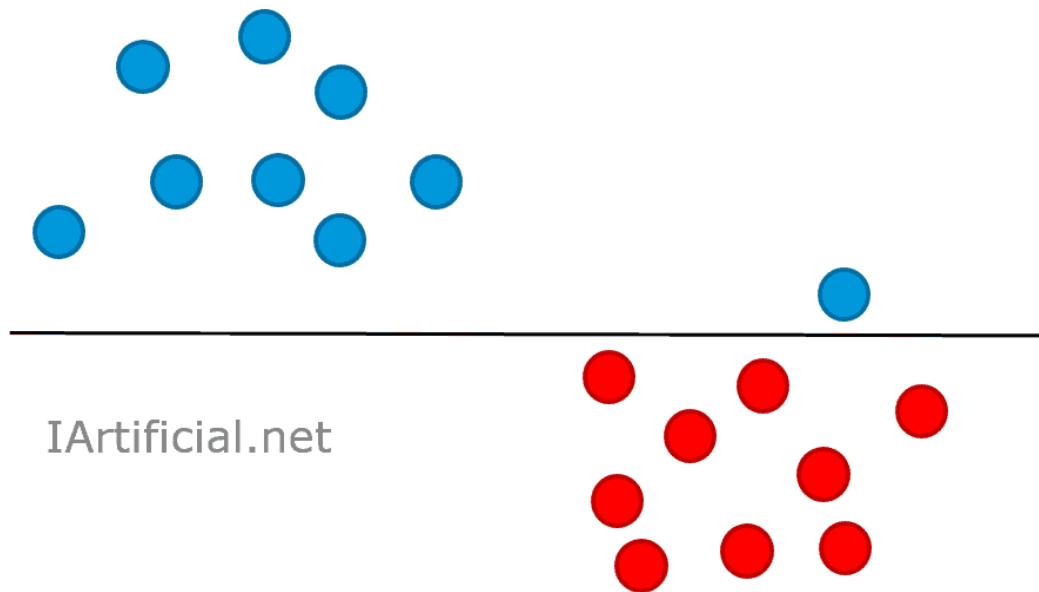
Ejemplo I

En la siguiente figura, podemos decir que lo que esté a la izquierda de la línea, es azul y lo que esté a la derecha, es rojo. Sin embargo, el punto nuevo abajo a la izquierda es clasificado como azul. Intuitivamente, está mal clasificar este punto nuevo como azul.



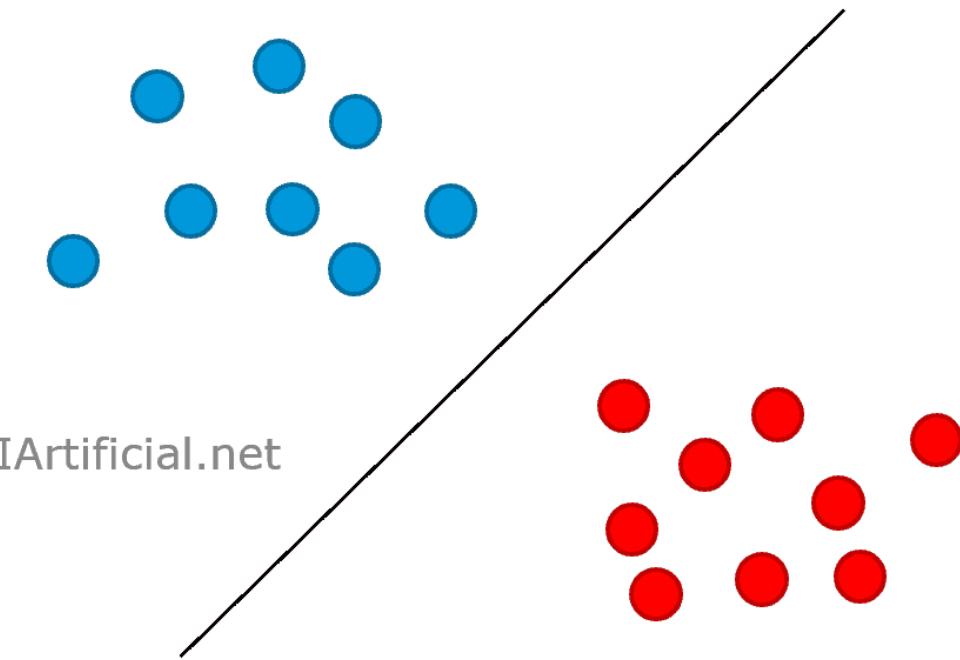
Ejemplo II

La siguiente figura muestra otra forma equivocada de clasificar estos puntos. Podemos decir que cualquier punto que esté por lo alto de la línea será azul. Cualquier punto que esté por debajo será rojo. Sin embargo, el nuevo punto a la derecha, ha sido «incorrectamente» clasificado como azul. Intuitivamente, diríamos que debería haber sido clasificado como rojo.



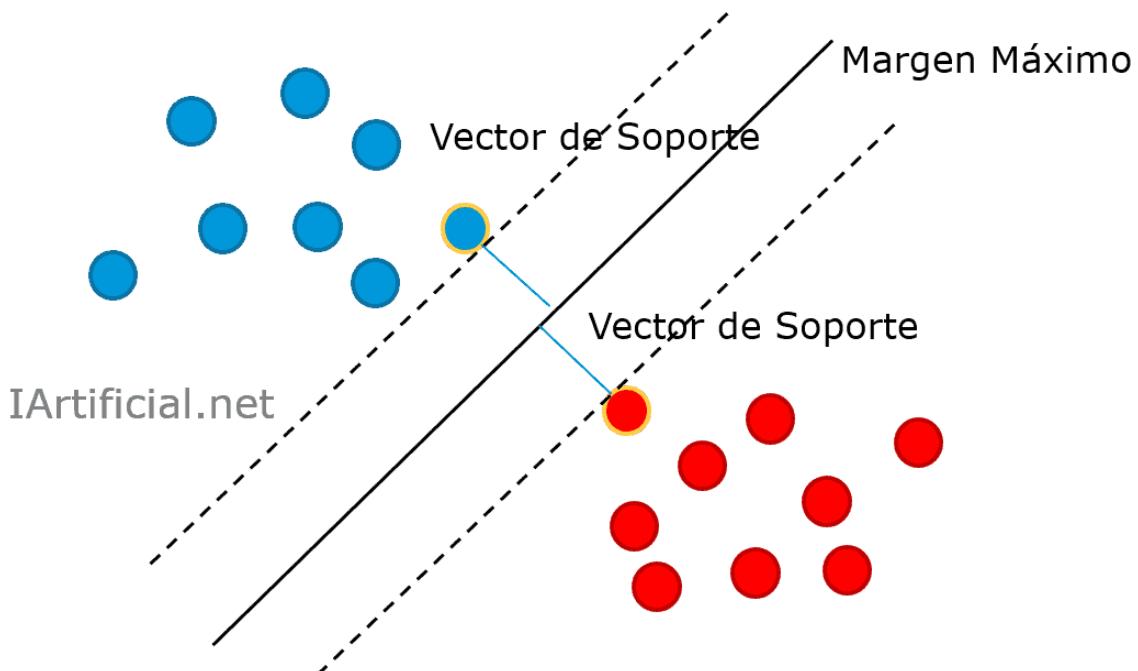
Clasificación óptima con máquinas de vectores de soporte

La línea que mejor distingue las zonas de los puntos azules de la zona de los puntos rojos es la línea que maximiza el margen entre ambos. Las máquinas de vectores de soporte son una técnica de machine learning que encuentra la mejor separación posible entre clases. Con dos dimensiones es fácil entender lo que está haciendo. Normalmente, los problemas de aprendizaje automático tienen muchísimas dimensiones. Así que en vez de encontrar la línea óptima, el SVM encuentra el hiperplano que maximiza el margen de separación entre clases.



¿Por qué se llaman Máquinas de Vectores de Soporte?

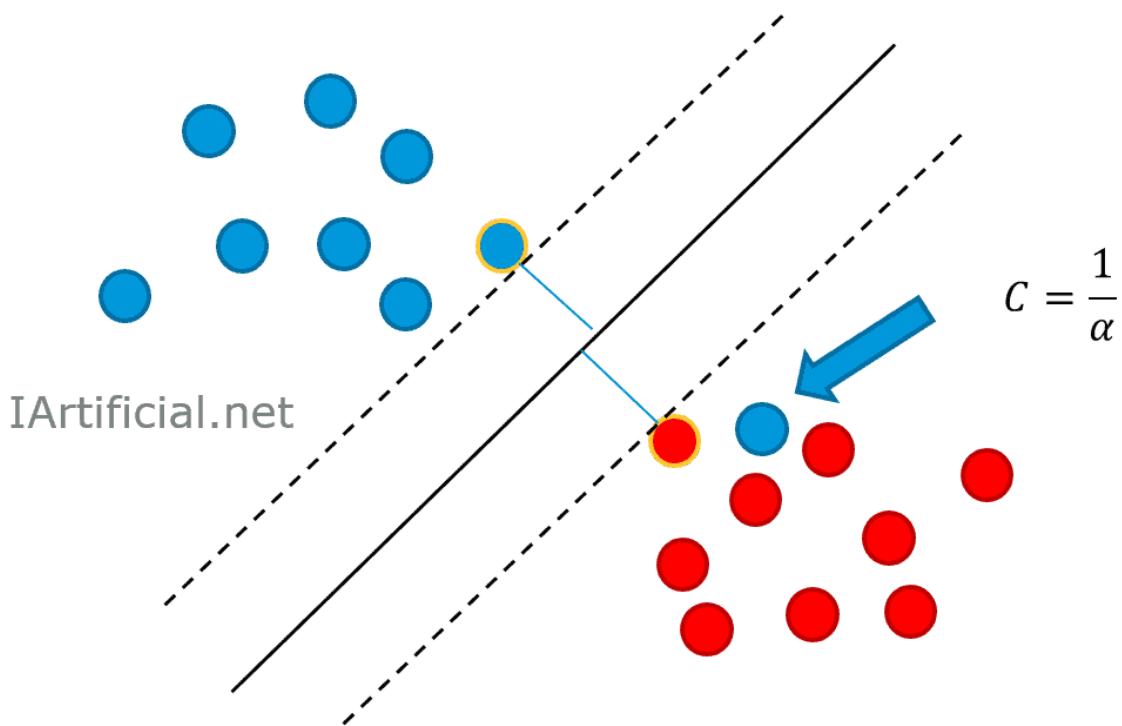
Se llama «máquina» en español por la parte de «machine» learning. Los vectores de soporte son los puntos que definen el margen máximo de separación del hiperplano que separa las clases. Se llaman vectores, en lugar de puntos, porque estos «puntos» tienen tantos elementos como dimensiones tenga nuestro espacio de entrada. Es decir, estos puntos multi-dimensionales se representan con un vector de n dimensiones.



Regularización

Es bastante frecuente que los datos tengan ruido, que no estén etiquetados perfectamente, o que el problema sea tan difícil que para unos pocos puntos, sea muy complicado clasificarlos correctamente. Para estos casos, podemos decirle al SVM (Support Vector Machine), que preferimos que generalice bien para la mayoría de los casos, aunque algunos pocos casos del conjunto de entrenamiento no estén perfectamente clasificados. Recuerda que lo que

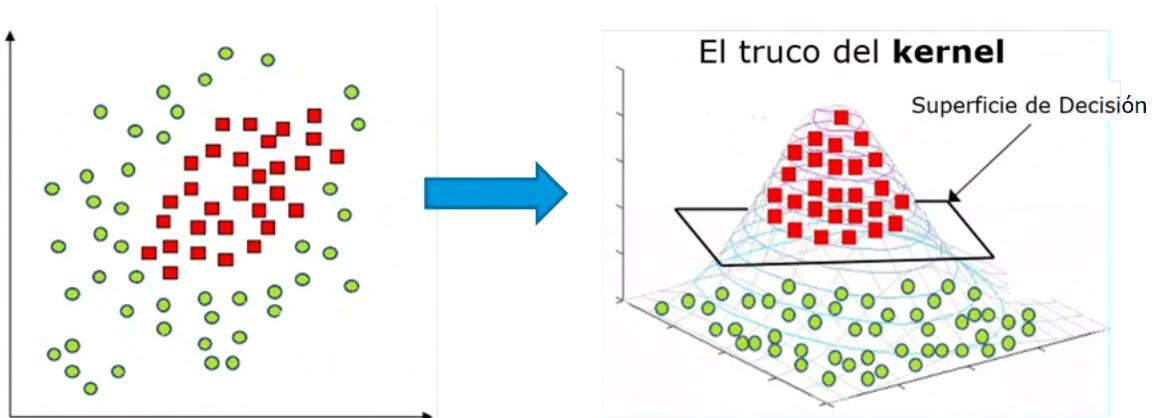
normalmente vamos buscando es la construcción de modelos de aprendizaje automático que generalicen bien. Para controlar la cantidad de regularización, podemos usar el hiper-parámetro C , que hace las veces del inverso del alfa que vimos en las regularizaciones Ridge, Lasso y ElasticNet.



El truco del kernel en SVM

Hay veces en las que no hay forma de encontrar una hiperplano que permita separar dos clases. En estos casos decimos que las clases no son linealmente separables. Para resolver este problema podemos usar el truco del kernel.

El truco del kernel consiste en inventar una dimensión nueva en la que podamos encontrar un hiperplano para separar las clases. En la siguiente figura vemos cómo al añadir una dimensión nueva, podemos separar fácilmente las dos clases con una superficie de decisión.



Algunas aplicaciones de las máquinas de vectores de soporte

Las máquinas de vectores de soporte eran muy utilizadas antes de la era del aprendizaje profundo. Para muchas aplicaciones se prefería el uso de SVM en lugar de redes neuronales. La

razón era que las matemáticas de los SVM se entiende muy bien y la propiedad de obtener el margen de separación máximo era muy atractivo. Las redes neuronales podrían realizar clasificación de forma equivocada como hemos visto en los ejemplos anteriores.

Algunos casos de éxito de las máquinas de vectores de soporte son:

- reconocimiento óptico de caracteres
- detección de caras para que las cámaras digitales enfoquen correctamente
- filtros de spam para correo electrónico
- reconocimiento de imágenes a bordo de satélites (saber qué partes de una imagen tienen nubes, tierra, agua, hielo, etc.)

Actualmente, las redes neuronales profundas tienen una mayor capacidad de aprendizaje y generalización que los SVM.

Resumen

Las Máquinas de Vectores de Soporte (Support Vector Machines) permiten encontrar la forma óptima de clasificar entre varias clases. La clasificación óptima se realiza maximizando el margen de separación entre las clases. Los vectores que definen el borde de esta separación son los vectores de soporte. En el caso de que las clases no sean linealmente separables, podemos usar el truco del kernel para añadir una dimensión nueva donde sí lo sean.

Recursos y Créditos

- [\[vídeo\]](#) donde explico cómo funciona las máquinas de vectores de soporte (en inglés)
- [Tutorial](#) de Máquinas de Vectores de Soporte en scikit-learn
- La figura del kernel: http://blog.csdn.net/sinat_35257860/article/details/58226823

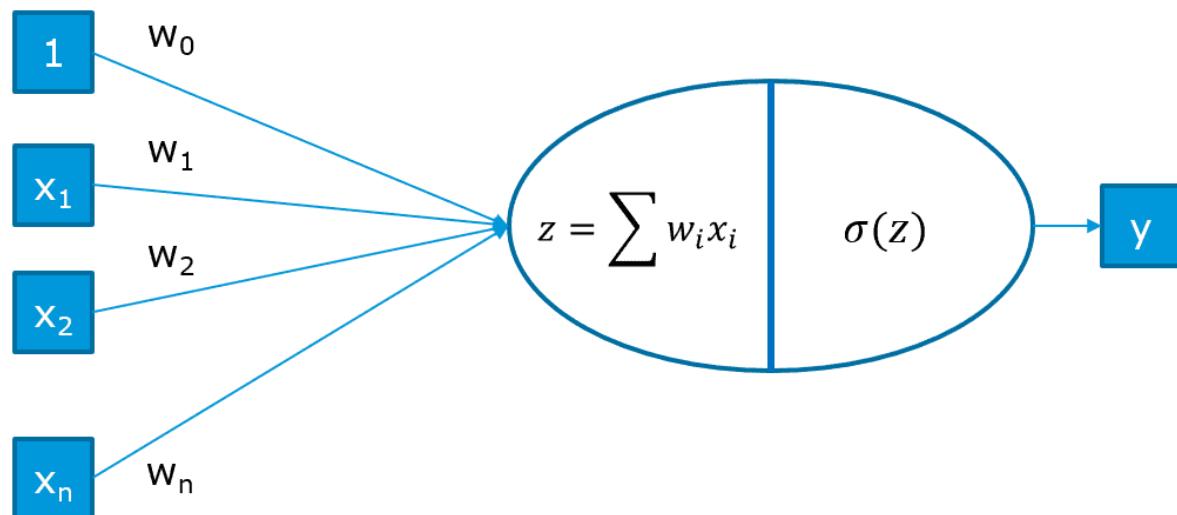
Regresión Logística para Clasificación

Por Jose Martinez Heras
23/04/2019

La Regresión Logística es una técnica de aprendizaje automático para clasificación. Es una red neuronal en miniatura. De hecho, la regresión logística, se trata de una red neuronal con exactamente una neurona.

Matemáticas de la Regresión Logística

Podemos representar lo que hace la regresión logística en la siguiente figura:



Los valores de x corresponden los distintos atributos de nuestro problema. Por ejemplo, si queremos saber si un correo electrónico es deseado o no deseado (spam), los valores de x podrían corresponder con cuántas veces aparece cada palabra en un texto. La predicción y sería la probabilidad de que el correo sea no deseado.

Matemáticamente, lo podemos formular de esta forma:

$$y = \sigma(z) = \sigma(WX) = \sigma(\sum(w_i x_i)) = \sigma(\sum(w_0 x_0 + w_1 x_1 + \dots + w_n x_n))$$

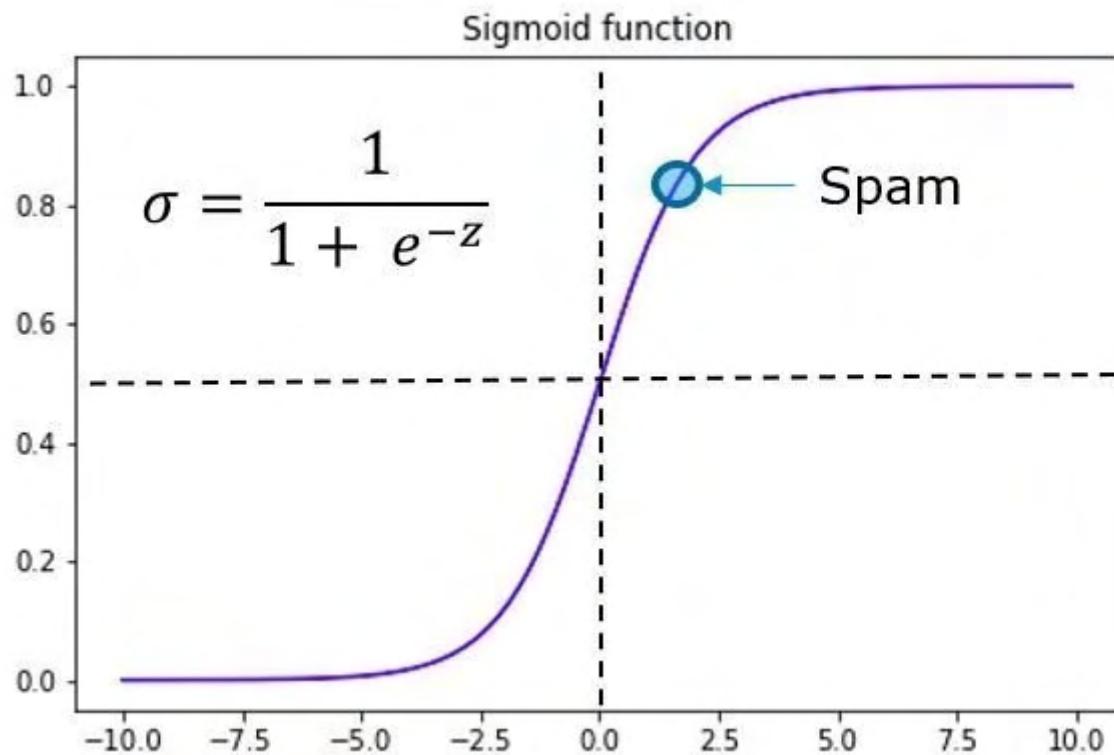
Como ves, la regresión logística tiene dos partes:

1. una combinación lineal (a la izquierda de la neurona)
2. aplicación de la función logística (a la derecha de la neurona)

Así que todas las entradas se combinan con una línea con los coeficientes w . Y luego se aplica la función logística (también llamada sigmoide) al resultado.

Función logística

La función logística, o sigmoide, se representa en la siguiente figura.



Matemáticamente, la función logística o sigmoide, se puede expresar:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Características de la función logística

- Está acotada entre 0 y 1. Su valor mínimo es 0 y el máximo es 1.
- Podemos interpretar sus resultados como probabilidades (por ejemplo, una probabilidad 0.85 de que el correo sea spam)
- Para problemas de clasificación binaria, podemos suponer que los valores menores de 0.5 corresponden a la clase 0 y los superiores a 0.5 a la clase 1.

Curiosidades

La parte de «regresión» en regresión logística podría dar la impresión de que se trata de una técnica de regresión. Sin embargo, recibe este nombre por motivos históricos. Al principio de la neurona, se hace un combinación lineal que se parece mucho a una regresión lineal. Y luego se aplica la función logística. Así que así surgió el nombre, regresión logística.

La función de coste de la Regresión Logística

El aprendizaje se realiza con optimización numérica, por ejemplo, con gradiente descendiente. No hay ninguna fórmula que nos de los coeficientes óptimos W , sino que tenemos que estimarlos. La función de coste que optimizamos es la siguiente:

$$J = -\frac{1}{m} \sum_i^m y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)$$

También podemos incluir términos de regularización en el proceso de aprendizaje.

Aplicaciones de la Regresión Logística

La regresión logística es un método de clasificación muy simple y efectiva para muchos problemas. Es fácil de interpretar y podemos inspeccionar qué factores contribuyen más a obtener qué tipo de resultados.

Cuando estuvimos analizando las noticias de portada de menéame, usamos regresión logística para predecir la probabilidad de que una noticia tuviese muchos meneos o muchas visitas. Puedes leer el artículo entero aquí.

Resumen

La regresión logística es una técnica de aprendizaje automático relativamente sencilla. Sus resultados son interpretables y es muy usada en industria. Funciona muy bien cuando hay muchísimos datos y las interrelaciones entre ellos no son muy complejas. Admiten el uso de regularización para generalizar mejor.

Recursos

- Instrucciones de cómo usar la Regresión Logística en Python
- [\[vídeo\]](#) donde explico cómo funciona la regresión logística (en inglés)
- La implementación de [LogisticRegression](#) en la librería de python scikit-learn

¿Cómo usar Regresión Logística en Python?

Por Jose Martinez Heras
01/05/2019

La regresión logística es una técnica de aprendizaje supervisado para clasificación. Es muy usada en muchas industrias debido a su escalabilidad y explicabilidad.

En este artículo vamos a ver cómo entrenar y usar un modelo de regresión logística. Si quieres repasar la teoría de esta técnica de machine learning, puedes consultar este artículo.

Instrucciones rápidas

¿Cómo usar Regresión Logística en Python con scikit-learn?

Total Time: 5 minutes

Importa la librería numérica NumPy

```
import numpy as np
```

Prepara los datos de entrenamiento

X serán los datos de entrada, y los de salida en este ejemplo

Importa el módulo LogisticRegression de la librería scikit-learn

```
from sklearn.linear_model import LogisticRegression
```

Crea una instancia de la Regresión Logística

```
regresion_logistica = LogisticRegression()
```

Entrena la regresión logística con los datos de entrenamiento

```
regresion_logistica.fit(X,y)
```

Usa el modelo entrenado para obtener las predicciones con datos nuevos

```
prediccion = regresion_logistica.predict(X_nuevo)
```

Opcionalmente, obtén las probabilidades de la predicción

```
probabilidades_prediccion = regresion_logistica.predict_proba(X_nuevo)
```

Ejemplo de Regresión Logística en Python

Datos

Vamos a suponer que queremos predecir cuál es la probabilidad que tiene un estudiante de aprobar un examen en función de las horas que ha estudiado. Date cuenta que para 1.75 horas de estudio, hay un estudiante que aprueba y el otro que no.

Horas	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50
Aprueba	0	0	0	0	0	0	1	0	1	0

Horas	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Aprueba	1	0	1	0	1	1	1	1	1	1

Podemos escribir el siguiente código python para representar estos datos:

```
# Paso 1: importamos la librería numérica NumPy
import numpy as np

# Paso 2: preparamos los datos
X = np.array([0.5, 0.75, 1, 1.25, 1.5, 1.75, 1.75, 2, 2.25, 2.5, 2.75, 3, 3.25,
3.5, 4, 4.25, 4.5, 4.75, 5, 5.5]).reshape(-1,1)
y = np.array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1])
```

Entrenando la regresión logística

Durante la fase de entrenamiento, el modelo aprende qué coeficientes minimizan la [función de coste](#).

```
# Paso 3: importamos la clase LogisticRegression de scikit-learn
from sklearn.linear_model import LogisticRegression

# Paso 4: Creamos una instancia de la Regresión Logística
regresion_logistica = LogisticRegression()

# Paso 5: Entrena la regresión logística con los datos de entrenamiento
regresion_logistica.fit(X,y)
```

Haciendo predicciones

Vamos a ver cómo podemos hacer predicciones una vez que el modelo está entrenado. Primero haremos predicciones absolutas y luego predicciones relativas. Vamos a ver qué pasa si estudiamos 1, 2, 3, 4, 5 ó 6 horas.

```
X_nuevo = np.array([1, 2, 3, 4, 5, 6]).reshape(-1,1)

# Paso 6: Usa el modelo entrenado para obtener las predicciones con datos nuevos
prediccion = regresion_logistica.predict(X_nuevo)
print(prediccion)
# produce el resultado: [0 0 1 1 1 1]
```

Como podemos ver, en el caso que el estudiemos 1 ó 2 horas, lo más probable es que suspendamos para este examen. Si estudiamos 3 o más horas, lo más probable es aprobar. Vamos a ver ahora cómo podemos calcular las probabilidades en estos casos.

```
# Paso 7: Opcionalmente, obtén las probabilidades de la predicción
probabilidades_prediccion = regresion_logistica.predict_proba(X_nuevo)
print(probabilidades_prediccion)

# produce el siguiente resultado (la primera columna es
# la probabilidad de suspender y la segunda columna es
# la probabilidad de aprobar)
# [[0.6801015  0.3198985 ]
# [0.53568295 0.46431705]
# [0.38502138 0.61497862]
# [0.25359079 0.74640921]
# [0.15566862 0.84433138]
# [0.09095092 0.90904908]]
```

Como seguramente estamos más interesados en la probabilidad de aprobar,

```
# podemos centrarnos en la segunda columna  
print(probabilidades_prediccion[:,1])  
  
# produce el resultado:  
[0.3198985 0.46431705 0.61497862 0.74640921 0.84433138 0.90904908]
```

Recursos

- Teoría sobre la regresión logística en artículos anteriores
- [LogisticRegression](#), la implementación en [scikit-learn](#)
- La librería [NumPy](#) para manipulación numérica
- El ejemplo que hemos visto (sin el código), viene de la [wikipedia](#)

Árboles de Decisión con ejemplos en Python

Por Jose Martinez Heras
07/05/2019

Los árboles de decisión son una técnica de aprendizaje automático supervisado muy utilizada en muchos negocios. Como su nombre indica, esta técnica de machine learning toma una serie de decisiones en forma de árbol. Los nodos intermedios (las ramas) representan soluciones. Los nodos finales (las hojas) nos dan la predicción que vamos buscando.

Los árboles de decisión pueden usarse para resolver problemas tanto de clasificación como de regresión. Veamos cómo se usan en cada caso con ejemplos.

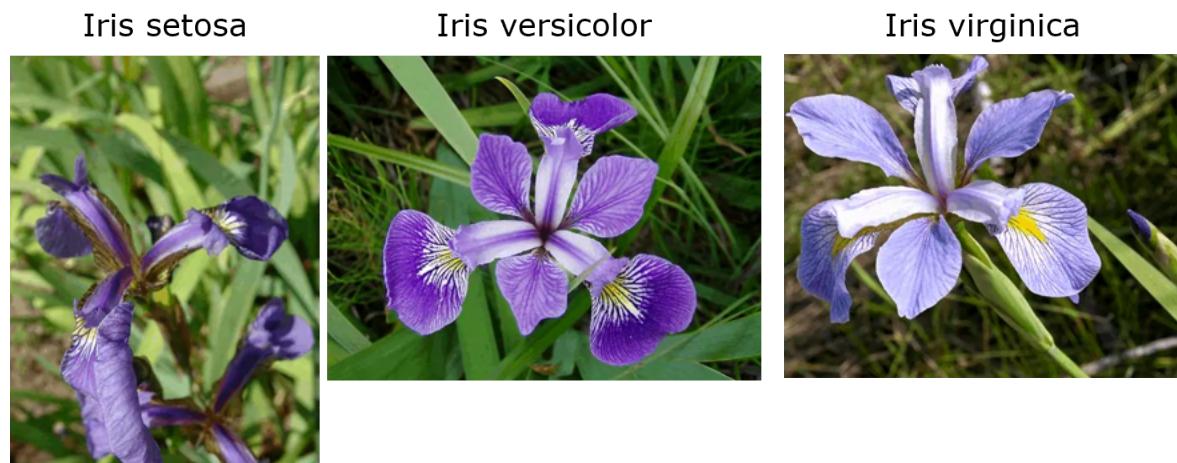
Árboles de Decisión para Clasificación

Datos para clasificación – Iris

Para explicar cómo funcionan los árboles de decisión con problemas de clasificación vamos a usar el conjunto de datos Iris. El problema consiste en clasificar correctamente la variedad de la flor *iris* a partir del ancho y largo de los pétalos y sépalos. Hay tres variedades de flor *iris*: *setosa*, *versicolor* y *virginica*.

Este conjunto de datos tiene 150 muestras:

- 50 iris setosa
- 50 iris versicolor
- 50 iris virginica



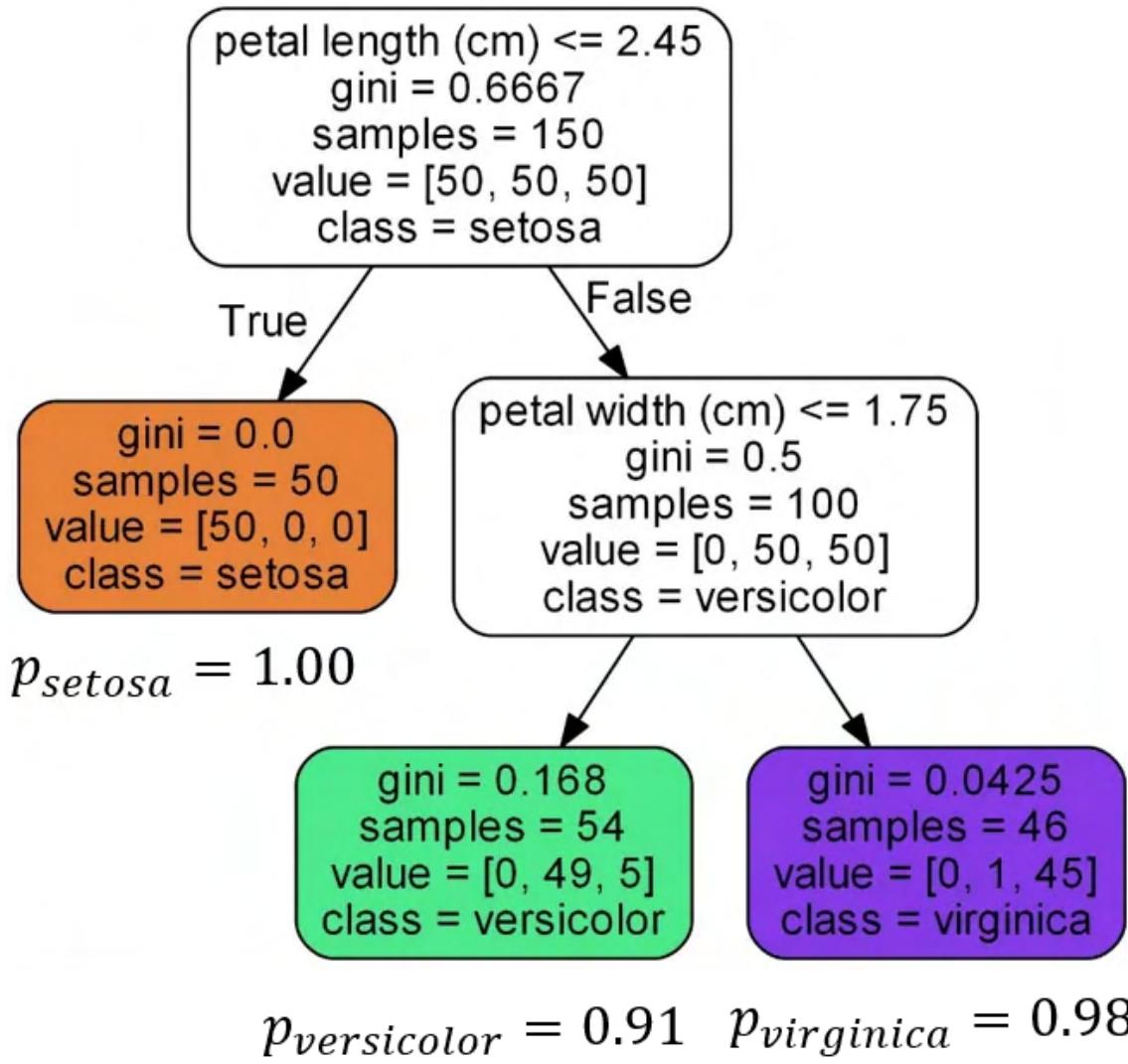
Teoría de los árboles de decisión para clasificación

Si le damos las 150 flores del conjunto de datos Iris a un árbol de decisión para que lo clasifique, nos quedaría un árbol como el que se muestra a continuación. Vamos a aprender a leerlo:

- cada color representa a una clase. El marrón para setosa, el verde para versicolor y el lila para virginica.
- el color es más intenso cuanto más seguros estamos que la clasificación es correcta
- los nodos blancos, por tanto, evidencia la falta de certeza
- Hay dos tipos de nodo:
 - Nodos de decisión: tienen una condición al principio y tienen más nodos debajo de ellos
 - Nodos de predicción: no tienen ninguna condición ni nodos debajo de ellos. También se denominan «nodos hijo»

La información de cada nodo es la siguiente:

- condición: si es un nodo donde se toma alguna decisión
- gini: es una medida de impureza. A continuación veremos cómo se calcula
- samples: número de muestras que satisfacen las condiciones necesarias para llegar a este nodo
- value: cuántas muestras de cada clase llegan a este nodo
- class: qué clase se le asigna a las muestras que llegan a este nodo



Interpretación

La interpretación del árbol de este árbol de decisión sería: si la longitud del pétalo es menos de 2.45 centímetros, entonces la flor iris pertenece a la variedad setosa. Si por el contrario, la longitud del pétalo es mayor que 2.45 centímetros, habría que mirar al ancho del pétalo. Cuando el ancho del pétalo es menor o igual a 1.75 centímetros, pertenece a la variedad versicolor con un 91% de probabilidad. Si no, parece que sería virginica con un 98% de probabilidad.

Gini: medida de impureza

Gini es una medida de impureza. Cuando Gini vale 0, significa que ese nodo es totalmente puro. La impureza se refiere a cómo de mezcladas están las clases en cada nodo. Para calcular la impureza gini, usamos la siguiente fórmula:

$$gini = 1 - \sum_{k=1}^n p_c^2$$

p_c se refiere a la probabilidad de cada clase. Podemos calcularla dividiendo el número de muestras de cada clase en cada nodo por el número de muestras totales por nodo.

Por ejemplo, para el caso del nodo donde la clasificación es *versicolor*, el cálculo sería el siguiente:

$$gini_{versicolor} = 1 - \sum_{k=1}^n p_c^2 = 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 = 0.168$$

¿Cómo se construyen los árboles de decisión para problemas de clasificación?

Los árboles de decisión se construyen usando un [algoritmo voraz](#) que optimiza la siguiente función de coste:

$$J(a, l_a) = \frac{m_{izquierdo}}{m} Gini_{izquierdo} + \frac{m_{derecho}}{m} Gini_{derecho}$$

- a es la abreviatura de atributo (también llamado característica o feature)
- l_a significa el límite del atributo
- m se refiere al número de muestras

Por ejemplo, al comprobar si la longitud del pétalo es menor o igual a 2.45 centímetros, tenemos:

- a = longitud del pétalo
- l_a = 2.45
- m = 150

El algoritmo voraz elige qué atributos y qué límites son los mejores para tomar las decisiones. Al usar un algoritmo voraz, no tenemos la garantía de que este sea el mejor árbol posible. Sin embargo, en la práctica, obtenemos un árbol bastante bueno mucho más rápidamente que si necesitáramos el «mejor árbol posible».

Código python para entrenar y predecir con árboles de decisión para clasificación

```

1. # importamos las librerías que necesitamos
2. from sklearn.datasets import load_iris # datos de iris
3. from sklearn.tree import DecisionTreeClassifier # árbol de decisión para
   clasificación
4.
5. iris = load_iris()
6. print(iris.DESCR) # información sobre del conjunto de datos iris
7.
8. # lo más relevante es:
9. #      :Number of Instances: 150 (50 in each of three classes)
10. #      :Number of Attributes: 4 numeric, predictive attributes and the class
11. #      :Attribute Information:
12. #          - sepal length in cm
13. #          - sepal width in cm
14. #          - petal length in cm
15. #          - petal width in cm
16. #          - class:
17. #              - Iris-Setosa

```

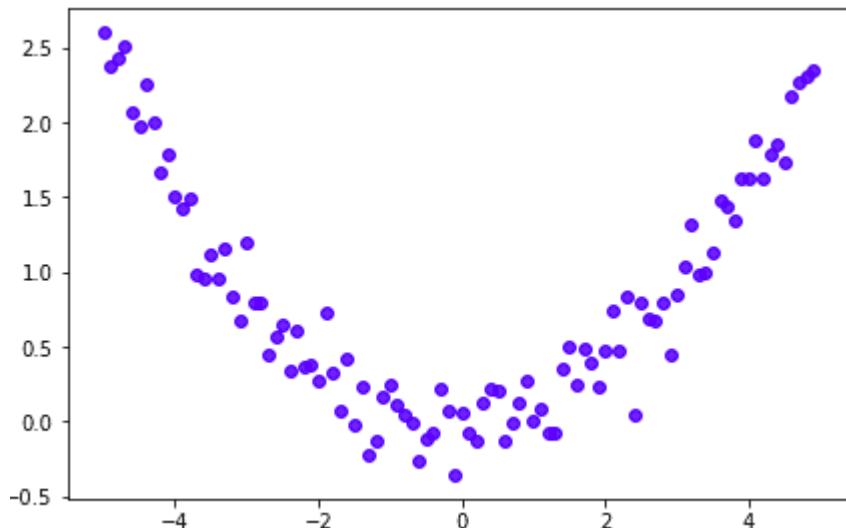
```

18.      #           - Iris-Versicolour
19.      #           - Iris-Virginica
20.
21.      # veamos 4 filas donde ocurre un cambio de clase
22.      print(iris.data[48:52,:])
23.
24.      # da el resultado
25.      # [[5.3 3.7 1.5 0.2]
26.      # [5.  3.3 1.4 0.2]
27.      # [7.  3.2 4.7 1.4]
28.      # [6.4 3.2 4.5 1.5]]
29.
30.      # para la variable a predecir también hacemos lo mismo
31.      print(iris.target[48:52])
32.
33.      # La clase 0 es Iris-Setosa, la 1 es Iris-Versicolor y la 2 es Iris-
34.      # Virginica
35.      # [0 0 1 1]
36.
37.      # Vamos a crear y entrenar un árbol de decisión para clasificar los datos
38.      # de Iris
39.      tree = DecisionTreeClassifier(max_depth=2, random_state=42) # vamos a usar
40.      # un árbol de profundidad 2
41.      tree.fit(iris.data, iris.target) # entrenamiento del árbol
42.
43.      # resulta en
44.      # [0 0 0 1 1 1]
45.
46.      # si queremos saber las probabilidades podemos usar el método predict_proba
47.      print( tree.predict_proba(iris.data[47:53]) )
48.
49.      # la primera clase (Setosa) es la primera columna, la segunda clase en la
50.      # segunda, etc..
51.      # este es el resultado:
52.      # [[1.          0.          0.          ]
53.      # [1.          0.          0.          ]
54.      # [1.          0.          0.          ]
55.      # [0.          0.90740741 0.09259259]
56.      # [0.          0.90740741 0.09259259]
57.      # [0.          0.90740741 0.09259259]]
```

Árboles de Decisión para Regresión

Datos para regresión

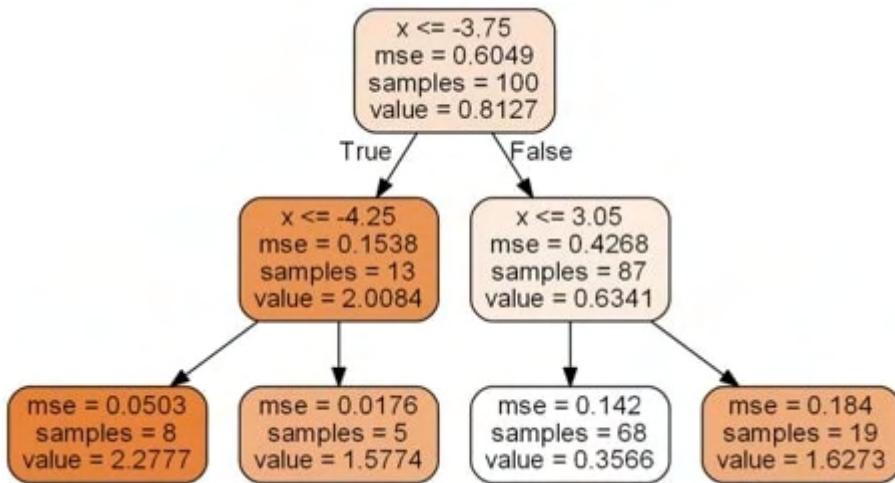
Para explicar cómo funcionan los árboles de decisión para problemas de regresión vamos a usar los datos que se presentan en la siguiente gráfica. Para generarlos he usado la siguiente fórmula en el intervalo [-5, 5]:



$$y = 0.1x^2 + 0.2(\text{Ruido Gaussiano})$$

Teoría de los árboles de decisión para regresión

En el caso de regresión, en lugar de usar Gini como medida de impureza, usamos MSE, el error cuadrático medio. Para este problema, si usamos un árbol de decisión de profundidad 2, obtenemos el siguiente árbol.



Interpretación

La interpretación del árbol de este árbol de decisión sería: si el valor de x es menor que -4.25, predice 2.2777; si está en el intervalo (-4.25, -3.75] predice 1.5774; si está en el intervalo (-3.75, 3.05] predice 0.3566 y si es mayor que 3.05 predice 1.6273.

¿Cómo se construyen los árboles de decisión para problemas de regresión?

Los árboles de decisión para regresión también se construyen usando un [algoritmo voraz](#). Para regresión, la función de coste es la siguiente:

$$J(a, l_a) = \frac{m_{izquierdo}}{m} MSE_{izquierdo} + \frac{m_{derecho}}{m} MSE_{derecho}$$

- a es la abreviatura de atributo (también llamado característica o feature)

- l_A significa el límite del atributo
- m se refiere al número de muestras

MSE es el error cuadrático medio por sus siglas en inglés (Mean Squared Error)

Código python para entrenar y predecir con árboles de decisión para regresión

```

1. # importamos las librerías que necesitamos
2. import numpy as np # NumPy para manipulación numérica
3. np.random.seed(42) # para hacer el código reproducible
4. from sklearn.tree import DecisionTreeRegressor # árbol de decisión para
   regresión
5.
6. #función y = 0.1x^2 + 0.2(Ruido Gaussiano)
7. def f(x):
8.     y = 0.1*np.square(x) + 0.2*np.random.randn(x.size)
9.     return y
10.
11. # Generamos los datos x, y con la función f(x)
12. x = np.arange(-5,5,0.1) # x = [-5, -4.9, -4.8, ... 4.8, 4.9, 5]
13. y = f(x)
14.
15. # Vamos a crear y entrenar un árbol de decisión para regresión
16. tree = DecisionTreeRegressor(max_depth=2, random_state=42) # máxima
   profundidad 2
17. tree.fit(x.reshape(-1,1), y) # entrenamos el árbol de regresión
18.
19. # Ahora predecimos qué valores de y tendríamos para x2 = [-0.7, 0.5, 2.3]
20. x2 = np.array([-0.7, 0.5, 2.3]).reshape(-1,1)
21. print(tree.predict(x2))
22.
23. # obtenemos el siguiente resultado:
24. # [0.35655791 0.35655791 0.35655791]
```

A tener en cuenta cuando usamos Árboles de Decisión

Regularización

Los árboles de decisión de [scikit-learn](#) no están regularizados por defecto. Esto significa que para obtener el menor error posible, pueden crear tantos nodos como necesiten. Esto puede resultar en un modelo muy complejo que funcione muy bien en el conjunto de entrenamiento pero muy mal con datos nuevos.

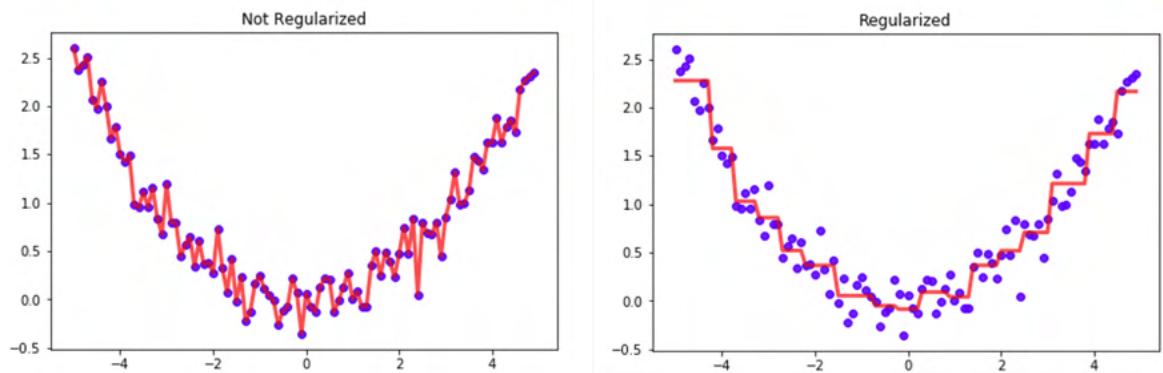
La regularización consiste en limitar de alguna forma las capacidades del modelo para obtener un modelo de aprendizaje automático que sea más simple y generalice mejor.

En scikit-learn podemos usar varios hiper-parámetros para configurar cómo regularizamos los árboles de decisión. Veamos los más usados:

- **max_depth**: la profundidad máxima del árbol. En los ejemplos anteriores hemos usado `max_depth = 2`
- **min_samples_split**: número mínimo de muestras necesarias antes de dividir este nodo. También se puede expresar en porcentaje.
- **min_samples_leaf**: número mínimo de muestras que debe haber en un nodo final (hoja). También se puede expresar en porcentaje.

- **max_leaf_nodes**: número máximo de nodos finales

El siguiente gráfico muestra los datos (en azul) y la predicción (en rojo) de dos árboles. El de la izquierda no está regularizado. El de la derecha está regularizado. La regularización, en este caso, consiste en que los nodos finales (las hojas del árbol) deben cubrir al menos un 5% de las muestras. Como puedes comprobar visualmente, el árbol regularizado va a generalizar mejor y no sufre de [sobreajuste](#).

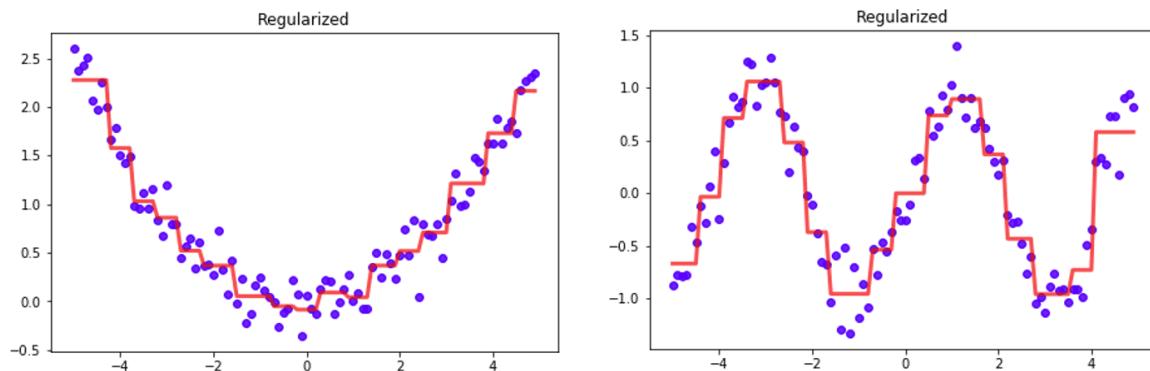


Generalización de los árboles de decisión

Los árboles de decisión son modelos no-paramétricos. Esto significa que no sabemos cuántos parámetros vamos a tener hasta que no veamos los datos y construyamos el árbol. La regresión lineal es un modelo paramétrico porque antes de ver los datos podemos decir con toda certeza cuántos parámetros vamos a necesitar.

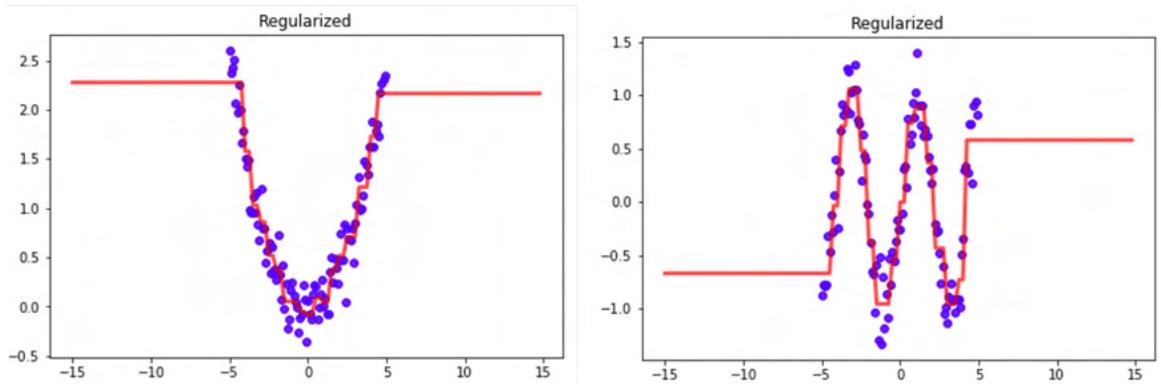
Ventaja de los modelos no-paramétricos

La principal ventaja de los árboles de decisión, como modelo no-paramétrico, es que pueden aprender cualquier cosa. En el siguiente gráfico vemos cómo aprende sin problemas una parábola con ruido (a la izquierda) y una función sinusoidal con ruido (a la derecha).



Desventaja de los modelos no-paramétricos

Por otra parte, al ser un modelo no-paramétrico, no sabe qué es lo que se supone que debe pasar fuera del rango de entrenamiento. Al contrario que en el caso de los modelos paramétricos, no «sabe» extrapolar. En el siguiente gráfico vemos que aunque parece que es capaz de modelar una función parabólica o una sinusoidal, en realidad, no «sabe» lo que está haciendo.



En realidad este problema no es tan malo como pudiera parecer en este gráfico. Normalmente, los problemas para los que usamos aprendizaje automático, tienen muchos atributos y habrá muchos de ellos para los que sí tendremos datos en el rango de entrenamiento. En cualquier caso, siempre conviene asegurarnos que la distribución de los datos que queremos predecir es similar a la distribución de datos que usamos para entrenar el modelo.

Importancia de los atributos

Podemos obtener una estimación de cómo de importante es cada atributo para realizar la predicción. Esto nos va a permitir entender mejor el problema que estemos resolviendo. Por ejemplo, en el caso de la clasificación de las distintas variedades de las flores Iris, tenemos esta importancia estimada:

- sepal length (cm): 0.0
- sepal width (cm): 0.0
- petal length (cm): 0.5619909502262443
- petal width (cm): 0.4380090497737556

Según estos resultados, no necesitamos los datos de los sépalos. El ancho y el largo de los pétalos son suficientes para clasificar las flores Iris en su variedad correcta.

Código python para obtener la importancia de los atributos en un árbol de decisión

```

1. # tree.feature_importances_ es un vector con la importancia estimada de
   cada atributo
2. for name, importance in zip(iris.feature_names,
   tree.feature_importances_):
3.     print(name + ': ' + str(importance))
4.
5. # resultado:
6. # sepal length (cm): 0.0
7. # sepal width (cm): 0.0
8. # petal length (cm): 0.5619909502262443
9. # petal width (cm): 0.4380090497737556

```

Resumen

Los árboles de decisión son una técnica de aprendizaje automático muy utilizada. Su ventajas principales son:

- son fáciles de entender y explicar a personas que todavía no están familiarizadas con técnicas de Inteligencia Artificial
- se adaptan a cualquier tipo de datos
- descubren cuáles son los atributos relevantes

También tienen sus desventajas:

- no extrapolan bien fuera del rango de entrenamiento
- tienen la tendencia a sobre-ajustar, sobre todo si no se regularizan

Recursos

- En este [vídeo](#) explico cómo funcionan los árboles de decisión (en inglés)
- Árbol de Decisión para Clasificación en scikit-learn: [DecisionTreeClassifier](#)
- Árbol de Decisión para Regresión en scikit-learn: [DecisionTreeRegressor](#)

Datos Fundamentales de Empresas Cotizadas en Bolsa

Por Jose Martinez Heras
19/05/2019

SimFin es una plataforma que proporciona datos financieros fundamentales y precios de acciones de empresas cotizadas en mercados estadounidenses (NYSE, NASDAQ, etc.). En el momento de escribir este artículo, SimFin ofrece datos de 2,452 empresas diferentes.

En este artículo vamos a hablar de SimFin, los datos y servicios que ofrece, cómo los puedes descargar y cómo usar python para transformar los datos. Para este análisis, usaremos los datos históricos que son gratuitos.

Servicios Ofrecidos por SimFin

[SimFin](#) ofrece los siguientes servicios:

- **Buscador de acciones:** permite encontrar fácilmente qué empresas cumplen con los criterios de búsqueda que te interesen. Por ejemplo, puedes buscar qué empresas ofrecen un dividendo mayor del 3% y que además hayan incrementado sus ventas en los últimos años. Ofrece la posibilidad de exportar estos datos en un formato compatible con excel.
- **Perfiles detallados de compañías:** si te interesa saber más acerca de una compañía, puedes leer su perfil detallado. Ofrece la posibilidad de exportar estos datos en un formato compatible con excel.
- **Transparencia de datos:** si te interesa, puedes ver las fórmulas de cómo se calculan los ratios y valores contables que no procede directamente de los informes financieros.

Online Research Platform

Stock Screener [go there](#)

- > Filter and screen companies according to 70+ ratios/metrics
- > Filter for growth rates of any fundamental indicator
- > Instant Excel download of all search results

Detailed Company Profiles [go there](#)

- > General company information, TTM financials and ratios
- > Original and standardised financial statements (10+ years back, quarterly and annual data)
- > Instant Excel download of all data

Full Data Transparency

- > All our calculations for ratios and the standardisation of financial statements are 100% transparent, check out all calculation steps for diluted EPS of Apple by clicking [here](#) for example
- > Click on any number in the finder or the company profiles to see all calculation steps as well as the used sources

Datos Fundamentales Financieros (incluye precio de acciones) en SimFin

Ya hay varios sitios online donde podemos obtener automáticamente datos de precios e indicadores técnicos. Lo que hace único a SimFin es que además de los precios históricos,

proporciona también los datos fundamentales de cada empresa a lo largo del tiempo. Así, si nos interesa, también podemos ver cómo los datos fundamentales han cambiado en los últimos años.

SimFin ofrece dos formas de acceder a sus datos financieros: en tiempo real o como descarga de datos históricos

Direct Data Access

API Access [go there](#)

- > 2,000 calls per day available for all SimFin users, more/unlimited calls available for SimFin+ users (read more [here](#))
- > All company and share data accessible, as well as API access to the stock screener

Bulk Download [go there](#)

- > Download the entire SimFin dataset, so that you can get creative and use it for machine learning applications for example
- > All fundamentals, stock prices and ratios available
- > No SimFin+ subscription required (just normal log-in)

En tiempo real (necesita SimFin+)

Si accedemos a los datos financieros de SimFin en tiempo real, podemos saber las últimas novedades de las compañías que nos interesen. Por ejemplo, el precio en tiempo real y los valores fundamentales más recientes. El acceso a estos datos se hace a través de una API (Application Programming Interface), que en español vendría a decir: Interface de Programación de Aplicaciones. Esta API nos va a permitir acceder tanto a los datos como al buscador de acciones.

Para usar este interfaz en tiempo real de acceso a datos financieros necesitamos ser usuarios de SimFin+. Es un servicio de suscripción (de pago). Sin embargo, para probar el interfaz podemos usar la API 2,000 veces al día gratuitamente.

Datos Históricos Fundamentales (gratis)

Los datos históricos fundamentales son ¡gratis! SimFin los ofrece gratis para que podamos experimentar con ellos y podamos evaluar si nuestras ideas funcionan. Para acceder a estos datos hay que registrarse en su web. Por lo demás es bastante fácil. Sólo tenemos que elegir qué datos queremos y recibiremos un fichero comprimido como descarga directa.

En el siguiente ejemplo vemos cómo he seleccionado tantos los datos fundamentales (incluyendo ratios) como los precios. El formato que he elegido, para este ejemplo, ha sido el del uso de los ratios calculados al final del período, usando datos fundamentales anuales, en formato estrecho y separados por comas.

Dataset

Stock prices + Fundamentals / **Stock prices + Fundamentals (Detailed)** / Fundamentals / Fundamentals (Detailed)

This dataset includes both stock prices, shares outstanding and detailed fundamental indicators, including ratios.

Options

Update fundamentals & ratios on

Publishing date / **Period end-date**

(show explanation)

General data format

Wide / **Narrow**

(show explanation)

Time periods fundamentals

TTM / Quarters

(show explanation)

Delimiter string of CSV

Semicolon / **Comma**

Notes

General information on the structure of the bulk download files can be found on the [help page](#).

A class for Python to extract the data from the CSV files (wide data format) as well as usage examples can be found [on Github](#).

 [Download CSV file](#)

Last update on Apr 30, 2019; file size (compressed): 140.4 MB

Lectura de datos fundamentales de SimFin en python

```
1. # importamos las librerías
2. import pandas as pd
3.
4. # lectura del fichero
5. simfin = pd.read_csv('output-mixeddet-ttm-gaps-period-comma-narrow.zip')
6.
7. # inspeccionamos el fichero
8. simfin.head()
```

	Ticker	SimFin ID	Company Industry Classification Code	Indicator Name	date	Indicator Value
0	BRCD	186307		101001 Common Shares Outstanding	2007-10-27	387406.000
1	HPQ	184360		101001 Common Shares Outstanding	2007-10-31	2579714.000
2	SNPS	744723		101003 Common Shares Outstanding	2007-10-31	292730.000
3	CLC	57096		100001 Common Shares Outstanding	2007-11-30	49218.822
4	ALTR	442342		0 Common Shares Outstanding	2007-12-28	314019.000

Los campos disponibles son:

- **Ticker**: el símbolo con el que cotizan en bolsa. Por ejemplo, en la bolsa americana, Apple cotiza con el símbolo AAPL
- **SimFin ID**: código de identificación interno a SimFin
- **Company Industry Classification Code**: qué tipo de industria es de acuerdo con el [estándar de clasificación industrial americano](#)
- **Indicator Name**: el nombre del indicador. Por ejemplo, el número de acciones disponible, retorno de los activos, etc.
- **date**: la fecha
- **Indicator Value**: el valor numérico correspondiente al indicador de esta fila

A lo mejor no necesitamos el identificador de SimFin (SimFin ID). Así que lo podemos eliminar. También podemos renombrar las otras columnas para que sea más fácil trabajar con ellas y convertir la fecha a representación de fecha en [pandas](#).

```
1. del(simfin['SimFin ID']) # borramos la columna "SimFin ID"
2.
3. # usaremos estos nombres más cortos para los atributos
```

```

4. simfin.columns = ['ticker', 'industry', 'indicator', 'date', 'value']
5.
6. simfin['date'] = pd.to_datetime(simfin['date']) # convertimos a fecha de
pandas
7.
8. simfin.head() # visualizamos como queda

```

	ticker	industry	indicator	date	value
0	BRCD	101001	Common Shares Outstanding	2007-10-27	387406.000
1	HPQ	101001	Common Shares Outstanding	2007-10-31	2579714.000
2	SNPS	101003	Common Shares Outstanding	2007-10-31	292730.000
3	CLC	100001	Common Shares Outstanding	2007-11-30	49218.822
4	ALTR	0	Common Shares Outstanding	2007-12-28	314019.000

Como te habrás dado cuenta, cada fila corresponde con un indicador para una empresa. Dependiendo del tipo de aplicaciones que estemos desarrollando, puede ser más práctico reestructurar los datos. Por ejemplo, podríamos reestructurar estos datos para que cada columna correspondiese a un indicador diferente. Vamos a usar las tablas *pivot* en pandas para conseguirlo.

```

1. # usaremos como índice el ticker y la fecha
2. # las columnas serán los indicadores
3. # los valores de cada celda en la tabla, los valores de los indicadores
4. pivot = simfin.pivot_table(index=['ticker', 'date'], columns='indicator',
values='value')
5.
6. # veamos como quedan los últimos valores disponibles para Apple
7. pivot.loc['AAPL'].ffill().tail()

```

indicator	Abnormal Gains/Losses	Accounts Payable	Avg. Basic Shares Outstanding	Avg. Diluted Shares Outstanding	Book to Market	COGS	Cash & Cash Equivalents	Cash From Financing Activities	Cash From Investing Activities	Cash From Operating Activities	... Share Capital	Share Price	Short Term Investments
date													
2019-04-23	0.0	44293.0	4861112.75	4903975.25	0.1587	161654.0	44771.0	-94051.0	35500.0	75831.0	... 40970.0	207.48	41656.0
2019-04-24	0.0	44293.0	4861112.75	4903975.25	0.1587	161654.0	44771.0	-94051.0	35500.0	75831.0	... 40970.0	207.16	41656.0
2019-04-25	0.0	44293.0	4861112.75	4903975.25	0.1587	161654.0	44771.0	-94051.0	35500.0	75831.0	... 40970.0	205.28	41656.0
2019-04-26	0.0	44293.0	4861112.75	4903975.25	0.1587	161654.0	44771.0	-94051.0	35500.0	75831.0	... 40970.0	204.30	41656.0
2019-04-29	0.0	44293.0	4861112.75	4903975.25	0.1587	161654.0	44771.0	-94051.0	35500.0	75831.0	... 40970.0	204.61	41656.0

Ideas para sacarle partido a estos datos

- Análisis Fundamental: puedes intentar determinar el auténtico valor de una acción con [análisis fundamental](#)
- Análisis Técnico: como SimFin también proporciona los datos diarios, puedes también probar a hacer [análisis técnico](#)
- Entender el pasado: puedes usar alguna técnica de aprendizaje automático para resumir de forma concisa qué características tenían las empresas que más subieron en la bolsa estadounidense en los últimos años. Por ejemplo, puedes usar un [árbol de decisión para](#)

[regresión](#) que prediga el retorno de la inversión dentro de un año. Otra posibilidad es usar un [árbol de decisión para clasificación](#) para predecir si el precio de una acción será mayor o menor al cabo de un año.

- Predecir el futuro: este es sin duda la idea más difícil que te propongo. Puedes intentar entrenar un modelo con estos datos para predecir qué va a pasar en el futuro. Como siempre, ten en cuenta su capacidad de generalización. Si generaliza bien, pruébalo durante un tiempo con datos nuevos (por ejemplo, los datos del próximo mes) para ver cómo se comporta de verdad antes de invertir. La idea es estar seguros de que no has usado algún dato futuro accidentalmente en la evaluación del modelo que podría hacer que los resultados parezcan mejor de lo que en realidad son. A este fenómeno se le denomina en inglés *data leakage*. Es español lo podríamos traducir por *filtración de datos*.

Si te decides a hacer algún análisis con los datos fundamentales ofrecidos por SimFin, puedes contárnoslo en los comentarios.

Actualización 29.03.2020

- simfin ha cambiando el formato de los datos, por lo que los ejemplos seguramente dejen de funcionar con el formato nuevo.
- además ha empezado a cobrar por los datos más recientes. Los datos históricos antiguos (con un retraso de 12 meses) son gratuitos. Los puedes descargar en [este enlace](#).

Aguathon: mi solución al primer Hackathon del Agua

Por Jose Martinez Heras
20/05/2019

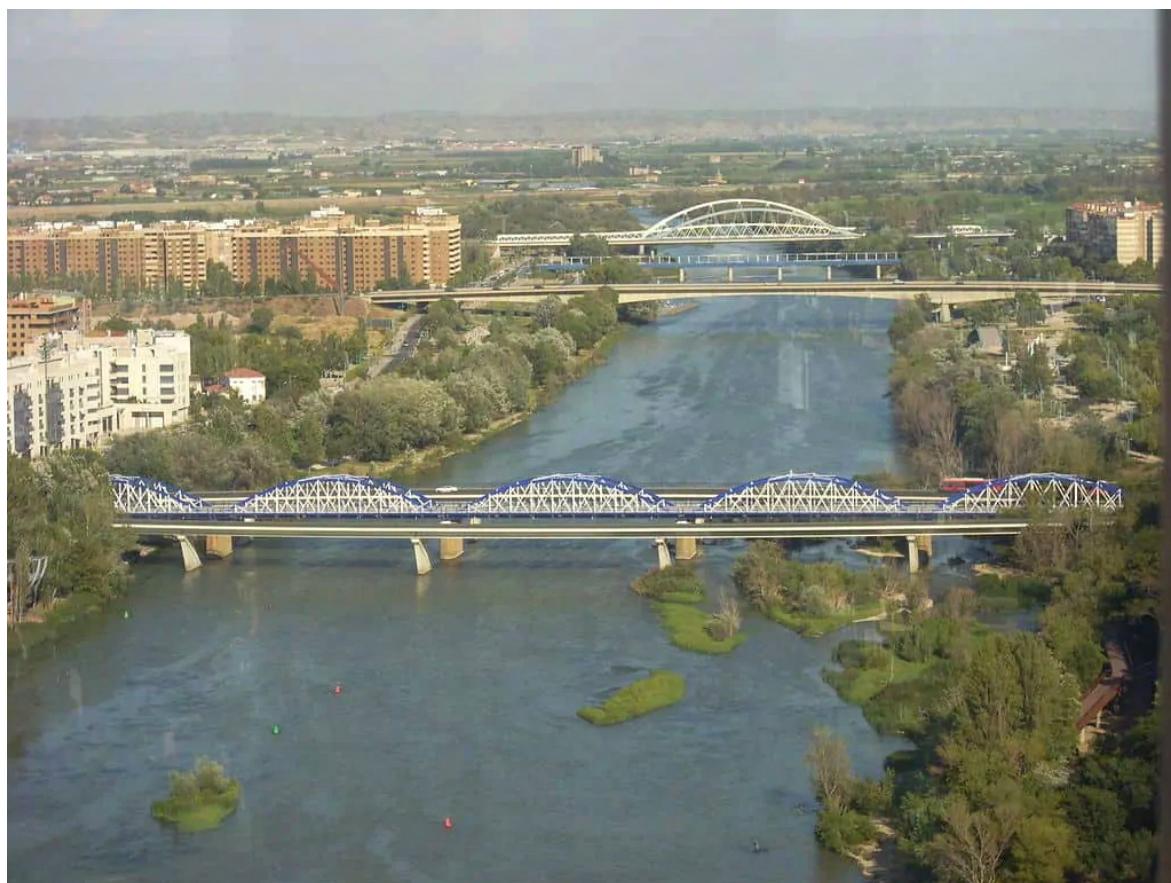
En este artículo explico mi solución al Aguathon: el primer Hackathon del Agua, organizado por ITAINNOVA.

Introducción al Aguathon: el Hackathon del Agua de ITAINNOVA

El Instituto Tecnológico de Aragón ha organizado el [1er Hackathon del Agua, "AGUATHON"](#). A continuación vamos a ver las partes más relevantes para entender en qué consiste este reto, por qué es tan importante, qué tipo de solución necesitan, qué datos proporcionan y cómo evalúan las soluciones propuestas.

Objetivo

El objetivo de este Hackathon es **modelizar el comportamiento del nivel del río Ebro** a su paso por la ciudad de **Zaragoza** para obtener **predicciones realistas** de su variación en cada instante. Con estas predicciones, la Administración Pública (sistemas de emergencia y etc.) podrá **generar alertas automáticas de posibles desbordamientos** (inundaciones) y activar los sistemas de emergencia y supervisión para evitar posibles daños que pueden llegar a ocasionar al ciudadano de la ciudad de Zaragoza.



Río Ebro a su paso por Zaragoza [[wikipedia](#)]

¿Qué forma tiene la predicción?

En concreto el modelo ganador será el que realice predicciones del nivel del río en Zaragoza con mayor precisión. Las predicciones que debe devolver el modelo son a 24, 48 y 72 horas de antelación.

Evaluación de las soluciones

Para evaluar las soluciones, se considerarán dos valoraciones:

- 90% de peso: la media del error cuadrático medio de las predicciones a 24h, 48h y 72h
- 10% de peso: la novedad de los algoritmos entregados comparado con las técnicas del estado del arte y su complejidad

Datos proporcionados

Los datos comprenden desde la fecha-hora: 2008-01-01 00:00:00 a 2018-12-07 23:00:00. En total el archivo contiene 95857 filas. Las columnas son las siguientes:

- *time*: fecha y hora de la medición (horas exactas, minutos y segundos igual cero).
- *ALAGON_NR*: Nivel del río Ebro al paso por Alagón. Unidades: metros.
- *GRISEN_NR*: Nivel del río Jalón al paso por Grisén. Unidades: metros.
- *NOVILLAS_NR*: Nivel del río Ebro al paso por Novillas. Unidades: metros.
- *TAUSTE_NR*: Nivel del río Arba al paso por Tauste. Unidades: metros.
- *TUDELA_NR*: Nivel del río Ebro al paso por Tudela. Unidades: metros.
- *ZGZ_NR*: Nivel del río Ebro al paso por Zaragoza. Unidades: metros.
- *RIESGO*: Variable booleana representando si hubo riesgo de inundación en Zaragoza o no. Sin unidades, es un valor lógico: True/False.
- *pred_24h*: Nivel del río Ebro al paso por Zaragoza 24 horas después del tiempo especificado en la variable time. Unidades: metros.
- *pred_48h*: Nivel del río Ebro al paso por Zaragoza 48 horas después del tiempo especificado en la variable time. Unidades: metros.
- *pred_72h*: Nivel del río Ebro al paso por Zaragoza 72 horas después del tiempo especificado en la variable time. Unidades: metros.

Para que os hagáis una idea, así es la forma que tienen los datos (puedes ver más información [aquí](#)):

```
time,ALAGON_NR,GRISEN_NR,NOVILLAS_NR,TAUSTE_NR,TUDELA_NR,ZGZ_NR,RIESGO,pred_24h,pred_48h,pred_72h
2008-01-01 00:00:00,0.81,0.4375,1.6,0.2675,0.7875,0.74,False,0.75,0.74,0.76
2008-01-01 01:00:00,0.81,0.4725,1.6075,0.265,0.79,0.74,False,0.745,0.7325,0.76
2008-01-01 02:00:00,0.81,0.5425,1.61,0.2675,0.79,0.74,False,0.74,0.73,0.76
2008-01-01 03:00:00,0.8075,0.55,1.61,0.26,0.79,0.74,False,0.74,0.72,0.76
2008-01-01 04:00:00,0.8,0.5525,1.6025,0.265,0.79,0.74,False,0.74,0.72,0.76
2008-01-01 05:00:00,0.8,0.565,1.595,0.2575,0.79,0.74,False,0.74,0.7275,0.76
2008-01-01 06:00:00,0.8,0.57,1.59,0.26,0.7825,0.7425,False,0.74,0.7375,0.76
2008-01-01 07:00:00,0.795,0.5775,1.59,0.2525,0.78,0.75,False,0.74,0.7475,0.76
2008-01-01 08:00:00,0.79,0.58,1.595,0.255,0.775,0.75,False,0.74,0.7525,0.76
2008-01-01 09:00:00,0.79,0.575,1.6,0.2525,0.78,0.75,False,0.74,0.7575,0.76
2008-01-01 10:00:00,0.79,0.57,1.61,0.255,0.78,0.75,False,0.74,0.76,0.76
2008-01-01 11:00:00,0.79,0.57,1.61,0.26,0.78,0.75,False,0.74,0.76,0.76
2008-01-01 12:00:00,0.79,0.57,1.61,0.2525,0.7825,0.75,False,0.74,0.76,0.7575
2008-01-01 13:00:00,0.7825,0.5675,1.615,0.255,0.79,0.75,False,0.74,0.76,0.7575
2008-01-01 14:00:00,0.7875,0.56,1.61,0.2425,0.79,0.75,False,0.74,0.76,0.76
2008-01-01 15:00:00,0.79,0.5575,1.6075,0.2525,0.79,0.75,False,0.74,0.76,0.755
2008-01-01 16:00:00,0.79,0.5475,1.605,0.24,0.79,0.75,False,0.74,0.76,0.75
2008-01-01 17:00:00,0.79,0.535,1.6,0.2475,0.79,0.75,False,0.74,0.77,0.75
2008-01-01 18:00:00,0.79,0.53,1.6,0.2525,0.79,0.75,False,0.74,0.77,0.75
2008-01-01 19:00:00,0.79,0.54,1.6,0.2425,0.79,0.75,False,0.74,0.77,0.75
2008-01-01 20:00:00,0.7875,0.55,1.5925,0.25,0.775,0.75,False,0.74,0.7675,0.75
2008-01-01 21:00:00,0.79,0.5425,1.6,0.24,0.765,0.75,False,0.74,0.77,0.75
2008-01-01 22:00:00,0.79,0.53,1.6075,0.2375,0.7725,0.75,False,0.74,0.76,0.75
2008-01-01 23:00:00,0.79,0.52,1.62,0.245,0.77,0.75,False,0.74,0.76,0.75
2008-01-02 00:00:00,0.79,0.525,1.6225,0.24,0.78,0.75,False,0.74,0.76,0.75
2008-01-02 01:00:00,0.79,0.5325,1.6225,0.2425,0.7725,0.745,False,0.7325,0.76,0.75
```

¿Quién está detrás del Aguathon?

Organiza: Instituto Tecnológico de Aragón ITAINNOVA (Gobierno de Aragón)

Patrocina: Instituto Aragonés del Agua (Gobierno de Aragón)

Colaboran: Confederación Hidrográfica del Ebro (CHE) – Ministerio para la Transición Ecológica (Gobierno de España); Sistema Automático de Información Hidrológica SAIH; Universidad de Zaragoza; Grupo de Investigación COSMOS y Cluster ZINNAE.

Datos y características para crear los modelos del Aguathon

Una vez descrito el problema a resolver en este hackathon del agua, explico qué datos he usado y las características (features) que he calculado.

Datos que he usado para el entrenamiento

No he usado todas las filas que había disponibles. Como el modelo sólo se evalúa en las filas donde hay riesgo, sólo he usado esas filas. Además, he incluido hasta 16 filas anteriores (correspondientes a 16 horas antes de cada “bloque de riesgo”) porque las necesito para calcular las características.

Características (features) que he creado

Características relativas a la cantidad de agua

- sumNR = suma de todos los niveles de agua
- sumNR-TUDELA_NR = sumNR restándole el nivel de TUDELA_NR
- sumNR-TUDELA_NR-NOVILLAS_NR = sumNR-TUDELA_NR restándole NOVILLAS_NR
- NOVILLAS_NR_y_TUDELA_NR = NOVILLAS_NR + TUDELA_NR
- ebroNR_SUM = TUDELA_NR + NOVILLAS_NR + ALAGON_NR + ZGZ_NR
- no_ebroNR_SUM = GRISEN_NR + TAUSTE_NR

Características polinómicas de grado 2

Cálculo de todas las interacciones polinómicas de grado 2 entre los atributos originales y las características relativas a la cantidad de agua. Sin embargo no considero cada elemento al cuadrado.

Características por diferencias

Para todas las características calculadas anteriormente F, calculamos las diferencias a diferentes intervalos: $F(t) - F(t-1)$, $F(t) - F(t-2)$, $F(t) - F(t-4)$, $F(t) - F(t-8)$, $F(t) - F(t-16)$

Características de ventana deslizante

Para todas las características calculadas hasta ahora, además de calcular en ventanas deslizantes de tamaño 2, 4, 8, 16 las siguientes características:

- Medias móviles exponenciales
- Valor mínimo
- Valor máximo
- Valor normalizado = $(\text{valor} - \text{mínimo}) / (\text{máximo} - \text{mínimo})$

Características por fechas

La única característica de fecha que he incluido es el día del año de cada muestra

Datos externos

La organización del hackathon permite usar datos externos. Al mismo tiempo recomienda que los datos externos que se usen «no sean dependientes de un instante de tiempo específico». En su lugar, recomiendan datos del tipo «media de las precipitaciones en la primera quincena de Abril en los últimos 10 años, tiempo máximo sin precipitaciones en otoño en los últimos 50 años, distancias, ...»

Con el tipo de algoritmos de aprendizaje automático que tengo pensado utilizar, estos datos estadísticos no serían muy relevantes porque serían constantes para cada muestra. Como estos algoritmos pueden también aprender estos sesgos, no necesitamos este tipo de información. Así que no he usado ningún tipo de dato externo.

Modelos de Aprendizaje Automático que he usado en el Aguathon

He usado la media aritmética de dos modelos de aprendizaje automático.

Predicción del nivel del Ebro a su paso por Zaragoza

Este modelo predice el nivel del río Ebro a su paso por Zaragoza con 24h, 48h y 72h de antelación

Predicción relativa al nivel actual del Ebro a su paso por Zaragoza

Este modelo predice cuánto va a subir (o bajar) el nivel del río Ebro con respecto al nivel actual. Así que predice:

- ZGZ_NR 24h – ZGZ_NR
- ZGZ_NR 48h – ZGZ_NR
- ZGZ_NR 72h – ZGZ_NR

Para usar este modelo necesitamos sumarle a las predicciones, el nivel actual del río Ebro a su paso por Zaragoza. Por ejemplo, para la predicción a 24 horas, le sumamos el nivel del río Ebro a su paso por Zaragoza actualmente más la predicción de cuánto va a subir (o bajar) el nivel en las siguientes 24 horas.

Predicción combinando el modelo de predicción del nivel del Ebro y el modelo de predicción relativa

La predicción final se hace haciendo la media aritmética de esos 2 modelos. Los dos modelos usan los mismos datos de entrada descritos en la sección anterior. Sólo la salida es diferente.

La ventaja de usar estos dos modelos es que son complementarios. La características (features) que son importantes para un modelo no lo son tanto para el otro, y viceversa. Esto hace que los errores de los modelos no estén correlados y que al combinarlos sus errores se compensen, dando lugar a resultados mejores. La siguiente tabla pone en evidencia qué características son importantes para cada modelo y permite comprobar cómo se complementan.

Importancia de las características (features) para cada uno de los 2 modelos [10 atributos más importantes]

Predicción del Nivel del Ebro en Zaragoza

0.0728 TUDELA_NR
0.0658 max2(TUDELA_NR)

Predicción del cambio en el Nivel del Ebro en Zaragoza

0.5788 normalized2(ALAGON_NR
NOVILLAS_NR_y_TUDELA_NR)

0.0523 min2(TUDELA_NR)	0.0125 ewma16(diff1(NOVILLAS_NR TUDELA_NR))
0.0494 max4(TUDELA_NR)	0.0125 ewma16(diff2(NOVILLAS_NR TUDELA_NR))
0.0435 max2(TUDELA_NR NOVILLAS_NR_y_TUDELA_NR)	0.011 ewma16(diff2(TUDELA_NR))
0.0383 min2(TUDELA_NR sumNR)	0.0084 diff1(sumNR ebroNR_SUM)
0.0352 ewma2(TUDELA_NR)	0.0081 doy
0.0347 TUDELA_NR NOVILLAS_NR_y_TUDELA_NR	0.0079 ewma16(diff4(TUDELA_NR))
0.0336 min4(TUDELA_NR NOVILLAS_NR_y_TUDELA_NR)	0.0072 diff1(sumNR-TUDELA_NR-NOVILLAS_NR NOVILLAS_NR_y_TUDELA_NR)
0.029 min2(NOVILLAS_NR TUDELA_NR)	0.0072 ewma16(diff8(TUDELA_NR))
	0.0067 ewma8(diff4(TUDELA_NR ebroNR_SUM))

Técnicas de Machine Learning

Como técnica de aprendizaje automático he usado la implementación de [ExtraTreesRegressor](#) de [scikit-learn](#) para los 2 modelos.

Quería utilizar alguna técnica que usara “ensembles”. La ventaja del uso de ensembles es que mejoran la generalización. La generalización es la capacidad que tienen los modelos de machine learning de obtener buenas predicciones con datos nuevos.

En muchas competiciones de machine learning, los bosques aleatorios (random forests) son la elección por defecto. Es un ensemble de varios árboles de decisión que generaliza bastante bien.

Sin embargo, como la velocidad de entrenamiento y predicción eran relevantes, me he decantado por el uso de ExtraTrees. Los ExtraTrees son más rápidos que los RandomForests porque no intentan encontrar el valor óptimo donde tomar la decisión, sino que esta decisión se toma al azar. Esto resulta en un entrenamiento más rápido y, normalmente, en una generalización mejor ya que también ayuda a disminuir el sesgo durante el entrenamiento.

Evaluación: estimación del rendimiento de los modelos para el Hackathon del Agua

Para evaluar el rendimiento del modelo, he dividido los datos proporcionados en 2 conjuntos:

- Entrenamiento: para construir el modelo
- Validación: para evaluar el modelo

La práctica habitual para construir el conjunto de validación es la elección de muestras aleatoriamente. Sin embargo, para el problema del Aguathon esto no sería una buena idea porque los resultados de evaluación sería artificialmente buenos. Esto se debe a que el modelo estaría entrenado con datos futuros con respecto a algunas muestras del conjunto de validación. Para evitar este fenómeno, he usado el primer 80% de los datos proporcionados como conjunto de entrenamiento y el 20% final como validación.

He usado este conjunto de validación para ir viendo cómo mejoraba el rendimiento a medida que iba añadiendo más características y también para ver qué otras características que había pensando no añadían ningún valor predictivo al modelo.

Resultados de Evaluación

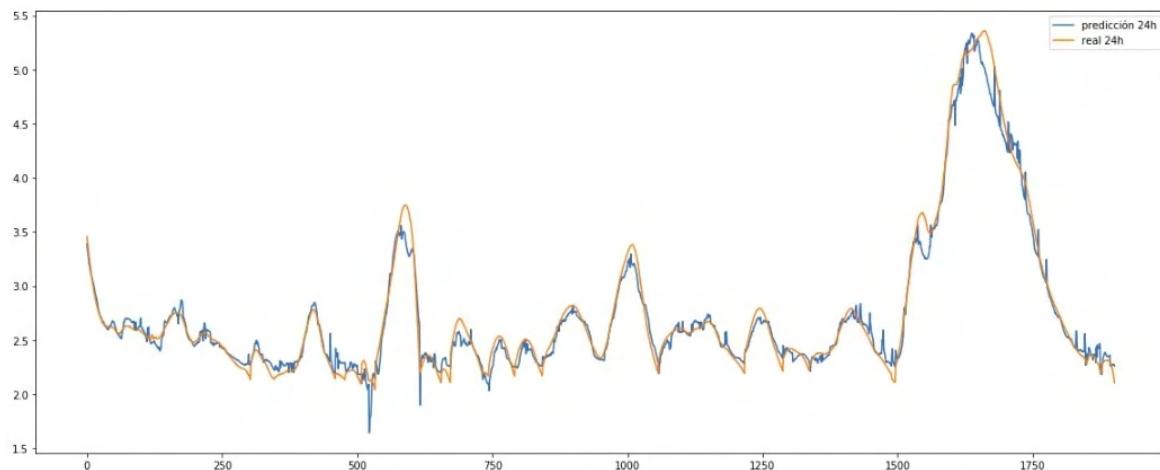
A continuación, muestro los resultados de evaluar los dos modelos. También podemos apreciar cómo los resultados mejoran al combinar los dos modelos.

En el conjunto de validación los errores cuadráticos medios que obtengo son:

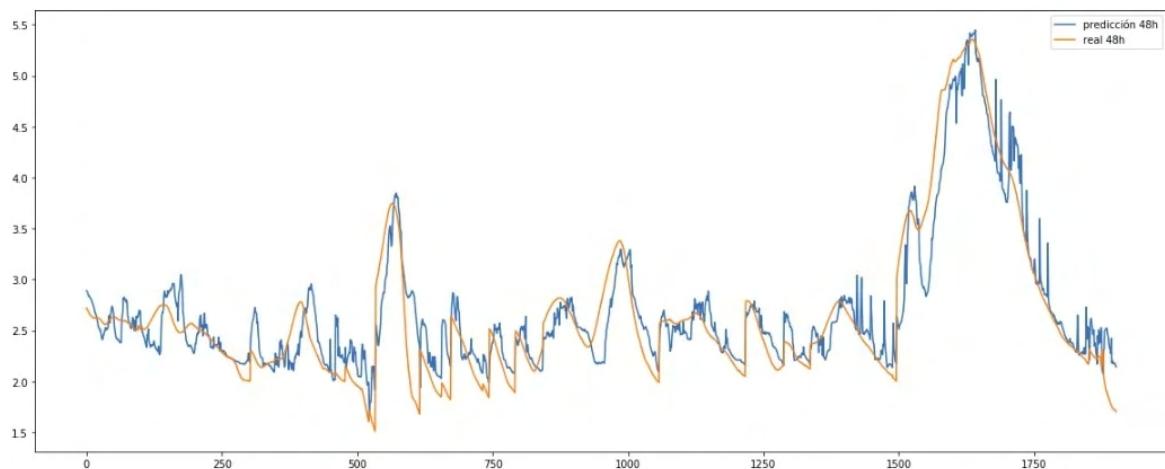
Error Cuadrático Medio	Predicción Nivel	Predicción Relativa	Predicción combinada
24h	0.015962	0.039450	0.010716
48h	0.086630	0.105484	0.076918
72h	0.259210	0.272680	0.243048

A continuación muestro cómo quedan las predicciones combinadas en el conjunto de validación. Las gráficas sólo consideran los datos que contienen riesgo (y las 16 horas anteriores para cada bloque de riesgo)

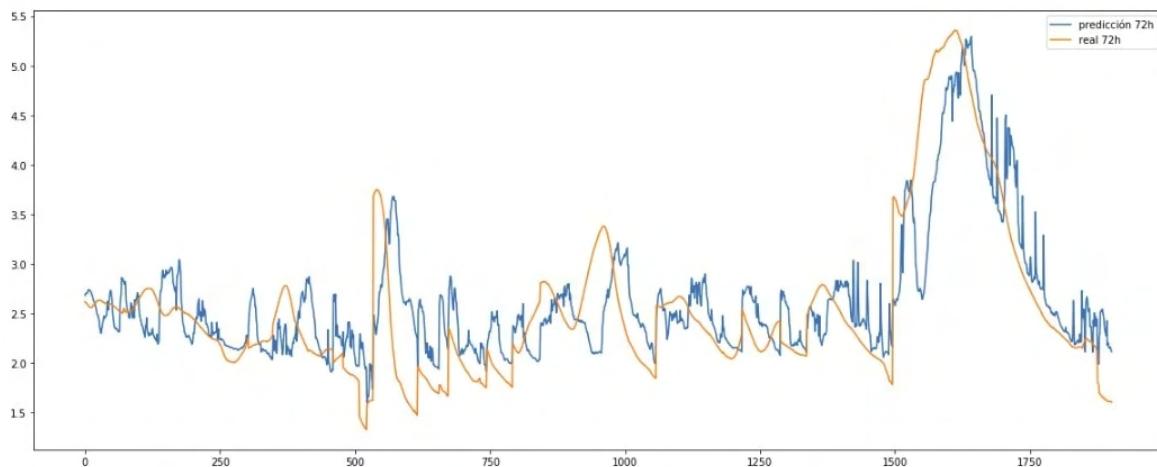
Ejemplo de la calidad de predicción a 24h



Ejemplo de la calidad de predicción a 48h



Ejemplo de la calidad de predicción a 72h



Modelo final enviado al Aguathon

El modelo final consiste en la media aritmética de los dos modelos entrenados con todos los datos disponibles (tanto los que he usado para entrenamiento como para validación).

Modelo Sustituto (surrogate) para explicar qué hacen los modelos del Aguathon

Los ExtraTrees son un modelo de caja negra. Teóricamente podemos ver lo que hace cada árbol ... pero en la práctica es difícil entender lo que hace. Para poder ganar visibilidad en lo que hace los modelos de caja negra, vamos a usar un modelo sustituto (surrogate en inglés).

El objetivo de ITAINNOVA es el de considerar la incorporación de los modelos del Aguathon en un sistema automático de alarma de inundaciones que puede avisar a los sistemas de emergencia. Para justificar las decisiones del modelo de caja negra, podemos usar modelos sustitutos de caja blanca ... aunque su rendimiento sean algo peor.

¿Cómo funcionan los modelos sustitutos (surrogates)?

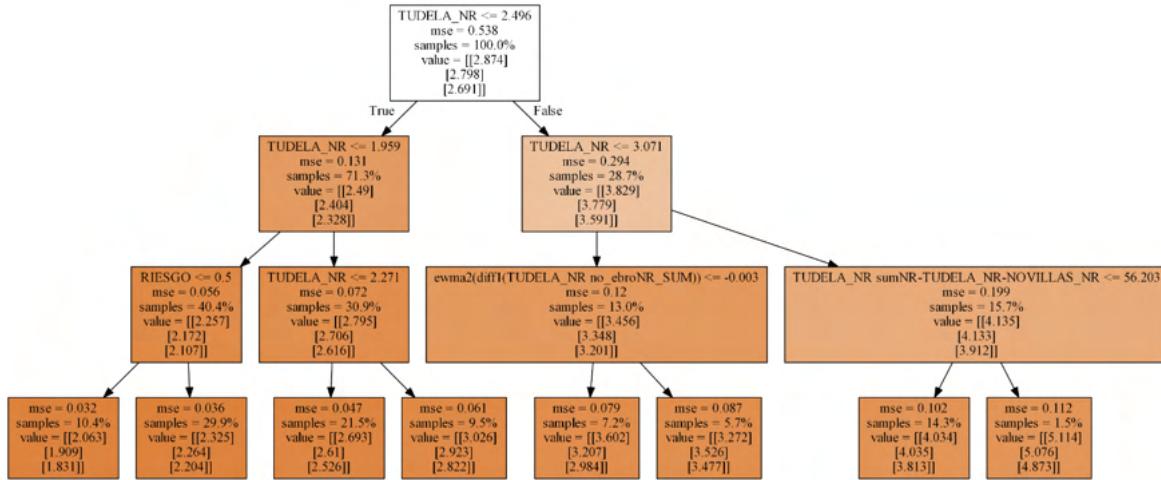
El modelo sustituto es un modelo de caja blanca que se usa para describir lo que hace un modelo de caja negra (posiblemente, con pérdida de rendimiento).

El modelo sustituto se construye usando:

- Como entrada, los datos que queramos
- Como salida, la predicción del modelo de caja negra para los mismos datos de entrada

El modelo sustituto

El modelo sustituto que elegido es un árbol de decisión de profundidad 3. El árbol lo he construido usando como entrada el conjunto de entrenamiento y como salida, las predicciones del ExtraTreeRegressor para los mismos datos. Aquí tenéis el árbol de decisión sustituto:



En cada nodo final podéis observar el vector de predicción para [24h, 48h, 72h]

Nota: la única razón de usar profundidad 3 en el modelo sustituto es porque queda mejor en la documentación. Un árbol de decisión sustituto más profundo daría mejores resultados y seguiría siendo explicable para pocas profundidades. Podéis hacer click en el árbol para obtener una imagen ampliada.

Evaluación del modelo sustituto

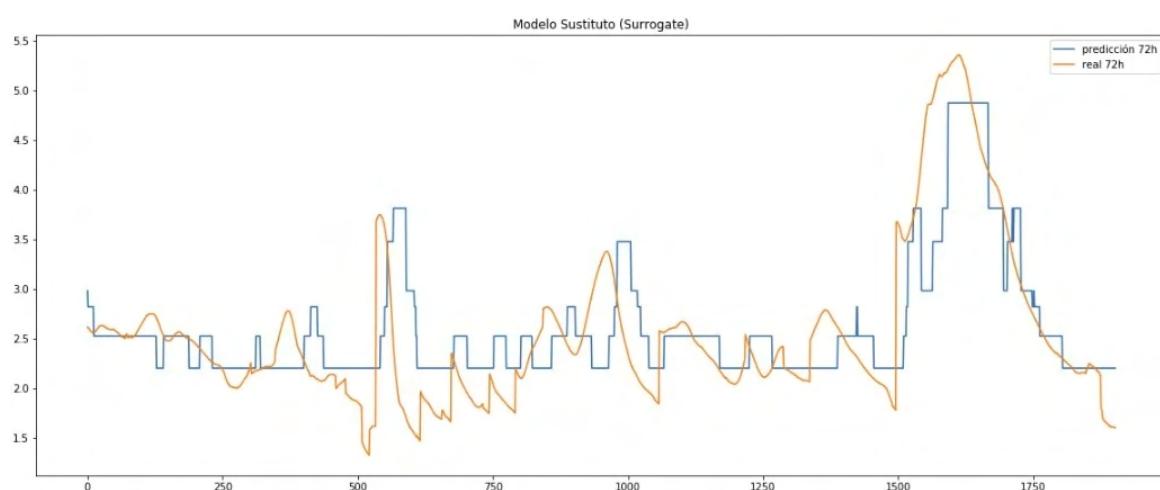
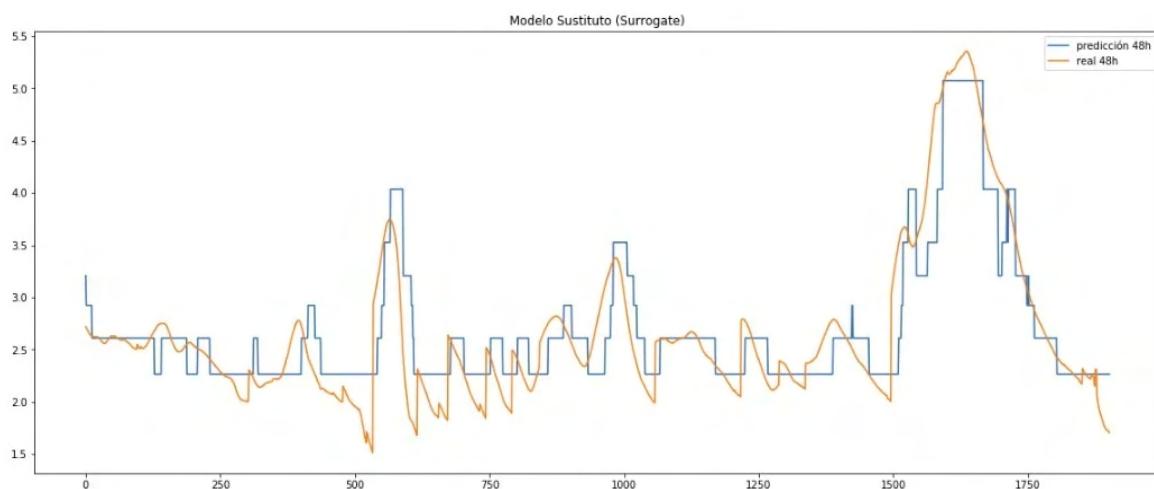
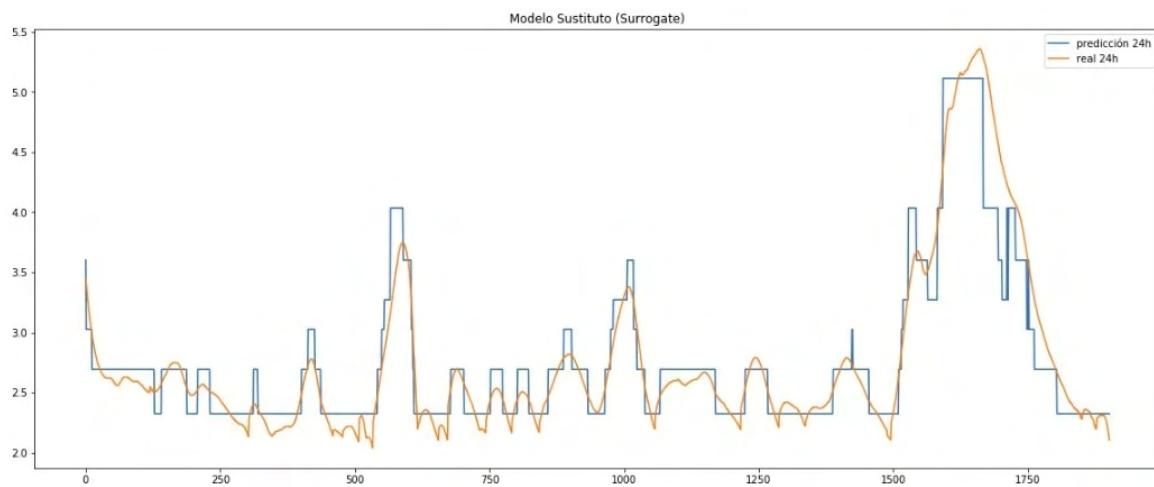
El árbol de decisión sustituto es muy simple y fácil de entender. Vamos a comparar el rendimiento en el conjunto de validación de combinar los dos modelos y su modelo sustituto.

Error Cuadrático Medio (mejor cuanto más bajo)	Predicción Combinada (Caja Negra)	Árbol de Decisión Sustituto (Caja Blanca)
Predicción a 24 horas	0.010716	0.050637
Predicción a 48 horas	0.076918	0.124144
Predicción a 72 horas	0.243048	0.292900

Como era de esperar, el rendimiento del modelo sustituto de caja blanca es peor. Aun así, su rendimiento es sorprendentemente bueno si tenemos en cuenta su simplicidad. Si queremos mejorar el rendimiento del modelo sustituto conservando la explicabilidad, podríamos aumentar su profundidad.

Ejemplos con el modelo sustituto

A continuación vemos las predicciones del modelo sustituto en el conjunto de validación para predicción a 24h, 48h y 72h.



Recomendaciones para mejorar la predicción del nivel del río Ebro a su paso por Zaragoza

Uso de datos del nivel del agua en otras ciudades

Como podéis ver en el modelo sustituto de caja blanca, el nivel del Ebro a su paso por Tudela es lo que más influencia tiene en el modelo. Esto funciona bastante bien para las predicciones a 24h y 48h, pero no es suficiente para predicciones a 72h.

Sin ser un entendido en hidrología, me da la impresión de que el agua que hay en Tudela llega a Zaragoza por el Ebro en menos de 2 días ... y que por eso es tan difícil predecir a 3 días. Si tuviésemos acceso al nivel del río en otros puntos del Ebro anteriores a Tudela, quizás las predicciones serían mejores.

Recomendación 1: añadir nivel del Ebro y afluentes de Ebro en otros lugares anteriores a Tudela

Uso de datos de datos meteorológicos

Creo que el uso de datos meteorológicos (lluvia sobre todo) en distintos puntos geográficos ayudaría a mejorar la predicción.

Una posibilidad sería usar los datos abiertos de la Agencia Estatal de Meteorología (AEMET) disponibles como [datos abiertos meteorológicos](#). La resolución de estos datos es diaria. Si se pudiese establecer alguna colaboración con la AEMET, quizá se podría usar los datos por horas

Recomendación 2: añadir al modelo datos meteorológicos

Uso de previsiones meteorológicas

Las predicciones meteorológicas están mejorando mucho últimamente. Si añadiésemos a los datos disponibles, el pronóstico de lluvia en 24h, 48h, 72h seguramente podríamos obtener mejores resultados.

El problema podría ser que el histórico de predicciones meteorológicas mezcle metodologías de predicción. Aun así, merece la pena investigarlo y, eventualmente, reducir el conjunto de entrenamiento para ser compatible con la metodología de predicción meteorológica actual.

Recomendación 3: añadir al modelo predicciones meteorológicas

Conclusiones de mi participación en el Aguathon

Por mi parte, quiero creer que he ayudado a mejorar el sistema de predicción de posibles inundaciones. Espero que las predicciones sean lo bastante buenas como para que puedan usarse para generar alertas automáticas. También espero que mi propuesta de usar modelos sustitutos explicables ayude a mejorar la aceptación de técnicas de Inteligencia Artificial en la ayuda a la toma de decisiones.

Agradecimientos al equipo de ITAINNOVA

Me imagino el esfuerzo que requiere organizar este hackathon: preparar los datos, decidir qué datos dar y cuáles no, publicitar el Aguathon, dar soporte a los participantes, construir una plataforma para evaluar automáticamente las soluciones, leer los documentos de las soluciones más prometedoras ...

Me gustaría dar las gracias al equipo de ITAINNOVA por la organización de este primer Hackathon del Agua tan interesante.

Ensembles: voting, bagging, boosting, stacking

Por Jose Martinez Heras
31/05/2019

Un **ensemble** es un conjunto de modelos de machine learning. Cada modelo produce una predicción diferente. Las predicciones de los distintos modelos se combinan para obtener una única predicción.

La ventaja que obtenemos al combinar modelos diferentes es que como cada modelo funciona de forma diferente, sus errores tienden a compensarse. Esto resulta en un mejor error de generalización.

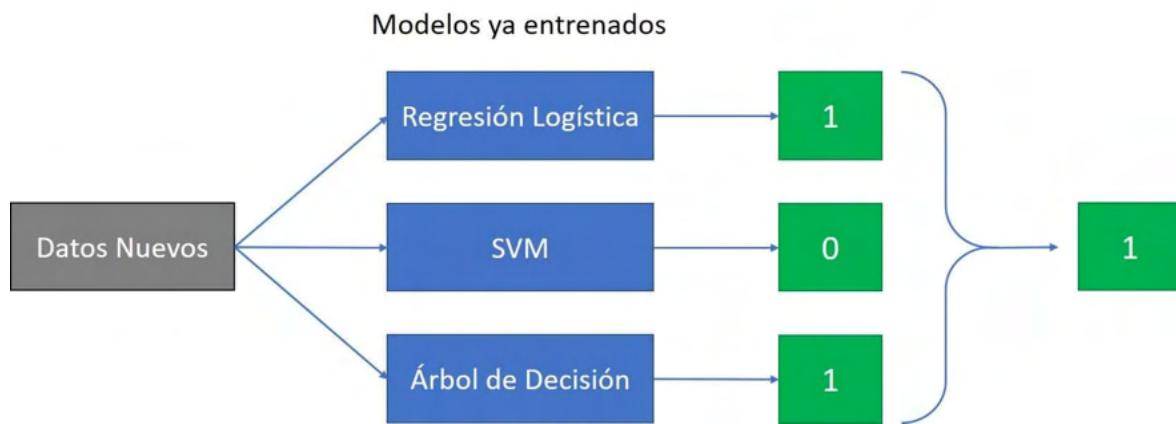
Hay varias formas de construir estos ensembles:

- votación por mayoría
- bagging
- boosting
- stacking

Votación por mayoría

Podemos entrenar varios modelos de aprendizaje automático con los mismos datos. Cuando tengamos datos nuevos, obtendremos una predicción de cada modelo. Cada modelo tendrá asociado un voto. De esta forma, propondremos como predicción final lo que voten la mayoría de los modelos.

Hay otra forma de combinar las votaciones. Cuando los modelos de machine learning dan una probabilidad, podemos usar el «voto suave» (*soft-voting*). En el voto suave, se le da más importancia a los resultados en los que algún modelo esté muy seguro. Es decir, cuando la predicción está muy cercana a la probabilidad 0 ó a 1, se le da más peso a la predicción de ese modelo.



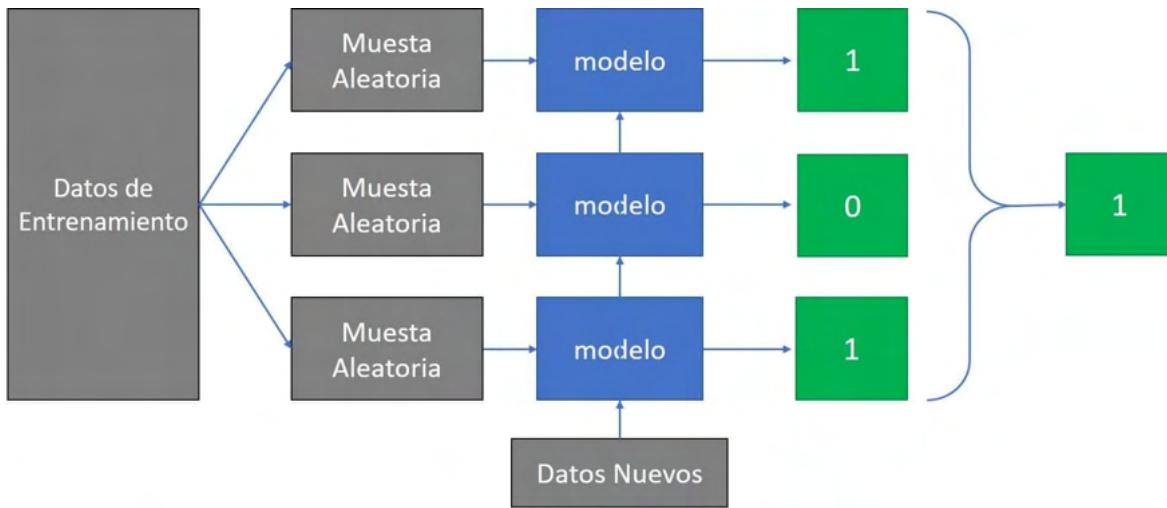
Cuando usamos modelos diferentes, los errores se compensan y la predicción combinada generaliza mejor. Por eso, no tiene sentido hacer un ensemble de votación por mayoría con el mismo tipo de modelo. Por ejemplo, si entrenamos 3 árboles de decisión con los mismos datos y combinamos sus resultados con votación por mayoría, vamos a obtener los mismos resultados que si sólo usásemos un sólo árbol. Para poder combinar muchos árboles de decisión necesitamos *bagging*.

Bagging

Cuando usamos bagging, también combinamos varios modelos de machine learning. A diferencia del voto por mayoría, la forma de conseguir que los errores se compensen entre sí, es que cada

modelo se entrena con subconjuntos del conjunto de entrenamiento. Estos subconjuntos se forman eligiendo muestras aleatoriamente (con repetición) del conjunto de entrenamiento.

Los resultados se combinan, para problemas de clasificación, igual que hemos visto en la votación por mayoría, con el voto suave para los modelos que den probabilidades. Para problemas de regresión, normalmente se utiliza la media aritmética.



Los bosques aleatorios (random forests), no son ni más ni menos, que un ensemble de árboles de decisión combinados con bagging. Aunque bagging se puede usar con cualquier modelo, la opción de usarlo con árboles de decisión es muy popular por motivos de rendimiento. Normalmente, es muy rápido construir un árbol de decisión.

Boosting

En el boosting, cada modelo intenta arreglar los errores de los modelos anteriores. Por ejemplo, en el caso de clasificación, el primer modelo tratará de aprender la relación entre los atributos de entrada y el resultado. Seguramente cometerá algunos errores. Así que el segundo modelo intentará reducir estos errores. Esto se consigue dándole más peso a las muestras mal clasificadas y menos peso a las muestras bien clasificadas. Para problemas de regresión, las predicciones con un mayor error cuadrático medio tendrán más peso para el siguiente modelo.

Hay muchas implementaciones de ensambles que usan boosting. El primero fue el [AdaBoost](#). Los más usados actualmente son [xgboost](#), [CatBoost](#) y [LightGBM](#).

Stacking (modelos apilados)

Cuando hablamos de un ensemble de stacking, nos referimos a que estamos apilando modelos. Cuando apilamos modelos, lo que en realidad estamos haciendo, es usar la salida de varios modelos como la entrada de varios modelos. Por ejemplo, el ganador del Aguathón, «ensambló» varios modelos apilándolos para conseguir un rendimiento mejor. El problema consistía en predecir el nivel del río Ebro a su paso por Zaragoza a 24h, 48h y 72h. [César Montenegro](#) usó bosques aleatorios para predecir el nivel del Ebro a 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72 horas y luego uso otro bosque aleatorio que usaba estas predicciones como entrada y predecía el nivel del río a 24h, 48h y 72h. Esta forma de combinar los modelos, sin duda contribuyó a un mejor rendimiento de su solución.

Resumen de Ensembles

Los ensambles son estrategias de combinación de modelos de machine learning para mejorar su generalización. Hemos visto los cuatro tipos de ensambles más usados: votación por mayoría, bagging, boosting y stacking.

Recursos para Ensembles

- [\[vídeo\]](#) donde explico voting y bagging aplicado y su aplicación a bosques aleatorios
- [Ensembles en scikit-learn](#) incluyendo votación por mayoría, AdaBoost, Bagging genérico, Bosques Aleatorios, etc.
- [xgboost](#): eXtreme Gradient Boosting
- [CatBoost](#): Yandex CatBoost
- [LightGBM](#): Light Gradient Boosting Machine

Random Forest (Bosque Aleatorio): combinando árboles

Por Jose Martinez Heras
10/06/2019

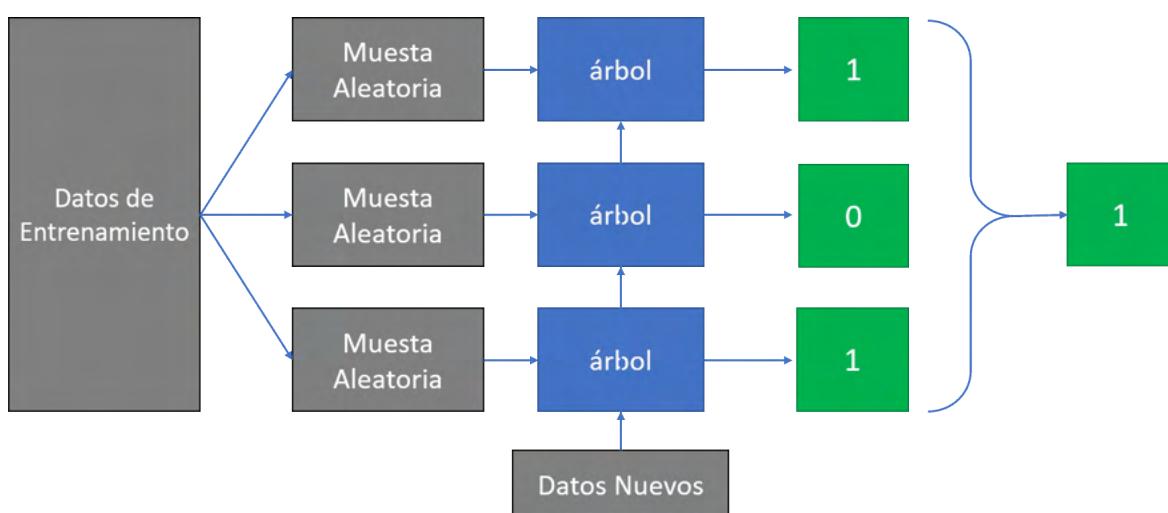
Un Random Forest (Bosque Aleatorio), es una técnica de aprendizaje automático muy popular. Los Random Forests tienen una capacidad de generalización muy alta para muchos problemas.

Limitaciones de los Árboles de Decisión

Los árboles de decisión tienen la tendencia de [sobre-ajustar \(overfit\)](#). Esto quiere decir que tienden a aprender muy bien los datos de entrenamiento pero su generalización no es tan buena. Una forma de mejorar la generalización de los árboles de decisión es usar [regularización](#). Para mejorar mucho más la capacidad de generalización de los árboles de decisión, deberemos combinar varios árboles.

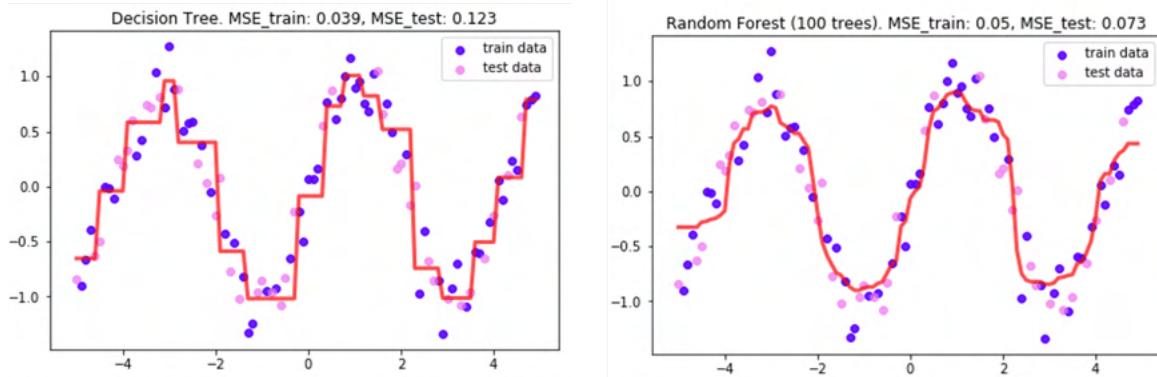
¿Qué es un Random Forest?

Un **Random Forest** es un conjunto (*ensemble*) de árboles de decisión combinados con [bagging](#). Al usar bagging, lo que en realidad está pasando, es que distintos árboles ven distintas porciones de los datos. Ningún árbol ve todos los datos de entrenamiento. Esto hace que cada árbol se entrene con distintas muestras de datos para un mismo problema. De esta forma, al combinar sus resultados, unos errores se compensan con otros y tenemos una predicción que generaliza mejor.



Diferencia intuitiva entre un árbol de decisión y un random forest

En la siguiente imagen puedes ver la diferencia entre el modelo aprendido por un árbol de decisión y un random forest cuando resuelven el mismo problema de regresión. Este random forest en particular, utiliza 100 árboles.



Como puedes ver, tanto el árbol de decisión como el bosque aleatorio, tienen un error cuadrático medio (MSE) de entrenamiento pequeño y similar. Sin embargo, el error de generalización del random forest es mucho mejor. Esto se puede ver también en que el modelo aprendido es mucho más suave como se puede observar en la figura.

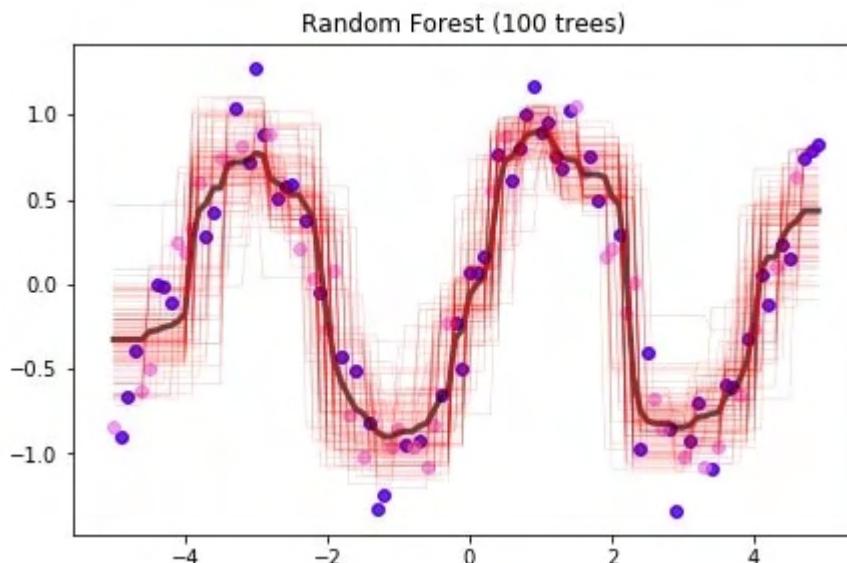
¿Cómo se combinan las predicciones?

Para problemas de [clasificación](#), se suelen combinar los resultados de los árboles de decisión usando soft-voting (voto suave). En el voto suave, se le da más importancia a los resultados en los que los árboles estén muy seguros.

Para problemas de [regresión](#), la forma más habitual de combinar los resultados de los árboles de decisión, es tomando su media aritmética.

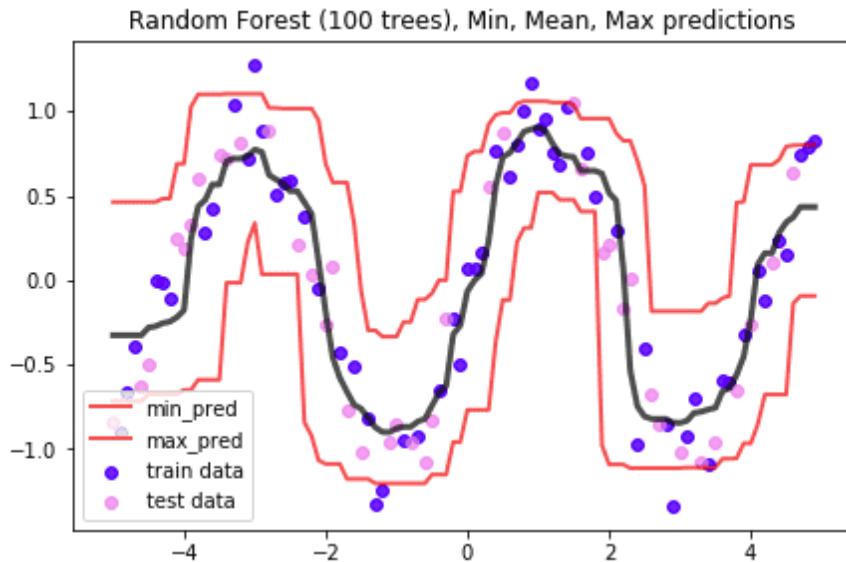
Podemos inventar nuestra propia forma de combinar predicciones

También podemos inventarnos nuestra propia forma de combinar los resultados de un bosque aleatorio. Esto es posible porque podemos acceder a las predicciones individuales de cada árbol. Así, podemos hacer el uso de estas predicciones que nos interesen.



Predicción (en rojo) de los 100 árboles de decisión de este bosque aleatorio. En negro puedes ver la media aritmética.

Por ejemplo, imagínate que en vez de querer predecir un resultado, queremos predecir el rango de valores probables en el futuro. Podemos obtener los valores mínimo y máximo de cada predicción para definir el intervalo más probable.



Predicción del rango más probable (mínimo y máximo en rojo) y la media aritmética de las predicciones de los 100 árboles en negro. El rango se corresponde con la predicción máxima y mínima para cada muestra.

Random Forest en scikit-learn: hiper-parámetros más útiles

[scikit-learn](#) ofrece dos implementaciones de random forests:

- Para clasificación: [RandomForestClassifier](#)
- Para regressión: [RandomForestRegressor](#)

Estos son los hiper-parámetros más útiles:

- Propios del Bosque Aleatorio:
 - **n_estimators**: número de árboles que va a tener el bosque aleatorio. Normalmente cuantos más mejor, pero a partir de cierto punto deja de mejorar y sólo hace que vaya más lento. Un buen valor por defecto puede ser el uso de 100 árboles.
 - **n_jobs**: número de cores que se pueden usar para entrenar los árboles. Cada árbol es independiente del resto, así que entrenar un bosque aleatorio es una tarea muy paralelizable. Por defecto sólo utiliza 1 core de la CPU. Para mejorar el rendimiento puedes usar tantos cores como estimes necesario. Si usas `n_jobs = -1`, estás indicando que quieres usar tantos cores como tenga tu máquina.
 - **max_features**: usa forma de garantizar que los árboles son diferentes, es que todos se entranan con una muestra aleatoria de los datos. Si queremos que todavía sean más diferentes, podemos hacer que distintos árboles usen distintos atributos. Esto puede ser útil especialmente cuando algunos atributos están relacionados entre sí. Hay varias estrategias para elegir el número máximo de atributos que se pueden usar; mira la [documentación](#) para saber más.
- Regularización (también disponibles para Decision Trees):
 - **max_depth**: la profundidad máxima del árbol. En los ejemplos anteriores hemos usado `max_depth = 2`
 - **min_samples_split**: número mínimo de muestras necesarias antes de dividir este nodo. También se puede expresar en porcentaje.
 - **min_samples_leaf**: número mínimo de muestras que debe haber en un nodo final (hoja). También se puede expresar en porcentaje.
 - **max_leaf_nodes**: número máximo de nodos finales

Resumen

Random Forest es una técnica de aprendizaje automático supervisada basada en árboles de decisión. Su principal ventaja es que obtiene un mejor rendimiento de generalización para un rendimiento durante entrenamiento similar. Esta mejora en la generalización la consigue compensando los errores de las predicciones de los distintos árboles de decisión. Para asegurarnos que los árboles sean distintos, lo que hacemos es que cada uno se entrena con una muestra aleatoria de los datos de entrenamiento. Esta estrategia se denomina [bagging](#).

Dado que un random forest es un conjunto de árboles de decisión, y los árboles son modelos no-paramétricos, los random forests tienen las mismas ventajas y desventajas de los modelos no-paramétricos:

- ventaja: pueden aprender cualquier correspondencia entre datos de entrada y resultado a predecir
- desventaja: no son buenos extrapolando ... porque no siguen un modelo conocido

Recursos

- En este [vídeo](#) explico cómo funcionan los árboles de decisión, bagging y los random forests (en inglés)
- Implementación de Random Forest en scikit-learn para clasificación y regresión: [RandomForestClassifier](#) y [RandomForestRegressor](#)
- Ventajas y desventajas de los árboles de decisión como [modelos no-paramétricos](#)

Créditos Imagen: [pixabay](#)

Avances en la generación de caras con GANs

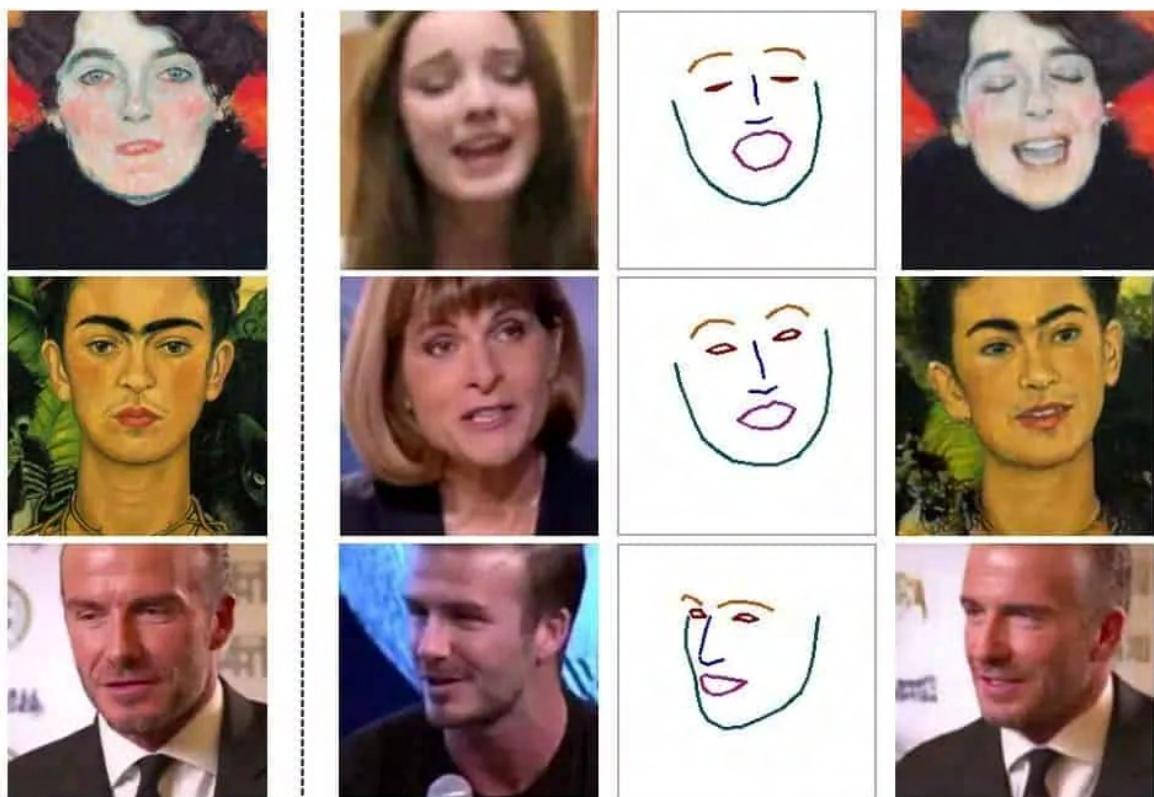
Por Jose Martinez Heras
20/06/2019

Los modelos generativos, también llamados GANs (Generative Adversarial Networks), permiten la creación de datos (e.g. imágenes) que no existen. En un artículo anterior vimos como los modelos generativos se habían utilizado para generar fotos de habitaciones de hoteles, caras humanas e incluso música.

El último avance en la generación de caras nos viene de la mano de ingenieros de Samsung (en Moscú) y del Instituto de Ciencia y Tecnología de Skolkovo.

Su trabajo es muy novedoso porque permite generar diferentes poses de la cara que queramos a partir de muy pocas imágenes, incluso a partir de sólo una imagen. Sin embargo, han necesitado una cantidad ingente de datos para entrenarlo de forma que sea tan genérico.

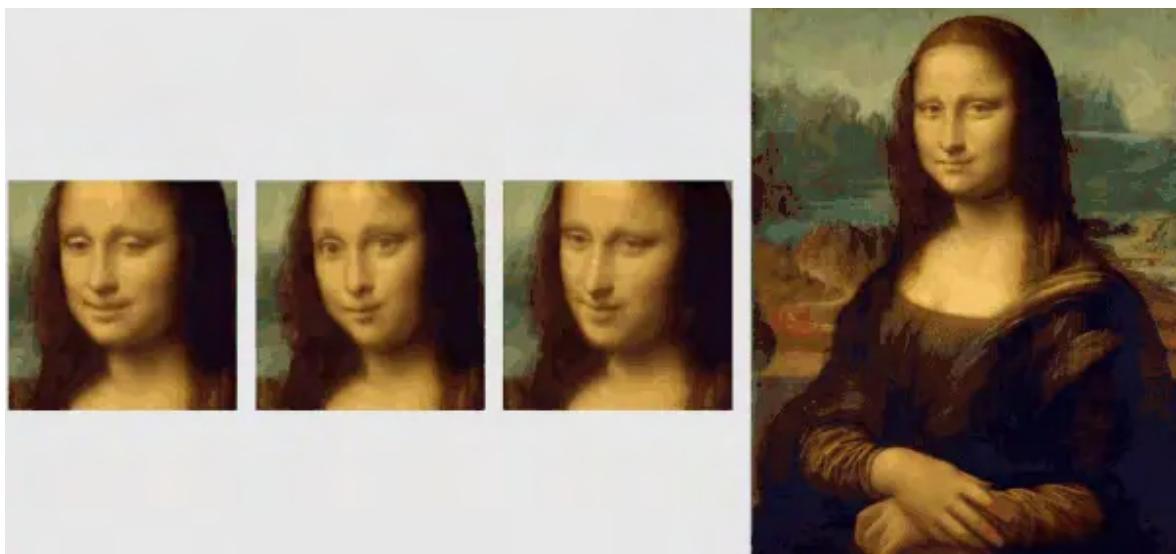
En este trabajo, primero detectan los puntos de referencia de la cara, tales como la nariz, ojos, boca, cejas y perímetro de la cara. Estos puntos de referencia son los que van a permitir cambiar la pose de la cara que queramos. El modelo generativo está condicionado a estos punto de referencia.



Aunque el modelo generativo se haya entrenado con vídeos y con fotos, también funciona con imágenes. En particular, a mi me ha gustado cómo le han dado vida a la Mona Lisa de Leonardo da Vinci.

Los autores hablan de las aplicaciones posibles de esta tecnología:

- Telepresencia: teleconferencias y juegos multi-jugador
- Efectos especiales



La tecnología funciona bastante bien, aunque todavía necesita mejorarse. Todavía es fácil darse cuenta que las imágenes no son del todo reales. Además, de vez en cuando, también aparecen aberraciones en las imágenes que lo delatan. De cualquier forma, el progreso de este trabajo es fascinante y sólo nos permite imaginar lo que vendrá en el futuro.

Para saber más puedes leer el artículo de investigación completo [aquí](#).

Basura Espacial: competición con machine learning

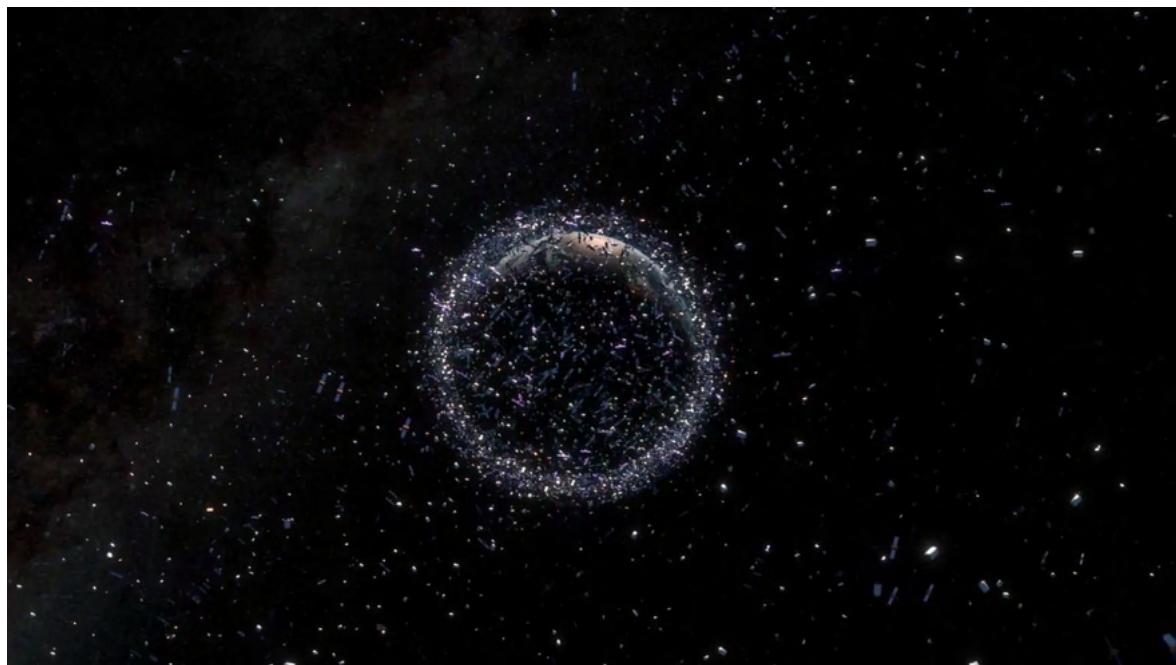
Por Jose Martinez Heras
11/10/2019

La Agencia Espacial Europea ha creado una competición de machine learning para predecir el riesgo de colisión entre basura espacial y satélites de observación de la Tierra. La competición empieza oficialmente el 16 de Octubre. A partir de ese momento será posible descargar los datos y enviar soluciones. La competición finaliza el 16 de Diciembre de 2019.

Basura Espacial

El espacio no está tan vacío como solía estar. Más de 34.000 elementos de desechos espaciales de más de 10 cm orbitan nuestro planeta. La Red de Vigilancia Espacial de Estados Unidos mantiene un catálogo de unos 22.300 de estos objetos y sus trayectorias y los monitorea con ayuda de telescopios y radares.

En órbitas con mucho tráfico, la prevención activa de colisiones se ha convertido en una tarea rutinaria en las operaciones espaciales. Los datos de vigilancia espacial revelan riesgos potenciales de que un satélite colisione con otro objeto espacial a varios kilómetros por segundo, ya sea con otro satélite o con basura espacial.



Basura Espacial alrededor de la Tierra (no está a escala).
Créditos: [ESA](#)

El número de objetos según su tamaño es el siguiente:

- > 1 metro: 5.400 objetos
- > 10 centímetros : 34.000 objetos
- > 1 centímetro: 900.000 objetos
- > 1 milímetro: 130.000.000 objetos

Descripción del problema

Cuando se propagan las órbitas de los satélites, también se comprueba si hay riesgo de colisión con alguno de los objetos del catálogo de basura espacial. En el caso de que haya algún riesgo,

se prepara un mensaje de conjunción (CDM por sus siglas en inglés: *Conjunction Data Message*). Cada CDM contiene varios atributos acerca de la posible colisión tales como la identidad del satélite afectado, el tipo de basura espacial, cuándo se prevé que sea la colisión, la incertidumbre, etc. Además también contiene el riesgo estimado en ese momento, que se calcula con algunos de los atributos del CDM. En los días posteriores al primer CDM, a medida que las incertidumbres de las posiciones de los objetos van disminuyendo, se crean otros CDM que van refinando el conocimiento que vamos adquiriendo sobre la posible colisión.

Típicamente, la serie temporal de CDMs de cada posible colisión cubre una semana. En media tiene unos 3 CDMs por día. Para cada posible colisión, el último CDM, incluyendo el riesgo calculado, es la información más exacta que se tiene sobre la posible colisión entre los dos objetos en cuestión. In la mayoría de los casos, la Oficina de Basura Espacial alertará a los equipos de control 2 días antes de la colisión potencial para empezar a pensar en hacer maniobras que eviten la colisión. La decisión final sobre la maniobra se hará típicamente 1 día antes de la posible colisión.

En esta competición de machine learning, se pide construir un modelo que use la historia de los CDM hasta 2 días antes de la posible colisión y se prediga cuál será el riesgo final (el riesgo final es el riesgo que aparece en el último CDM antes del tiempo de colisión).

Evaluación de la solución

La calidad de la solución de mide considerando el error cuadrático medio (MSE) y [F2](#). En particular, con el ratio MSE / F2.

- El error cuadrático medio (MSE) va a indicar cómo de cerca la predicción del riesgo en comparación con el riesgo actual.
- La medida F2 es una variante de [F1](#) que penaliza más los falsos negativos.

En principio, el error cuadrático medio sería suficiente. Lo que ocurre, es que es el mismo error tiene significados diferentes dependiendo en qué parte de la escala de riesgo se encuentre. Por ejemplo, si en realidad el riesgo es 10^{-20} y el modelo predice que el riesgo será 10^{-23} , no tiene ningún impacto en la práctica. Sin embargo, si el riesgo en realidad fuera 10^{-3} y el modelo predice 10^{-6} sería más grave.

Por eso se usa F2 usando 10^{-6} como límite. De esta forma, resulta muy importante clasificar si tendremos un error alto (mayor que 10^{-6}) o bajo (menor que 10^{-6}).

Además, el error cuadrático medio sólo se calcula para los casos en el que el riesgo real sea alto.

Participación

Para participar, sólo tienes que inscribirte (gratuitamente) en la [plataforma kelvins](#). Es muy parecida a [kaggle](#) en cuanto a su funcionamiento.



A tener en cuenta

Una de las dificultades de este problema es que hay muy pocos eventos que tengan un riesgo muy alto de colisión. La mayoría de las posibles colisiones llevan poco riesgo. Esto está muy bien para los satélites, pero supone que el conjunto de datos está muy desbalanceado.

Suerte!

Esperamos que os divirtáis tanto resolviendo este desafío como nosotros preparándolo. Como os podéis imaginar, hemos empleado un tiempo en preparar los datos, decidiendo qué predecir, decidiendo cómo evaluar los resultados ... y también intentando resolver este problema como si fuésemos uno de los participantes en este desafío de machine learning para asegurarnos de que todo iba a ir estupendamente.

Mucha suerte y espero veros pronto en el «Leaderboard».

Actualización

La competición terminó. Puedes leer nuestro análisis de la competición y los resultados en el paper [Spacecraft Collision Avoidance Challenge: design and results of a machine learning competition](#)

Recursos

- [Web de la competición](#)
- [Nota de Prensa](#)

Precision, Recall, F1, Accuracy en clasificación

Por Jose Martinez Heras
17/11/2019

Cuando necesitamos evaluar el rendimiento en clasificación, podemos usar las métricas de precision, recall, F1, accuracy y la matriz de confusión. Vamos a explicar cada uno de ellos y ver su utilidad práctica con un ejemplo.

Términos es Español

Estas métricas también tienen su correspondiente nombre en español, pero es importante que sepas su nombre en inglés porque muchas librerías ([scikit-learn](#)), las tienen ya implementadas. En esta tabla puedes encontrar la correspondencia.

Inglés	Español
Precision	Precisión
Recall	Exhaustividad
F1-score	Valor-F
Accuracy	Exactitud
Confusion Matrix	Matriz de Confusión
True Positive	Positivos Verdaderos
True Negative	Negativos Verdaderos
False Positive	Positivos Falsos
False Negative	Negativos Falsos

Ejemplo de Marketing

Vamos a utilizar un ejemplo de marketing para entender mejor qué es lo que miden cada una de estas métricas y su significado en el ámbito de los negocios.

Imagínate que estamos llevando la campaña de marketing para un banco. Al banco le interesa vender un fondo de inversión a sus clientes, porque así pueden ganar dinero por la comisión de gestión.

Podríamos contactar con todos los clientes del banco y ofrecerles el fondo de inversión. Esto es bastante ineficiente porque la mayoría de los clientes no estarán interesados. Sería más eficiente contactar con unos pocos, recoger datos, hacer machine learning y predecir qué otros clientes tienen más probabilidad de aceptar la oferta del banco.

Sin embargo, (casi) ningún modelo de Machine Learning es perfecto. Esto quiere decir que:

- habrá clientes con los que contactaremos porque el modelo ha predicho que aceptarían y en realidad no lo hacen (*False Positive [FP], Positivos Falsos*).
- habrá también clientes con los que no contactaremos porque el modelo ha predicho que no aceptarían que en realidad si lo hubieran hecho (*False Negative [FN], Negativos Falsos*).

El modelo de Machine Learning también acertará (esperemos que mucho). A efectos prácticos esto significa que:

- habrá clientes con los que contactaremos porque el modelo ha predicho que aceptarían y en realidad sí que lo hacen (*True Positive [TP]*, *Positivos Verdaderos*).
- habrá clientes que no contactaremos porque el modelo ha predicho que no aceptarían la oferta y en realidad no lo hacen (*True Negative [TN]*, *Negativos Verdaderos*).

Confusion Matrix (Matriz de Confusión)

Vamos a explicar como funciona la matriz de confusión con un ejemplo hipotético de marketing. En este ejemplo, contactamos a 100 clientes y 80 de ellos nos dicen que no están interesados y 20 de ellos que sí.

Nuestro modelo (en el ejemplo) no es muy bueno, aunque dependiendo de qué métrica usemos podría parecer que es mejor de lo que es.

Hemos utilizado como valores de la clasificación binaria:

- 0: no está interesado
- 1: sí está interesado

		predicción				predicción	
		0	1			0	1
realidad	0	70	10	realidad	0	TN	FP
	1	15	5		1	FN	TP

Matriz de Confusión con un ejemplo de marketing

En la matriz de confusión de la izquierda podéis ver los valores para este ejemplo. En la matriz de confusión de la derecha, los nombres genéricos cuando usamos la nomenclatura inglesa: *True Negative [TN]*, *True Positive [TP]*, *False Positive [FP]*, *False Negative [FN]*.

Truco: para recordar fácilmente la matriz de confusión:

- *Positivo (Positive)* o *Negativo (Negative)*: se refiere a la predicción. Si el modelo predice 1 entonces será *positivo*, y se predice 0 será *negativo*.
- *Verdadero (True)* o *Falso (False)*: se refiere si la predicción es correcta o no.

Precision (Precisión)

Con la métrica de precisión podemos medir la **calidad** del modelo de machine learning en tareas de clasificación. En el ejemplo, se refiere a que la precisión es la respuesta a la pregunta ¿qué porcentaje de los clientes que contactemos estarán interesados?

Para calcular la precisión usaremos la siguiente fórmula:

$$precision = \frac{TP}{TP + FP}$$

		predicción	
		0	1
realidad	0	TN	FP
	1	FN	TP

Precisión (precision)

En el ejemplo de marketing, siguiendo los datos de la matriz de confusión, tenemos que:

$$precision = \frac{TP}{TP + FP} = \frac{5}{5 + 10} = 0.33$$

Es decir, que sólo un 33% de los clientes a los que contactemos estarán realmente interesados. Esto significa que el modelo del ejemplo se equivocará un 66% de las veces cuando prediga que un cliente va a estar interesado.

Recall (Exhaustividad)

La métrica de exhaustividad nos va a informar sobre la **cantidad** que el modelo de machine learning es capaz de identificar. En el ejemplo, se refiere a que la exhaustividad (recall) es la respuesta a la pregunta ¿qué porcentaje de los clientes están interesados somos capaces de identificar?

Para calcular la exhaustividad (recall) usaremos la siguiente fórmula:

$$recall = \frac{TP}{TP + FN}$$

		predicción	
		0	1
realidad	0	TN	FP
	1	FN	TP

Exhaustividad (recall)

En el ejemplo de marketing, siguiendo los datos de la matriz de confusión, tenemos que:

$$recall = \frac{TP}{TP + FN} = \frac{5}{5 + 15} = 0.25$$

Es decir, el modelo sólo es capaz de identificar un 25% de los clientes que estarían interesados en adquirir el producto. Esto significa que el modelo del ejemplo sólo es capaz de identificar 1 de cada 4 de los clientes que sí aceptarían la oferta.

F1

El valor F1 se utiliza para combinar las medidas de precision y recall en un sólo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

F1 se calcula haciendo la media armónica entre la precisión y la exhaustividad:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

En el ejemplo de marketing, combinando precision y recall en F1 nos quedaría:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 2 \cdot \frac{0.33 \cdot 0.25}{0.33 + 0.25} = 0.28$$

El valor F1 asume que nos importa de igual forma la precisión y la exhaustividad. Esto no tiene que ser así en todos los problemas. Por ejemplo, cuando necesitamos predecir si hay riesgo de que un trozo de basura espacial se choque con un satélite, podemos valorar más la exhaustividad a riesgo de tener una peor precisión. Por eso elegimos F2 en lugar de F1 para esa competición de machine learning.

En este caso podemos usar F2, que es la fórmula genérica del valor F para beta = 2. La fórmula genérica de F es la siguiente:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

Accuracy (Exactitud)

La exactitud (accuracy) mide el porcentaje de casos que el modelo ha acertado. Esta es una de las métricas más usadas y favoritas ... que te recomiendo evitar! El problema con la exactitud es que nos puede llevar al engaño, es decir, puede hacer que un modelo malo (como el del ejemplo) parezca que es mucho mejor de lo que es.

El accuracy (exactitud) se calcula con la siguiente fórmula:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

La exactitud del ejemplo de marketing sería:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{5 + 70}{5 + 70 + 10 + 15} = 0.75$$

Es decir, el modelo acierta el 75% de las veces. Como ves, la exactitud es una métrica muy engañosa. De hecho, si tuviésemos un modelo que siempre predijera que el cliente nunca va a estar interesado, su accuracy sería del 80%.

Peligro: la métrica accuracy (exactitud) no funciona bien cuando las clases están desbalanceadas como es en este caso. La mayoría de los clientes no están interesados en la oferta, así que es muy fácil acertar diciendo que no lo van a estar. Para problemas

con clases desbalanceadas es mucho mejor usar precision, recall y F1. Estas métricas dan una mejor idea de la calidad del modelo.

Calculando precision, recall, F1, accuracy en python con scikit-learn

La librería de python [scikit-learn](#) implementa todas estas métricas. Para usarlas sólo tienes que seguir sus instrucciones:

- [sklearn.metrics.confusion_matrix](#)
- [sklearn.metrics.precision_score](#)
- [sklearn.metrics.recall_score](#)
- [sklearn.metrics.f1_score](#)
- [sklearn.metrics.accuracy_score](#)

Precision, Recall, F1, Accuracy y la Matriz de Confusión son métricas de **clasificación**. El artículo **Error Cuadrático Medio para Regresión** explica las métricas de **regresión**.

Resumen

En este artículo hemos visto cuáles son las métricas más extendidas para evaluar el rendimiento de modelo supervisado en tareas de clasificación.

Hemos destacado que la métrica **accuracy** (exactitud) es engañosa cuando las clases están desbalanceadas. Cuando decimos que es engañosa nos referimos a que nos hace creer que el modelo es mejor de lo que en realidad es. De hecho, he puesto un ejemplo de un modelo relativamente malo para poner de manifiesto este ejemplo.

Las medidas de precision, recall y F1 son mucho más representativas y funcionan tanto si las clases están balanceadas como si no:

- **Precision** nos da la calidad de la predicción: ¿qué porcentaje de los que hemos dicho que son la clase positiva, en realidad lo son?
- **Recall** nos da la cantidad: ¿qué porcentaje de la clase positiva hemos sido capaces de identificar?
- **F1** combina Precision y Recall en una sola medida
- La **Matriz de Confusión** indica qué tipos de errores se cometan

Recursos

- [Métricas](#) en scikit-learn que incluyen las descritas y muchas más
- [Vídeo](#) (en inglés) donde explico estas métricas
- [Artículo](#) en la wikipedia
- Error Cuadrático Medio como métrica en problemas de aprendizaje supervisado de regresión

Clustering (Agrupamiento), K-Means con ejemplos en python

Por Jose Martinez Heras
25/01/2020

El **clustering** consiste en la agrupación automática de datos. Es un tipo de aprendizaje automático no-supervisado. En castellano se denomina *agrupamiento*. Vamos a ver en más detalle en qué consiste el clustering, el algoritmo de agrupamiento más popular: K-Means y algunos ejemplos en python.

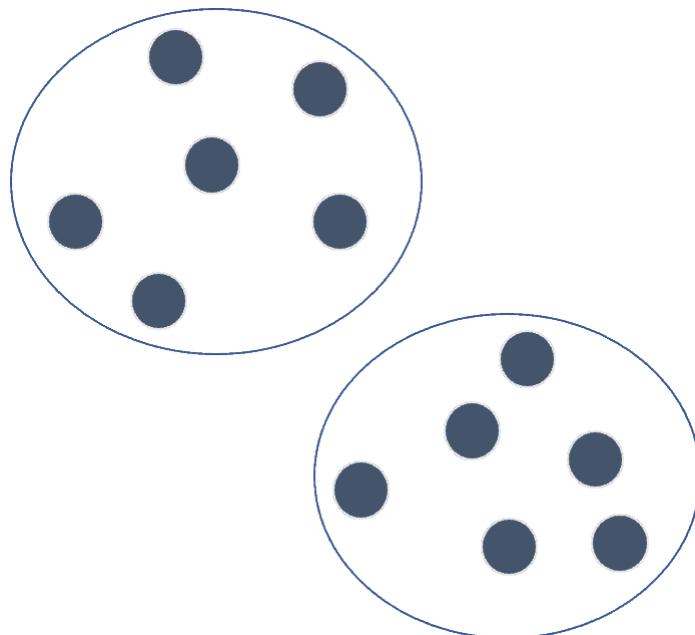
Clustering – Agrupamiento

El clustering consiste en la agrupación automática de datos. Al ser un [aprendizaje no-supervisado](#), no hay una respuesta correcta. Esto hace que la evaluación de los grupos identificados sea un poco subjetiva. Vamos a verlo con un ejemplo:

En el siguiente gráfico podemos ver un conjunto de datos, representados por puntos en 2 dimensiones.

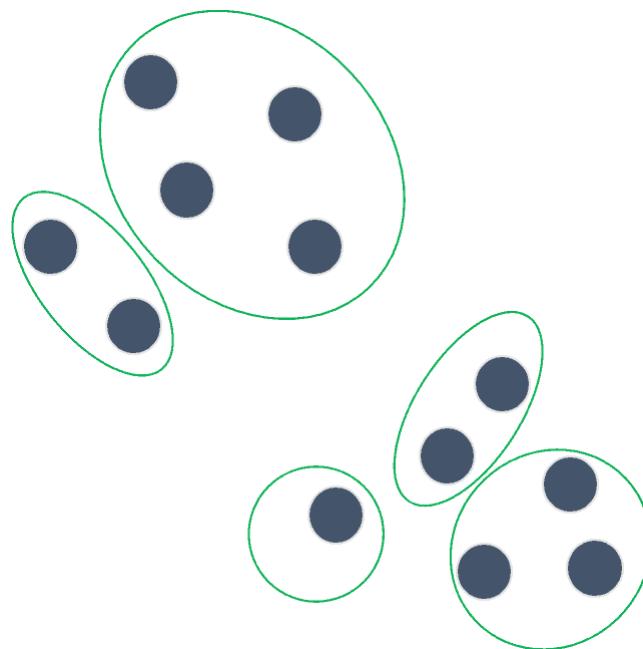


Con estos datos, algunas personas dirían que hay 2 grupos, tal y como se muestra en la siguiente figura.



Datos agrupados en 2 clusters

Sin embargo, otras personas podrían decir que, en realidad, hay 5 grupos.



Datos agrupados en 5 clusters

Esto no significa haya alguien equivocado. Tampoco que todos tengan razón. Tan sólo que los datos son tal y como son pero podemos elegir verlos como más nos convengan.

Las técnicas de clustering intentan descubrir cuál es el mejor agrupamiento de los datos. Algunas de estas técnicas necesitan que especifiquemos el número de grupos de queremos encontrar. Otras técnicas necesitan otros hiper-parámetros.

Ejemplos de Clustering

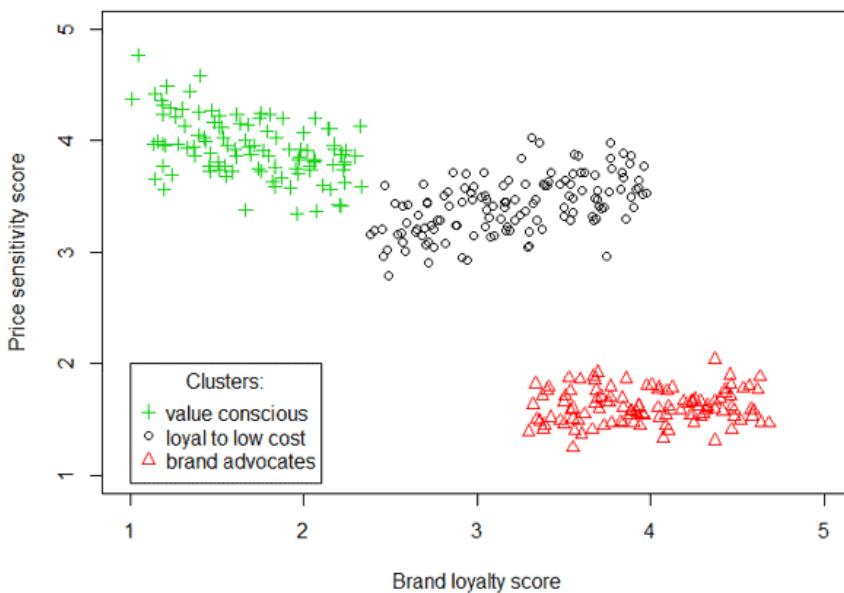
Vamos a ver 2 ejemplos en esta sección: segmentación de clientes y compresión de datos. En el ejemplo de K-Means con python agruparemos las acciones del Dow Jones de la bolsa americana que tengan un comportamiento similar.

Segmentación de clientes

En la segmentación de clientes intentamos comprender qué hace a los clientes diferentes para poder ofrecerles los productos y servicios que necesiten.

En este ejemplo podemos ver cómo una técnica de agrupamiento ha encontrado 3 grupos cuando considera 2 variables:

- Lealtad a la marca (eje X)
- Sensibilidad al precio (eje Y)



Fuente: <https://select-statistics.co.uk/blog/customer-segmentation/>

Los 3 grupos (clusters) que se han encontrado son:

- Preocupados por el precio (en verde): no son leales a la marca y son muy sensativos al precio
- Leales a precios bajos (en negro): son leales a la marca pero sólo si es barato
- Defensores de la marca (en rojo): son leales a la marca sin importar demasiado el precio

Por supuesto, el análisis de clustering no nos da ninguna de estas explicaciones. Sólo nos da los grupos. Una persona con conocimiento del dominio del problema será la encargada de interpretar los resultados. En el problema de segmentación de clientes, esta persona posiblemente trabajará en marketing o ventas.

Compresión de datos

También podemos usar clustering para comprimir imágenes con pérdida de información. La compresión, en este ejemplo, se hace en el número de colores diferentes que se usan.

Vamos a suponer que la imagen original utiliza una paleta de 255 colores. Para comprimir la imagen, podemos decidir usar menos bits por pixel, es decir, usar menos colores.

En la siguiente figura puedes ver cómo quedaría la foto original a medida que vamos usando menos y menos colores. Primero 10 colores, luego 3 y por último sólo 2.

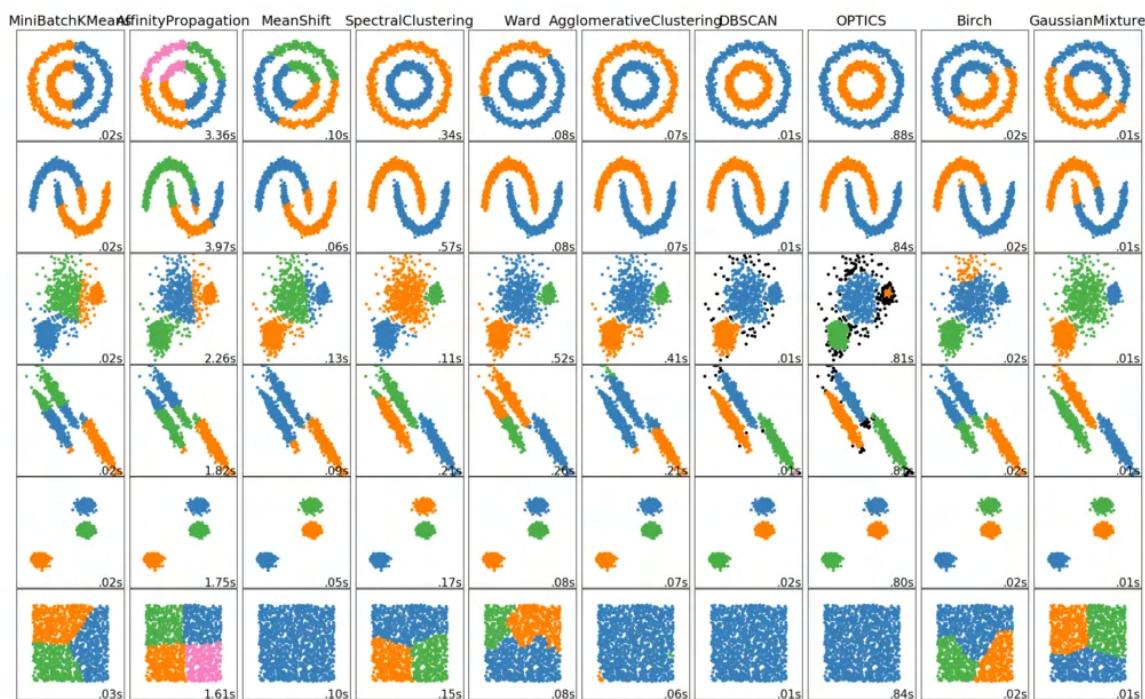
La técnica de clustering nos ayuda a decidir qué nuevos colores pueden representar mejor la imagen original cuando limitamos el número posible de colores a usar (K).



Fuente: <https://rpubs.com/yujingma45/155921>

Algoritmos de Clustering

La librería de python [scikit-learn](#) ofrece implementaciones eficientes de varias técnicas de agrupamiento. Esta figura muestra cómo distintos algoritmos se comportan con varios tipos de datos.



Fuente: <https://scikit-learn.org/stable/modules/clustering.html>

Scikit-learn ofrece la siguiente tabla que nos ayuda a comparar los diferentes algoritmos de clustering. La comparación de los algoritmos de clustering está hecha en función a los parámetros que necesitan, su escalabilidad, caso de uso y geometría (métrica usada). Algunas preguntas útiles pueden ser:

- ¿Tengo una idea del número de grupos que quiero encontrar? ... ¿o prefiero que el algoritmo lo encuentre?
- ¿Tengo muchísimos datos? En este caso, deberemos tener en cuenta la escalabilidad del algoritmo.

Method name	Parameters	Scalability	Use case	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large n_samples, large n_clusters	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Fuente: <https://scikit-learn.org/stable/modules/clustering.html>

K-Means

El algoritmo de clustering más usado es K-Means. Tiene una muy buena escalabilidad con la cantidad de datos. Para utilizar K-Means debemos especificar el número de grupos que queremos encontrar. A este número de grupos se le denomina K.

El algoritmo K-Means sigue los siguientes pasos:

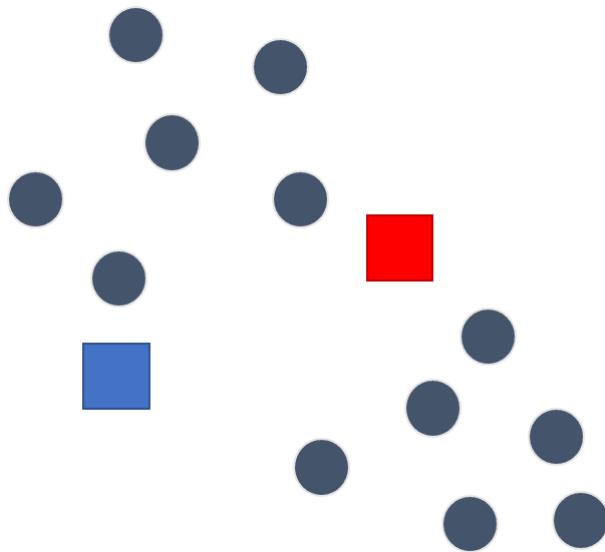
1. **Inicialización:** se elige la localización de los centroides de los K grupos aleatoriamente
2. **Asignación:** se asigna cada dato al centroide más cercano
3. **Actualización:** se actualiza la posición del centroide a la media aritmética de las posiciones de los datos asignados al grupo

Los pasos 2 y 3 se siguen iterativamente hasta que no haya más cambios.

Vamos a verlo con un ejemplo.

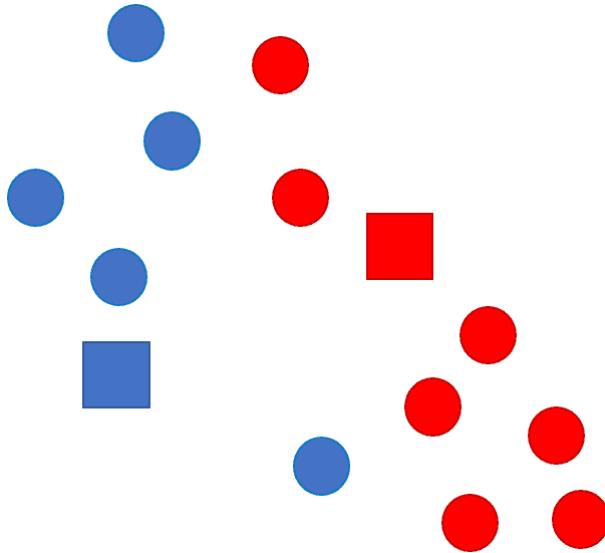
Inicialización

Se elige la localización de los centroides de los K grupos aleatoriamente. La figura muestra los datos como círculos y los centroides como cuadrados.



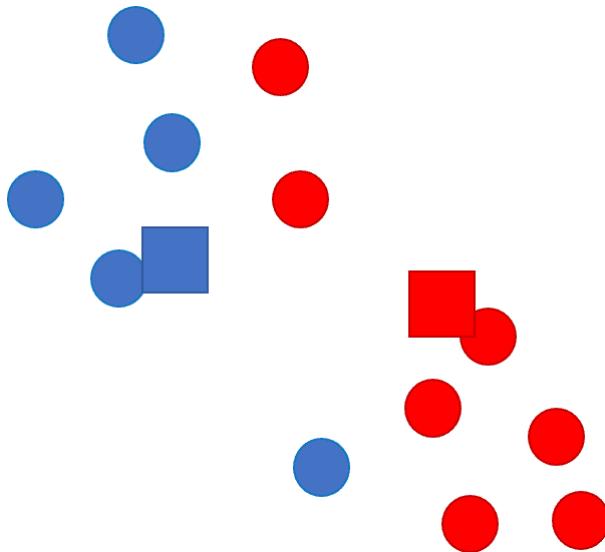
Asignación

A continuación, se asigna cada dato al centroide más cercano. En el ejemplo, los círculos cambian de color para indicar a qué centroide han sido asignados.



Actualización

Ahora se actualiza la posición del centroide a la media aritmética de las posiciones de los datos asignados al grupo. Observa cómo la posición de los centroides (cuadrados) cambia.



A continuación irían las fases: asignación, actualización, asignación, actualización, etc. hasta que las posiciones de los centroides no cambien.

Recomendaciones

Normalización

A la hora de usar cualquier algoritmo de clustering, es buena idea normalizar nuestros datos. En este contexto, «normalizar» se refiere a que los valores de cada atributo estén en escalas similares. Normalizar ayuda al clustering porque los grupos se forman a partir de distancias. Si hay atributos con escalas muy diferentes, los atributos de escala mayor dominarán las distancias.

Para poner un ejemplo del problema de no normalizar, imagina el caso en el que queramos encontrar grupos de inmuebles similares. Si los atributos que tenemos son el precio en euros y el número de habitaciones, seguramente obtendremos el mismo resultado de agrupamiento si eliminamos el número de habitaciones. Sin embargo, al poner ambos atributos en la misma escala, los dos se convierten en importantes.

Las técnicas más comunes de normalización son:

- re-escalar cada atributo en el rango [0, 1]
- suponer que cada atributo sigue una distribución normal y hacer que los datos tengan una media de 0 y una desviación típica de 1.

Hay, por supuesto, más tipos de normalización. Por ejemplo: transformación logarítmica, hacer ratios, cuantizar los datos, etc.

Selección de características

El clustering es una técnica de aprendizaje automático no-supervisada. Esto implica que no es capaz de establecer la relación entre los atributos de entrada y los resultados ... sencillamente porque no hay resultados. Así que la responsabilidad de identificar qué atributos son relevantes recae sobre nosotros.

Por ejemplo, si estamos usando clustering para segmentar clientes ... seguramente no es buena idea incluir el color de los ojos de los clientes. Esta es una característica posiblemente irrelevante que hará que las distancias entre los puntos multidimensionales que representan cada dato sea menos informativa ... excepto si estamos vendiendo lentes para cambiar el color de ojos!

Además, siempre es buena idea usar el menor número atributos posible debido a los que se conoce como *la maldición de la dimensionalidad*. En esencia se refiere a que a medida que el número de dimensiones (atributos) aumenta, la distancia discrimina cada vez menos. Una práctica

común antes de hacer clustering es reducir la dimensionalidad del problema. Hay varias técnicas para este propósito tales como *Principal Components Analysis (PCA)*, *auto-encoders*, etc.

Aprender más sobre la Maldición de la Dimensión

Número de clusters

Algunas técnicas de agrupamiento, tales como K-Means, necesitan que especifiquemos el número de clusters (grupos) que queremos encontrar. No es obvio, a priori, saber qué número de grupos es mejor.

Por ejemplo, si queremos encontrar grupos de tallas de ropa y elegimos tener 5 grupos, podríamos llamarlos: XS, S, M, L, XL. Sin embargo, si buscamos 7 grupos, tendríamos: XXS, XS, S, M, L, XL, XXL.

Si lo que estás buscando es un método más objetivo, te recomiendo que profundices en este tema con alguna de estas dos técnicas:

- [Método de la silueta](#)
- [Método del codo](#)

Ejemplo de Clustering K-Means en Python

En este ejemplo vamos a agrupar las acciones del mercado bursátil estadounidense. El objetivo es encontrar automáticamente qué acciones tienen comportamientos similares.

Los datos están disponibles en [kaggle](#). Para simplificar el problema, vamos a usar sólo las 30 acciones de están recogidas en el índice Dow Jones.

Las técnicas de clustering están disponibles en scikit-learn en el paquete [sklearn.cluster](#)

```
1. # KMeans está en el paquete sklearn.cluster
2. from sklearn.cluster import KMeans
```

Vamos a usar datos semanales de los últimos 5 años. Este extracto muestra qué forma tienen los datos. Cada columna representa a cada empresa.

symbol	A	AAL	AAP	AAPL	ABBV	ABC	ABT	ACN	ADBE	ADI	...	XL	XLNX	XOM	XRAY	XRX	XYL	YUM	ZBH	ZION
date																				
2013-02-10	45.08	14.75	78.90	67.8542	36.25	46.89	34.41	73.31	39.120	45.700	...	28.24	37.51	88.61	42.87	31.84	27.09	65.30	75.85	24.14
2013-02-17	42.25	14.50	79.00	65.7371	37.58	46.60	35.08	74.16	38.635	46.175	...	28.67	38.12	88.36	42.80	31.88	28.28	63.99	75.90	24.34
2013-02-24	41.80	13.57	79.21	64.4014	38.46	46.95	34.55	74.80	38.550	45.520	...	28.99	37.96	89.20	41.78	32.48	27.79	65.45	74.14	24.04
2013-03-03	41.93	13.61	76.37	61.4957	37.81	47.98	33.60	74.82	39.830	45.230	...	28.82	36.65	89.43	41.16	32.60	27.49	65.21	74.99	24.04
2013-03-10	43.03	14.92	76.84	61.6742	37.34	48.48	34.68	78.35	41.500	46.050	...	29.66	38.37	88.97	42.74	34.84	28.43	67.72	75.40	25.30

Cada acción tiene su propio rango de valores. Por ejemplo Apple (APPL) empieza en \$67.85 y Adobe (ADBE) en \$39.12.

La intención de este ejercicio es agrupar las acciones por el comportamiento de sus ganancias, en vez de por su precio de mercado. De esta forma el rendimiento de cada acción es comparable con los demás.

Vamos a calcular el rendimiento de cada acción desde el principio del periodo que estamos considerando.

```

1. start = stocks.iloc[0]                      # start es el valor al principio del
   periodo
2. returns = (stocks - start) / start      # returns es el valor relativo al
   inicio
3.
4. # dow_returns contendrá la información de returns limitada a las acciones
   del Dow Jones

```

Ahora vamos a hacer clustering con estos datos con K-Means. Instruimos a K-Means a que use 8 grupos, es decir K=8.

```

1. kmeans = KMeans(n_clusters=8, random_state=42) # K-Means con K=8
2. kmeans.fit(dow_returns.T);                  # haz 8 grupos con nuestros
   datos

```

Los resultados que obtenemos son los siguientes:

```

1. Cluster 0:
2.     - American Express Co. (AXP)
3.     - General Electric (GE)
4.     - Goldman Sachs Group (GS)
5.     - Coca-Cola Company (The) (KO)
6.     - McDonald's Corp. (MCD)
7.     - Pfizer Inc. (PFE)
8.     - Procter & Gamble (PG)
9.     - United Technologies (UTX)
10.    - Verizon Communications (VZ)
11.    - Wal-Mart Stores (WMT)
12.
13. Cluster 1:
14.    - Apple Inc. (AAPL)
15.    - 3M Company (MMM)
16.
17. Cluster 2:
18.    - United Health Group Inc. (UNH)
19.
20. Cluster 3:
21.    - Caterpillar Inc. (CAT)
22.    - Chevron Corp. (CVX)
23.    - International Business Machines (IBM)
24.    - Exxon Mobil Corp. (XOM)
25.
26. Cluster 4:
27.    - Home Depot (HD)
28.    - Microsoft Corp. (MSFT)
29.    - Visa Inc. (V)
30.
31. Cluster 5:
32.    - Cisco Systems (CSCO)
33.    - Intel Corp. (INTC)
34.    - Johnson & Johnson (JNJ)
35.    - JPMorgan Chase & Co. (JPM)
36.    - Merck & Co. (MRK)
37.    - The Travelers Companies Inc. (TRV)

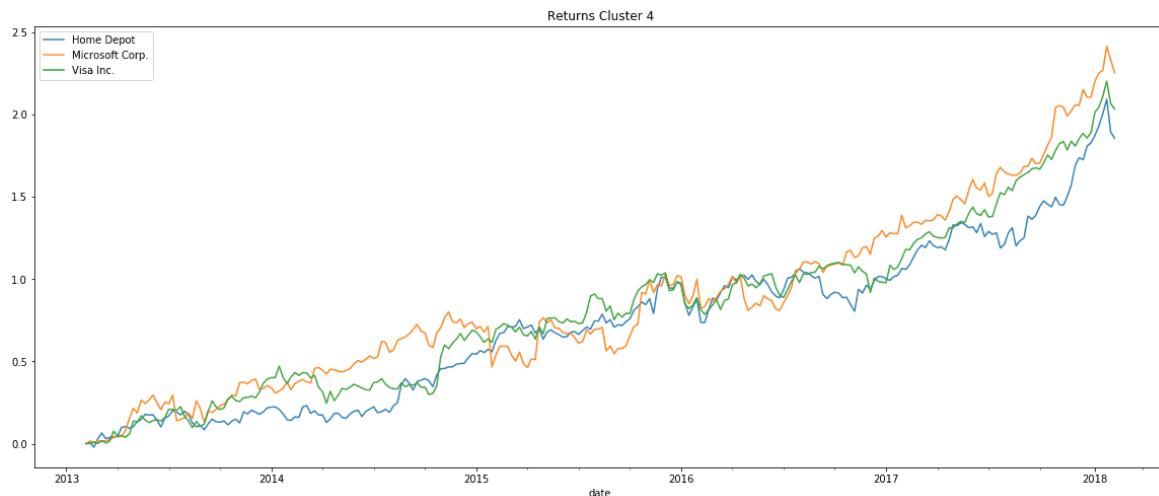
```

```

38.
39.     Cluster 6:
40.         - The Walt Disney Company (DIS)
41.         - Nike (NKE)
42.
43.     Cluster 7:
44.         - Boeing Company (BA)

```

Por ejemplo, la siguiente figura muestra los componentes del Cluster 4 que contiene: Home Depot, Microsoft y Visa.



Para ver el resto de los gráficos y el código completo, puedes consultar este [jupyter notebook](#).

Resumen

Agrupar datos es muy útil para aplicaciones tales como segmentación de clientes, compresión con pérdida de datos. También para encontrar entidades con comportamientos similares (en mercados bursátiles, por ejemplo).

K-Means es la técnica de clustering más usada pero hay muchas más. Te animo a usar la tabla comparativa de los métodos de clustering que ofrece scikit-learn para decidir cuál es la más apropiada para tu problema.

Para que el clustering sea efectivo es recomendable reducir al mínimo el número de atributos, usar atributos relevantes y poner los datos en una escala similar.

Recursos

- [Video donde explico clustering y K-Means](#) (en inglés)
- [Video donde explico el ejemplo de las acciones del Dow Jones](#) (en inglés)
- [Jupyter Notebook](#) con el ejemplo de las acciones del Dow Jones al completo (en inglés)
- [Clustering](#) en scikit-learn

Google Dataset Search – descubre conjuntos de datos

Por Jose Martinez Heras
01/02/2020

En toda la web, hay millones de conjuntos de datos sobre casi cualquier tema que le interese. [Google DataSet Search](#) nos pone fácil encontrar estos datos. Tiene ya acceso a casi 25 millones de conjuntos de datos y este número sigue creciendo. Los datos no los tiene Google, sino que DataSet Search ofrece un lugar para buscar y encontrar enlaces a los datos que buscamos.

Las técnicas de Machine Learning necesitan datos para aprender. De hecho, cuantos más datos mejor. Cuando estemos resolviendo algún problema, podemos recurrir a DataSet Search para encontrar datos adicionales que puedan ser relevantes y de ayuda para resolverlo.

Por ejemplo, si estamos haciendo predicciones relacionadas con el esquí, podemos encontrar datos sobre los beneficios de las estaciones de esquí o el porcentaje de lesiones.

The screenshot shows four search results for datasets related to skiing:

- Skiing velocities and kinematic parameters for 36 world (fast)...** (F) - figshare.com. Updated Nov 9, 2017. Description: Total Revenue for Skiing Facilities, All Establishments, Employer Firms (REVEF71392ALLEST). [Explore at FRED](#). License: https://research.stlouisfed.org/fred_terms.html#copyright-public-domain.
- Total Revenue for Skiing Facilities, All Establishments, Employer...** (F) - fred.stlouisfed.org. Updated Dec 26, 2018. Description: Graph and download economic data for Total Revenue for Skiing Facilities, All Establishments, Employer Firms (REVEF71392ALLEST) from 1998 to 2017 about sport, recreation, employer firms, accounting, revenue, establishments, services, and USA.
- OC Cross County Skiing** (A) - hub.arcgis.com, accesoakland.oakgov.com, +1more. Updated Dec 21, 2015. Description: Bianchi 2015 National Strategy for Preventing Injuries From Skiing... www.researchgate.net. Updated Oct 8, 2015.
- Bianchi 2015 National Strategy for Preventing Injuries From Skiing...** (R) - www.researchgate.net. Updated Oct 8, 2015.

Algunos de los resultados de búsqueda para la consulta «skiing», que incluyen conjuntos de datos que van desde las velocidades de los esquiadores más rápidos hasta los ingresos de las estaciones de esquí.

Novedades en Google Dataset Search

Durante su fase beta, muchos usuarios lo probaron y proporcionaron comentarios que permitieron a Google mejorar su DataSet Search. Desde el 23 de Enero de 2020 está oficialmente fuera de la fase beta.

Algunas de la novedades en la versión oficial incluyen:

- **Filtrado:** ahora puede filtrar los resultados según los tipos de conjunto de datos que desee (por ejemplo, tablas, imágenes, texto). Otra posibilidad es filtrar sólo los conjuntos de datos que estén disponibles de forma gratuita.
- **Visualización geográfica:** si un conjunto de datos se trata de un área geográfica, puede ver el mapa.

- **Dispositivos móviles:** ahora Google DataSet Search funciona también en dispositivos móviles
- **Calidad de metadatos:** la nueva versión incorpora una mejora significativa en la calidad de las descripciones de los conjuntos de datos.

Además, Google ofrece la posibilidad de que los datos sean fácilmente indexados por Google DataSet Search a todas las entidades que publiquen datos. Para ello, sólo hay que usar un [estándar abierto](#) para describir las propiedades de del conjunto de datos en página web de dicha entidad.

¿Quién usa DataSet Search?

Aunque lo puede utilizar todo el mundo, el perfil de los usuarios más comunes son:

- **investigadores académicos:** así encuentran datos para desarrollar sus hipótesis
- **estudiantes:** que buscan datos gratuitos para su proyecto fin de carrera, fin de grado, etc.
- **analistas de negocios y científicos de datos:** buscan, por ejemplo, información sobre aplicaciones móviles o establecimientos de comida rápida

Las consultas más comunes incluyen *educación, clima, cáncer, crimen, fútbol y perros*.

The screenshot shows search results for the query "fast food establishment". There are four main items displayed:

- OAHU Food Establishments** (D): Data from data.hawaii.gov, catalog.data.gov, +2more. Updated Oct 13, 2012. Includes a link to explore at openICPSR Self-Deposit Archive.
- National Neighborhood Data Archive (NaNDA): Fast-Food Restaurants by Census Tract, United States, 2006-2015** (O): Data from www.openicpsr.org, datasearch.gesis.org, +2more. Includes three explore links: openICPSR Self-Deposit Archive, datasearch.gesis.org, and search.datacite.org. It also lists a unique identifier (DOI: <https://doi.org/10.3886/E115323V1>) and the dataset provider (University of Michigan, Institute for Social Research).
- Sample by reported visits to fast food restaurants and...** (F): Data from figshare.com. Updated Apr 8, 2017. Includes a link to explore at www.da-ra.de.
- kaggle Fast food restaurants US**: Data from www.kaggle.com. Updated Nov 21, 2018. Includes a link to explore at search.datacite.org. It also lists the time period covered (Jan 1, 2006 - Dec 31, 2015) and the license (Attribution 4.0 (CC BY 4.0)).

Algunos de los resultados de búsqueda para la consulta «fast food establishment» («establecimiento de comida rápida»)

¿Qué conjuntos de datos puedes encontrar?

Dataset Search nos brinda una instantánea de los datos que existen en la Web. Los temas más destacados son las geociencias, la biología y la agricultura. La mayoría de los gobiernos del mundo publican sus datos y los describen con [schema.org](#). Estados Unidos lidera el número de conjuntos de datos de gobierno abiertos disponibles, con más de 2 millones. El formato de datos más popular son las tablas. Puede encontrar más de 6 millones de ellas en Dataset Search.

La cantidad de conjuntos de datos que puede encontrar en la búsqueda de conjuntos de datos continúa creciendo. Si tienes un conjunto de datos en tu web y lo describes usando schema.org, un estándar abierto, otros pueden encontrarlo en el DataSet Search de Google.

Fuente: <https://blog.google/products/search/discovering-millions-datasets-web/>

Feliz San Valentín menéame

Por Jose Martinez Heras
13/02/2020

¡Feliz San Valentín! Vamos a celebrarlo haciendo un análisis de todas las noticias de portada de [menéame](#) desde el último San Valentín. Para el análisis, usaremos técnicas de Procesamiento del Lenguaje Natural y Visualización de datos.

Quisiera agradecer a [Alfonso Martínez Heras](#) su colaboración en este proyecto. Alfonso se ha encargado de crear un *web scrapper* para obtener estos artículos automáticamente.

Regalo de San Valentín para menéame

Este es el regalo de San Valentín de [IArtificial](#) para [menéame](#). Es un corazón hecho con las palabras de los artículos en portada desde el último San Valentín. El tamaño de las palabras corresponde a la frecuencia de su uso. La posición y color son aleatorios. ¡Espero que os guste!



Los 14 artículos más interesantes

Vamos descubrir, automáticamente, cuáles son los 14 artículos más interesantes. Normalmente son los 10 más interesantes ... pero como San Valentín es el 14 de Febrero, me decanto por 14 !

Para encontrar los artículos más interesantes he usado técnicas de procesamiento del lenguaje natural [no-supervisado](#) (modelo bolsa de palabras, tf-idf, similitud coseno, modelado de temas, etc...). En otros posts más técnicos explicaré cómo funcionan. De momento, nos podemos quedar con que interpreto «más interesante» como «más diferente» en comparación con el resto de artículos».

Estos son los 14 artículos más interesantes (diferentes):

1. [Columbine, 20 años después](#)
2. [Bhopal 35 años después \[ENG\]](#)
3. [La «magnífica animalidad de una mulata del Misisipi»: Josephine Baker en España](#)
4. [La España que madruga](#)
5. [Acuerdo de gobierno entre PSOE y Podemos incluye a los animales por primera vez en la historia de España](#)
6. [Historia de España desde 718-2019](#)
7. [El 95% de hombres menores de 30 años en España no tienen hijos](#)
8. [Auge y caída de Lupín, el mayor ciberestafador de España \(de solo 23 años\)](#)
9. [¿Dónde estaba situada tu ciudad hace millones de años?](#)
10. [Vox anuncia un acuerdo con el PP para entrar en el Gobierno de la Comunidad de Madrid](#)
11. [El Ayuntamiento de Barcelona pide ser capital de España junto a Madrid](#)
12. [España, 27 años encabezando la donación de órganos en todo el mundo](#)
13. [Europa aprueba una ayuda pública de 400 millones de euros para instalar banda ancha en España](#)
14. [El Banco de España da por perdido el 75% del rescate a la banca: 42.561 millones de euros](#)

¡Feliz San Valentín!

Espero que te haya gustado este artículo dedicado al día de San Valentín. Feliz día de San Valentín y hasta el siguiente post.

La Maldición de la Dimensión en Machine Learning

Por Jose Martinez Heras
08/03/2020

¿Sabías que a medida que aumenta el número de dimensiones, las distancias se vuelven menos discriminativas? A este efecto se le conoce con el nombre de la Maldición de la Dimensión (*Curse of Dimensionality*) y tiene un gran impacto en técnicas de aprendizaje automático basadas en distancias tales como K-Means y KNN (los k vecinos más cercanos).

La Maldición de la Dimensión

En Machine Learning, el número de dimensiones se puede equiparar al número de variables o características (*features*) que estemos utilizando.

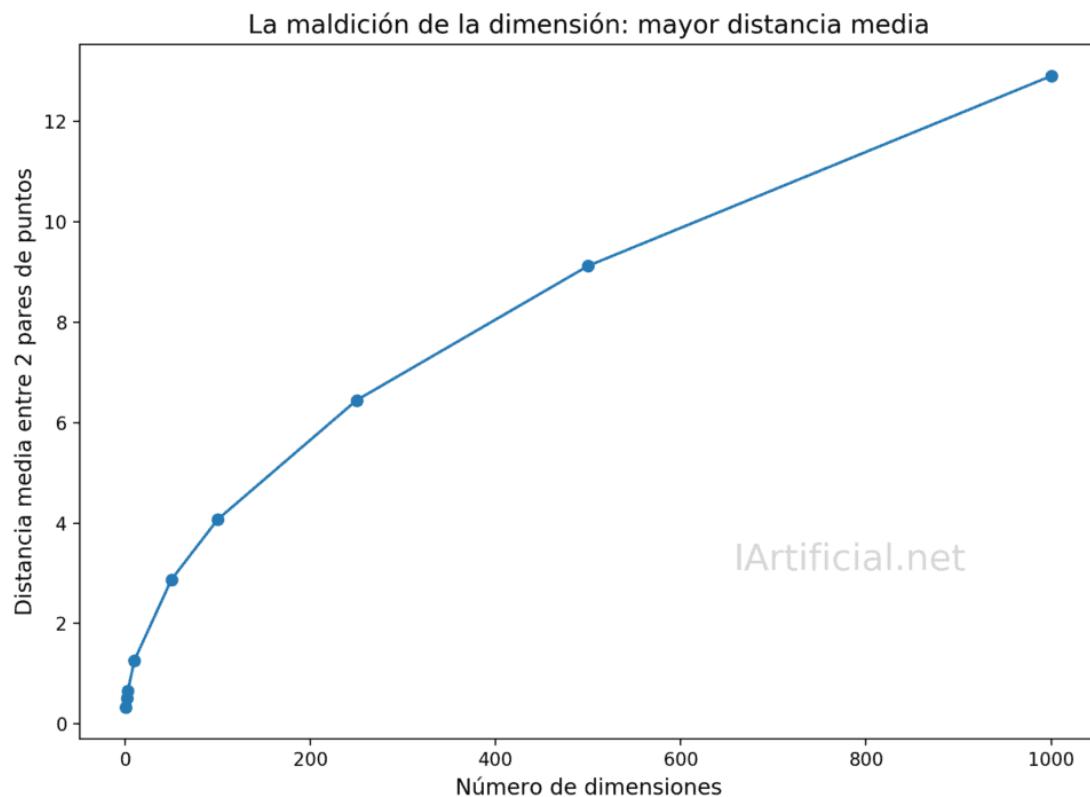
La maldición de la dimensión se «manifiesta» de dos maneras:

- La distancia media entre los datos aumenta con el número de dimensiones
- La variabilidad de la distancia disminuye exponencialmente con el número de dimensiones (éste es el verdadero problema)

Vamos a verlo en detalle. Para darte una intuición gráfica he generado puntos aleatorios en varias dimensiones siempre en el rango [0, 1].

La distancia media aumenta con el número de dimensiones

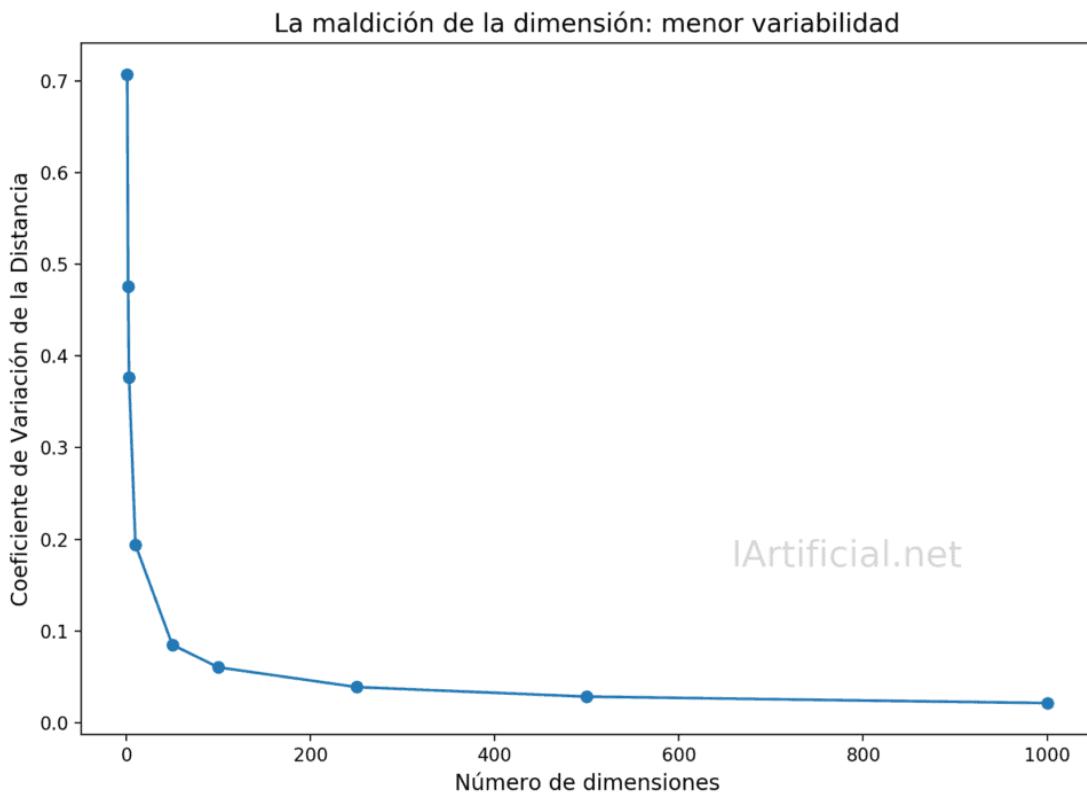
A medida que el número de dimensiones aumenta, la distancia media entre ellos también aumenta. Esto en sí mismo no es muy problemático para el aprendizaje automático. El siguiente gráfico muestra este efecto (hasta 1000 dimensiones).



La variabilidad disminuye exponencialmente

Lo verdaderamente importante es que a medida que el número de dimensiones aumenta ... ¡la variabilidad de las distancias entre los datos disminuye exponencialmente!

El siguiente gráfico muestra el [coeficiente de variación](#) para varias dimensiones (hasta 1000). El coeficiente de variación se utiliza para medir la variabilidad.

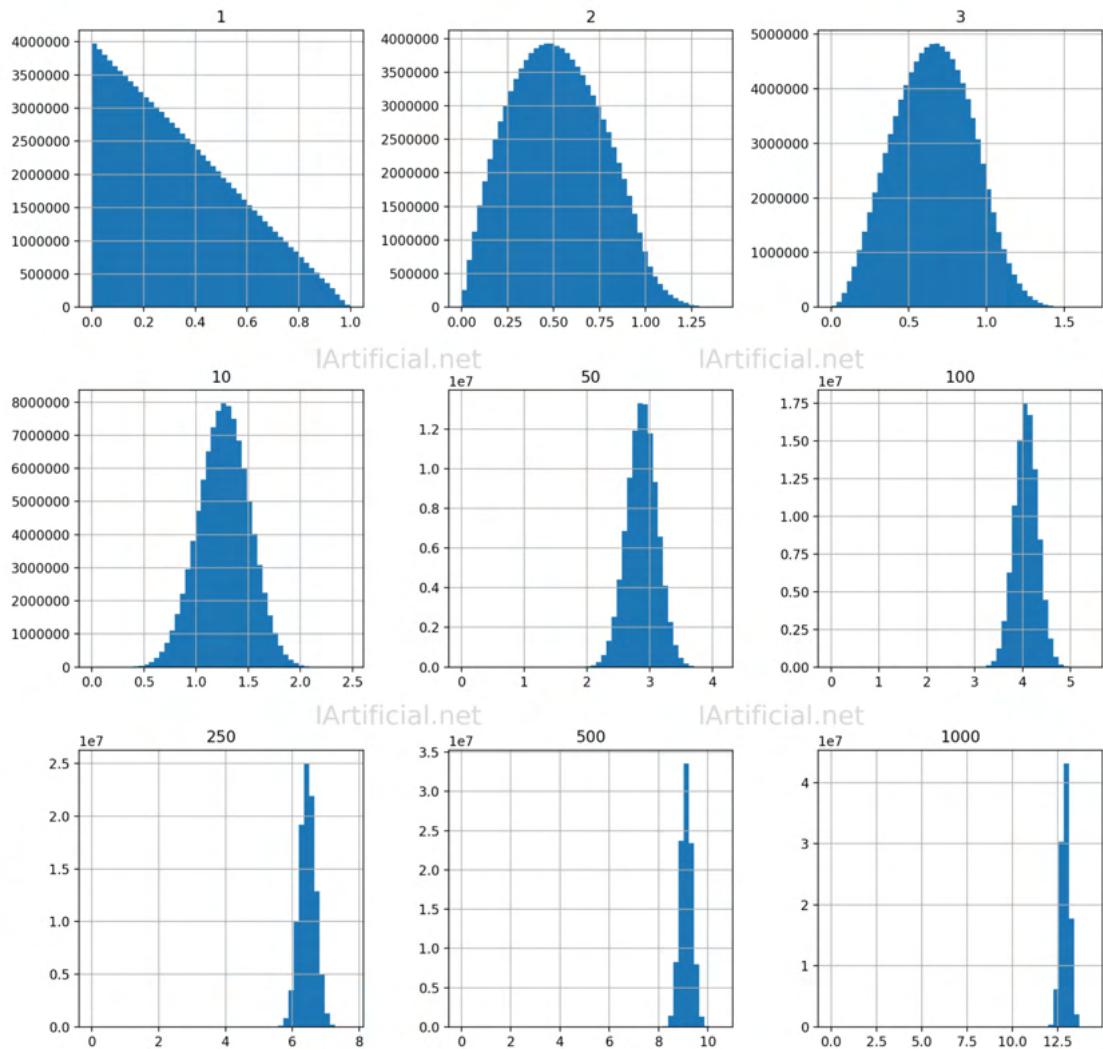


Esto es terrible para los métodos de aprendizaje automático basados en distancias tales como clustering o los vecinos más cercanos. El problema es que cuando hay un gran número de atributos (*features*), los datos están todos a casi la misma distancia. Es decir, no hay variabilidad entre sus distancias.

Distribución de distancias a medida que aumenta la dimensión

La maldición de la dimensión es un concepto un tanto abstracto. Por eso, creo que una intuición gráfica que nos explique qué está pasando cuando cuando aumenta el número de variables puede resultar esclarecedora.

La siguiente figura muestra la distribución de distancias euclídeas entre 10000 puntos aleatorios en el rango $[0, 1]$ en varias dimensiones: 1, 2, 3, 10, 50, 100, 250, 500 y 1000.



Distribución de las distancias en distintas dimensiones

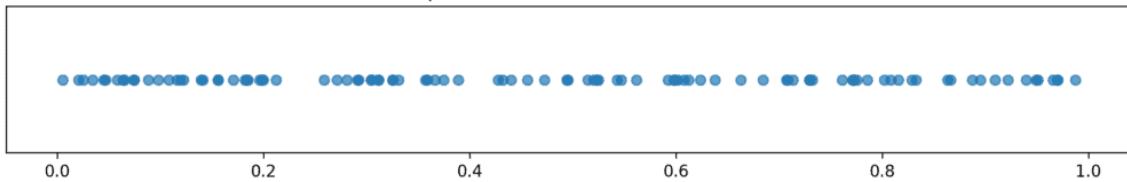
Como puedes apreciar, conforme van aumentando las dimensiones la variabilidad va desapareciendo. Esto hace que los algoritmos de machine learning basados en distancias tengan muy difícil el saber qué puntos están más cerca que otros.

Intuición Gráfica de la Maldición de la Dimensión

¿Por qué aumenta la distancia media con el número de dimensiones? ¿y por qué se reduce la variabilidad entre las distancias? Vamos a tratar de verlo con un ejemplo.

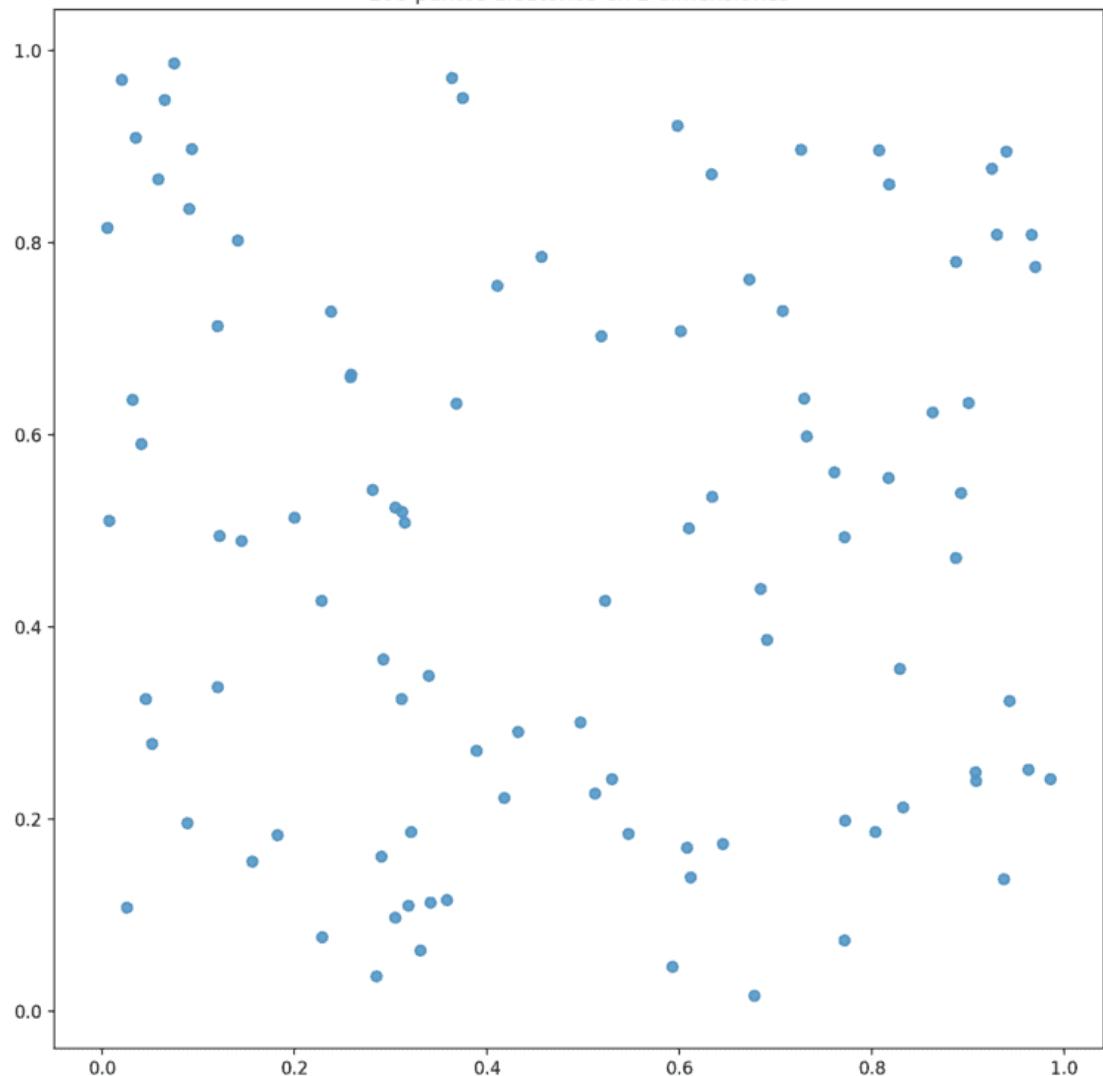
En la siguiente figura podemos ver 100 puntos elegidos aleatoriamente en el rango [0,1] para 1 dimensión. Como ves, es bastante fácil encontrar puntos que estén muy cerca (por ejemplo: 0.21 y 0.25) y puntos que estén lejos (0.21 y 0.99). Para cualquier punto, en 1 dimensión, es fácil encontrar muchos puntos que estén cerca y bastantes puntos que estén lejos.

100 puntos aleatorios en 1 dimensión



Sin embargo, encontrar muchos puntos que estén cerca de uno dado ya no es tan fácil cuando los 100 puntos los ponemos aleatoriamente en 2 dimensiones. En 2 dimensiones, la mayoría de los puntos están a una distancia media de otros ... mientras que para 1 dimensión, la mayoría de los puntos están cerca de otros puntos.

100 puntos aleatorios en 2 dimensiones



Este efecto se acentúa a medida que el número de dimensiones va subiendo.

Impacto de la Maldición de la Dimensión en Machine Learning

En una representación de datos de alta dimensionalidad, cada dato está aproximadamente a la misma distancia de cualquier otro. Es decir, la distancia discrimina menos conforme aumenta el número de dimensiones.

Esto no siempre es un problema, ya que la maldición de la dimensión sólo afecta a las técnicas basadas en distancias.

Estas son algunas de las técnicas de aprendizaje automático afectadas por la maldición de la dimensión:

- Agrupamiento (clustering): para encontrar automáticamente grupos en los datos
- K vecinos más cercanos (KNN): ofrece una predicción combinando los resultados de instancias similares.
- Técnicas de detección de anomalías basadas en distancias tales como [Local Outlier Factor](#).

Mitigando el efecto de la Maldición de la Dimensión

Para mitigar el efecto de la maldición de la dimensión tenemos dos opciones:

- reducir el número de dimensiones
- aumentar la cantidad de datos

Técnicas de reducción de la dimensionalidad

Hay varias técnicas para reducir la dimensión de nuestro problema. La más conocida es el Análisis de Componentes Principales (*PCA* por sus siglas en inglés, *Principal Component Analysis*). Otras técnicas incluyen el uso de *auto-encoders*, [variedades](#) (*manifold*), etc.

Las técnicas de reducción de dimensionalidad asumen que pueden capturar bastante variabilidad de los datos en menos dimensiones. Así crean un conjunto nuevo de dimensiones que representan nuestros mismos datos con menor dimensionalidad.

Esto permite utilizar técnicas de machine learning en una dimensionalidad menor. Para técnicas sensibles a la maldición de la dimensión, los resultados serán probablemente mucho mejores. En cualquier caso, al tener menos atributos, hay una mejora de rendimiento, tanto en el uso de memoria como en el uso de la CPU.

El problema ... es que no resulta muy interpretable lo que significan el conjunto nuevo de dimensiones. Sin embargo, si no nos preocupa la interpretabilidad para algún modelo, es un enfoque aconsejable.

Aumentar (exponencialmente) la cantidad de datos

A más dimensiones, más datos se necesitan para llenar el espacio. Cuando el número de dimensiones es muy alto, el espacio está casi vacío. Por eso es tan difícil encontrar puntos que estén cerca.

Por esta razón, también podemos mitigar el efecto de la maldición de la dimensión aumentando la cantidad de datos. Este aumento debe ser exponencial con el número de dimensiones, para contrarrestar el efecto exponencial de la pérdida de variabilidad de las distancias.

Normalmente es mucho más difícil (y más caro) aumentar exponencialmente la cantidad de datos. Así que las técnicas de reducción de la dimensionalidad se usan con frecuencia en estos casos. Por otra parte, si para un problema en particular, es fácil (y barato) conseguir muchos más datos, puede ser una buena opción.

Resumen

La maldición de la dimensión nos enseña que la variabilidad de las distancias entre nuestros datos disminuye exponencialmente con el número de dimensiones (*features*).

Esta maldición afecta a las técnicas que estén basadas en distancias tales como agrupamiento (*clustering*), los vecinos más cercanos (k-NN), algunas técnicas de detección de *outliers*, etc.

Para mitigar los efectos de esta «maldición» podemos utilizar técnicas de reducción de la dimensionalidad y / o aumentar (exponencialmente) la cantidad de datos.

Fuente imagen del espejo: pixbay

Redes neuronales desde cero (II): algo de matemáticas

Por José David Villanueva García
26/05/2020

En este segundo post de la serie nos adentraremos un poco en las matemáticas que hay detrás de las redes neuronales. Ya vimos algo de matemáticas en el anterior post, donde hablamos sobre las diferentes funciones de activación de las neuronas. Ahora veremos una de las partes más importantes de las redes neuronales: la asignación de los pesos para cada conexión neuronal. Entre los algoritmos más usados para asignar estos pesos se encuentra el llamado **back propagation** (propagación hacia atrás).

Introducción

Como ya intuimos por el post anterior, la programación clásica y las redes neuronales son aproximaciones distintas para resolver problemas. En la programación clásica, escribimos código fuente indicando al ordenador qué decisiones han de tomarse de forma absolutamente determinista para resolver un problema, una descripción detallada de todo lo que hay que hacer, similar a una receta. Por ejemplo, si lo que queremos es sumar dos números, tiene sentido programar en nuestro lenguaje favorito el algoritmo de la suma. Sin embargo, supongamos que nuestro problema es distinguir si una fotografía es un coche o no. En principio, no parece complicado cómo estructurar un algoritmo, un coche siempre tiene cuatro ruedas, ventanas, parabrisas, faros.

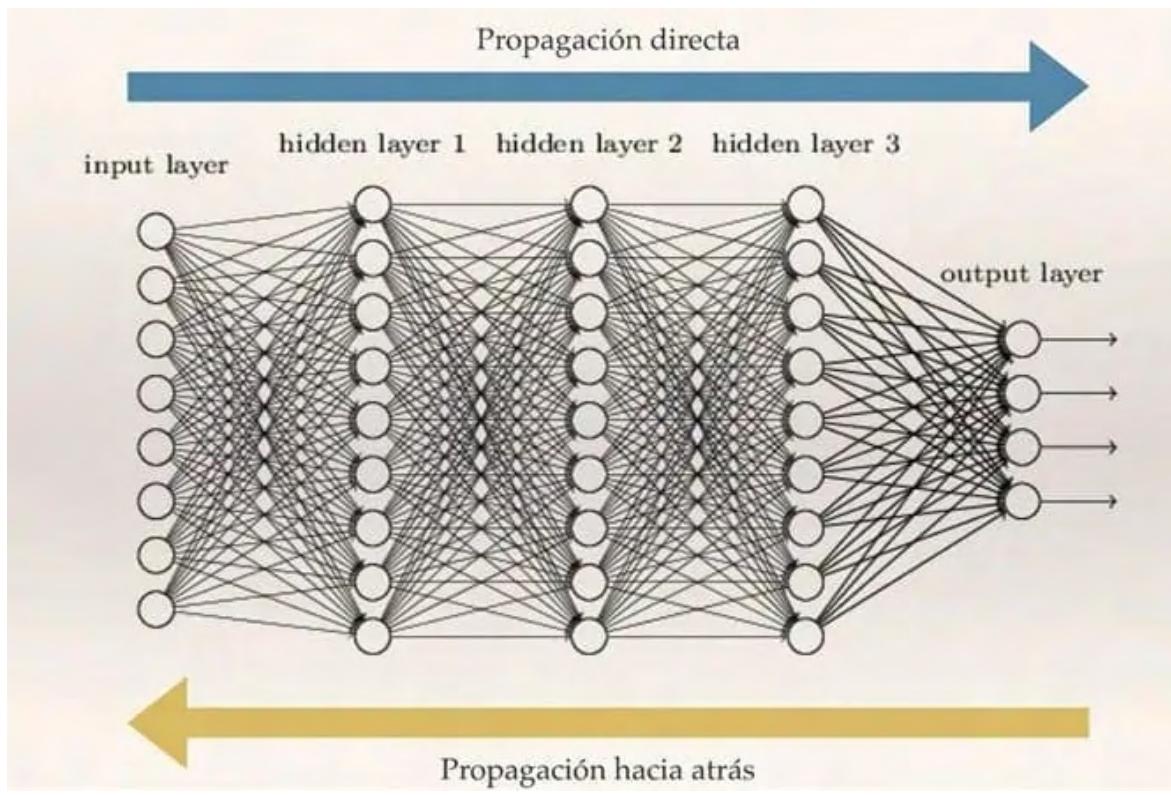
Pero algunas de estas características pueden no aparecer en la fotografía, la cual puede mostrar parte de coche, en lugar del coche entero. Además, diferenciar entre ruedas y ventanas, por ejemplo, puede ser igual de complicado que reconocer el propio coche. Por lo tanto, lo que hacemos en mostrarle a la red neuronal la mayor cantidad de coches diferentes posibles, y con estos ejemplos lo que la máquina aprende es el algoritmo por si mismo para identificar coches. Lo que hacemos es acudir a las redes neuronales como vimos en el post anterior. Lo único que nos falta ahora es asignar los pesos a cada una de las conexiones entre neuronas.

Asignación de pesos en Redes Neuronales

Una vez que tenemos el diseño de nuestra red neuronal, necesitamos asignar pesos a cada conexión neuronal, de manera que la salida de la función de activación de cada neurona se vea afectada por el paso de cada conexión. Existen varios algoritmos de asignación de pesos, como el gradiente de descenso o el back propagation. Veremos en detalle este último, pero primero echemos un vistazo a un problema general que tenemos al asignar los pesos, clave para entender la existencia de estos algoritmos.

Problema general de la asignación de pesos

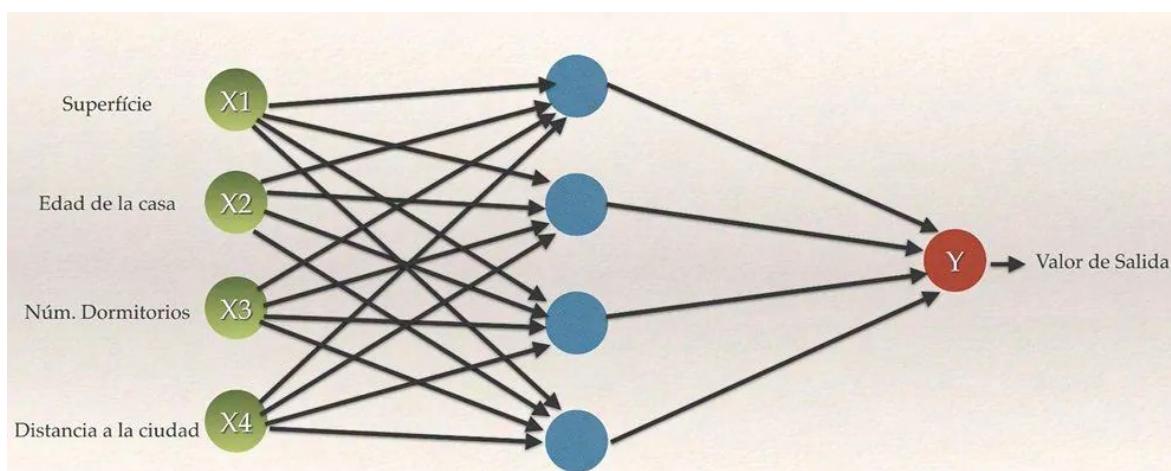
Supongamos entonces que tenemos la red neuronal que de la siguiente imagen:



Sabemos que la información se introduce en la primera capa de neuronas por la izquierda, propagándose hacia delante atravesando las diferentes capas ocultas, hasta que llega a la capa de neuronas de salida, dando el resultado final. Este resultado se compara con los valores reales para calcular el error que se comete, y propagar este error hacia atrás a través de la red neuronal, lo que nos va a permitir ajustar los pesos durante el proceso de entrenamiento. Veremos algunas de las matemáticas involucradas en este proceso un poco más adelante, pero es muy importante recalcar que en la propagación hacia atrás **todos los pesos se ajustan de manera simultánea**.

Pero, ¿por qué complicarnos la vida con algoritmos de ajuste de pesos aparentemente tan difíciles, pudiendo asignar los pesos y probar con todas las combinaciones posibles para al final quedarnos con la que minimice el error? Esto es lo que se denomina **ajuste de pesos por fuerza bruta**. Veamos como esto no es viable, incluso teniendo una red neuronal muy simple, como la que usamos en el post anterior.

Algoritmo de fuerza bruta



Como vemos en la imagen, tenemos las cuatro neuronas de la capa de entrada conectadas con las cuatro neuronas de la capa oculta, lo que nos da un total de 16 conexiones. Además, las cuatro neuronas de la capa oculta están conectadas con la única neurona de salida, lo que nos da 4 conexiones más, por lo que en total tenemos 20 conexiones, es decir, debemos que calcular **20 pesos**. En principio, Podemos pensar que estamos hablando de muy pocos pesos, únicamente 20, por lo que decidimos intentarlo por el método de la fuerza bruta, es decir, asignar pesos iniciales e ir probando todas las combinaciones obteniendo el error que cometemos, quedándonos con la combinación de pesos que minimice este error.

Imposible de realizar

Supongamos que tenemos 1000 valores posibles para cada uno de estos 20 pesos. Es decir, el número de combinaciones es:

$$1000 \times 1000 \dots \times 1000 = 1000^{20} = 10^{60}$$

De todas estas combinaciones, tendríamos que escoger aquella que minimizara el error, por lo que tendríamos que probar **absolutamente todas ellas**.

El ordenador más potente que existe sobre la faz de la Tierra mientras lee este post, es [Summit](#), capaz de realizar 200 cuatrillones de cálculos por segundo. Para probar todas las combinaciones, Summit tardaría miles de millones de veces la edad que tiene nuestro propio universo. Por ello, recurrimos a algoritmos complejos, como el *back propagation* o propagación hacia atrás.

Algoritmo back propagation

A continuación, antes de describir matemáticamente con un ejemplo el algoritmo, vemos cuáles son los pasos necesarios para implementar este algoritmo.

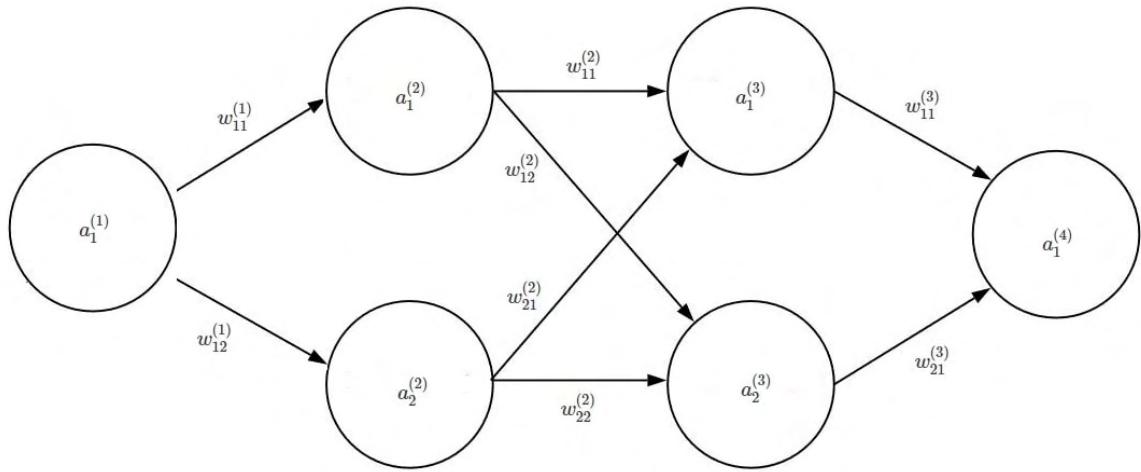
1. Asignamos a cada conexión neuronal un peso con un valor pequeño, pero no nulo.
2. Introducimos la primera observación de nuestro conjunto de entrenamiento por la capa inicial de la red neuronal.
3. La información se propaga de izquierda a derecha, activando cada neurona que ahora es afectada por el peso de cada conexión, hasta llegar a la capa de neuronas de salida, obteniendo el resultado final para esa observación en concreto.
4. Medimos el error que hemos cometido para esa observación.
5. Comienza la propagación hacia atrás de derecha a izquierda, actualizando los pesos de cada conexión neuronal, dependiendo de la responsabilidad del peso actualizado en el error cometido.
6. Repetimos los pasos desde el paso 2, actualizando todos los pesos para cada observación o conjunto de observaciones de nuestro conjunto de entrenamiento.
7. Cuando todas las observaciones del conjunto de entrenamiento ha pasado por la red neuronal, hemos completado lo que se denomina un Epoch. Podemos realizar tantos Epochs como creamos convenientes.

Matemáticas del algoritmo back propagation

Veamos ahora con un ejemplo simple las matemáticas que hay detrás del algoritmo back propagation.

Cálculos preliminares

Partamos de una configuración inicial de una red neuronal compuesta por una capa de entrada de una neurona, una capa oculta de dos neuronas y una capa de salida de una única neurona.



Las neuronas se indican con la letra «a» y las conexiones neuronales con la letra «w». Para las neuronas, el superíndice entre paréntesis indica la capa a la que pertenece la neurona , y el índice indica el número de neurona dentro de la capa. Así, por ejemplo,

$$a_2^{(3)}$$

indica la neurona 2 de capa 3.

Sabemos que las conexiones neuronales conectan neuronas de una capa con las neuronas de la capa siguiente. El superíndice del peso indica el número de la capa inicial de la conexión. El subíndice está compuesto de dos números: el primero indica el número de la neurona de la capa inicial de la conexión y el segundo el número de neurona dentro de la capa al final de la conexión, es decir,

$$w_{21}^{(2)}$$

indica que la conexión empieza en la capa 2, y conecta la segunda neurona de esta capa con la neurona 1 de la capa siguiente.

La salida de la neurona 1 dentro de la capa 3 sería entonces:

$$a_1^{(3)} = f(u_1^{(3)} + w_{11}^{(2)} a_1^{(2)} + w_{21}^{(2)} a_2^{(2)})$$

Ahora, hacemos lo mismo con la salida de las neuronas de la capa oculta anterior, y tenemos que:

$$a_1^{(2)} = f(u_1^{(2)} + w_{11}^{(1)} a_1^{(1)})$$

$$a_2^{(2)} = f(u_2^{(2)} + w_{12}^{(1)} a_1^{(1)})$$

Sustituimos estas dos últimas expresiones en la expresión inicial, y nos queda:

$$a_1^{(3)} = f(u_1^{(3)} + w_{11}^{(2)} f(u_1^{(2)} + w_{11}^{(1)} a_1^{(1)}) + w_{21}^{(2)} f(u_2^{(2)} + w_{12}^{(1)} a_1^{(1)}))$$

Generalizando la fórmula

Por lo tanto, generalizando la fórmula anterior para cualquier numero de capas k y cualquier neurona i dentro de la capa, tenemos que:

$$a_i^{(k)} = f(u_i^{(k)} + \sum_{j=1}^{n_k-1} a_j^{(k-1)} w_{ji}^{(k-1)}), i = 1 \dots n_k$$

Recordemos que las variables en toda esta expresión son los pesos, por lo que para hacer mínimo el error tenemos que derivar con respecto de cada uno de ellos, teniendo en cuenta que estamos usando una función de activación sigmoide, es decir, tenemos que hacer derivadas parciales con respecto a cada peso.

Una vez obtenido el valor mínimo, repetimos el proceso tantas veces como necesitemos para que las conexiones de la red queden ajustadas de manera óptima.

Resumen

En este post hemos visto la base de uno de los algoritmos mas utilizados para asignar pesos a las conexiones neuronales de nuestra red. Por supuesto, este es uno de entre muchos algoritmos que se usan para tal fin. Saber algo de las matemáticas básicas que se encuentran detrás de estos algoritmos nos dará una ventaja extra a la hora de elegir cual de ellos es mas adecuado usar en nuestra red neuronal. Por supuesto, aunque es un ejercicio muy conveniente y didáctico, no tenemos que programar explícitamente estos algoritmos al construir la red neuronal, prácticamente todas las librerías y frameworks que usamos para programar la red nos proporcionan métodos a los cuales únicamente hay que dotarles de los parámetros adecuados, y se encargaran de ejecutar el algoritmo por sí mismos.

Recursos

- [Tutorial](#) de red neuronal perceptron con algoritmo back propagation.
- [Video](#) fantástico (en inglés) sobre el algoritmo back propagation.

Créditos

Algunas de las imágenes de este artículo han sido reproducidas, con permiso, del [Curso completo de Inteligencia Artificial con Python](#).

Acerca del autor

Este es un post invitado por [José David Villanueva García](#). José David es Ingeniero Técnico en Informática de Sistema por la Universidad Rey Juan Carlos, Graduado en Matemáticas por la UNED, y Máster en Matemáticas Avanzadas por la UNED ([Máster Thesis](#)).

Actualmente trabaja como ingeniero en Darmstadt, Alemania, en diferentes proyectos para la ESA (European Space Agency) y EUMETSAT (European Organisation for the Exploitation of Meteorological Satellites).

Correlación, Covarianza e IBEX-35

Por Jose Martinez Heras
29/06/2020

Tanto la correlación como la covarianza son técnicas que nos ayudan a entender mejor cómo varía una variable en función de otra. Vamos a ver 3 tipos de correlación: Pearson, Spearman y Kendall. Veremos qué son, cómo se calculan y en qué casos funcionan mejor. Como ejemplo en python encontraremos qué compañías del IBEX-35 están más y menos correladas.

La correlación y la covarianza tienen muchas aplicaciones en campos tan diversos como genética, biología molecular, finanzas, etc. Además nos ayudan a [entender mejor los datos](#) y pueden ser útiles como [características en la fase de preparación de datos](#).

Covarianza

La covarianza es una medida de variabilidad entre dos variables. Es decir, mide cómo varía una variable en relación a otra variable. Para calcular la covarianza usaremos la siguiente fórmula.

$$\text{cov}(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$$

μ_X y μ_Y se refieren a la media aritmética de X e Y respectivamente

E se refiere a la esperanza matemática, es decir, a la media aritmética

Si a valores mayores de X le corresponden valores mayores de Y , y a valores menores de X también le corresponden valores menores de Y , entonces la covarianza será positiva. Es decir, la covarianza será positiva si las variables están relacionadas linealmente y se mueven en la misma dirección. La covarianza será negativa si a valores mayores de X le corresponden valores menores de Y y a valores menores de X le corresponden valores mayores de Y .

Las limitaciones de la covarianza son:

- La magnitud de la covarianza no es fácil de interpretar porque sus unidades son las unidades de X multiplicadas por las unidades de Y . Veremos que la correlación de Pearson resuelve este problema.
- La covarianza sólo es capaz de reconocer relaciones lineales. Como sabemos, muchas de las relaciones entre variables son, en la práctica, no lineales.
- Asume que las variables siguen una [distribución normal](#).

Correlación

Existen varios tipos de correlación. La más conocida es, sin duda, la de Pearson. De hecho, si no decimos cuál estamos usando, se asume que usamos la de Pearson. Además de la de Pearson veremos también la correlación de Spearman y la de Kendall.

Correlación de Pearson

La correlación de Pearson es una versión normalizada de la covarianza. Una de las desventajas de la covarianza es que sus unidades son difíciles de interpretar y comparar. Es decir, con la covarianza no podemos decir que la relación lineal entre A y B es mayor que entre A y C de forma intuitiva.

Para calcularla tendremos en cuenta la covarianza y las desviaciones típicas tanto de X como de Y . La expresión es la siguiente:

$$\text{pearson}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

La correlación de Pearson resuelve el problema de la interpretabilidad de la magnitud de la relación entre las variables X e Y. No tiene dimensiones. Dicho de otra forma, es adimensional.

Si la correlación es +1 las variables varían en la misma dirección en la misma cantidad relativa; si la correlación es -1 lo hacen en direcciones opuestas en cantidades relativas iguales. Así, el rango para la correlación de Pearson está siempre en [-1, 1]. Un valor de 0 indicaría que las dos variables no están relacionadas.

Las limitaciones de la correlación de Pearson son:

- Sólo puede detectar correctamente relaciones lineales.
- Asume que las variables siguen una distribución normal.

Veremos cómo las correlaciones de Spearman y Kendall superan estas limitaciones.

Correlación de Spearman

La correlación de Spearman es la misma que la de Pearson pero con un preprocesado. En lugar de aplicar la correlación a los datos, se aplica al ranking de los valores de X e Y.

Por ejemplo, si $X = [3.28, -4.12, 8.21, 6.65, -0.94]$, obtendremos su ranking r_X ordenando sus valores de menor a mayor y remplazándolos por su posición en el nuevo orden: $r_X = [3, 1, 5, 4, 2]$.

La fórmula sería:

$$spearman = \frac{cov(r_X, r_Y)}{\sigma_{r_X} \sigma_{r_Y}}$$

En el caso de Spearman, la correlación es positiva si tanto X como Y varían en la misma dirección y negativa si varían en dirección contraria. Su magnitud depende de cómo de fuerte sea esa relación.

Por ejemplo, si los valores más altos de X se corresponden con los valores de Y en todas las situaciones obtendremos un valor de +1. La correlación de Spearman también está normalizada en el rango [-1, +1]

Las ventajas de la correlación de Spearman son:

- No necesita asumir ningún tipo de distribución de las variables
- Puede encontrar relaciones no lineales. De hecho se centra en encontrar relaciones monotónicas.

Correlación de Kendall

La correlación de Kendall está indicada para encontrar relaciones entre variables que representen secuencias tales como series temporales. Se fija en cómo varían los pares de X e Y. Para ello define pares concordantes y discordantes:

- Par concordante: si X aumenta de valor, Y también aumenta de valor, o si X disminuye de valor, Y también disminuye de valor.
- Par discordante: si X aumenta de valor Y disminuye de valor, o si X disminuye de valor, Y aumenta de valor.

Date cuenta que sólo tienen que aumentar o disminuir simultáneamente para que los pares sean considerados concordantes. Es decir, no se fija para nada en la cantidad o de si son proporcionales. Esto le permite encontrar relaciones tanto lineales como no lineales.

$$kendall = \frac{(pares\ concordantes) - (pares\ discordantes)}{\frac{n(n-1)}{2}}$$

Como vemos, estos aumentos y disminuciones en valor son ideales para secuencias de datos tales como series temporales. La correlación de Spearman también está normalizada en el rango [-1, +1].

Las ventajas de la correlación de Kendall son:

- No necesita asumir ningún tipo de distribución de las variables
- Puede encontrar relaciones no lineales.
- Está pensada para secuencias de datos tales como las series temporales

Su desventaja sería:

- No tiene sentido usar la correlación de Kendall si los datos no se corresponden con secuencias tales como series temporales.

Diversificación en el IBEX35

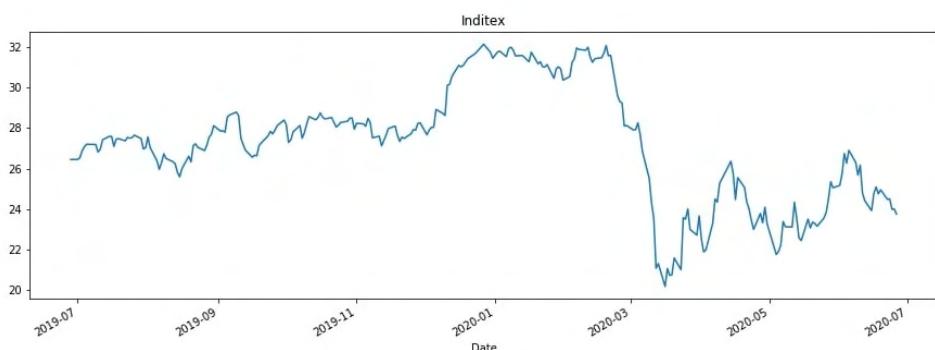
Datos

Los datos para este ejercicio son las cotizaciones diarias de las acciones que cotizan en el IBEX35 desde 28/06/2019 hasta 28/06/2020. Es decir, justo un año de cotizaciones. Los datos los he obtenido usando la librería [investpy](#)

investpy – a Python package for Financial Historical Data Extraction developed by Álvaro Bartolomé del Canto @ [alvarobartt](#) at GitHub

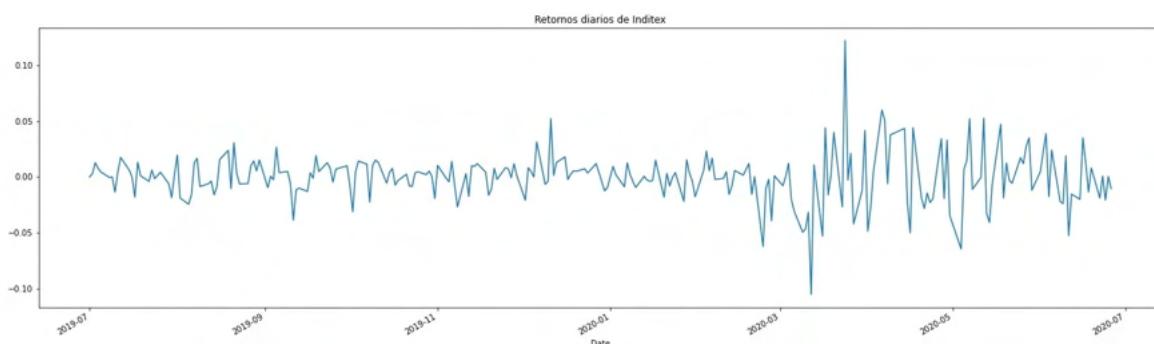
<https://pypi.org/project/investpy/>

Por ejemplo, esta sería la cotización de Inditex en este periodo.



Transformación de datos

Para este análisis he usado las ganancias diarias, es decir, qué porcentaje ha subido (o bajado) la cotización de una acción con respecto al precio del día anterior. Así por ejemplo, el porcentaje diario de la cotización de Inditex quedaría de esta forma.



IBEX35 con la correlación Kendall

Me he decidido a hacer una correlación Kendall con los datos de los retornos diarios de las cotizaciones del IBEX35 por estar optimizada para series temporales. Así podemos examinar la relación temporal de los rendimientos diarios.

Además tiene la ventaja de no asumir ninguna distribución (esto sería un problema para Pearson porque es sabido que la distribución de los retornos bursátiles no siguen una distribución normal). De esta forma podremos descubrir relaciones tanto lineales como no lineales.

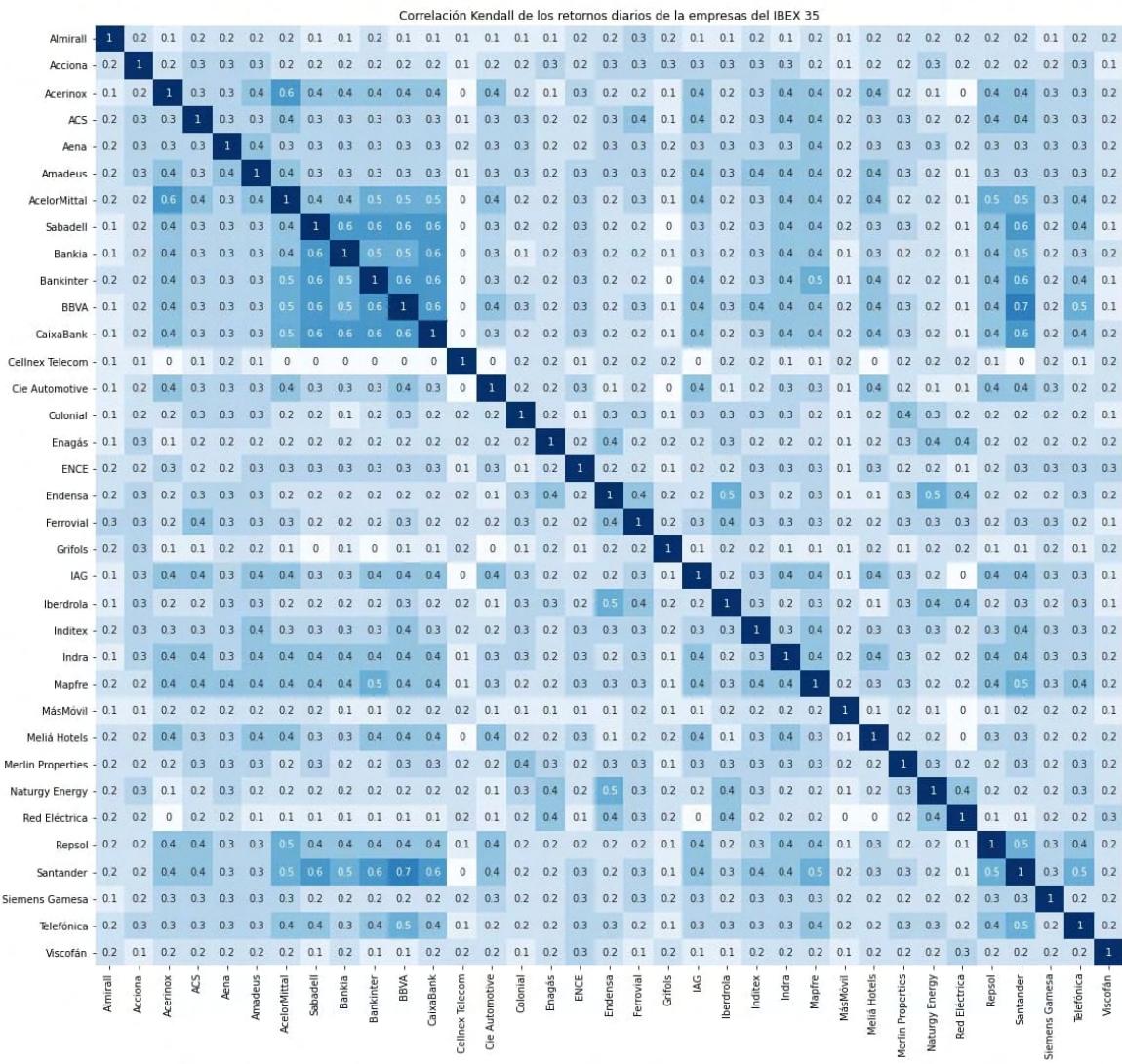
Para calcular la correlación en python he usado la librería [pandas](#). Es lo más cómodo porque tenía los datos en un [DataFrame](#) de pandas.

```
1.     kendall = retornos_diarios.corr(method='kendall').abs()
2.     #pearson = retornos_diarios.corr().abs()
3.     #pearson = retornos_diarios.corr(method='pearson').abs()
4.     #spearman = retornos_diarios.corr(method='spearman').abs()
```

Sólo vamos a usar la correlación de Kendall, pero también te he indicado en el código comentado cómo puedes calcular la correlación de Pearson (que es la que usa pandas por defecto) y la de Spearman.

Al final tomo el valor absoluto porque para este análisis me interesa saber qué acciones están relacionadas con otras, independientemente de si la relación es positiva o negativa.

La correlación de Kendall de los retornos diarios de las empresas que componen el IBEX 35 se puede ver en el siguiente gráfico.



Como puedes ver, se puede apreciar que, normalmente, las empresas que pertenecen a los mismo sectores están más correlacionadas entre sí que con empresas de otros sectores.

Podemos ahora encontrar cuáles son las empresas que menos correlación tienen, en media, con el resto de empresas del IBEX 35.

```
1. # Calculamos el valor medio para cada empresa y ordenamos
```

```
2. kendall.mean().sort_values()
```

Nos daría el siguiente resultado

1.	Cellnex Telecom	0.134929
2.	Grifols	0.151353
3.	MásMóvil	0.176032
4.	Almirall	0.185148
5.	Red Eléctrica	0.194319
6.	Viscofán	0.199901
7.	Enagás	0.235660
8.	Colonial	0.237525
9.	Siemens Gamesa	0.245824
10.	Acciona	0.254782
11.	ENCE	0.258932

12.	Naturgy Energy	0.259595
13.	Cie Automotive	0.265624
14.	Iberdrola	0.265870
15.	Merlin Properties	0.267845
16.	Endensa	0.272461
17.	Meliá Hotels	0.287074
18.	Ferrovial	0.294401
19.	Aena	0.302596
20.	IAG	0.303259
21.	Bankia	0.306136
22.	Telefónica	0.306876
23.	Acerinox	0.309142
24.	ACS	0.309831
25.	Inditex	0.309949
26.	Amadeus	0.309978
27.	Repsol	0.311501
28.	Sabadell	0.314201
29.	Indra	0.316982
30.	Bankinter	0.323318
31.	CaixaBank	0.330371
32.	Mapfre	0.334976
33.	AcelorMittal	0.338454
34.	Santander	0.353874
35.	BBVA	0.355677

Así vemos que los rendimientos diarios de Cellnex Telecom, Grifols, MásMóvil, Almirall y Red Eléctrica están entre los menos correlados con el resto de las empresas. Sin embargo, BBVA, Santander, AcelorMittal, Mapfre y CaixaBank están entre los más correlados.

Observación

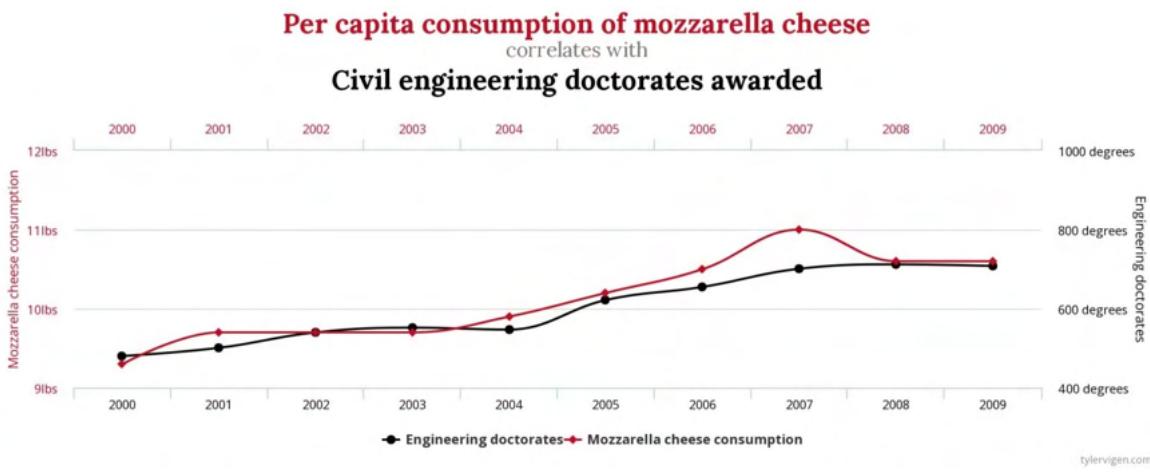
Si quieras invertir en acciones del IBEX 35 y asegurarte que en realidad estás diversificando, comprueba en la matriz de correlación que sus retornos no estén excesivamente correlados.

Por cierto, esto no es ningún consejo para invertir, sólo un ejercicio para demostrar los resultados que se pueden obtener haciendo un análisis de correlación.

Te dejamos como ejercicio calcular covarianzas y los distintos tipos de correlaciones con datos fundamentales.

La correlación no implica causalidad

La correlación indica que dos variables están relacionadas pero esto no tiene por qué significar que una sea la causa de otra. Por ejemplo, parece razonable creer que el aumento del consumo per cápita de queso mozzarella no es la causa del aumento de los ingenieros civiles, pero la correlación está ahí.



[Fuente](#)

Para más ejemplos de correlaciones divertidas puedes echar un vistazo a [tylervigen](#).

Quería despedirme con esta viñeta de [xkcd](#). Que luego no digas que no te avisé que la correlación no sirve para determinar causalidad.



[Fuente](#)

Resumen

Hemos visto cómo estudiar la interrelación entre variables con la covarianza y los distintos tipos de correlación. Hemos visto las ventajas y desventajas de cada técnica y cómo se han ido superando su limitaciones.

También, a modo de ejemplo, hemos estudiado cómo están interrelacionados los retornos diarios de las empresas del IBEX 35 y cuáles son las empresas más y menos correladas.

Por último hemos hecho hincapié en el hecho de que la correlación no implica causalidad.

Recursos

- Cálculo de covarianza en: [pandas](#), [numpy](#)
- Cálculo de correlación en: [pandas](#), [numpy](#)
- Librería para descargar cotizaciones bursátiles: [investpy](#)
- Correlaciones [divertidas](#)

10 Blogs de Inteligencia Artificial y Machine Learning

Por Jose Martinez Heras
16/08/2020

Hemos recopilado una lista de los blogs de Inteligencia Artificial, Machine Learning y Análisis de Datos en español. Nuestro objetivo es poner a tu alcance la información disponible sobre IA en castellano.

Listado de Blogs de Inteligencia Artificial en español

- [Analytics Lane](#)
- [Aprende Machine Learning – antes de que sea demasiado tarde](#)
- [Ciencia de Datos, Estadística, Programación y Machine Learning](#)
- [Gaussianos – porque todo tiende a infinito](#)
- [IArtificial.net – Inteligencia Artificial, Machine Learning y Análisis de Datos](#)
- [Ligdi González – Aprende todo sobre Inteligencia Artificial](#)
- [Inteligencia Analítica](#)
- [Machine Learning para todos – Machine Learning, Data Science y Analítica Avanzada](#)
- [Raul E. Lopez Briega – Matemáticas, análisis de datos y python](#)
- [sitiobigdata](#)

Estamos seguros que nos hemos olvidado de muchos blogs, así que si tu blog favorito en español sobre machine learning no está en la lista, ponlo en los comentarios y lo añadiremos más tarde.

Nota: la lista está ordenada alfabéticamente (por url)

Antenas de Espacio Profundo & Inteligencia Artificial

Por Jose Martinez Heras
22/08/2020

Hoy quería poneros al tanto sobre un proyecto de machine learning que estamos llevando a cabo con la Agencia Espacial Europea. Va sobre antenas e inteligencia artificial.

Utilizamos antenas de espacio profundo para comunicarnos con satélites que estén en el «espacio profundo», por ejemplo, en Mercurio, Venus, Marte, etc. En la foto se puede apreciar la [estación de Marlargüe](#). Tiene 35 metros de diámetro y para poner el tamaño en perspectiva, puedes fijarte que abajo a la izquierda hay una persona.

Al estar estos satélites tan lejos y ser la señal de radio tan débil, además de necesitar antenas de gran tamaño, necesitamos que el apuntamiento de estas antenas sea muy preciso, en el orden de miligrados. Cuando el **viento** es muy fuerte, el apuntamiento tiene errores. Es decir, apunta a donde no debería, aunque sea un error muy pequeño. Estos errores pueden causar que se interrumpa el enlace y se pierdan datos.

Hemos comenzado un proyecto donde usamos machine learning para predecir el error de apuntamiento de antenas de espacio profundo. Los primeros de resultados de nuestro prototipo, con un horizonte de predicción de 500ms, son excelentes. Las predicciones hechas por la IA servirán para compensar en gran medida los errores de apuntamiento. En la práctica, esto significa que aumentará la cantidad de datos que obtengamos de los satélites científicos debido al incremento de disponibilidad del enlace.

Estoy realizando este proyecto junto con Peter Droll en el marco de la primera copa de innovación en operaciones espaciales (1st OPS Innovation Cup) en la Agencia Espacial Europea.

Detección de anomalías en espacio

Por Jose Martinez Heras
19/09/2020

La detección de anomalías es un aspecto muy importante en la monitorización de satélites. Detectar una anomalía a tiempo nos va a permitir, en muchos casos, tomar las medidas necesarias para que la anomalía no se produzca o para minimizar sus efectos.

Te presentamos los retos, una solución sin machine learning, nuestra solución con machine learning usando aprendizaje semi-supervisado, la «salsa secreta», algún ejemplo, soluciones de otros operadores y qué librerías de python te pueden ayudar a detectar anomalías.

A lo mejor no trabajas con satélites o con espacio. Pero seguro que este artículo te da que pensar, y te ofrece soluciones que a lo mejor no habías tenido en cuenta. Te puede venir muy bien si trabajas en monitorización, mantenimiento predictivo o necesitas detectar comportamientos atípicos en datos.

Retos de la detección de anomalías en Operaciones Espaciales

¿Qué es una anomalía?

El primer reto al que nos enfrentamos es la propia definición de anomalía. ¿Qué es una anomalía? Hay varias definiciones, pero la que considero más útil es la que usamos, informalmente, en operaciones de espacio.

No queríamos que esto hubiera pasado

También está, por supuesto, la definición complementaria «queríamos que esto hubiera pasado y no pasó», pero en el 99% de los casos, cuando hablamos de anomalías nos referimos a «no queríamos que esto hubiera pasado».

Es muy difícil instruir a un sistema automático sobre qué es lo que queríamos que no hubiera pasado. Así que, en la práctica, resolvemos un problema más fácil: detectar comportamientos inusuales («novelty detection» en la terminología de machine learning).

Al detectar comportamientos inusuales, no tenemos en cuenta lo que se esperaba ... pero es un problema mucho más fácil de afrontar. De hecho, basada en nuestra experiencia podemos afirmar que **comportamientos inusuales son en muchas ocasiones la pista de que una anomalía está a punto de ocurrir**.

Localización de la anomalía

Saber cuándo hay una anomalía es muy útil. Pero también necesitamos saber dónde está ocurriendo la anomalía.

Los satélites actuales tienen entre 20,000 y 40,000 sensores (sin tener en cuenta los datos científicos). Estos sensores miden cosas tan diversas como el voltaje de la batería, el porcentaje usado de la CPU de ordenador de a bordo, la velocidad a la que se mueven las ruedas de inercia, la presión en los tanques de combustible, etc.

Cuando hay una anomalía, es muy importante que podamos indicar a los ingenieros de operaciones dónde (en qué sensores) está la anomalía para que puedan investigarlo más a fondo.

Anomalías falsas

Hay muchas técnicas de detección de anomalías y detección de outliers. Muchas de ellas tienen el problema de detectar demasiadas anomalías falsas. Demasiados falsos positivos.

Para ponerte un ejemplo, imagina un sistema de monitorización de comportamiento atípico que sea correcto un 99% de las veces. Este sistema sería terrible! Tan malo, que nadie en operaciones espaciales tendría la paciencia de usarlo. ¿Por qué? Porque un 1% de las veces daría una alarma falsa. Con unos 30,000 sensores, esto correspondería a 300 alarmas falsas que alguien tendría que comprobar.

Los falsos positivos en la detección de anomalías tienen un coste. Los ingenieros de operaciones tienen que comprobarlas y esto les lleva tiempo. Sin embargo, el mayor coste es el psicológico. Cuando hay muchos falsos positivos, los ingenieros de operaciones dejarían de confiar en estos resultados.

Así que es mejor tener pocas detecciones pero que las pocas que se detecten sean comportamientos inusuales que merezca la pena investigar.

Obviedad de la anomalía

Al detectar una anomalía, el sistema alerta a los ingenieros de operaciones. Ellos deben considerar obvio por qué el sistema los ha avisado.

Lo que vamos buscando es obviedad. Que cualquier persona al ver un gráfico de la serie temporal histórica diga, hay algo raro aquí. Cualquier persona podría hacerlo, el problema es que hay unos 30,000 sensores que comprobar.

Queremos tener esa intuición humana en un software que pueda hacer esta comprobación automáticamente.

Anomalías que ocurren por primera vez

Las peores anomalías son las que ocurren por primera vez. Nadie esperaba que esto ocurriese.

Por supuesto, sabemos que habrá anomalías en los satélites que operamos. Esto es normal porque muchos satélites son construidos pensando que tendrán una vida útil de cierto tiempo y se siguen usando mucho después. Sabemos que algo fallará, lo que no sabemos es en qué parte, cuándo y cómo.

Saber que hay un comportamiento inusual en alguno(s) de los sensores nos da la pista de que a lo mejor vamos a tener una anomalía de aquí a poco. Esto nos da tiempo a investigar antes de que se convierta en un problema.

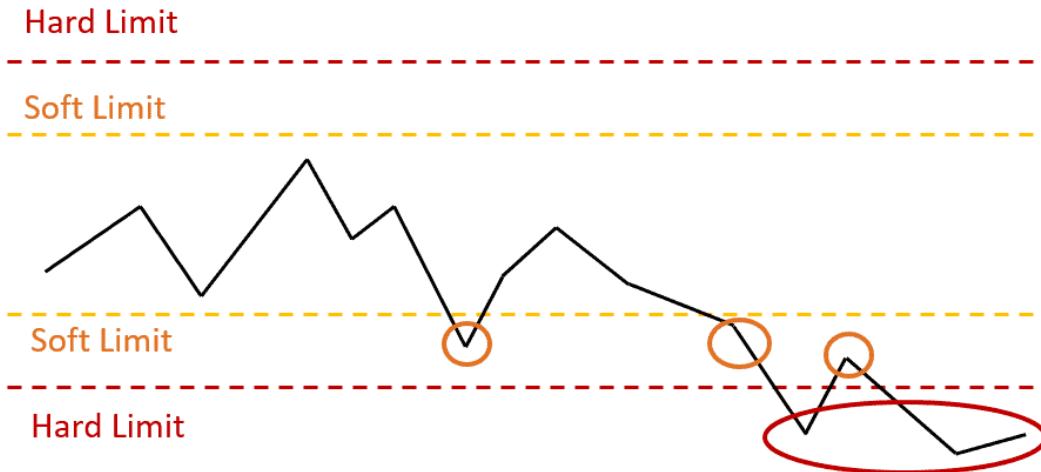
En las operaciones espaciales, muchas veces sólo hay anomalías que ocurren una vez. Esto es así porque los ingenieros de operaciones, una vez que entienden por qué ha ocurrido la anomalía, operan el satélite de otra forma. Es decir, evitan las condiciones que harían que la anomalía ocurriese otra vez. Algunas veces la anomalía es inevitable y tenemos que vivir con ella. Aun así, podemos diseñar estrategias para minimizar su impacto.

Desde el punto de vista de aprendizaje automático, esto quiere decir que no podemos utilizar aprendizaje supervisado. Al menos no en el sentido de predecir directamente si cualquier comportamiento es anómalo.

Control de límites

El control de límites es la solución clásica en sistemas de control.

A cada sensor se le asignan límites inferiores y superiores. Se usan normalmente dos tipos de límites: límites blandos (soft limits) y límites duros (hard limits). Cuando un sensor alcanza alguno de estos límites decimos que está «fuera de límites» (Out-of-Limits: OOL). En este caso, el sistema de control alerta a los operadores.



Aunque parezca simple es bastante efectivo. En su sencillez estriba su eficacia.

El control de límites tiene las siguientes ventajas:

- Puede detectar que un sensor ha salido de límites tan pronto como salga. Es muy rápido detectando.
- Los límites pueden ser condicionales. Por ejemplo, si una unidad está en el modo «A», aplica unos límites. Si está en el modo «B», aplica estos otros.

El control de límites también tiene sus desventajas:

- No todos los sensores tienen asociados control de límites. Así que hay muchos que se quedan sin ser monitorizados.
- El comportamiento de la telemetría puede ser anómalo y estar dentro de sus límites. Con el control de límites esta anomalía pasaría desapercibida. Veremos después un ejemplo.
- A medida que pasa el tiempo, los componentes del satélite se van degradando y hay que ajustar los límites manualmente.

Nuestra solución para la detección de anomalías en operaciones espaciales

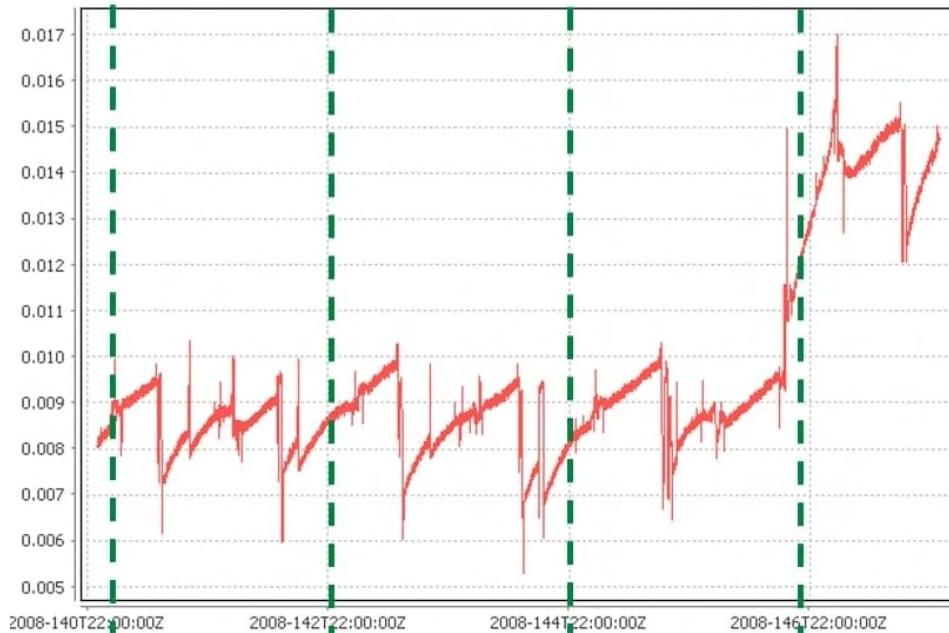
Os presentamos la solución que estamos usando ahora en el [Centro de Operaciones en la Agencia Espacial Europea](#). Es una «working solution» en el sentido de que funciona lo suficientemente bien para seguir usándola. Por supuesto, estamos abiertos a revisarla, mejorarla o usar otras soluciones que den resultados mejores a nuestros esfuerzos de monitorización automática de telemetría.

Nuestra solución usa un aprendizaje semi-supervisado. Es decir, damos ejemplos de comportamientos nominales, pero no damos ejemplos de comportamientos inusuales. Con estos datos se detectará qué comportamientos son raros y merecen la atención de los operadores. Utilizamos este enfoque para poder detectar comportamientos atípicos nuevos.

A continuación explico cómo lo hacemos. Puedes leer el artículo científico completo aquí: [Martínez-Heras, José-Antonio, and Alessandro Donati. «Enhanced telemetry monitoring with novelty detection.» *AI Magazine* 35, no. 4 \(2014\): 37-46.](#)

Preparación de datos

Los datos consisten en series temporales de miles de sensores. Para preparar los datos calculamos estadísticas simples en períodos de tiempo (un día normalmente). Los valores estadísticos que calculamos son: media, desviación típica, máximo y mínimo. De esta forma, cada periodo se puede representar como un punto de 4 dimensiones.



Particionamos una serie temporal en períodos y calculamos estadísticas para cada período

¿Por qué calculamos estas 4 dimensiones y no otras? ¿Por qué no calculamos otras características (*features*)?

Resulta que tenemos en mente usar técnicas semi-supervisadas basadas en distancias. Para este tipo de técnicas debemos intentar reducir al máximo el número de dimensiones para que la maldición de la dimensión nos afecte lo menos posible. Además resulta que tenemos un sistema que ofrece estas estadísticas ya precalculadas.

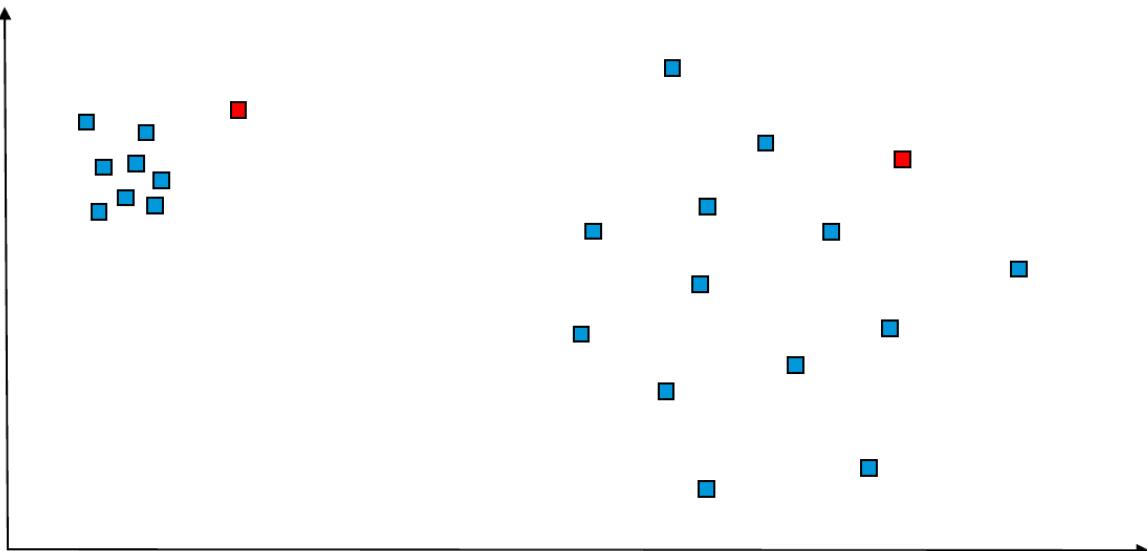
Intuición de detección de anomalías

Así que representamos cada periodo de tiempo de una serie temporal como un punto en 4 dimensiones. Para proporcionar una intuición de lo que vamos buscando, os traigo un ejemplo con 2 dimensiones.

En la siguiente figura, los puntos azules corresponden a comportamiento nominal. Lo sabemos porque los operadores del satélite en cuestión nos lo han comunicado. Podemos ver que el sensor de este ejemplo tiene dos modos de funcionamiento diferentes (izquierda y derecha) de la gráfica. Esto es bastante normal en espacio, donde los instrumentos pueden funcionar de formas diferentes según lo que estén haciendo (por ejemplo: apagado, encendido, estabilizándose, transmitiendo, etc.)

Calculamos dónde estaría el punto del periodo que nos interesa investigar. Normalmente las últimas 24 horas. Representamos este punto en rojo en la figura. Podemos ver que hay dos casos:

- El punto rojo de la izquierda corresponde claramente a un comportamiento raro, inusual. Este comportamiento debe ser investigado por si pudiera desembocar en una anomalía.
- El punto rojo de la derecha corresponde a un comportamiento nominal. De hecho, si le indicásemos a un operador que hemos detectado un comportamiento extraño, el operador no sabría a lo que nos referimos.



Intuición de la técnica que necesitamos: alerta en el ejemplo de la izquierda y todo normal a la derecha

Para dramatizar más este ejemplo, me he asegurado de que el punto rojo esté exactamente a la misma distancia del punto azul más cercano en los dos casos (izquierda y derecha).

Esto quiere decir que no podemos poner un umbral de distancia e informar a los operadores que si la distancia entre el punto nuevo y el más cercano conocido sobrepasa este umbral debe ser investigado. Esto provocaría un número muy alto de alarmas falsas o que no se detectaran comportamientos inusuales. Dicho de otra forma, cualquier umbral de distancia que escogiésemos sería el equivocado.

La intuición de por qué es un comportamiento raro en el caso de la izquierda y un comportamiento normal en el caso de la derecha (en la figura) estriba en la densidad.

En el caso de la izquierda, los puntos están muy cerca entre ellos. Así que si algún punto está un poco más lejos, ya podemos decir que es raro. En el caso de la derecha, los puntos están más dispersos. Así que necesitamos una distancia mayor para decir que el comportamiento es inusual.

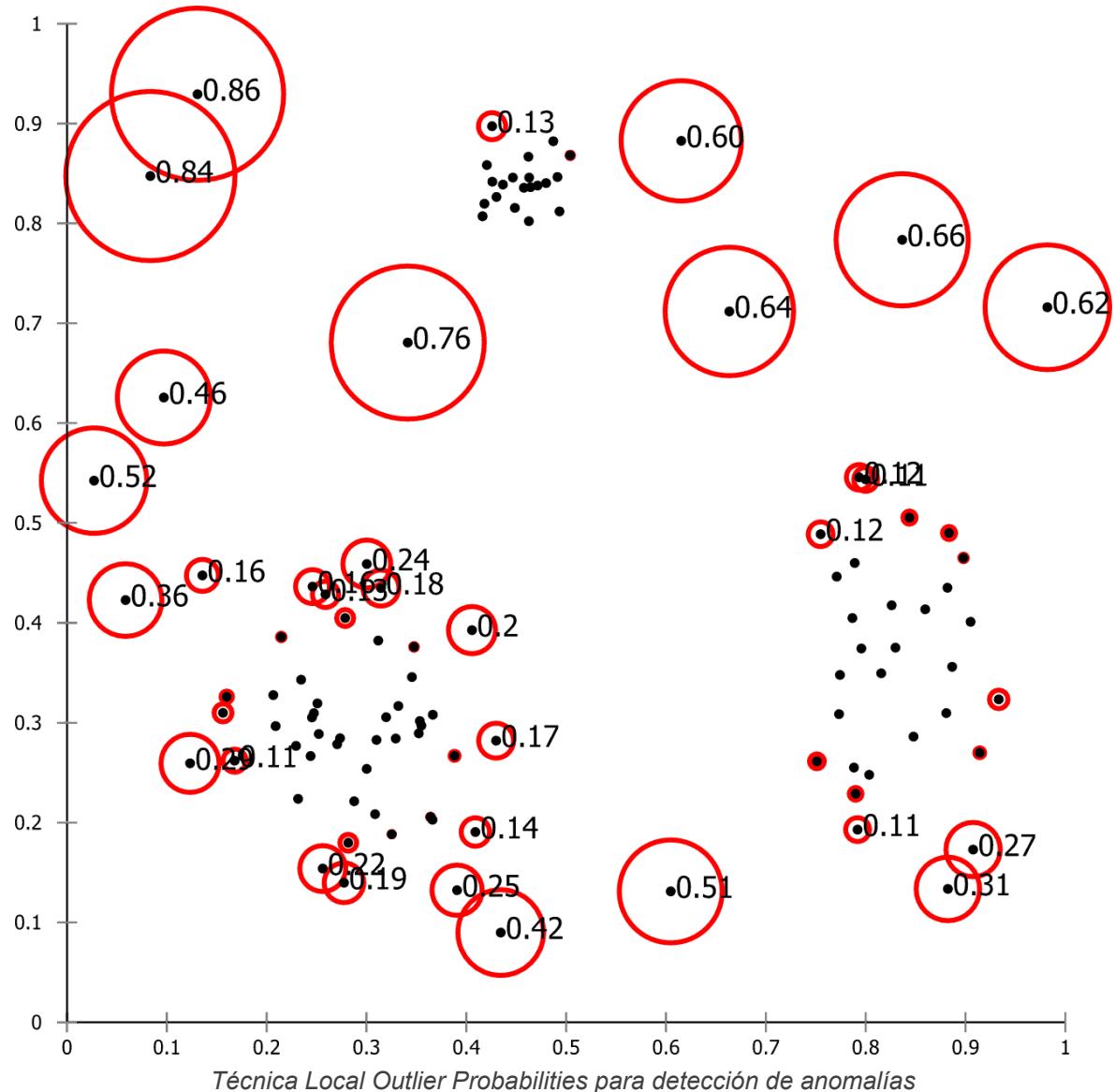
Algunas veces, cuando he explicado esta intuición me han preguntado si se podría solucionar con clustering. En principio, esto es diferente. No nos interesa saber que grupos hay, ni cuántos grupos hay, ni sabríamos qué hacer con la información de los grupos. No, no es clustering.

Local Outlier Probabilities

La técnica «[Local Outliers Probabilities](#)» implementa la intuición que teníamos en mente. Calcula cuál es la probabilidad de que un punto sea un «outlier». La traducción de *outlier* podría ser «*dato atípico*».

Para calcular la probabilidad de que un dato sea atípico, primero mira a los k datos más cercanos y calcula su densidad. Para zonas muy densas, una distancia pequeña bastará para tener una mayor probabilidad de comportamiento inusual. Para zonas dispersas, se necesitará una mayor distancia para una alta probabilidad de comportamiento raro.

En la figura vemos una intuición gráfica en 2 dimensiones. Los números que vemos se corresponden a la probabilidad de que un dato sea un *outlier*. Las probabilidades están en el rango [0, 1].



La técnica de *Local Outlier Factors* tiene muchas ventajas para nuestro problema de detección de anomalías:

1. **No asume ninguna distribución:** algunas técnicas de detección de outliers sólo funcionan si asumimos que los datos siguen una [distribución normal](#). Este no es nuestro caso. Muchos de los sensores que queremos monitorizar no siguen una distribución Gaussiana.
2. **Multimodal:** permite que los datos nominales ocurran en varias partes. Este es un comportamiento que también necesitamos porque muchos de los sensores toman valores en distintas regiones cuando los instrumentos están trabajando en modos diferentes.
3. **Adimensionalidad:** la probabilidad, al ser adimensional, nos permite comparar sensores que de otra forma no serían comparable entre ellos. Esto es posible porque lo que comparamos el grado de inusualidad. Así podemos comparar cómo de raro es el comportamiento de una temperatura con cómo de raro es el comportamiento de un voltaje, el uso de CPU, la presión en un tanque de combustible, etc.
4. **Pocas anomalías falsas:** al calcular las probabilidades basándose en la densidad, no tenemos que especificar ningún umbral de distancia. Esto hace que el número de anomalías falsas se reduzca dramáticamente.

Local Outlier Probabilities es una técnica de detección de *outliers*. La hemos adaptado para detectar comportamiento nuevo. La adaptación es simple: solo calculamos la probabilidad de comportamiento atípico en los datos nuevos.

«Salsa secreta» para reducir el número de anomalías falsas

Si usaras la técnica de *Local Outlier Probabilities* tal y como hemos descrito hasta ahora te iría bastante bien, pero todavía tendrías anomalías falsas. Así que ahora te revelamos la «salsa secreta» para reducir al mínimo el número de anomalías.

Estadísticas más estables

Cuanto más estables sean las estadísticas, más fácil es que tengas pocas anomalías falsas con relativamente pocos datos.

En algunos casos, elegimos para algunos satélites periodicidades de 90 – 100 minutos (que viene a ser una órbita para los satélites de observación de la Tierra). Pero teníamos varias anomalías falsas. Nos dimos cuenta que nos iba mucho mejor con períodos de 1 día.

Las estadísticas diarias son más estables. Además las estadísticas diarias capturan mejor la rutina en operaciones.

Esperar a que los datos estén completos

Como hemos indicado anteriormente, los datos los preparamos calculando valores estadísticos diarios. Si los datos no están completos, esto puede llevarnos a estadísticas incorrectas que pueden interpretarse como un dato atípico.

Los datos de telemetría no están disponibles inmediatamente. Hay que esperar a que el satélite se comunique con alguna estación de tierra. Aun así, puede que los datos que se descargen no estén completos. Así que habrá que esperar hasta la próxima oportunidad de comunicación hasta que los datos estén «consolidados»

Varios k para calcular la probabilidad de anomalía

Local Outlier Probabilities calcula las probabilidades estimando la densidad de los k puntos más cercanos. El valor de k es un poco arbitrario y podemos obtener valores diferentes probabilidades dependiendo del k que elijamos.

Queremos conseguir reducir al máximo posible el número de anomalías falsas. También queremos que los resultados sea obvios para los operadores. Así que hemos decidido que si un comportamiento es inusual, lo será independientemente del hiper-parámetro k que elijamos.

En la práctica, esto se traduce que calcular las probabilidades de que un dato sea atípico para varios valores de k (por ejemplo, $k=\{5, 10, 20, 30\}$). La probabilidad final será la probabilidad mínima.

Poca frecuencia en la detección de anomalías

Cada vez que intentemos detectar anomalías corremos el riesgo de detectar anomalías falsas. A riesgo de decir una obviedad, cuantas menos veces detectemos anomalías, menos oportunidades tendremos de encontrar anomalías falsas.

En nuestro caso detectamos anomalías una vez al día. Esto es suficiente porque el tipo de anomalías que vamos buscando son las anomalías que son difíciles de identificar con el control de límites (*out-of-limits*). El control de límites detecta las anomalías tan pronto como los datos están disponibles.

Añadir datos y, eventualmente, borrar los antiguos

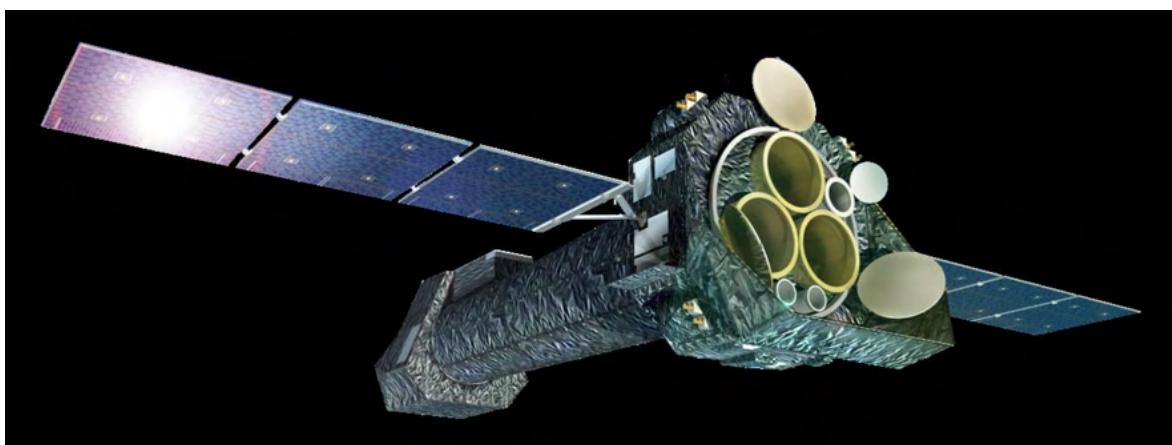
Si el número de anomalías falsas aumenta, probablemente sea hora de añadir como ejemplos de datos nominales datos nuevos. Por ejemplo, si hay un satélite que va camino de Marte, llega a Marte, habrá muchas cosas que cambien en la forma en la que es operado. Nos irá mejor

añadiendo ejemplos nominales orbitando alrededor de Marte y, eventualmente, eliminando los datos nominales correspondientes a la trayectoria.

Algunos ejemplos

Termistor en XMM-Newton

El **XMM-Newton** (X-ray Multi-mirror Mission – **Newton**) es un observatorio espacial de rayos X nombrado en honor de [Isaac Newton](#). XMM es el mayor satélite científico construido en Europa hasta el momento, pesa 3800 kg, mide 10 metros de largo y unos 16 metros de ancho con los paneles solares desplegados. Tiene tres telescopios de rayos X, cada uno con 58 espejos concéntricos, diseñados de manera que se maximiza su área colectora, focalizan los rayos X en las cámaras CCD de los detectores. Esto le hace capaz de detectar fuentes de rayos X extremadamente débiles. Fuente: [wikipedia](#).



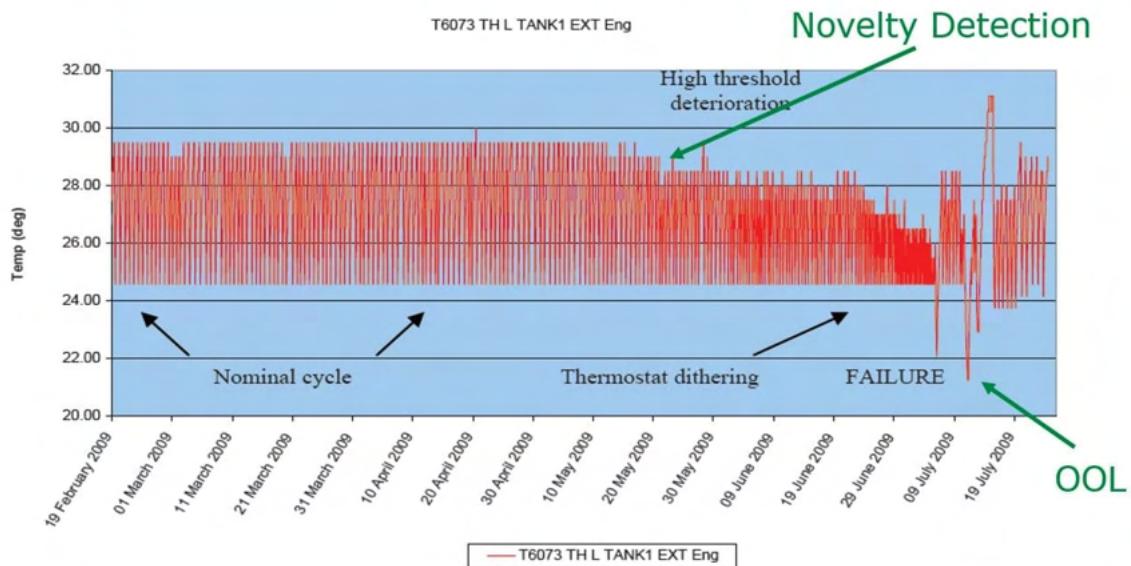
By National Aeronautics and Space Administration (NASA) – Published source': «Spacecraft Icons» at NASA ScienceDirect source: Image hosted by [smd-prod.s3.amazonaws.com](#), Public Domain, <https://commons.wikimedia.org/w/index.php?curid=60276937>

Muchos de los instrumentos de los satélites necesitan estar en unas condiciones de temperatura controladas para que funcionen correctamente. Esto se consigue usando termistores. Los termistores permiten calentar una parte del satélite cerrando un circuito cuando la temperatura baja de un cierto umbral inferior. Cuando la temperatura sube de un umbral superior, el termistor abre el circuito y deja de calentar.

A continuación vamos a ver una anomalía que ocurrió en el satélite XMM-Newton, descrita por el equipo de control de vuelo (la traducción al español y la negrita son mías):

«Notamos que el termistor T6073 empezó a tener un comportamiento extraño desde mediados de mayo de 2009, 2 meses antes que de la anomalía se detectara. El rango del termistor redució la temperatura cuando se abría y empezó a disminuir, un signo del deterioro del límite superior, incluso si el límite inferior era respetado, hasta mediados de Julio, cuando el límite superior y el inferior se situaron uno muy cerca del otro.

El termistor empezó a oscilar, en una rango estrecho de temperatura, hasta que no se cerró más a la temperatura correcta, y dejó que la temperatura bajara casi hasta los 22 grados. **La primera bajada de temperatura no fue detectada porque no generó ninguna alerta en el control de límites.** Después de esta bajada, el termistor tuvo algunos ciclos nominales, pero el 13 de julio de 2009, la temperatura bajó de nuevo hasta 21.25 grados, causando una alarma en el control de límites y permitiendo que el equipo de control de vuelo detectara el problema.»



Como vemos, nuestro sistema de detección de anomalías detectó la anomalía a finales de mayo, 2 meses antes de que el sistema de control de límites la detectara. Este ejemplo demuestra que puede haber comportamientos anómalos aunque ocurran dentro de los límites monitorizados. De hecho, en este caso la anomalía era que el comportamiento medido por el sensor estaba más en límites que en el caso nominal.

En el momento de la detección no hubiésemos sabido si las lecturas del sensor hubieran salido de límites en una semana o en 2 meses, como en este caso. Pero sí es suficiente para alertar al equipo de control de vuelo para que investiguen si esto representa un problema.

Seguramente al mirar esta figura, y este comportamiento se te ocurrán varias ideas simples para haber detectado este anomalía. Si te está pasando, no eres el primero. El reto consiste en encontrar un sistema de detección de anomalías que detecte todos los comportamiento inusuales, sin saber cómo van a ser, en cualquier tipo de sensor. Y con muy pocas anomalías falsas!

Otros ejemplos

El caso de XMM-Newton fue el primero donde probamos la eficacia de este método de detección de anomalías. Desde entonces lo hemos seguido usando en operaciones espaciales y tenemos bastantes más ejemplos de detecciones que ha resultado en investigaciones de comportamientos inusuales.

No los hemos publicado de forma oficial. Si alguna vez lo hacemos actualizaré este artículo.

Otras soluciones de detección de anomalías en espacio

Por supuesto, no somos los únicos en abordar el problema de la detección automática de anomalías en operaciones espaciales. Aquí tienes una lista de trabajos también muy interesantes en este campo:

- Fuertes, Sylvain, Gilles Picart, Jean-Yves Tourneret, Lotfi Chaari, André Ferrari, and Cédric Richard. «[Improving spacecraft health monitoring with automatic anomaly detection techniques.](#)» In *14th International Conference on Space Operations*, p. 2430. 2016.
- OMeara, Corey, Leonard Schlag, Luisa Faltenbacher, and Martin Wickler. «[ATHMoS: Automated telemetry health monitoring system at GSOC using outlier detection and supervised machine learning.](#)» In *14th International Conference on Space Operations*, p. 2347. 2016.
- Verzola, Ivano, Alessandro Donati, Jose Martinez, Matthias Schubert, and Laszlo Somodi. «[Project Sibyl: A Novelty Detection System for Human Spaceflight Operations.](#)» In *14th International Conference on Space Operations*, p. 2405. 2016.

- OMeara, Corey, Leonard Schlag, and Martin Wickler. «[Applications of deep learning neural networks to satellite telemetry monitoring.](#)» In *2018 SpaceOps Conference*, p. 2558. 2018.
- Hundman, Kyle, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. «[Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding.](#)» In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387-395. 2018.
- Trollope, Ed, Richard Dyer, Tiago Francisco, James Miller, Mauro Pagan Griso, and Alessandro Argemandy. «[Analysis of automated techniques for routine monitoring and contingency detection of in-flight LEO operations at EUMETSAT.](#)» In *2018 SpaceOps Conference*, p. 2532. 2018.
- Pilastre, Barbara, Loic Boussouf, Stéphane d'Escrivan, and Jean-Yves Tourneret. «[Anomaly detection in mixed telemetry data using a sparse representation and dictionary learning.](#)» *Signal Processing* 168 (2020): 107320.
- Y muchos más ...

¿Cuál funciona mejor?

Te estarás preguntando, ¿cuál funciona mejor?

La verdad es que es difícil saberlo. El «problema» reside en que hay muy pocas anomalías en realidad. Esto es estupendo para las operaciones espaciales, pero no está tan bien para aprendizaje automático.

Cuando hay tan pocas anomalías es muy difícil comparar el rendimiento de distintas soluciones. Por este motivo, estamos trabajando en la creación de un conjunto de datos de anomalías en operaciones espaciales, posiblemente combinando datos de varios satélites, para que se convierta en una especie de «benchmark dataset».

Recursos

- [vídeo] donde explico (en inglés) cómo funciona nuestra detección de anomalías entre otras cosas.
- El artículo donde explicamos nuestra solución: Martínez-Heras, José-Antonio, and Alessandro Donati. «[Enhanced telemetry monitoring with novelty detection.](#)» *AI Magazine* 35, no. 4 (2014): 37-46.
- El artículo que explica la técnica *Local Outliers Probabilities*: Kriegel, Hans-Peter, Peer Kröger, Erich Schubert, and Arthur Zimek. «[LoOP: local outlier probabilities.](#)» In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1649-1652. 2009.

Detección de anomalías en python

Si estás buscando cómo puedes detectar anomalías en python, te propongo que le eches un vistazo a estas tres librerías:

- scikit-learn tiene funcionalidades específicas para la detección de outliers y anomalías: [Novelty and Outlier Detection](#)
- [pyOD](#) es una librería que implementa 30+ métodos de detección de outliers y anomalías. Cada vez van añadiendo más.
- [pyNomaly](#) es una librería de python que implementa la técnica de Local Outlier Probabilities que usamos en nuestro sistema de detección de anomalías. En el momento de necesitarla, esta librería no estaba todavía disponible, así que usamos nuestra propia implementación.

Resumen

En este artículo hemos presentado los retos que tenemos que afrontar al intentar detectar anomalías automáticamente en operaciones espaciales y cómo los hemos afrontado. Así mismo, proporcionamos algunas recomendaciones (la «salsa secreta»).

También hemos visto un ejemplo de detección de anomalía con el satélite XMM-Newton y hemos listado otras soluciones usadas por diferentes operadores de satélites.

Por último hemos visto algunos recursos que incluyen cómo detectar anomalías usando librerías en python.

TorchServe para servir modelos de PyTorch

Por Álvaro Bartolomé del Canto
21/01/2021



TorchServe es el framework para servir modelos de ML desarrollado por PyTorch.

A lo largo de este repositorio podrás encontrar una guía sobre cómo entrenar y desplegar/servir un modelo de *transfer learning* basado en una red neuronal convolucional (CNN) con [ResNet](#) como *backbone*, cuyo objetivo es clasificar imágenes, del popular conjunto de datos [Food101](#), en categorías.

AVISO: TorchServe está aún en fase experimental y, por tanto, sujeto a cambios.

Requisitos

Antes de comenzar, tendrás que asegurarte de que tienes todas las dependencias necesarias instaladas o, en caso de no tenerlas, instalarlas.

Primero tienes que comprobar que tienes el JDK 11 de Java instalado, ya que es un requisito de `torchserve` a la hora de desplegar los modelos, ya que expone las APIs utilizando Java.

```
1. pip install torch==1.7.0+cpu torchvision==0.8.1+cpu -f
   https://download.pytorch.org/whl/torch_stable.html
2. pip install torchserve==0.2.0 torch-model-archiver==0.2.0
```

A continuación, puedes proceder con la instalación de los paquetes de Python necesarios tanto para entrenar como para servir el modelo de [PyTorch](#). De este modo, para instalarlo puedes utilizar el siguiente comando:

```
1. pip install torch==1.7.0+cpu torchvision==0.8.1+cpu -f
   https://download.pytorch.org/whl/torch_stable.html
2. pip install torchserve==0.2.0 torch-model-archiver==0.2.0
```

O bien puedes instalarlo desde el fichero de requisitos llamado `requirements.txt`, con el comando:

```
1. pip install -r requirements.txt
```

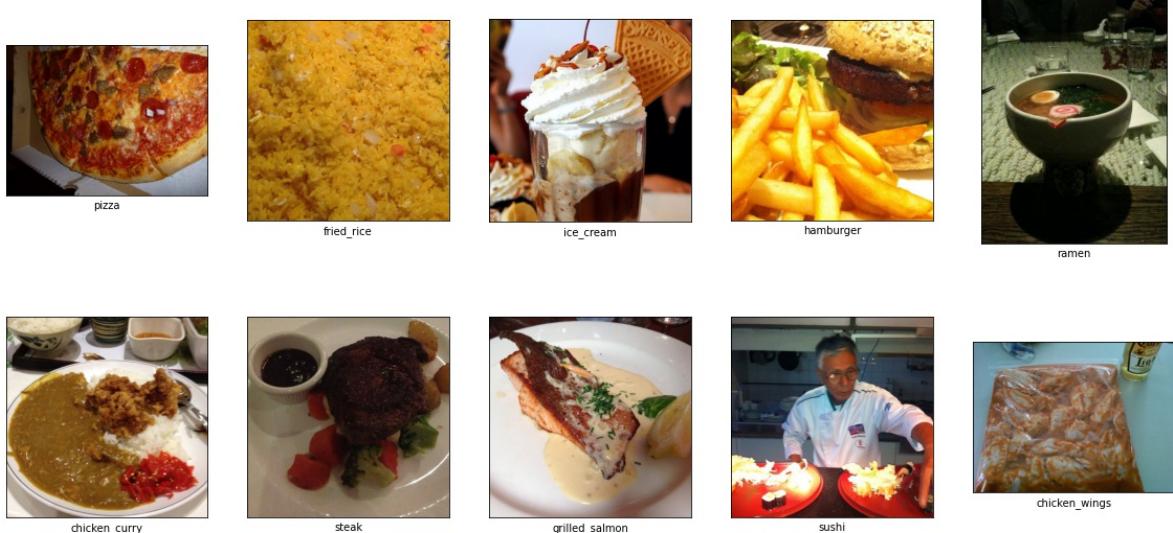
En caso de que tengas algún problema durante la instalación, visita [PyTorch – Get Started Locally](#).

Conjunto de Datos

El conjunto de datos a utilizar para el entrenamiento del modelo para la clasificación de imágenes en categorías es [Food101](#). En este caso, dado que esto es una guía, no se utilizará el conjunto de datos completo, sino que se utilizará un fragmento reducido del mismo, en este caso,

aproximadamente el 10% del total, abarcando tan solo 10 clases/categorías de las 101 disponibles.

El conjunto de datos original contiene imágenes de 101 categorías de comida distintas, con un total de 101.000 imágenes. Así, para cada clase, hay 750 imágenes para el entrenamiento y 250 imágenes para la evaluación del modelo. Las imágenes del conjunto de datos de evaluación han sido etiquetadas manualmente, mientras que en las de entrenamiento puede existir algo de ruido, principalmente en forma de imágenes con colores intensos o etiquetas erróneas. Por último mencionar que todas las imágenes han sido rescaladas con el fin de que tengan un tamaño máximo de 512 píxeles (bien de largo o bien de ancho).



Modelado

Comenzaremos con la creación de un modelo de *transfer learning* a partir del *backbone* [ResNet](#) con un conjunto pre-entrenado de pesos, entrenado y evaluado con el conjunto de datos [ImageNet](#), que es el estado del arte en lo que a clasificación de imágenes se refiere.

En este caso, queremos servir un modelo de PyTorch por lo que partiremos de [la implementación de ResNet de PyTorch](#) y, más concretamente, ResNet18, que es la implementación de ResNet que contiene 18 capas convolucionales.

Por tanto, cargaremos dicho modelo a partir de los pesos pre-entrenados desde el Hub de PyTorch y los congelaremos puesto que no nos interesa modificarlos dado que la idea del *transfer learning* es que ya se han ajustado para obtener el mejor resultado posible sobre el conjunto de datos ImageNet. Para cargar el modelo de PyTorch desde el Hub podemos utilizar el siguiente fragmento de código:

```
1.  from torchvision import models  
2.  
3.  model = models.resnet18(pretrained=True)  
4.  model.eval()  
5.  
6.  for param in model.parameters():  
7.      param.requires_grad = False
```

Una vez cargado el modelo, necesitamos actualizar la capa fc, cuyas siglas del inglés significan *Fully Connected*, y es la última capa del modelo que define las neuronas de salida. En este caso concreto, incluimos una capa secuencial que se añadirá tras las capas convolucionales del modelo original, dado que el objetivo es optimizar los pesos de dicha capa para obtener los mejores resultados sobre el conjunto de datos de evaluación que estamos utilizando. Así la capa secuencial incluida es la que se muestra en el siguiente bloque de código:

```

1. import torch.nn as nn
2.
3. sequential_layer = nn.Sequential(
4.     nn.Linear(model.fc.in_features, 128),
5.     nn.ReLU(),
6.     nn.Dropout(.2),
7.     nn.Linear(128, 10),
8.     nn.LogSoftmax(dim=1)
9. )
10.
11. model.fc = sequential_layer

```

Tras determinar la arquitectura de la red, se procederá a entrenar dicho modelo con el conjunto de datos de entrenamiento que contiene 750 imágenes y que ha sido dividido en dos sub-conjuntos, uno para el entrenamiento y uno para la validación con una separación del 80-20%, respectivamente. Dicha separación se realiza para poder estimar durante el entrenamiento del modelo cómo se comportará el modelo ante ejemplos no vistos previamente y, por tanto, para poder estimar cómo funcionará el modelo cuando se le pase el conjunto de prueba que contiene 2500 imágenes.

Nota: para más detalles en lo que a la creación y entrenamiento del modelo de *transfer learning* se refiere, puedes observar el código desarrollado en [notebooks/transfer-learning.ipynb](#) a modo de ejemplo.

Tras entrenar el modelo se procederá a exportar el modelo desde el *state_dict* a un fichero «.pth», el cual contiene el conjunto pre-entrenado de pesos que más adelante se podrá cargar de nuevo, con el código mostrado a continuación:

```
1. torch.save(model.state_dict(), '../model/foodnet_resnet18.pth')
```

Tras generar fichero exportable del modelo ya entrenado, por lo que tenemos a asegurarnos de que se ha exportado previamente, comprobando que se carga correctamente. Para poder realizar esta comprobación es importante que la arquitectura del modelo esté propiamente definida, dado que se requiere de la arquitectura de la red para poder cargar los pesos pre-entrenados sobre dicha red.

Puesto que hemos utilizado *transfer learning* a partir de un modelo para clasificación de imágenes pre-entrenado pero modificando tanto la última capa como ajustado los pesos de la misma, tenemos que modificar también la arquitectura original de ResNet18 definida en una clase de Python en [torchvision/models/segmentation](#). El código original de PyTorch del modelo ResNet18 es el siguiente:

```

1. def resnet18(pretrained: bool = False, progress: bool = True, **kwargs: Any) -> ResNet:
2.     r"""ResNet-18 model from
3.     `Deep Residual Learning for Image Recognition`_
4.     <https://arxiv.org/pdf/1512.03385.pdf>_"""
5.     Args:
6.         pretrained (bool): If True, returns a model pre-trained on ImageNet
7.         progress (bool): If True, displays a progress bar of the download
8.             to stderr
9.         """
10.
11.        return _resnet('resnet18', BasicBlock, [2, 2, 2, 2], pretrained,
12.                      progress,
13.                      **kwargs)

```

Que, traducido a la arquitectura completa de nuestro modelo, será de la forma:

```

1. import torch.nn as nn
2.
3. from torchvision.models.resnet import ResNet, BasicBlock
4.
5.
6. class ImageClassifier(ResNet):
7.     def __init__(self):
8.         super(ImageClassifier, self).__init__(BasicBlock, [2,2,2,2],
9.         num_classes=10)
10.
11.        self.fc = nn.Sequential(
12.            nn.Linear(512 * BasicBlock.expansion, 128),
13.            nn.ReLU(),
14.            nn.Dropout(.2),
15.            nn.Linear(128, 10),
16.            nn.LogSoftmax(dim=1)
17.    )

```

De este modo creamos una nueva clase de Python llamada **ImageClassifier** que hereda de la clase base de **ResNet** definida por PyTorch en **torchvision**. Tenemos que inicializar dicha clase con nuestra arquitectura, puesto que modificamos la última capa de la arquitectura de la red original ResNet18, pero antes hay que definir la arquitectura base de ResNet18 aunque modifiquemos el número clases de salida, que en este caso serán 10 clases de Food101 como se ha mencionado previamente.

Finalmente, tras definir la arquitectura base de **ResNet18**, procedemos a sobreescribir la capa **self.fc** con la definida previamente y sobre la cual hemos optimizado los pesos de la red para nuestro conjunto de datos. En este caso, la arquitectura resultante será la de ResNet pero con la última capa *Fully Connected* modificada con la capa secuencial pre definida.

Así ahora ya podremos comprobar si los pesos del modelo que hemos exportado se pueden cargar propiamente en la clase **ImageClassifier** con el siguiente fragmento de código:

```

1. model = ImageClassifier()
2. model.load_state_dict(torch.load("../model/foodnet_resnet18.pth"))

```

Así, la salida del fragmento de código anterior tras la carga del modelo, ha de ser <All keys matched successfully>, en caso de haber tenido éxito.

Puedes encontrar más modelos pre-entrenados de PyTorch para la clasificación de imágenes en [PyTorch Image Classification Models](#), y probar así distintos modelos de *transfer learning*.

Nota: el modelo ha sido entrenado con una tarjeta gráfica **NVIDIA GeForce GTX 1070 8GB GPU con CUDA 11**. En caso de no conocer los requisitos de la tarjeta gráfica de tu sistema, puedes utilizar el comando `nvidia-smi`, que también te indicará si los *drivers* de NVIDIA y CUDA están correctamente instalados. Además, para comprobar si PyTorch está haciendo uso de la GPU disponible en el sistema o no, puedes utilizar el fragmento de código presentado a continuación:

```

1. import torch
2. device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
3. torch.cuda.get_device_name(0)

```

Despliegue

Finalmente, de cara a desplegar el modelo necesitarás reproducir la siguiente secuencia de pasos tras haber instalado todos los requisitos previamente mencionados y disponer del modelo entrenado, tal y como se ha descrito previamente.

1. Generar el fichero MAR

Inicialmente se ha de generar el fichero MAR, que es un fichero listo para servir y que contiene el modelo completo generado con `torch-model-archiver` a partir del `state_dict` exportado previamente. Para generar así el fichero MAR tienes que utilizar el siguiente comando:

```
1. torch-model-archiver --model-name foodnet_resnet18 \
2.                               --version 1.0 \
3.                               --model-file model/model.py \
4.                               --serialized-file model/foodnet_resnet18.pth \
5.                               --handler model/handler.py \
6.                               --extra-files model/index_to_name.json
```

El significado de los flags de `torch-model-archiver` es el siguiente:

- `--model-name`: indica el nombre del modelo en formato MAR que vamos a generar.
- `--version`: se refiere a la versión del modelo, lo cual es una buena práctica a la hora de mantener los modelos puesto que se degradan con el tiempo.
- `--model-file`: contiene el fichero de Python que contiene la clase con la arquitectura del modelo que vamos a servir.
- `--serialized-file`: contiene el `state dict` con los pesos del modelo ya entrenado.
- `--handler`: especifica el cómo se van a manejar los datos en las llamadas a dicho modelo, por lo que incluye tanto el preprocesamiento como el postprocesamiento.
- `--extra-files`: dado que este es un problema de clasificación de imágenes, se puede incluir un fichero JSON que contenga las relaciones entre los IDs que asigna el modelo con los nombres de las categorías o etiquetas asignadas a cada uno de los IDs.

Por último, mencionar que no se requiere crear `handlers` personalizados puesto que los disponibles en TorchServeson bastante útiles, pero en caso de necesitar redefinir cualquiera de los procesos, preprocesamiento o postprocesamiento, se podrá crear uno personalizado como el presentado en este proyecto.

Una vez generado el fichero MAR, tienes que moverlo al directorio [`deployment/model-store`](#) que contendrá tanto este modelo como el resto de modelos puesto que a TorchServe se le indica el directorio sobre el cual ha de leer los modelos de PyTorch para servirlos más adelante.

```
1. mv foodnet_resnet18.mar deployment/model-store/
```

Puedes encontrar más información sobre `torch-model-archiver` en [Torch Model Archiver for TorchServe](#).

2. Desplegar TorchServe

Una vez que se haya generado un fichero MAR para servir, tan solo se necesita proceder con el despliegue de TorchServe. Así el proceso de servir un modelo pre-entrenado de PyTorch en formato MAR comienza con el despliegue de las APIs REST de TorchServe, que son las llamadas: *Inference API*, *Management API* y *Metrics API*, que se despliegan en el localhost o, lo que es lo mismo, la IP 127.0.0.1, en los puertos 8080, 8081 y 8082, respectivamente.

De este modo, el comando para desplegar TorchServe junto con el modelo MAR generado previamente, disponible en el directorio [`deployment/model-store/`](#), es el siguiente:

```
1. torchserve --start --ncs --ts-config deployment/config.properties --model-
   store deployment/model-store --models foodnet=foodnet_resnet18.mar
```

El significado de los flags de `torchserve` es el siguiente:

- `--start`: indica que el servicio de TorchServe se va a desplegar (es decir, las APIs).

- `--ncs`: indica que se desactivará el *snapshot* para no hacer una copia del contenido de la API, lo cual reduce los tiempos de despliegue, pero por seguridad se puede activar (sin poner dicho flag).
- `--ts-config`: especifica la configuración del despliegue a utilizar desde el fichero de configuración.
- `--model-store`: indica el directorio desde el cual se van a leer los ficheros MAR listos para ser servidos (expuestos como un *endpoint* de las APIs).
- `--models`: especifica los nombres de los modelos a utilizar, de modo que a cada uno de los modelos disponibles, bajo el directorio mencionado previamente, se les podrá asignar un alias que será a través del cual se creará el *endpoint* para acceder a dicho modelo desde las APIs REST. Sino, se utilizarán los nombres por defecto del fichero, pero lo recomendable es especificar manualmente el nombre de todos los modelos a servir con el formato: `endpoint=model_name.mar`.

Nota: otra forma de proceder en el despliegue consiste en desplegar primero TorchServe sin ningún modelo indicado en tiempo de despliegue y, en su defecto, registrar el modelo o modelos a través de la API de *Management* (que también permite gestionar los *workers* asignados a cada modelo entre otras cosas).

```
1. torchserve --start --ncs --ts-config deployment/config.properties --model-store deployment/model-store
2. curl -X POST "http://localhost:8081/models?initial_workers=1&synchronous=true&url=foodnet_resnet18.mar"
3. curl -X PUT "http://localhost:8081/models/foodnet?min_worker=3"
```

Puedes encontrar más información sobre `torchserve` en [TorchServe CLI](#).

3. Comprobar el estado de TorchServe

Para comprobar la disponibilidad de TorchServe tras el despliegue, puedes enviar una petición HTTP GET a la API para la inferencia desplegada por defecto en el puerto 8080 con el comando presentado a continuación. Para conocer el puerto o puertos en los que se ha desplegado cada una de las APIs, puedes comprobar el fichero [`config.properties`](#) que contiene dicha información sobre los servicios desplegados por TorchServe.

```
1. curl http://localhost:8080/ping
```

Si todo ha ido como se esperaba, debería de mostrar una salida similar a la siguiente:

```
1. {
2.   "status": "Healthy"
3. }
```

Nota: si el estado del *health check* es "Unhealthy", deberías de comprobar los *logs* de TorchServe para comprobar que el despliegue fue correctamente y, en caso de no haber ido bien, identificar el error e intentar resolverlo.

Al desplegar o intentar desplegar TorchServe, se creará automáticamente un directorio, desde donde se usó el comando, llamado `logs/`, donde poder comprobar si el despliegue ha ido como se esperaba.

4. Parar TorchServe

Una vez se «termine» de utilizar TorchServe con idea de no utilizarlo más, puedes pararlo de forma elegante con el siguiente comando:

```
1. torchserve --stop
```

Así la próxima vez que despliegues TorchServe, tardará menos tiempo puesto que en el primer despliegue, tanto los modelos especificados durante el despliegue como los modelos registrados

más adelante, serán cacheados, de modo que en los siguientes despliegues de TorchServe, dichos modelos no requerirán ser registrados de nuevo por estar ya en caché.

Docker

Con el fin de reproducir el despliegue de TorchServe, tal y como se ha descrito antes, en una imagen de Docker sobre Ubuntu, tendrás que asegurarte de tener Docker instalado en tu máquina y proceder con la ejecución de los comandos presentados a continuación:

```
1. docker build -t ubuntu-torchserve:latest deployment/
2. docker run --rm --name torchserve_docker \
   -p8080:8080 -p8081:8081 -p8082:8082 \
   ubuntu-torchserve:latest \
   torchserve --model-store /home/model-server/model-store/ --
   models foodnet=foodnet_resnet18.mar
```

Para más información en lo referente al despliegue desde Docker, puedes acudir a la documentación de TorchServe en la que se detalla el uso de Docker para el despliegue y notas adicionales disponible en [pytorch/serve/docker](#).

Uso

Una vez que has completado con éxito todos los pasos descritos previamente, puedes probar las APIs desplegadas por TorchServe enviando peticiones de ejemplo al modelo que está siendo servido. En este caso, dado que es un problema de clasificación de imágenes, se utilizará una imagen que se pueda englobar en alguna de las categorías sobre las que hace la inferencia el modelo. De este modo, una vez dispongamos de una imagen válida, podremos enviar la petición HTTP POST con el contenido de la imagen en el cuerpo de la petición de la forma:

```
1. wget https://raw.githubusercontent.com/alvarobartt/pytorch-model-
   serving/master/images/sample.jpg
2. curl -X POST http://localhost:8080/predictions/foodnet -T sample.jpg
```

Que, si todo ha ido bien, debería de devolver una salida en formato JSON como la que se muestra a continuación:

```
1. {
2.   "hamburger": 0.6911126375198364,
3.   "grilled_salmon": 0.11039528995752335,
4.   "pizza": 0.039219316095113754,
5.   "steak": 0.03642556071281433,
6.   "chicken_curry": 0.03306535258889198,
7.   "sushi": 0.028345594182610512,
8.   "chicken_wings": 0.027532529085874557,
9.   "fried_rice": 0.01296720840036869,
10.  "ice_cream": 0.012180349789559841,
11.  "ramen": 0.008756187744438648
12. }
```

Recuerda: el hecho de que la respuesta de la petición HTTP POST a TorchServe esté formateada con los nombres originales de cada una de las categorías de comida, se debe a que durante la creación del fichero MAR se especificó el índice al que correspondía cada categoría, en el fichero `index_to_name.json`. Así TorchServe realiza la asignación de índices a categorías de forma automática al responder a la petición a la API para la inferencia, de modo que es más clara.

Así los comandos presentados anteriormente se traducen en código de Python de la forma presentada en el siguiente bloque:

```
1. # Descarga una imagen de ejemplo de alvarobartt/pytorch-model-
   serving/images
2. import urllib
3. url, filename = ("https://raw.githubusercontent.com/alvarobartt/pytorch-
   model-serving/master/images/sample.jpg", "sample.jpg")
4. try: urllib.URLopener().retrieve(url, filename)
5. except: urllib.request.urlretrieve(url, filename)
6.
7. # Transforma la imagen en un objeto de bytes
8. import cv2
9. from PIL import Image
10. from io import BytesIO
11.
12. image = Image.fromarray(cv2.imread(filename))
13. image2bytes = BytesIO()
14. image.save(image2bytes, format="PNG")
15. image2bytes.seek(0)
16. image_as_bytes = image2bytes.read()
17.
18. # Envía la petición HTTP POST a TorchServe
19. import requests
20.
21. req = requests.post("http://localhost:8080/predictions/foodnet",
   data=image_as_bytes)
22. if req.status_code == 200: res = req.json()
```

Nota: en caso de querer ejecutar un *script* con el código proporcionado anteriormente, se requieren más requisitos de los mencionados previamente en la sección de requisitos, con lo que para instalarlos, puedes utilizar el siguiente comando:

```
1. pip install opencv-python pillow requests --upgrade
```

Contacto

Se puede encontrar todo el código fuente y los recursos mencionados en: [alvarobartt/serving-pytorch-models](#). Y para contactar directamente conmigo podéis realizarlo bien a través de [Twitter](#) o bien a través de [GitHub](#).

¡Agradezco mucho todos los *follows* y comentarios!

Créditos

Finalmente, mencionar que el fragmento del conjunto de datos utilizado ha sido realizado por **Daniel Bourke**, y algunas tips referentes a servir modelos de PyTorch basados en Transfer Learning con TorchServe descritas por **Prashant Sail**.

Algoritmos Genéticos y Memoria Visual

Por Alan Lopez
26/01/2021

Los algoritmos genéticos forman parte de la llamada computación evolutiva, que a su vez suele clasificarse dentro de las técnicas de Inteligencia Artificial. ¿Qué aplicaciones tienen los algoritmos genéticos? En este artículo aprenderás dos cosas: 1. Cómo se construye un algoritmo genético simple, y 2. Cómo aplicar un algoritmo genético en la creación de un mapa visual para la navegación de robots.

Computación Evolutiva y Algoritmos Genéticos

La Computación Evolutiva (CE) engloba diversos algoritmos que están inspirados en la teoría Neo-Darwiniana de la evolución natural. Ésta es vista como un proceso de optimización, en el cual los individuos de una población mejoran gradualmente adaptándose a su ambiente. En los algoritmos pertenecientes a la CE, un individuo es una solución potencial a un problema de optimización y el ambiente al que pertenece es la función(es) objetivo y sus restricciones. Este ambiente determinará la capacidad de supervivencia del individuo. Más adelante veremos un ejemplo simple para dejar esto más claro. Antes veamos algunos conceptos básicos para seguir avanzando.

Un algoritmo genético es una clase de algoritmo perteneciente a la Computación Evolutiva [David E. G., 1998]. Es un algoritmo de búsqueda basado en la mecánica de la selección natural. Estos algoritmos hacen evolucionar una población de individuos a través de acciones aleatorias semejantes a las que actúan según la teoría de la evolución biológica como mutaciones y recombinaciones genéticas. Además, hacen una selección de acuerdo con alguna función de aptitud que decide cuáles son los individuos más aptos que sobreviven, y cuáles los menos aptos que son descartados.

En un algoritmo genético, los individuos pueden ser codificados como cadenas binarias, que representan el cromosoma o genotipo del individuo. Por otra parte, el valor real al que codifica el genotipo es llamado fenotipo. Por ejemplo la cadena binaria 1010 sería el genotipo, mientras que el fenotipo sería 10.

En este artículo usaremos la codificación binaria, aunque no es la única forma de representar a un individuo. Existe también la codificación real, que implica que el fenotipo y genotipo coincide. En otras palabras, la codificación real del individuo con fenotipo 10, sería 10 también.

Son las cadenas binarias las que pasan por operaciones de recombinación genética y mutaciones. De forma más particular, un algoritmo genético simple ejecuta tres operaciones básicas:

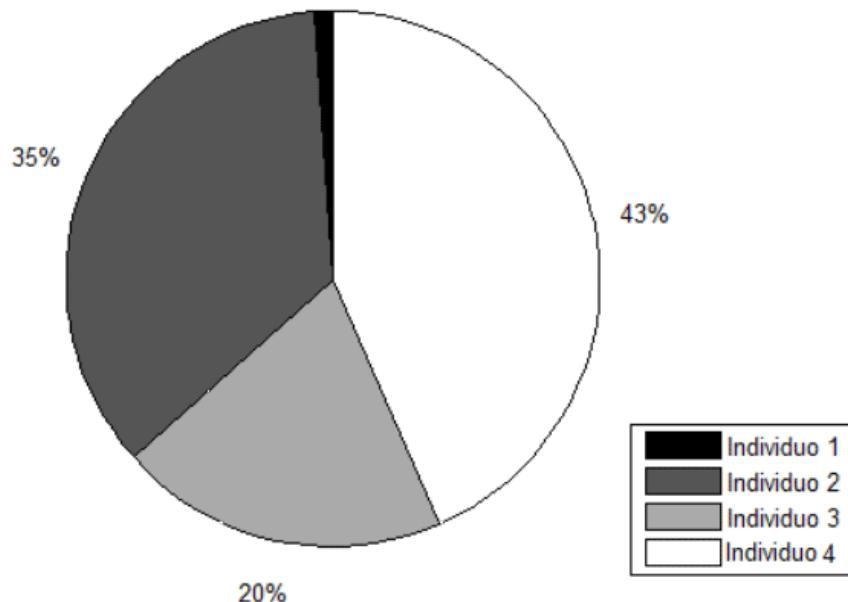
1. Reproducción
2. Cruza
3. Mutación

Reproducción

La reproducción es un proceso de selección de cromosomas para el posterior apareamiento e intercambio de genes, con base en el valor de su función de aptitud (la que indica qué tan apto es un individuo). Idealmente, se seleccionan con mayor probabilidad los individuos con una función de aptitud alta, logrando así una mayor probabilidad de contribuir con descendencia a la próxima generación. La función de aptitud determina entonces la supervivencia o muerte de algún individuo.

Uno de los métodos más usado para la selección de individuos es la llamada rueda de ruleta. Este método asigna un porcentaje de la ruleta, es decir una probabilidad, a cada individuo según su aptitud; donde el 100% de la ruleta es la suma de las aptitudes. Como ejemplo, observa la ruleta que se construye para una población de cuatro individuos con aptitudes: 25, 784, 441 y 961.

Ruleta para la población (Porcentaje de las funciones de aptitud)



Cruza

La cruza es un operador que lleva a cabo el intercambio de información genética de dos individuos de la población, a los que llamaremos padres, en el cual se combinan sus genes, que son los bits de las cadenas binarias, para generar descendencia o hijos. Toma por ejemplo la cruza de estas dos cadenas binarias.

$$\begin{array}{r} 10 | 10 \\ \Rightarrow 1000 \\ 11 | 00 \end{array}$$

Mutación

La mutación modifica bits de la cadena binaria de forma aleatoria con cierta probabilidad. Una mutación se vería así:

$$1000 \Rightarrow 1001$$

El objetivo de la mutación es generar nuevos individuos o hijos que formarán parte de la nueva población. Este operador permite la variabilidad en la población, con la finalidad de no quedar estancado en un máximo o mínimo local. Eso contribuye a la exploración de todo el espacio de búsqueda evitando así la convergencia prematura.

Los distintos algoritmos genéticos que se pueden formular responden a un esquema básico común, y comparten una serie de propiedades:

- Procesan simultáneamente, no una solución al problema, sino todo un conjunto de ellas. Estos algoritmos trabajan con una codificación de soluciones potenciales al problema, que se denominan individuos o cromosomas. El conjunto de todos ellos forman la población con la que trabaja el algoritmo.
- La composición de la población se va modificando a lo largo de las iteraciones del algoritmo que se denominan generaciones. De generación en generación, además de variar el número de copias de un mismo individuo en la población, también pueden aparecer nuevos individuos generados mediante operaciones de transformación sobre individuos de la

población anterior. Dichas operaciones se conocen como operadores genéticos, que ya han sido descritos.

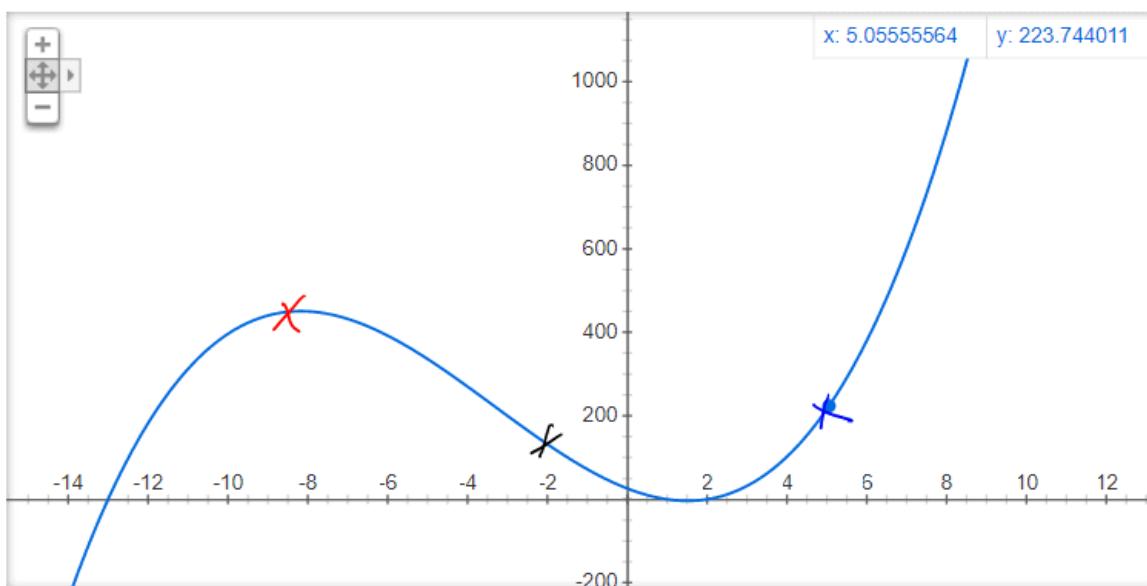
- Cada generación incluye un proceso de selección, que da mayor probabilidad de permanecer en la población y participar en las operaciones de reproducción a los mejores individuos. Los mejores individuos son aquellos que dan lugar a los mejores valores de la función de aptitud.

Es fundamental para el funcionamiento de un algoritmo genético que este proceso de selección tenga una componente aleatoria, de forma que individuos con bajo valor de aptitud también tengan oportunidades de sobrevivir, aunque la probabilidad asociada a éstos sea menor. Es esta componente aleatoria la que dota a los algoritmos genéticos de capacidad para escapar de óptimos locales y de explorar distintas zonas del espacio de búsqueda.

Requisitos para resolver un problema usando Algoritmos Genéticos

Para resolver un problema con algoritmos genéticos, el problema debe poder plantearse como un problema de optimización. Esto es, dada una función matemática se deben encontrar el o los parámetros que hagan que la función tenga su máximo (o mínimo) valor.

Observa la gráfica para la función $f(x) = x^3 + 10x^2 - 37x + 26$. En este caso, x es el único parámetro. El problema consiste en encontrar el valor de x que optimice (maximice) la función.



La gráfica representa los distintos valores de la función de aptitud también llamada función objetivo. Como puedes observar, de los tres valores señalados, el punto rojo tiene mejor función de aptitud.

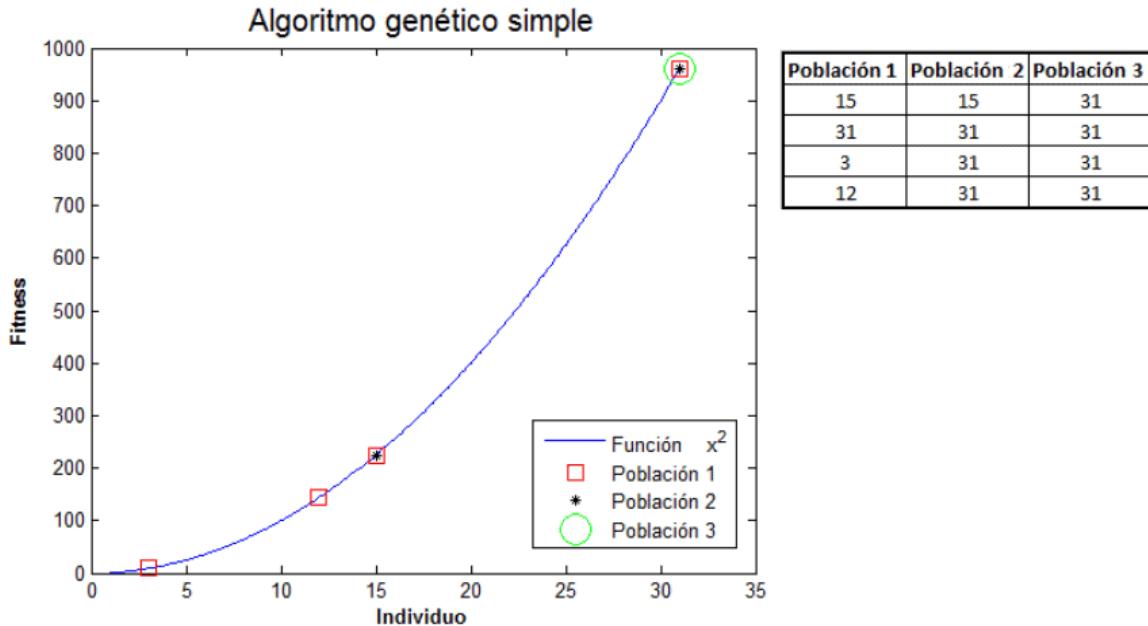
Ejemplo de un Algoritmo Genético Simple

Consideremos el problema de maximizar la función $f(x) = x^2$, donde x tiene la restricción de variar entre 0 y 31. Se codifican los individuos con una cadena de 5 elementos (bits) siendo 00000 = 0 y 11111 = 31. Para empezar se elige aleatoriamente una población de 4 individuos.

El algoritmo elige de forma aleatoria una población inicial con individuos: 15, 31, 3 y 12, cuya codificación, valor de aptitud y porcentaje en la ruleta se muestran en la siguiente tabla.

Individuo	Cadena / Cromosoma	Aptitud	% en ruleta
1	01111	225	16.8
2	11111	961	71.76
3	00011	9	0.67
4	01100	144	10.77

Observa una simulación de 4 iteraciones o generaciones.



En la primera iteración del algoritmo se encuentra una nueva población con individuos: 15, 31, 31 y 31. Se observa que la aptitud máxima se mantuvo y que los mejores individuos fueron seleccionados para la cría. La gráfica anterior muestra la evolución de la población, en cada iteración de los algoritmos de reproducción y cría la función de aptitud mejoró en términos generales. Sin embargo, al tratarse de un algoritmo basado en la probabilidad, la población no siempre converge al máximo en la tercera iteración. Es posible que sean necesarias más iteraciones para ciertas funciones objetivo.

Aplicación de Algoritmos Genéticos: Construcción de una Memoria Visual para la navegación autónoma de robots

El problema de navegación de robots consiste en el desplazamiento seguro de éstos dentro de un ambiente, sea éste estructurado o no. Para lograr la navegación autónoma tres problemas diferentes debes ser resueltos:

1. **Mapeo.** Determinación de una representación eficiente y adecuada del ambiente a navegar. Esto a través de un mapa que describa correctamente el espacio de navegación.
2. **Auto-localización.** Es necesario que el robot logre localizarse eficientemente en el mapa.
3. **Navegación.** Diseño de leyes de control. Debe ser posible el desplazamiento del robot de un punto A a un punto B con leyes de control adecuadas.

Resolvamos el primer problema relacionado con el mapeo del ambiente de navegación. Este trabajo fue publicado en *The International Symposium on Optomechatronic Technology (ISOT)* [López-Martínez, A., 2019].

En la teoría, los tipos de mapas pueden ser divididos en dos familias: las geométricas y las topológicas.

En las geométricas, el espacio de navegación se representa en un marco de referencia euclíadiano; se tiene información de las medidas métricas del ambiente de navegación.

En las representaciones topológicas, por otra parte, el ambiente es descrito de forma cualitativa, es decir, un mapa sin información métrica del ambiente. En esta parte del artículo, te mostraré la construcción de un mapa topológico en la forma de una memoria visual.

Memoria Visual

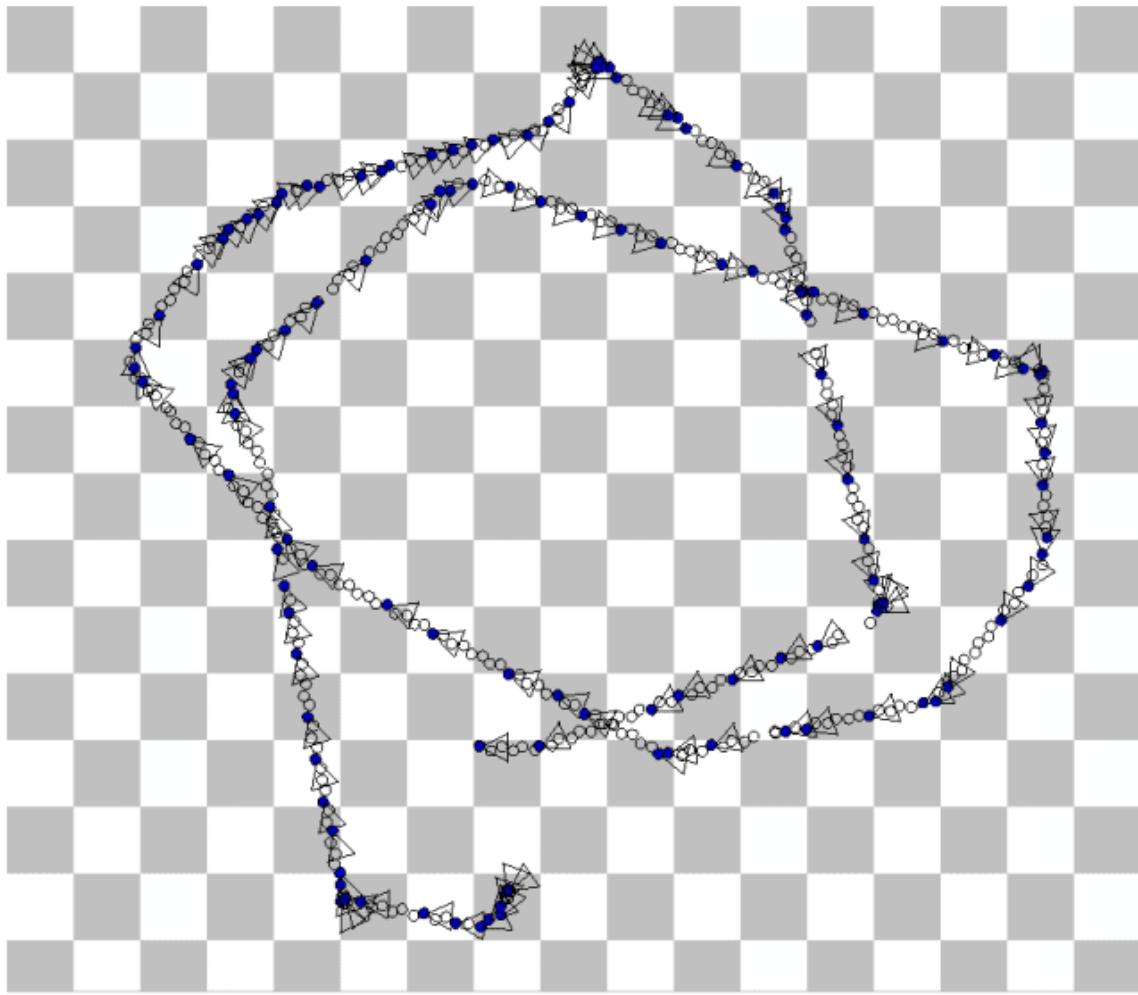
Una memoria visual es un mapa topológico que usa solamente información visual [Delfin, J., 2018]. El sensor es una cámara proyectiva a bordo del robot, y las trayectorias por las cuales el robot tiene permitido navegar se describen con un conjunto de imágenes (imágenes clave) que caracterizan o describen el ambiente. A esta colección de imágenes la llamamos memoria visual.

¿Por qué una Memoria Visual?

La motivación es simular la forma en que humanos y animales guardamos la información de lugares nuevos [Van Veen H., 1998]. Pongamos un ejemplo: Imagina que exploras un espacio nuevo. ¿Cómo guardas la información? De ninguna manera lo haces de forma métrica. Es decir, no sabes exactamente la distancia en metros de la puerta A a la puerta B, ¿cierto? Más bien, guardas información topológica o cualitativa. En otras palabras, guardas la relación de la posición de cada elemento de interés; la puerta A está antes de la puerta B.

Un ejemplo común de mapa topológico son los mapas de las estaciones de metro. Cuando ves el mapa, no observas información métrica con la distancia entre estaciones. Más bien, obtienes información sobre la posición relativa entre estaciones. Sin embargo, esta información parcial no te impide navegar ya sea dentro del espacio nuevo que has explorado o dentro del sistema del metro.

Comparemos el mapa de metro con la memoria visual. En la memoria visual cada imagen clave es como una estación de metro. En la siguiente imagen, cada círculo representa distintos frames del video capturado por la cámara del robot. Un círculo azul indica una imagen clave dentro de la memoria visual.



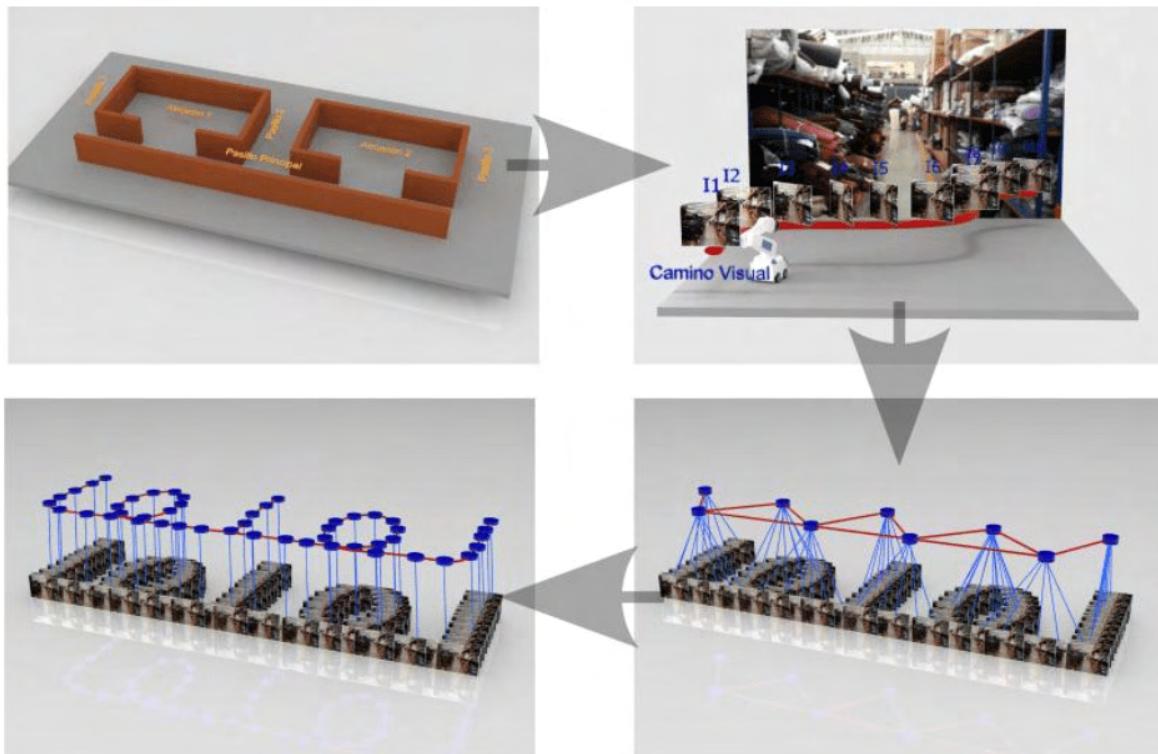
Una ventaja de este tipo de mapas topológicos está en la cantidad de información necesaria para navegar. Es mucho menos que la información de mapas 3D por ejemplo. Entre menos datos para procesar, mejor eficiencia de procesamiento, y de almacenamiento de información. Existen desventajas claro está ¿puedes pensar en algunas? Comenta este artículo.

¿Cómo construir una memoria visual?

La metodología para la construcción de la memoria visual se divide en tres etapas: la primera etapa inicia con una fase de aprendizaje, en ésta el robot es conducido por un humano a través del entorno, mientras el robot captura imágenes de entrenamiento.

Después, este conjunto de imágenes se procesa en la etapa dos, que busca reducir el número de imágenes que conformarán la MV. Para lograr tal reducción, se seleccionan sólo algunas de las imágenes del conjunto, que son llamadas imágenes clave (Los círculos azules que viste en la imagen anterior). Para ello se analiza la similitud entre imágenes, o su traslape, y se desechan las imágenes que no cumplen ciertos criterios. Tal selección también toma en cuenta el cumplimiento de algunas hipótesis necesarias para el control del robot durante la navegación. Es en esta etapa donde usaremos el algoritmo genético.

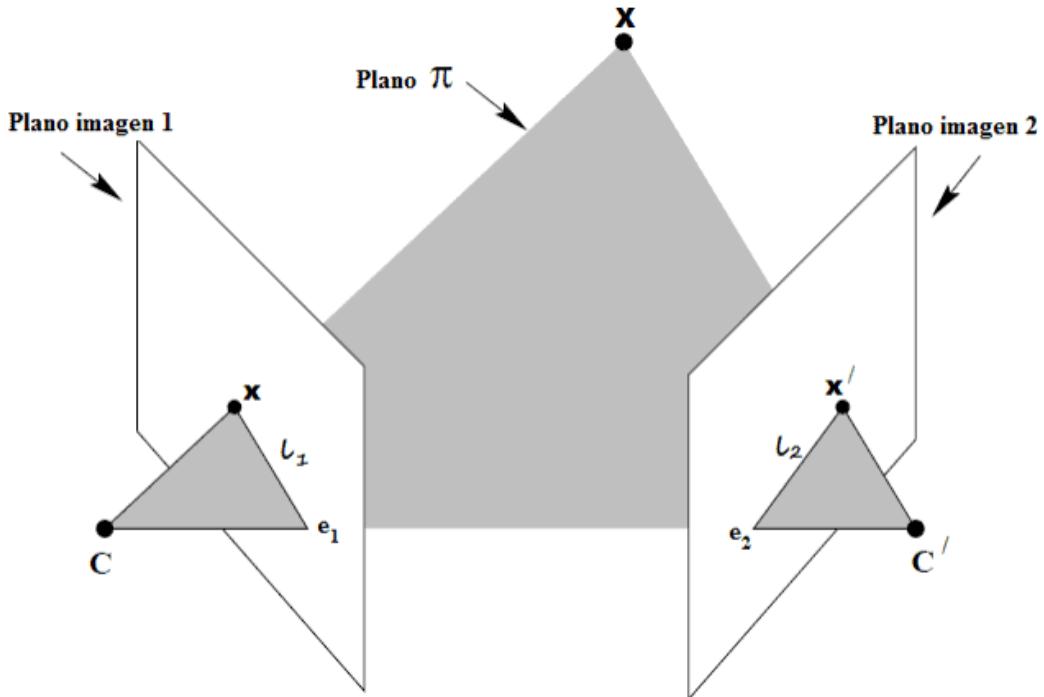
Una vez que se tiene un conjunto de imágenes clave, el cual denominamos conjunto M1, se organizan las imágenes pertenecientes a éste para formar una representación topológica del ambiente de navegación. Esto último ocurre en la etapa tres de la construcción de la memoria visual. La siguiente imagen ilustra estos pasos.



Construcción de una Memoria Visual con Algoritmos Genéticos

En esta sección nos concentraremos en la selección de imágenes clave. Queremos que en el mapa visual tengamos únicamente información relevante. Entonces, no debería haber imágenes similares. Para lograr esto, seleccionamos los frames del video que tengan solo un porcentaje de similitud o empalme. Cuando el robot haya avanzado cierta distancia, la cámara capturará información nueva. Seleccionamos una imagen clave nueva cuando tengamos un porcentaje previamente definido de información nueva. ¿Cómo logramos hacer esto con un algoritmo genético?

La clave está en la geometría epipolar [Zisserman, R. H. A., 2004]. Esta geometría describe la rotación y traslación de la cámara. Se pueden conocer estos movimientos cuando una imagen observa un punto, y después se mueve (rotación o traslación) un poco sin dejar de observar el mismo punto.



Para usar un algoritmo genético haremos lo siguiente. Cada individuo codificará una rotación y traslación candidata.

$$\text{Ind} = [\varphi, \theta, \psi, tx, ty],$$

donde

- $\varphi \in [0, 2\pi]$
- $\theta \in [0, 2\pi]$
- $\psi \in [0, 2\pi]$
- $tx \in [-5, 5]$
- $ty \in [-5, 5]$

Para calcular la aptitud de cada individuo se calcula qué tan bien describe la rotación y traslación real. ¿Cómo sabemos cuál fue el movimiento real? Se deduce indirectamente con puntos emparejados entre imágenes. Si la rotación y traslación codificada por el individuo explica correctamente la posición de cada punto en la imagen, entonces tendrá una buena aptitud. Desde luego que necesitamos un valor numérico para representar la aptitud, entonces se contará la cantidad de puntos que concuerdan con la rotación y traslación propuesta por el individuo.



Todos los individuos pasarán un número determinado de generaciones, y serán modificados con los operadores de selección, cruce y mutación. Al final, se escoge al mejor individuo como la solución final. Conocer cuál fue la rotación y traslación de la cámara permite saber indirectamente la cantidad de puntos nuevos en cada frame, y en consecuencia decidir qué imágenes aportan más información para la memoria visual.

Cuando te mostré el ejemplo simple con la función $f(x) = x^2$ o la función $f(x) = x^3 + 10x^2 - 37x + 26$, tal vez pensaste ¿para qué usar un algoritmo genético? Bueno para estos ejemplos sí que era innecesario. Sin embargo, para el problema de la selección de imágenes clave, ni siquiera podemos graficar la función de aptitud. En este caso, los algoritmos genéticos son de ayuda.

La anterior es una explicación de muy alto nivel de cómo usar un algoritmo genético para la construcción de un mapa visual topológico.

Sobre el autor



Alan es doctor en ciencias, y coordinador del área dedicada a la investigación e implementación del Machine Learning y Ciencia de

Datos en una empresa de consultoría TI. Le encanta aprender y enseñar sobre cómo lograr sistemas de aprendizaje de máquina prácticos y funcionales. Blog personal:
MachineLearningEnEspanol.com

Otros artículos del autor

- [Cómo es trabajar de Científico de Datos: Ejercicios Prácticos de Machine Learning](#)
- [Cómo empezar en Machine Learning y Encontrar Trabajo este 2021](#)
- [Cómo elegir un curso de Machine Learning con Python en Español en 2021](#)

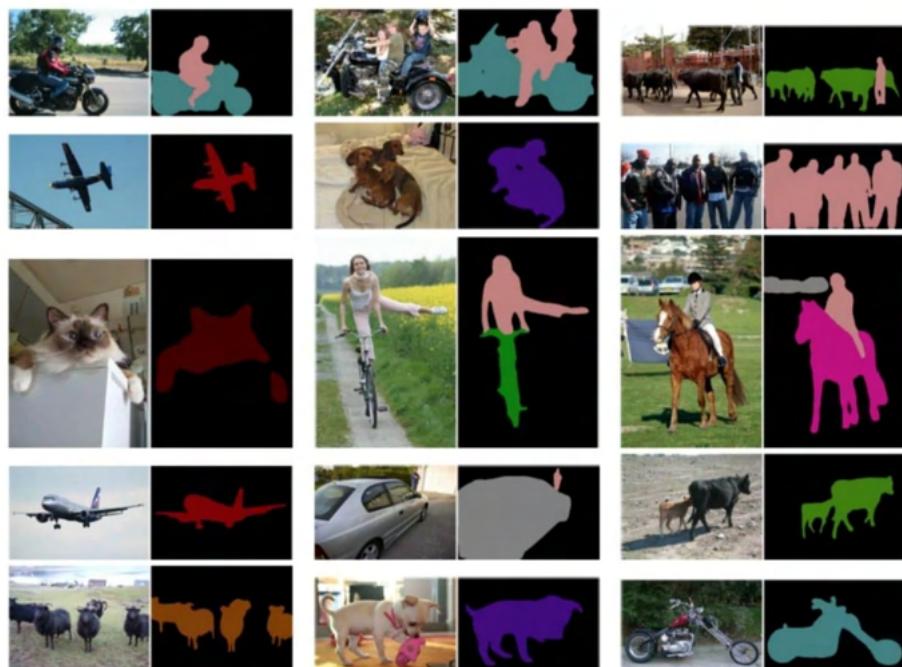
Referencias

1. López-Martínez, A., Cuevas, F. J., & Sosa-Balderas, J. V. (2019). Visual Memory Construction for Autonomous Humanoid Robot Navigation. In *Progress in Optomechatronic Technologies* (pp. 103-109). Springer, Singapore. Este artículo describe lo presentado en este post.
2. David E. G. (1998). Genetic Algorithms in search, optimization and machine learning. pp 1-25. Este libro fue escrito por el inventor de los algoritmos genéticos, ilustra a mayor detalle cómo funcionan.
3. Van Veen H., Distler H., Braun S., and Bulthoff H. (1998). Navigating through a virtual city: Using virtual reality technology to study human action and perception. Journal, Future Generation Computer Systems. Vol. 14. No. 3-4, pp 231-242. El artículo muestra cómo navegamos por un ambiente nuevo los humanos.
4. Delfin, J., Becerra, H. M., & Arechavaleta, G. (2018). Humanoid navigation using a visual memory with obstacle avoidance. *Robotics and Autonomous Systems*, 109, 109-124. En este artículo se explica cómo generar las leyes de control adecuadas para permitir la navegación autónoma dentro de una memoria visual con un robot humanoide Nao.
5. Zisserman, R. H. A. (2004). Multiple view geometry in computer vision. Este libro muestra cómo usar información visual (a partir de imágenes) para diversas aplicaciones como structure from motion, reconstrucción 3D o entendimiento de la escena.

Segmentación de Imágenes con Redes Convolucionales

Por Daniel Iglesias
18/04/2021

En este post vamos a hablar sobre un área muy influenciada por la inteligencia artificial como es la visión artificial, concretamente en lo que se denomina como segmentación de imágenes, para introducir posteriormente el uso del Deep Learning y, de forma más específica, las redes convolucionales y su potencial uso en este ámbito.



Segmentación de imágenes (Chen et. al, 2017)

La segmentación de imágenes, en su sentido más estricto y convencional, no tiene por qué recurrir a ninguna estrategia de aprendizaje automático. Sin embargo, en la actualidad, debido al enorme auge del Deep Learning, la segmentación de imágenes ha mejorado notablemente gracias a la utilización de arquitecturas de redes profundas como es el caso de las redes convolucionales, muy utilizadas en el ámbito de análisis de imágenes.

Concretamente, hablaremos sobre qué es la segmentación y de su enorme importancia en la visión artificial, a continuación, resumiremos brevemente algunos conceptos de redes neuronales y las ideas fundamentales del Deep Learning, para más tarde pasar a hablar de las redes convolucionales, sus ventajas frente a otras arquitecturas de red y de cómo se pueden utilizar en el contexto de la segmentación.

¿Qué es la segmentación de imágenes?

Imagínate que tienes un conjunto de imágenes del fondo del mar con peces y, suponiendo que saber el número de ejemplares que hay en una zona es relevante para un estudio científico, quieras obtener un método automático que te permita contarlos.



Imagen de Pixabay

Esto nos ahorraría muchísimo tiempo en el caso de tener, por ejemplo, 500 imágenes. En cada imagen, la cuenta nos podría llevar 2 minutos, por poner un tiempo razonable. Ahora imagínate escalar eso a 2 minutos por 500 imágenes. Probablemente te llevaría días e incluso semanas, porque una tarea tan monótona te acabaría agotando y no podrías hacer todo de una tirada.

Para conseguir esto, las estrategias de visión artificial son, sin ninguna duda, lo más apropiado para el problema. Pero... ¿cómo podríamos conseguir resolver este problema tan concreto? Teniendo en cuenta que en una imagen del fondo marino hay muchas cosas, para el método de visión no sería trivial encontrar los peces. Para ello, necesitaría eliminar todos los datos que no sean relevantes y centrarse exclusivamente en aquello que sea un pez.

Esto se puede conseguir por medio de la segmentación. La segmentación es básicamente obtener la silueta de los objetos de interés que queramos extraer en una imagen. En el caso de este ejemplo, estaríamos buscando obtener la silueta de los peces. Al centrarnos exclusivamente en la silueta de los peces, estaríamos eliminando todo aquello que no fuese relevante y que, de este modo, estaría introduciendo ruido al sistema.

Dificultades de la segmentación



Ejemplo de segmentación manual de los peces en las imágenes. Las siluetas blancas se corresponden con los píxeles que caen dentro de un pez.

La segmentación no es un proceso para nada sencillo, en ninguna de sus partes, es decir, ni en la de elegir el mejor método de segmentación ni a la hora de validar. De hecho, hemos hecho manualmente esa segmentación manual que se muestra más arriba y nos hemos dado cuenta de algunos problemas importantes que se pueden dar a la hora de analizar este tipo de imágenes, unas dificultades que podrían hacer incluso inviable el conteo preciso de los peces.

Observamos que estas imágenes se caracterizan por tener a muchos peces que se superponen sobre otros, lo cual hace difícil hacerle entender al método si ahí hay un pez o hay varios. Del mismo modo, hay peces que quedan parcialmente ocultos por algún elemento submarino, haciendo todavía más difícil ese conteo preciso.

Probablemente dirás «qué ejemplo tan mal escogido». Pues lo cierto es que nos ha parecido interesante, ya que representa muy bien los problemas a los que nos enfrentamos cuando queremos desarrollar un sistema de visión artificial. A menudo, nos encontramos ejemplos como un caballo en un campo, en una imagen muy bien hecha, con buen contraste y con el caballo bien encuadrado. Pero lo cierto es que, en la vida real, debemos prepararnos para cualquier problema que pueda surgir.

¿Qué métodos de segmentación existen en la literatura?

Existe una multitud de métodos de segmentación. Algunos de los más destacables son la umbralización, el algoritmo split and merge, los algoritmos de relleno de regiones, el algoritmo de watershed o los modelos deformables. Cada uno de ellos, funcionará bien o no en un problema concreto. La única forma de conocerlo es validándolo de una manera conveniente.

Cabe destacar que los principales métodos de segmentación de imágenes se pueden encontrar ya implementados en librerías como [OpenCV](#) o [Scikit-Image](#), las cuales están disponibles para

Python y son muy fácilmente instalables con sistemas de gestión de paquetes como es el caso de pip.



Ejemplo de segmentación de objetos en una imagen con un método de umbralización.
Fuente: [¿Cómo umbralizar una imagen con OpenCV en Python?](#) – AprendiendoInformatica.net

¿Cómo se valida y por qué es tan complicado?

Para validar la segmentación, tiene que haber una segmentación de referencia, lo que se denomina generalmente como un ground truth. Este ground truth se tiene que obtener de forma manual, pintando la silueta de los objetos de interés con un color uniforme. Pensarás que esto es un proceso muy tedioso... y efectivamente, lo es.

A menudo, la obtención de estos ground truths se denomina «etiquetado manual de imágenes» o, en este caso, «segmentación manual de imágenes» y, dado que es un proceso tan tedioso, existe una falta de datos de este tipo. Es muy común que, por ejemplo, de esas 500 imágenes mencionadas solo se hayan etiquetado 50. Esto es un grave problema para las redes de Deep Learning, como veremos más adelante.

Para que te hagas a la idea, nosotros hemos segmentado manualmente la imagen que se mostraba más arriba de una forma bastante burda y ya nos ha resultado bastante tedioso. Además, estamos seguros de que, durante el proceso, se nos ha escapado algún pez, que probablemente caiga en una zona de muy bajo brillo. Nuevamente, esto es significativo de lo complicado que resulta segmentar manualmente imágenes y, sobre todo, lo tedioso que llegaría a ser en caso de que no fuera 1 si no muchísimas más.

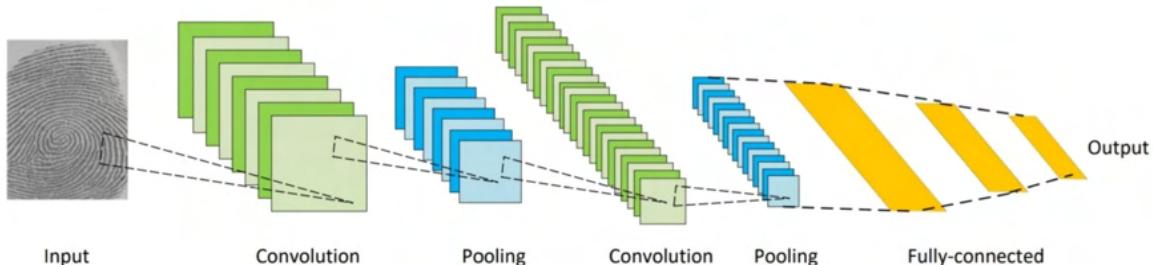
El caso es que en el hipotético caso de que tengamos un suficiente número de imágenes etiquetadas, este ground truth será comparado con la segmentación realizada por el método de visión artificial, por medio de una serie de métricas que evaluarán la precisión con la que el sistema segmenta.

La segmentación, el machine learning y la importancia de las redes convolucionales

Como en cualquier paso de la visión artificial, el machine learning puede ser utilizado como alternativa a otras estrategias más clásicas como las anteriormente mencionadas (umbralización, growing regions, modelos deformables...) en el contexto de la segmentación de imágenes.

Además, el Deep Learning se ha presentado como una solución mucho más óptima e inmediata para realizar la segmentación de una imagen. En visión artificial, concretamente, una arquitectura de red profunda muy utilizada es la de las redes convolucionales.

Esta estrategia intenta solventar el gran tamaño que podrían tener las redes totalmente conectadas en caso de utilizar alguna arquitectura profunda con una enorme cantidad de capas. En las redes fully connected, todas las neuronas de una capa se conectan con todas las neuronas de la capa siguiente. En lugar de esto, las redes convolucionales utilizan un esquema de conectividad local. Eso permite reducir el número de conexiones entre las neuronas y reducir el tamaño de las redes profundas, que ya de por sí consumen una enorme cantidad de memoria.



Ejemplo de arquitectura típica de red convolucional (LeCun et. al, 2021)

La figura que se muestra más arriba, de una típica arquitectura de red convolucional, tiene unas capas que se llaman «convolution», otras que se llaman «pooling» y una capa totalmente conectada al final. Esto es muy común en las redes convolucionales que se usan para clasificación.

Por una parte, las capas de «convolution» permiten extraer características de manera automática de las imágenes. Por otra parte, las capas de «pooling» reducen la dimensionalidad de la imagen poco a poco, para llegar finalmente a un vector de características que representará la entrada a una red «fully connected». La ventaja de hacer esto es que, aunque tengamos una red profunda, la única red fully connected tendrá el tamaño equiparable al de un perceptrón multicapa, es decir, algo mucho más asumible que una red fully connected con cientos de capas.

Capas convolucionales

Este esquema de conectividad local se consigue por medio de las convoluciones (de ahí recibe el nombre la arquitectura de red). Para simplificar, una [convolución](#) es una operación matricial que, en el contexto de las imágenes, permite realizar un filtrado sobre las mismas. En otras palabras, por medio de una convolución puedes realizar filtros muy útiles como eliminar toda la información que no sea un borde o difuminar la imagen.

La idea de la conectividad local es que un filtro convolucional es una matriz de tamaño relativamente pequeño que se aplica a modo de ventana deslizante sobre toda la imagen. De este modo, un solo kernel representará a un filtro.

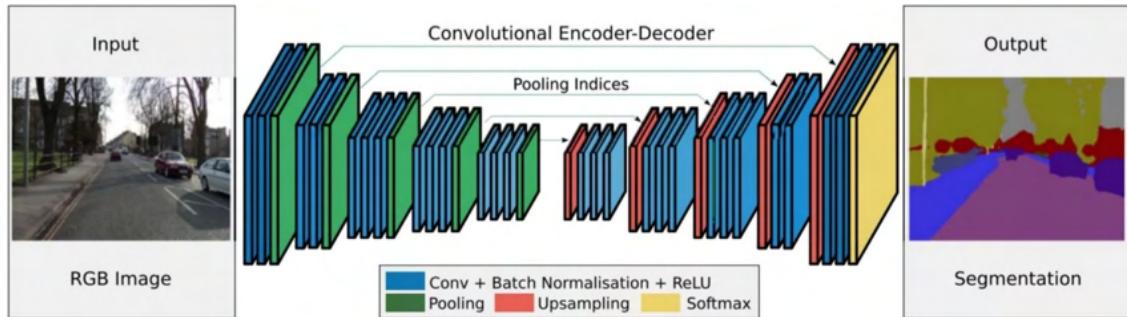
La idea de todo esto no es solo reducir el tamaño de la arquitectura a nivel de conexiones entre neuronas, si no también conseguir un modelo de aprendizaje que sea capaz de aprender a filtrar las características más relevantes de una imagen de modo totalmente automático. De hecho, las redes convolucionales se caracterizan por tener una enorme cantidad de filtros diferentes.

A nivel de optimización de red neuronal, todo esto significa que el modelo actualizará los coeficientes del kernel convolucional para filtrar las imágenes del modo que el modelo considera oportuno.

¿Cómo ayudan las redes convolucionales a segmentar imágenes?

La idea final de utilizar una red convolucional para segmentar es tener un sistema al que le pases una imagen y que te devuelva otra imagen con los objetos de interés segmentados. Esto es un ejemplo de estrategia end-to-end. Para este caso, es muy importante disponer de un ground truth, ya que el modelo aprenderá cómo debe segmentar la imagen a partir de este ground truth.

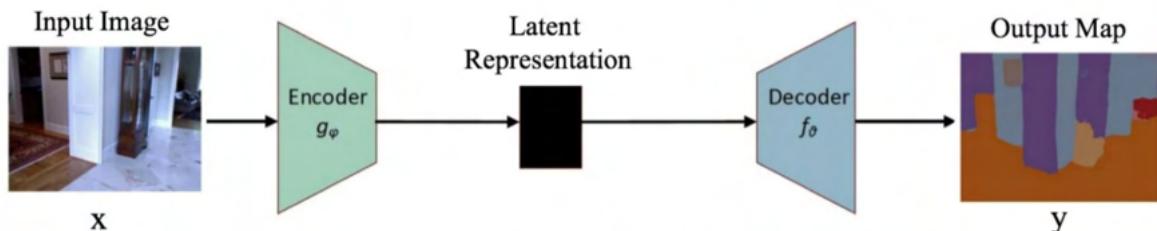
En contraste con la red convolucional presentada anteriormente, que estaba destinada a clasificación, la red que aquí se presenta es una fully convolutional, de tal forma que no tiene ninguna capa totalmente conectada.



Red fully convolutional (Badrinarayanan et. al, 2017)

Esto es muy ventajoso ya que, entre otras cosas, nos permite que el tamaño de la imagen de entrada sea variable ya que al tener una capa totalmente conectada lo más probable es que la red se entrene con un tamaño de imagen fijo y que solo se le pueda pasar una imagen de ese mismo tamaño en la fase de predicción (es decir, una vez que estamos usando el modelo ya entrenado).

Este tipo de arquitecturas totalmente convolucionales para tareas como segmentación, se caracterizan por tener dos partes: el encoder y el decoder. El encoder reduce la dimensionalidad de la imagen de entrada hasta llegar a un vector de características. Por su parte, el decoder utiliza ese vector de características como entrada (que en la figura de abajo se denomina como *latent representation*) y aumenta poco a poco su dimensionalidad hasta llegar a la imagen de salida, que será la correspondiente a la segmentación.



Típica arquitectura de encoder-decoder (Minaee et. al, 2021)

Escasez de datos etiquetados

Y aquí es donde entramos en el problema de la escasez de datos etiquetados manualmente que mencionábamos unos cuantos párrafos más atrás. Los modelos de aprendizaje profundo necesitan de una enorme cantidad de datos para ser entrenados, en general.

En los últimos años, se han hecho muchos esfuerzos para solventar ese problema. Para ello, se pueden utilizar estrategias como el [data augmentation](#), que básicamente pretende aumentar el tamaño de un conjunto de imágenes de manera artificial, es decir, sin necesidad de añadir ejemplos etiquetados nuevos.

El data augmentation clásico utiliza transformaciones triviales como rotaciones aleatorias, traslaciones o cambios de intensidad de los píxeles, pero cabe destacar que, en los últimos años, ha cobrado mucha importancia la generación de imagen sintética, con las redes generativas antagónicas (GAN) como una de las estrategias más utilizadas para tal fin.

Con todas las ideas mencionadas hasta ahora en mente y, conociendo los conceptos fundamentales sobre redes neuronales, el entrenamiento de una red convolucional es muy trivial. Sencillamente, le pasaremos una imagen al sistema, obtendremos una salida y la compararemos con su correspondiente ground truth. En esa comparación, se utilizará una función de error cuyo

valor será retropropagado a los pesos de la red para que poco a poco su salida se optimice y alcance el comportamiento deseado.

Resumen

En resumidas cuentas, la segmentación de imágenes es un proceso de la visión artificial con el cual nos quedamos con la información que nos importa de una imagen para un problema concreto, eliminando cualquier otra cosa. Generalmente, esta segmentación será representada con una nueva imagen donde solo se puedan ver las siluetas de los objetos de interés.

Para la segmentación de imágenes, es muy interesante poder utilizar diferentes modelos de machine learning. Existen algunas aproximaciones convencionales que son un poco más rudimentarias y menos intuitivas.

Sin embargo, en la actualidad, el auge se encuentra en el Deep Learning que, con las arquitecturas de red convolucionales, es capaz de recibir una imagen como entrada y devolver directamente la segmentación a la salida.

Es muy importante disponer de datos etiquetados manualmente pero, por desgracia, es muy común que esto no sea así. Para evitarlo, se puede recurrir a estrategias de data augmentation, para incrementar el tamaño del conjunto de imágenes original de manera totalmente artificial, es decir, sin necesidad de etiquetar nuevos ejemplos de imágenes.

Fuentes

(Chen et. al, 2017) – L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” arXiv preprint arXiv:1706.05587, 2017.

(LeCun et. al, 1998) – Y. LeCun, L. Bottou, Y. Bengio, P. Haffner et al., “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

(Badrinarayanan et. al, 2017) – V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 12, pp. 2481–2495, 2017

(Minaee et. al, 2021) – Minaee, Shervin, et al. «Image segmentation using deep learning: A survey.» *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

Créditos

Daniel Iglesias es informático y amplio conocedor del ámbito del aprendizaje automático y la visión artificial.

Machine Learning en Operaciones Espaciales

Por Jose Martinez Heras
13/12/2021



En noviembre de 2021, tuve la oportunidad de asistir a una charla sobre Machine Learning en Operaciones Espaciales. La charla se tituló "Pie & AI: Darmstadt - Spacecraft Anomaly Detection". El enlace para ver la grabación es <https://youtube.com/watch?v=UcS4JiVGnPY>. Aunque no pude asistir en persona, pude ver la grabación en YouTube.

Patrick, el autor de la charla, ha puesto la grabación en YouTube. Así que si te lo perdiste, aquí tienes la oportunidad de verlo.

El evento fue en inglés. Si te interesa el tema de Machine Learning en Operaciones espaciales en español, puedes consultar los siguientes artículos:

- Detección de Anomalías en Espacio
- Antenas de Espacio Profundo & Inteligencia Artificial
- Basura Espacial: competición con machine learning

Colaboradores

Esta es la lista de colaboradores en IArtificial.net

Álvaro Bartolomé del Canto

[Álvaro Bartolomé del Canto](#) es graduado en Ingeniería Informática por la Universidad de Salamanca (USAL) con un máster en Inteligencia Artificial por la Universidad Internacional de La Rioja (UNIR). Empezó trabajando como Investigador en Ciencia de Datos en BISITE Research Group y posteriormente dio el salto a Frontiers como Ingeniero de Machine Learning. Álvaro publicó su primer proyecto de código abierto en GitHub en 2019 y descubrió así una de sus grandes pasiones, por lo que desde entonces no ha dejado de crear y colaborar en este tipo de proyectos. Sus principales intereses son el Deep Learning y el MLOps.

Puedes contactar con Álvaro a través de [Twitter](#) o [GitHub](#).

Posts Invitados

- TorchServe para servir modelos de PyTorch

Daniel Iglesias

Daniel Iglesias es informático y amplio conocedor del ámbito del aprendizaje automático y la visión artificial.

Post Invitados

- Segmentación de Imágenes con Redes Convolucionales

Alan López

Alan es doctor en ciencias, y coordinador del área dedicada a la investigación e implementación del Machine Learning y Ciencia de Datos en una empresa de consultoría TI. Le encanta aprender y enseñar sobre cómo lograr sistemas de aprendizaje de máquina prácticos y funcionales. El blog personal de Alan es: [MachineLearningEnEspanol.com](#)

Posts Invitados

- Algoritmos Genéticos y Memoria Visual

José David Villanueva García

[José David](#) es Ingeniero Técnico en Informática de Sistema por la Universidad Rey Juan Carlos, Graduado en Matemáticas por la UNED, y Máster en Matemáticas Avanzadas por la UNED ([Máster Thesis](#)).

Actualmente trabaja como ingeniero en Darmstadt, Alemania, en diferentes proyectos para la ESA (European Space Agency) y EUMETSAT (European Organisation for the Exploitation of Meteorological Satellites).

Posts Invitados

- Contraste de Hipótesis 1 – ¿cómo no aceptar lo falso?
- Redes neuronales desde cero (I) – Introducción
- Redes neuronales desde cero (II): algo de matemáticas

¿Cómo puedo colaborar?

Una forma de colaborar en IArtificial.net es escribiendo post invitado sobre algún tema relacionado con Inteligencia Artificial, Machine Learning, Estadística, Análisis de Datos, Visualización de Datos, python, R, etc. Es una muy buena forma de darte a conocer a la comunidad de Inteligencia Artificial en castellano ya que tu artículo lo leerán miles de personas.

También puedes proponer tus ideas para colaborar con iartificial.net

Si quieres escribir un post invitado, o tienes otras ideas para colaborar, [contacta con nosotros](#) y hablamos.

Sobre mí

Hola, soy José Martínez Heras y desde hace más de 15 años estoy en el mundo de la Inteligencia Artificial y el Machine Learning. Sí, trabajo con datos antes de que el Big Data y el Análisis de Datos estuviese de moda.

Nací en Córdoba, España y llevo ya varios años viviendo en Alemania. Trabajo en [Solenix](#) como Senior Research Engineer, y muchos de nuestros proyectos son con la [Agencia Espacial Europea](#), concretamente, en el Centro de Operaciones. Mi trabajo consiste en ayudar a ejecutivos e ingenieros a tomar mejores decisiones usando técnicas de Inteligencia Artificial.

En 2018 organicé una serie de charlas sobre Machine Learning en la Agencia Espacial Europea. El objetivo era el ofrecer una introducción intuitiva a la Inteligencia Artificial. Estas charlas fueron todo un éxito y contribuyeron a un mejor entendimiento de lo que la Inteligencia Artificial puede hacer (y lo que no), y a dar una intuición de cómo funciona. Estas charlas y el material que preparé están disponible en inglés en [GitHub](#).

Hay mucho material sobre Inteligencia Artificial disponible en inglés. Para mi sorpresa, he visto que hay bastante poco en español. Algunos de los artículos que he visto en español sobre Inteligencia Artificial hablan de las implicaciones de la IA en la sociedad. Sin embargo, he encontrado muy poco sobre cómo funciona la Inteligencia Artificial en realidad.

Mis objetivos con [IArtificial.net](#) son:

1. Divulgar conocimientos de inteligencia artificial y machine learning a profesionales, negocios y empresas.
2. Facilitar el desarrollo de actividades en el sector de la Inteligencia Artificial, Machine Learning y Análisis de Datos

Si quieres saber más sobre mí, puedes conectar conmigo en [LinkedIn](#) o seguirme en [Twitter](#).



Clases de Machine Learning en la Agencia Espacial Europea