

Comprensión de las redes neuronales con TensorFlow Playground

26 de julio de 2016

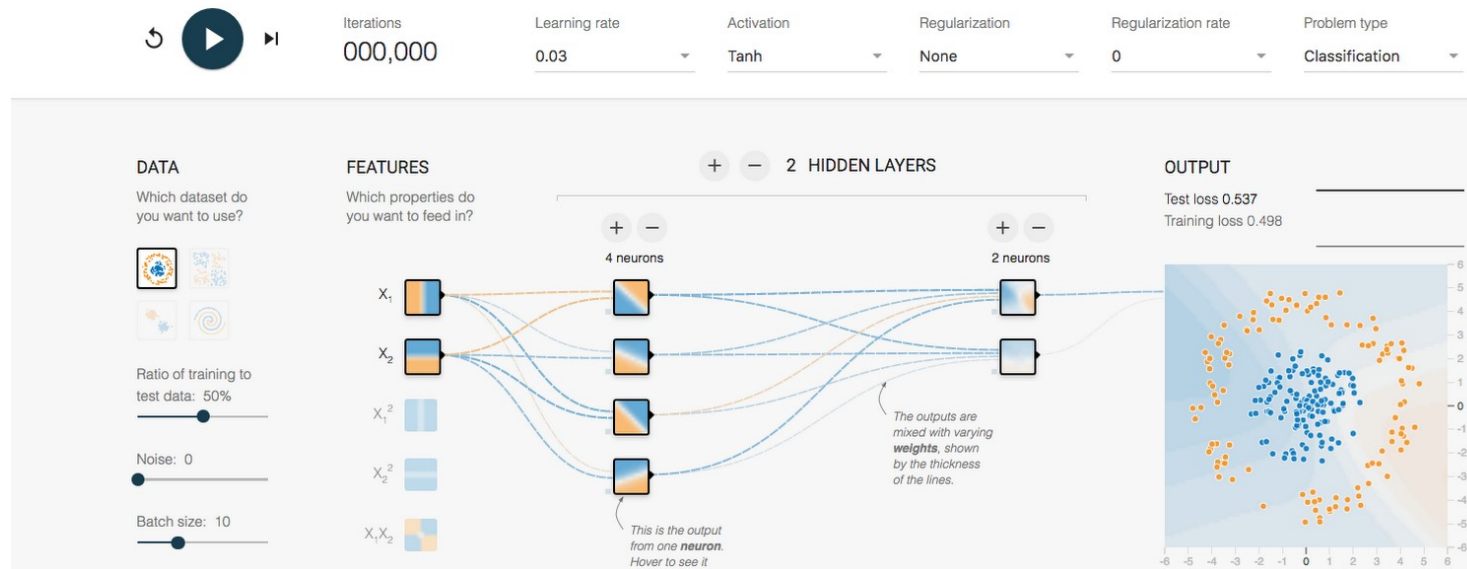
Kaz Sato

Defensor de desarrolladores, IA en la nube

Quizás hayas oído hablar mucho de las redes neuronales y el aprendizaje profundo, y quieras aprender más. Pero cuando aprendes sobre la tecnología en un libro de texto, muchas personas se sienten abrumadas por los modelos y fórmulas matemáticas. Yo, sin duda, lo estaba.

Para personas como yo, existe una herramienta increíble que puede ayudarles a comprender la idea de las redes neuronales sin necesidad de cálculos matemáticos complicados: [TensorFlow Playground](#), una aplicación web escrita en JavaScript que les permite jugar con una red neuronal real ejecutándose en su navegador, hacer clic en botones y ajustar parámetros para ver cómo funciona.

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

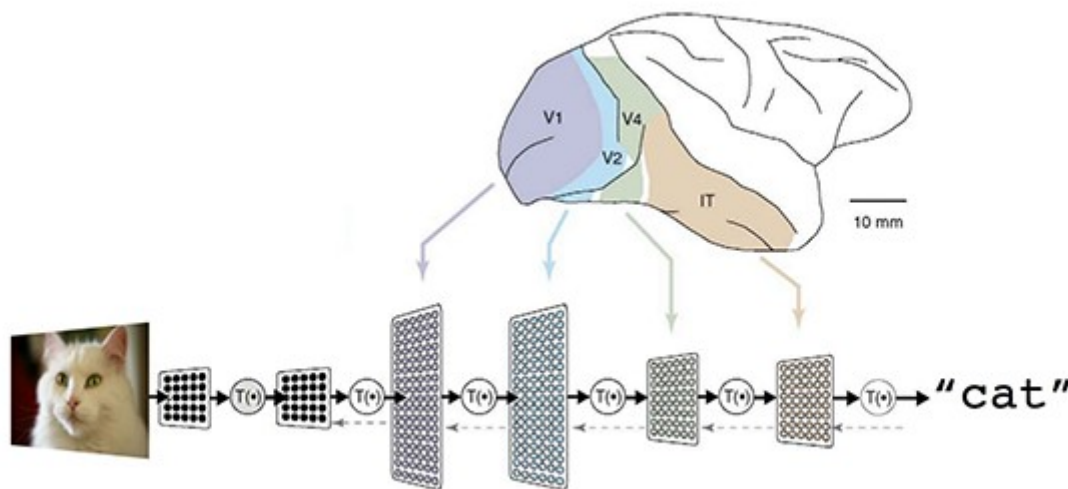


Zona de juegos de TensorFlow

En este artículo, me gustaría mostrarte cómo puedes experimentar con TensorFlow Playground para comprender las ideas centrales de las redes neuronales. Así, podrás entender por qué esta tecnología ha despertado tanto interés últimamente.

Deje que la computadora resuelva el problema

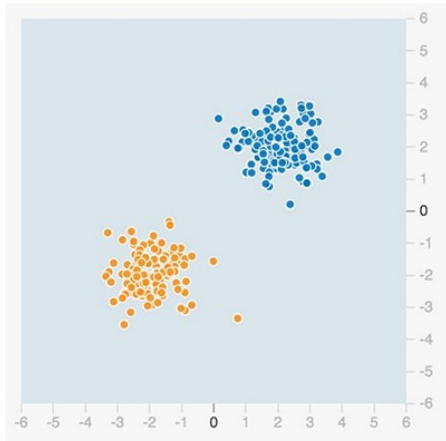
La programación informática requiere un programador. Los humanos le dan instrucciones a una computadora para que resuelva un problema especificando cada paso mediante muchas líneas de código. Pero con el aprendizaje automático y las redes neuronales, se puede dejar que la computadora intente resolver el problema por sí misma. Una red neuronal es una **función** que **aprende** el resultado esperado para una entrada dada a partir de **conjuntos de datos de entrenamiento**.



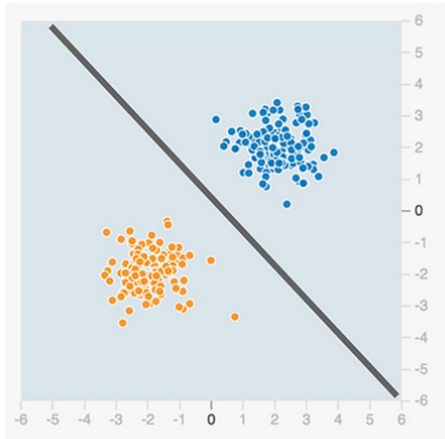
Una red neuronal es una función que aprende a partir de conjuntos de datos de entrenamiento (de: [Aprendizaje profundo a gran escala para sistemas informáticos inteligentes](#), Jeff Dean, WSDM 2016, adaptado de [Desenredar el reconocimiento de objetos invariantes](#), J DiCarlo et D Cox, 2007)

Por ejemplo, para construir una red neuronal que reconozca imágenes de un gato, se entrena con muchas imágenes de gatos de muestra. La red resultante funciona como una función que toma la imagen de un gato como entrada y genera la etiqueta "gato". O, para un ejemplo más práctico, se puede entrenar para que ingrese varios registros de actividad de usuarios de servidores de juegos y genere qué usuarios tienen una alta probabilidad de conversión.

¿Cómo funciona esto? Analicemos un problema de clasificación sencillo. Imaginemos que tiene un conjunto de datos como el que se muestra a continuación. Cada punto de datos tiene dos valores: x_1 (eje horizontal) y x_2 (eje vertical). Hay dos grupos de puntos de datos: el grupo naranja y el grupo azul.



¿Cómo se escribe código que clasifique si un punto de datos es naranja o azul? Quizás se dibuje una línea diagonal arbitraria entre los dos grupos, como se muestra a continuación, y se defina un umbral para determinar a qué grupo pertenece cada punto de datos.



La condición de su declaración IF se vería así.

$$x_1 + x_2 > b$$

Donde **b** es el umbral que determina la posición de la línea. Al asignar **w1** y **w2** como **pesos** a x_1 y x_2 , respectivamente, el código es más reutilizable.

$$w_1x_1 + w_2x_2 > b$$

Además, si ajusta los valores de w_1 y w_2 , puede rotar el ángulo de la línea a su gusto. También puede ajustar el valor "b" para mover la posición de la línea. De esta manera, puede reutilizar esta condición para clasificar cualquier conjunto de datos que pueda clasificarse mediante una sola línea recta.

Pero el problema es que el programador tiene que encontrar valores apropiados para w_1 , w_2 y b —los llamados parámetros— e indicarle al ordenador cómo clasificar los puntos de datos.

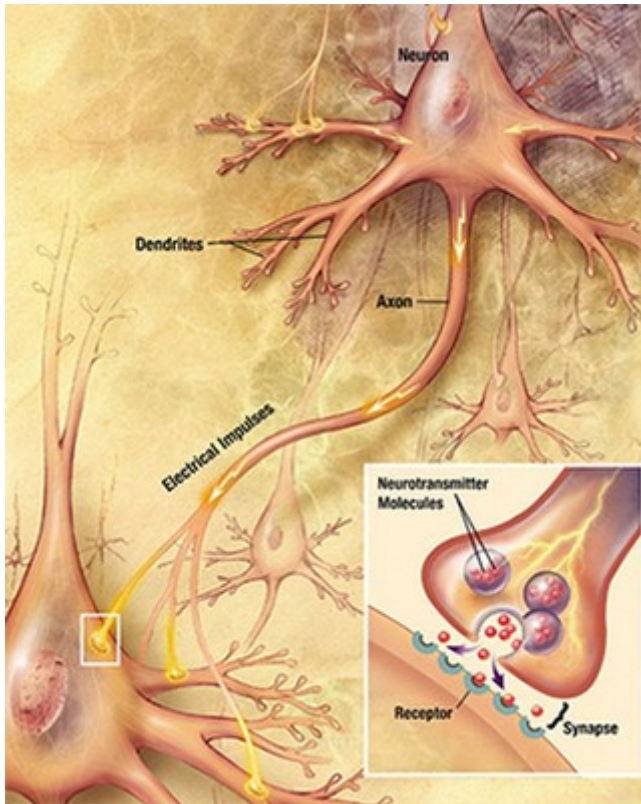
Una neurona: clasificar un punto de datos en dos tipos

Ahora, veamos cómo el ordenador de TensorFlow Playground resuelve [este problema](#). En Playground, haz clic en el botón Reproducir en la esquina superior izquierda. La línea entre los puntos de datos azul y naranja comienza a moverse lentamente. Pulsa el botón de reinicio y vuelve a pulsar Reproducir varias veces para ver cómo se mueve la línea con diferentes valores iniciales. Lo que ves es el ordenador intentando encontrar la mejor combinación de pesos y umbral para dibujar la línea recta entre dos grupos.



Un problema de clasificación simple en TensorFlow Playground.

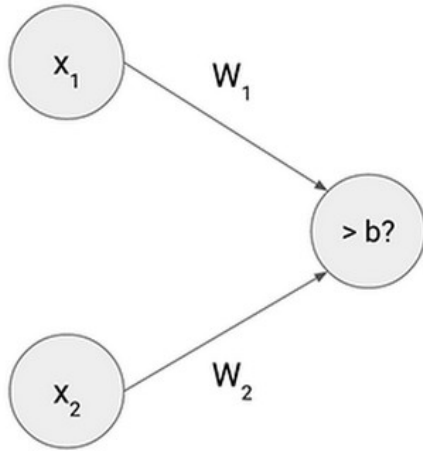
TensorFlow Playground utiliza una sola neurona artificial para esta clasificación. ¿Qué es una neurona artificial? Es una idea inspirada en el comportamiento de las neuronas biológicas del cerebro humano.



La red entre neuronas biológicas (De: [Wikipedia](#))

Para una descripción detallada del mecanismo de una red neuronal biológica, visite [la página de Wikipedia](#) : cada neurona se excita (activa) al recibir señales eléctricas de otras neuronas conectadas. Cada conexión entre neuronas tiene diferente intensidad. Algunas conexiones son lo suficientemente fuertes como para activar otras neuronas, mientras que otras inhiben la activación. En conjunto, los cientos de miles de millones de neuronas y conexiones de nuestro cerebro representan la inteligencia humana.

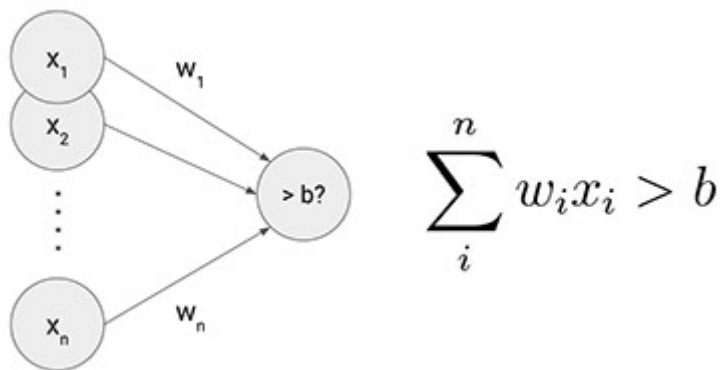
La investigación sobre neuronas biológicas condujo a la creación de un nuevo paradigma informático: la [red neuronal artificial](#) . Con las redes neuronales artificiales, imitamos el comportamiento de las neuronas biológicas con matemáticas sencillas. Para resolver el problema de clasificación mencionado, se puede utilizar la siguiente red neuronal simple, que consta de una sola neurona (también conocida como perceptrón).



x_1 y x_2 son los valores de entrada, y w_1 y w_2 son pesos que representan la fuerza de cada conexión a la neurona. b es el llamado **s sesgo** , que representa el umbral para determinar si una neurona se activa o no con las entradas. Esta neurona individual se puede calcular con la siguiente fórmula.

$$w_1x_1 + w_2x_2 > b$$

Sí, es exactamente la misma fórmula que usamos para clasificar los conjuntos de datos con una línea recta. De hecho, es lo único que una neurona artificial puede hacer: clasificar un punto de datos en uno de dos tipos examinando los valores de entrada con ponderaciones y sesgos. Con dos entradas, una neurona puede **clasificar los puntos de datos en un espacio bidimensional en dos tipos** con una línea recta. Con tres entradas, una neurona puede clasificar los puntos de datos en un espacio tridimensional en dos partes con un plano, y así sucesivamente. Esto se llama "dividir un espacio n -dimensional con un [hiperplano](#) ".



Una neurona clasifica cualquier punto de datos en uno de dos tipos

Utilizando una sola neurona para realizar el reconocimiento de imágenes

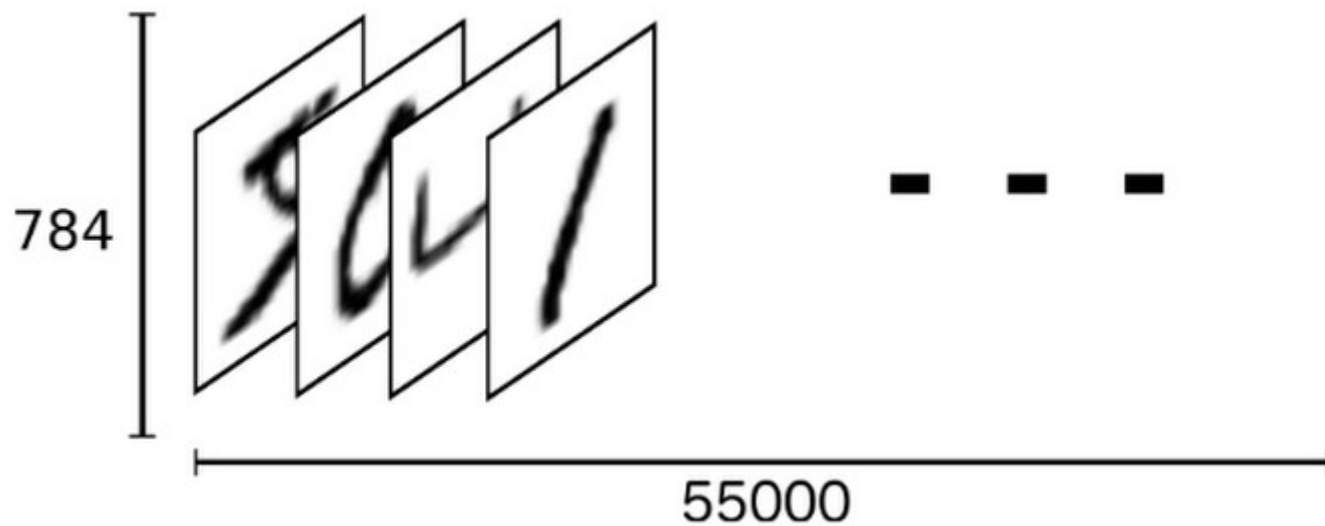
¿Cómo puede un "hiperplano" resolver problemas cotidianos? Por ejemplo, imagina que tienes muchas imágenes de texto escrito a mano como las que se muestran a continuación.



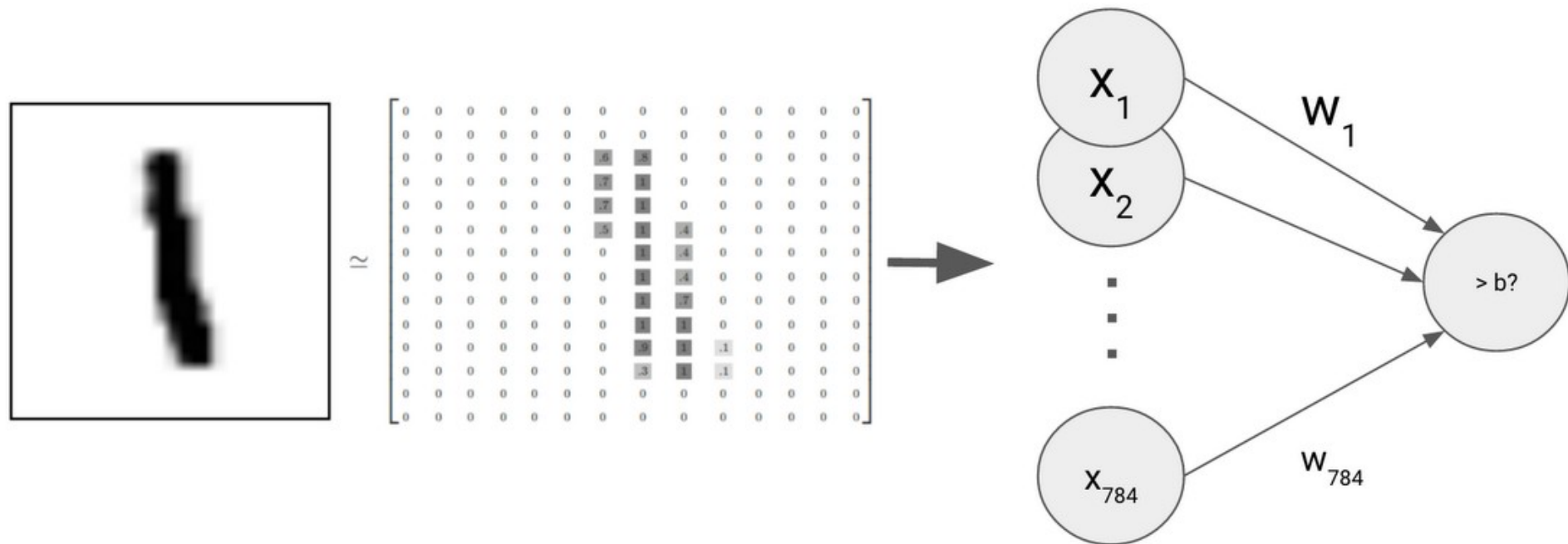
Imágenes de píxeles de textos escritos a mano (de: [MNIST para principiantes de ML](https://www.tensorflow.org/datasets/catalog/mnist) , tensorflow.org)

Puedes entrenar una sola neurona para clasificar un conjunto de imágenes como "imágenes del número 8" u "otras imágenes".

¿Cómo se hace? Primero, se deben preparar decenas de miles de imágenes de muestra para el entrenamiento. Supongamos que una sola imagen tiene 28 x 28 píxeles en escala de grises; cabrá en una matriz de 28 x 28 = 784 números. Con 55 000 imágenes de muestra, se obtendría una matriz de 784 x 55 000 números.

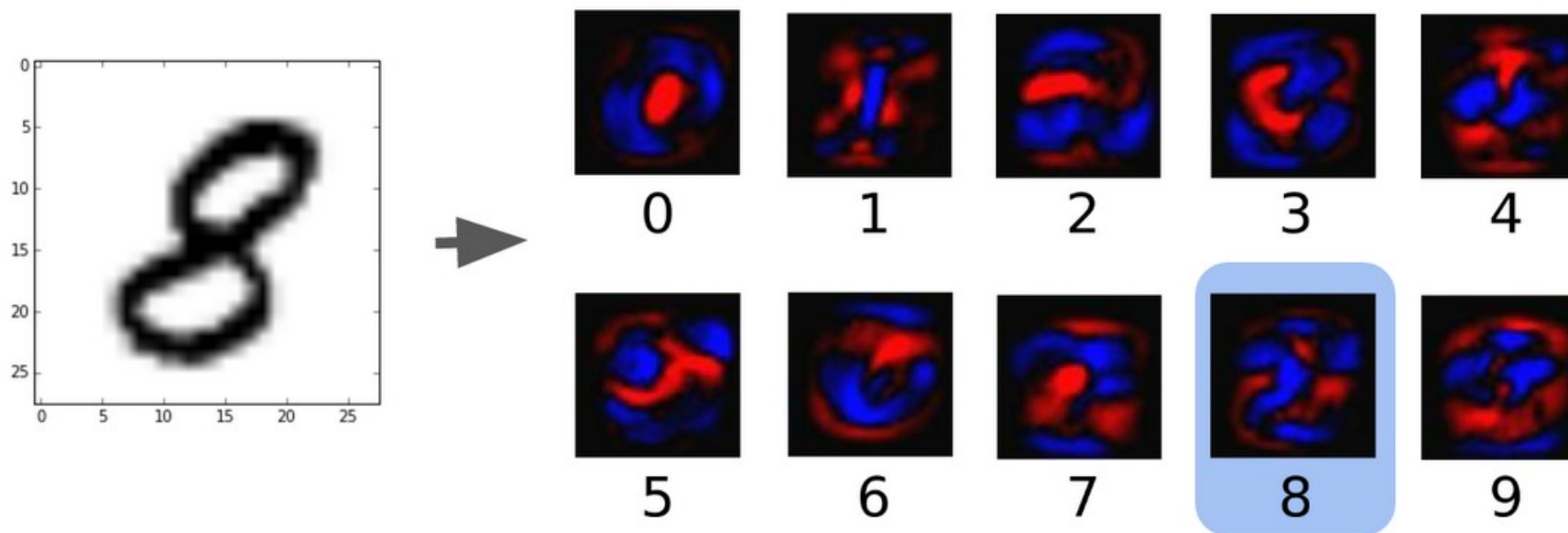


Para cada imagen de muestra en las 55 000 muestras, ingresa los 784 números en una sola neurona, junto con la etiqueta de entrenamiento que indica si la imagen representa o no un "8".



Como viste en la demostración de Playground, la computadora intenta encontrar un conjunto óptimo de pesos y sesgos para clasificar cada imagen como un "8" o no.

Después del entrenamiento con las 55K muestras, esta neurona habrá generado un conjunto de pesos como los que se muestran a continuación, donde el azul representa un valor positivo y el rojo un valor negativo.



Eso es todo. Incluso con esta neurona única tan primitiva, se puede lograr una precisión del 90 % al reconocer una imagen de texto escrito a mano [1](#). Para reconocer todos los dígitos del 0 al 9, se necesitarían solo diez neuronas con una precisión del 92 %.

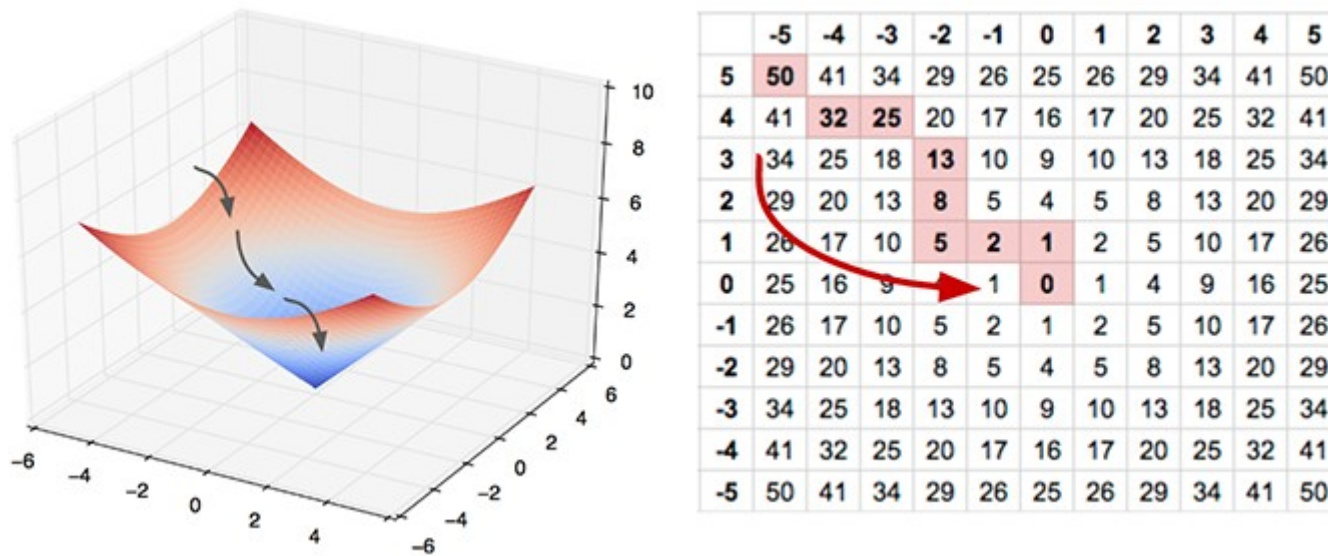
De nuevo, lo único que esta neurona puede hacer es clasificar un punto de datos como uno de dos tipos: "8" o no. ¿Qué se considera un "punto de datos" en este caso? En este caso, cada imagen contiene $28 \times 28 = 784$ números. En términos matemáticos, se podría decir que cada imagen representa un único punto en un espacio de 784 dimensiones. La neurona divide el espacio de 784 dimensiones en dos partes con un único hiperplano y clasifica cada punto de datos (o imagen) como "8" o no. (Sí, es casi imposible imaginar cómo serían ese espacio dimensional y ese hiperplano. Olvídalo).

En el ejemplo anterior, usamos la imagen 1 de texto manuscrito como datos de muestra, pero se puede usar una red neuronal para clasificar muchos tipos de datos. Por ejemplo, un proveedor de juegos en línea podría identificar a los jugadores que hacen trampa examinando sus registros de actividad. Un proveedor de comercio electrónico puede identificar a los clientes premium a partir de los registros de acceso al servidor web y el historial de transacciones. En otras palabras, se puede expresar cualquier dato que pueda convertirse y expresarse como un número como un punto de datos en un espacio n-dimensional, dejar que la neurona intente encontrar el hiperplano y ver si esto ayuda a clasificar eficazmente el problema.

¿Cómo se entrena una red neuronal?

Como puede ver, una red neuronal es un mecanismo simple que se implementa con matemáticas básicas. La única diferencia entre la programación tradicional y la red neuronal es, nuevamente, que se permite que la computadora determine los parámetros (pesos y sesgo) aprendiendo de los conjuntos de datos de entrenamiento. En otras palabras, el patrón de peso entrenado en nuestro ejemplo no fue programado por humanos.

En este artículo, no analizaré en detalle cómo entrenar los parámetros con algoritmos como [la retropropagación](#) y [el descenso de gradiente](#). Basta decir que el ordenador intenta aumentar o disminuir ligeramente cada parámetro para ver cómo reduce el error en comparación con el conjunto de datos de entrenamiento, con la esperanza de encontrar la combinación óptima de parámetros.



Piense en la computadora como un estudiante o un trabajador novato. Al principio, comete muchos errores y tarda un tiempo en encontrar una forma práctica de resolver problemas reales (incluidos posibles problemas futuros) y minimizar los errores (lo que se denomina generalización).

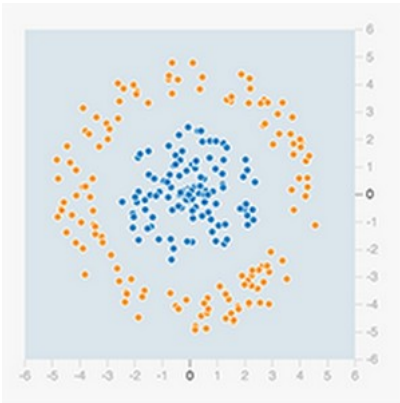


Una red neuronal necesita tiempo de entrenamiento antes de poder minimizar los errores (De: Irasutoya.com)

Podríamos retomar el tema en un próximo artículo. Por ahora, conténtete con el hecho de que una biblioteca de redes neuronales como [TensorFlow](https://www.tensorflow.org/) encapsula la mayor parte de los cálculos necesarios para el entrenamiento, y no tendrás que preocuparte demasiado por ello.

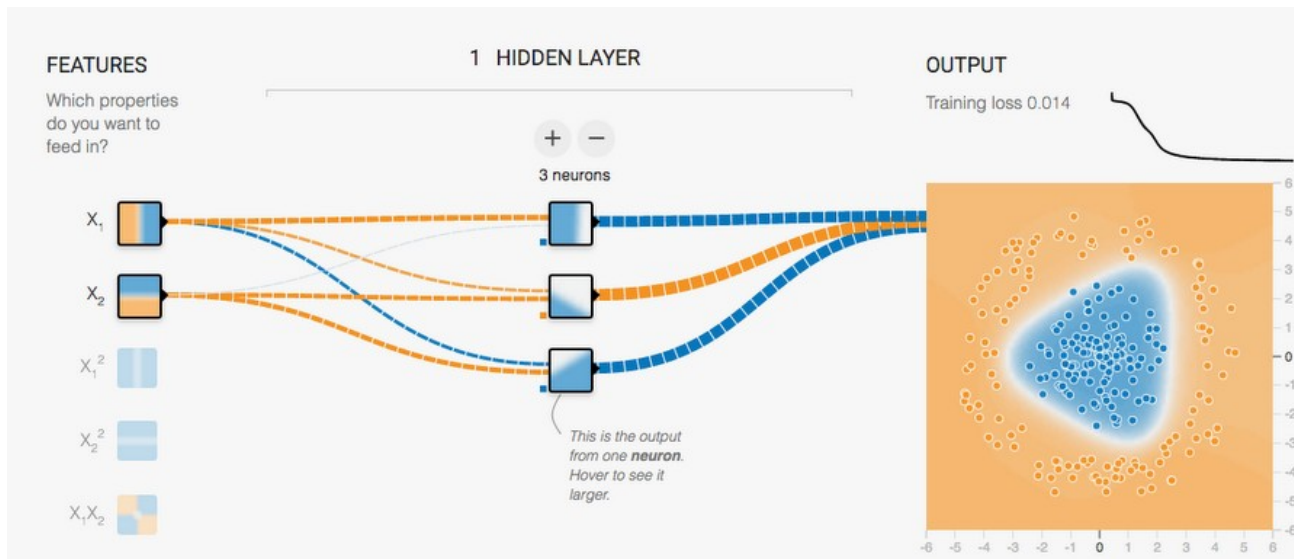
Más neuronas, más características para extraer

Hemos demostrado cómo una sola neurona puede realizar una clasificación simple, pero quizás te preguntes cómo se puede usar una neurona simple para construir una red capaz de reconocer miles de imágenes diferentes y competir con un jugador profesional de Go. Hay una razón por la que las redes neuronales pueden ser mucho más inteligentes que lo descrito anteriormente. Veamos otro ejemplo de TensorFlow Playground.



Este conjunto de datos no se puede clasificar con una sola neurona, ya que los dos grupos de puntos de datos no se pueden dividir con una sola línea. Este es un problema de clasificación no lineal. En el mundo real, la cantidad de conjuntos de datos no lineales y complejos como este es infinita, y la pregunta es cómo capturar este tipo de patrones complejos.

La solución es añadir una capa oculta entre los valores de entrada y la neurona de salida. [Haz clic aquí](#) para probarlo.



Problema de clasificación no lineal en TensorFlow Playground ([haga clic aquí](#) para probarlo)

¿Qué está pasando aquí? Si haces clic en cada neurona de la capa oculta, verás que cada una realiza una clasificación simple de una sola línea:

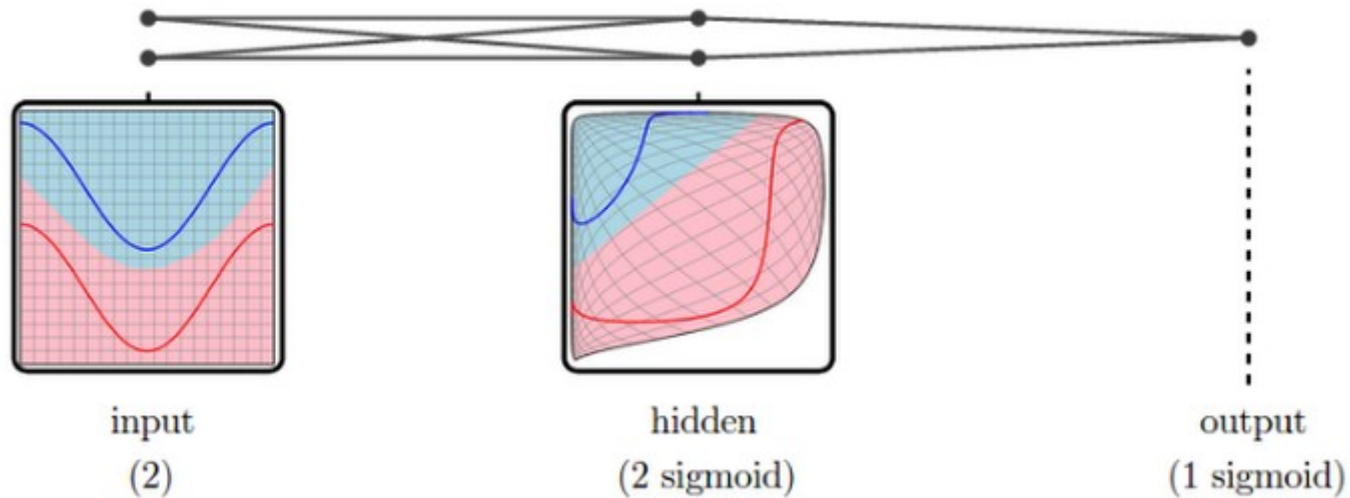
- La primera neurona comprueba si un punto de datos está a la izquierda o a la derecha.
- La segunda neurona comprueba si está en la parte superior derecha.

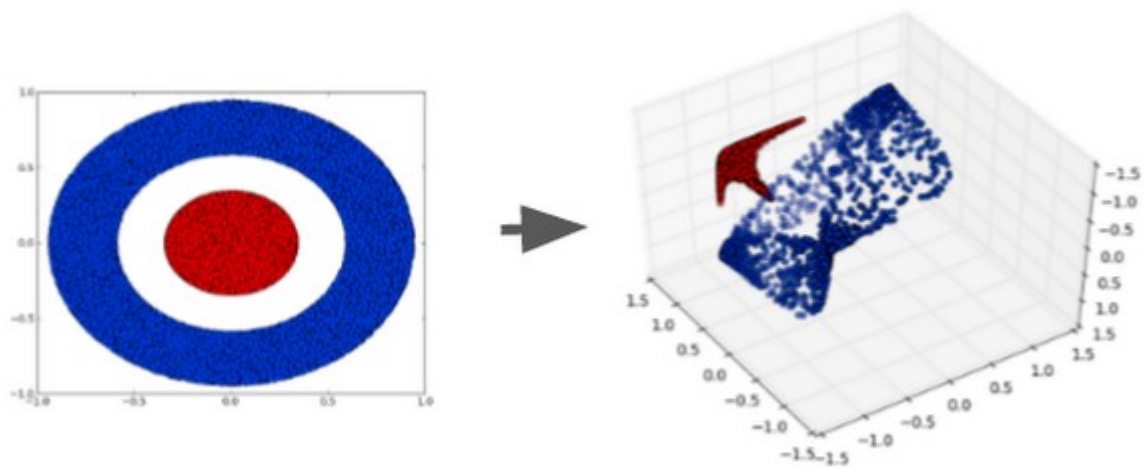
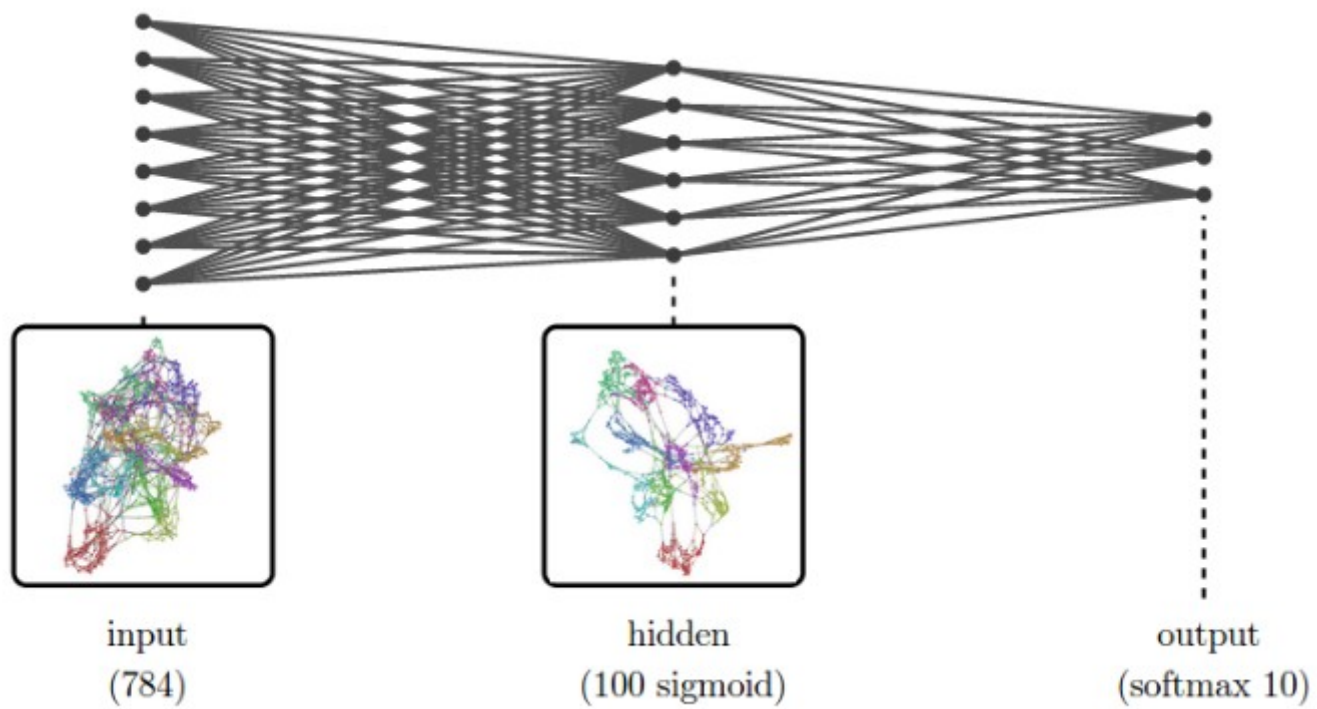
- El tercero comprueba si está en la parte inferior derecha.

Estos tres resultados se denominan características de los datos. Las salidas de estas neuronas indican la intensidad de sus características correspondientes.

Finalmente, la neurona de la capa de salida utiliza estas características para clasificar los datos. Si se dibuja un espacio tridimensional compuesto por los valores de las características, la neurona final puede simplemente dividir este espacio con una superficie plana. Este es un ejemplo de transformación de los datos originales en un espacio de características.

Para ver algunos excelentes ejemplos visuales de transformaciones, visita [el blog de colah](#).





Una capa oculta transforma las entradas en un espacio de características, lo que las hace linealmente clasificables (de: [Visualizing Representations: Deep Learning and Human Beings](#) and [Neural Networks, Manifolds and Topology](#) , Christopher Olah)

En el caso de la demostración de Playground, la transformación resulta en una composición de múltiples características que corresponden a un área triangular o rectangular. Si agrega más neuronas haciendo clic en el botón "más", verá que la neurona de salida puede capturar formas poligonales mucho más sofisticadas del conjunto de datos.

Volviendo a la analogía del oficinista, se puede decir que la transformación consiste en extraer los conocimientos que un profesional experimentado adquiere en su trabajo diario. Un empleado nuevo se confunde y se distrae con señales aleatorias provenientes de correos electrónicos, teléfonos, el jefe, clientes, etc., pero los empleados con experiencia son muy eficientes a la hora de extraer la señal esencial de esas entradas y organizan el caos según unos principios importantes.

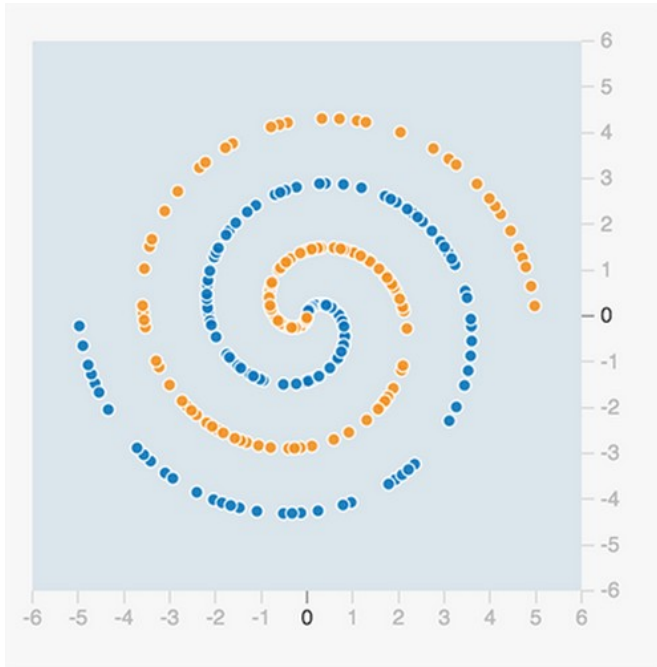
Las redes neuronales funcionan de la misma manera: intentan extraer las características más importantes de un conjunto de datos para resolver el problema. Por eso, a veces pueden ser lo suficientemente inteligentes como para gestionar tareas bastante complejas.



Una red neuronal puede extraer información de señales (aparentemente) aleatorias (De: [Irasutoya.com](#))

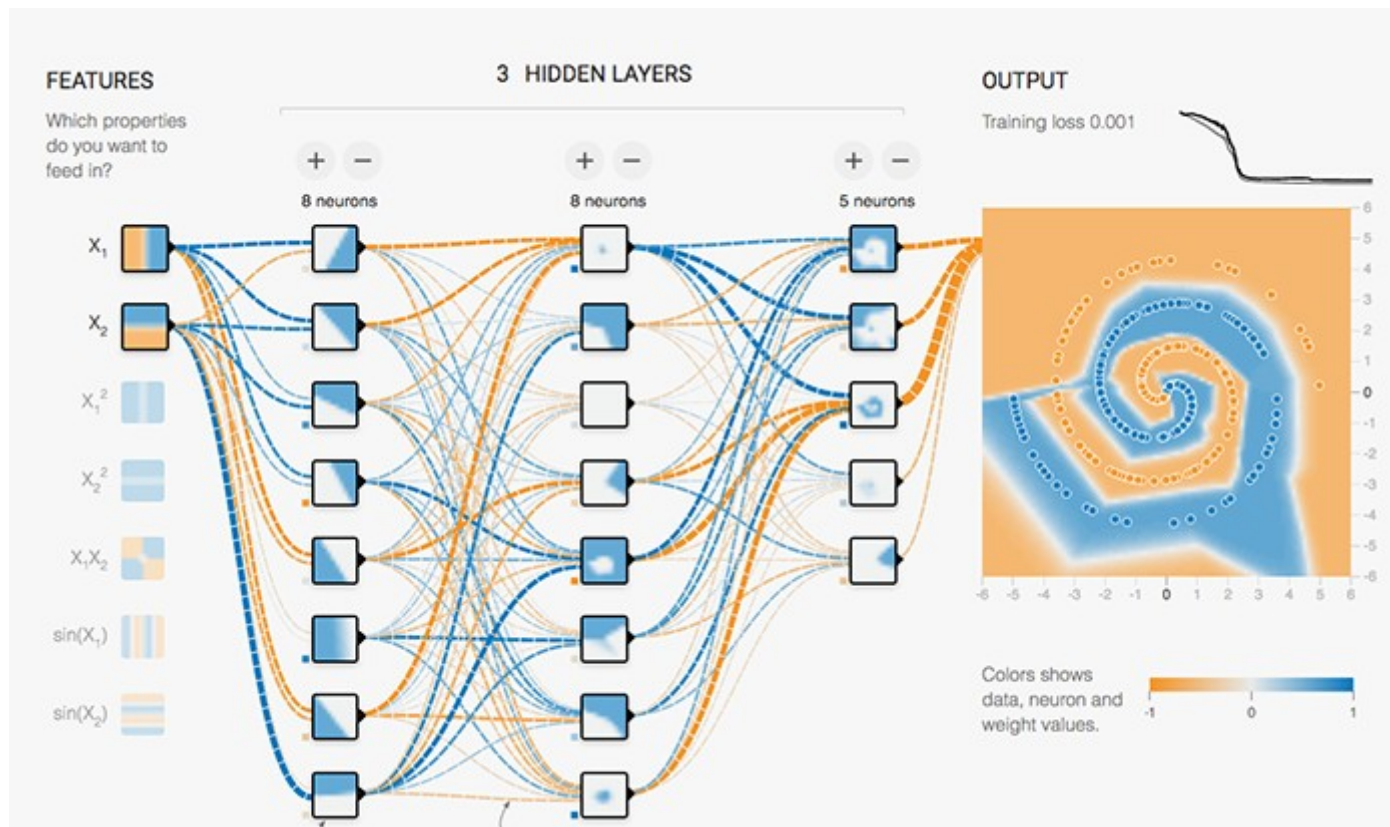
Necesitamos profundizar más: construir una jerarquía de abstracciones

Con más neuronas en una sola capa oculta, se pueden capturar más características. Y tener más capas ocultas implica que se pueden extraer construcciones más complejas del conjunto de datos. El siguiente ejemplo muestra su potencial.



¿Qué tipo de código escribirías para clasificar este conjunto de datos? ¿Docenas de sentencias IF con muchísimas condiciones y umbrales, cada una comprobando en qué área pequeña se encuentra un punto de datos dado? Personalmente, no querría hacer eso.

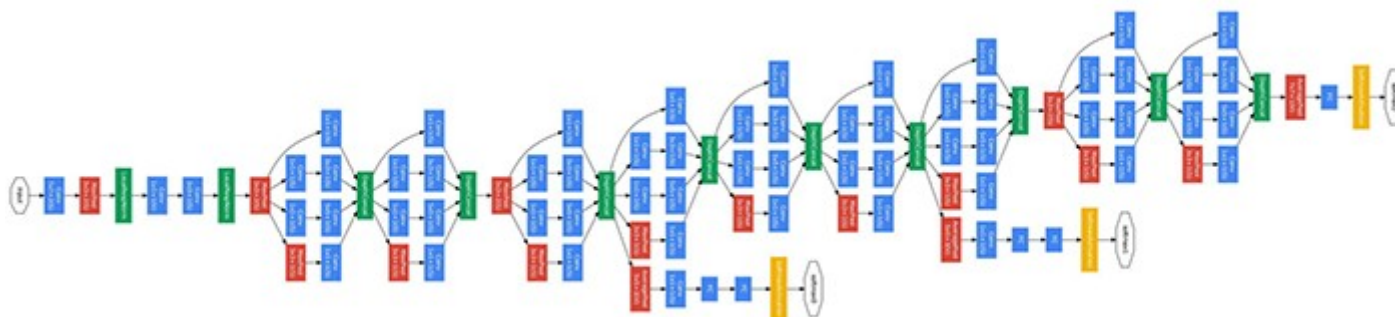
Aquí es donde el aprendizaje automático y las redes neuronales superan el rendimiento de un programador humano. [Haz clic aquí](#) para verlo en acción (el entrenamiento te llevará un par de minutos).



Problema de doble espiral en TensorFlow Playground ([haga clic aquí](#) para probarlo)

Genial, ¿verdad? Lo que acabas de ver es la computadora intentando construir una jerarquía de abstracción con una red neuronal profunda. Las neuronas de las primeras capas ocultas realizan las mismas clasificaciones simples, mientras que las neuronas de la segunda y tercera capa componen características complejas a partir de las simples, creando finalmente el patrón de doble espiral.

Más neuronas + una red más profunda = abstracción más sofisticada. Así es como las neuronas simples se vuelven más inteligentes y funcionan tan bien en ciertos problemas, como el reconocimiento de imágenes y el juego de Go.



Inception: un modelo de reconocimiento de imágenes publicado por Google (De: [Profundizando con las convoluciones](#) , Christian Szegedy et al.)

[Algunos ejemplos publicados de visualización por redes profundas](#) muestran cómo se las entrena para construir la jerarquía de patrones reconocidos, desde simples bordes y manchas hasta partes y clases de objetos.

Dos desafíos: potencia computacional y datos de entrenamiento

En este artículo, analizamos algunas demostraciones de TensorFlow Playground y cómo explican el mecanismo y el poder de las redes neuronales. Como has visto, los fundamentos de la tecnología son bastante simples. Cada neurona simplemente clasifica un punto de datos en uno de dos tipos. Sin embargo, al tener más neuronas y capas profundas, una red neuronal puede extraer información oculta y patrones complejos de un conjunto de datos de entrenamiento y construir una jerarquía de abstracción.

La pregunta entonces es, ¿por qué no todo el mundo utiliza todavía esta magnífica tecnología? Actualmente, las redes neuronales se enfrentan a dos grandes retos. El primero es que entrenar redes neuronales profundas requiere mucha potencia de cálculo, y el segundo es que requieren grandes conjuntos de datos de entrenamiento. Un potente servidor GPU puede tardar varios días o incluso semanas en entrenar una red profunda con un conjunto de datos de millones de imágenes.

Además, se requiere mucho ensayo y error para obtener los mejores resultados de entrenamiento con diversas combinaciones de diferentes diseños de red y algoritmos. Hoy en día, algunos investigadores utilizan decenas de servidores GPU o incluso supercomputadoras para realizar entrenamiento distribuido a gran escala.

Pero en un futuro muy cercano, los servicios de predicción y entrenamiento distribuidos totalmente administrados como [Google Cloud AI Platform](#) con TensorFlow pueden resolver estos problemas con la disponibilidad de CPU y GPU basadas en la nube a un costo asequible, y pueden abrir el poder de las redes neuronales grandes y profundas a todos.

Expresiones de gratitud

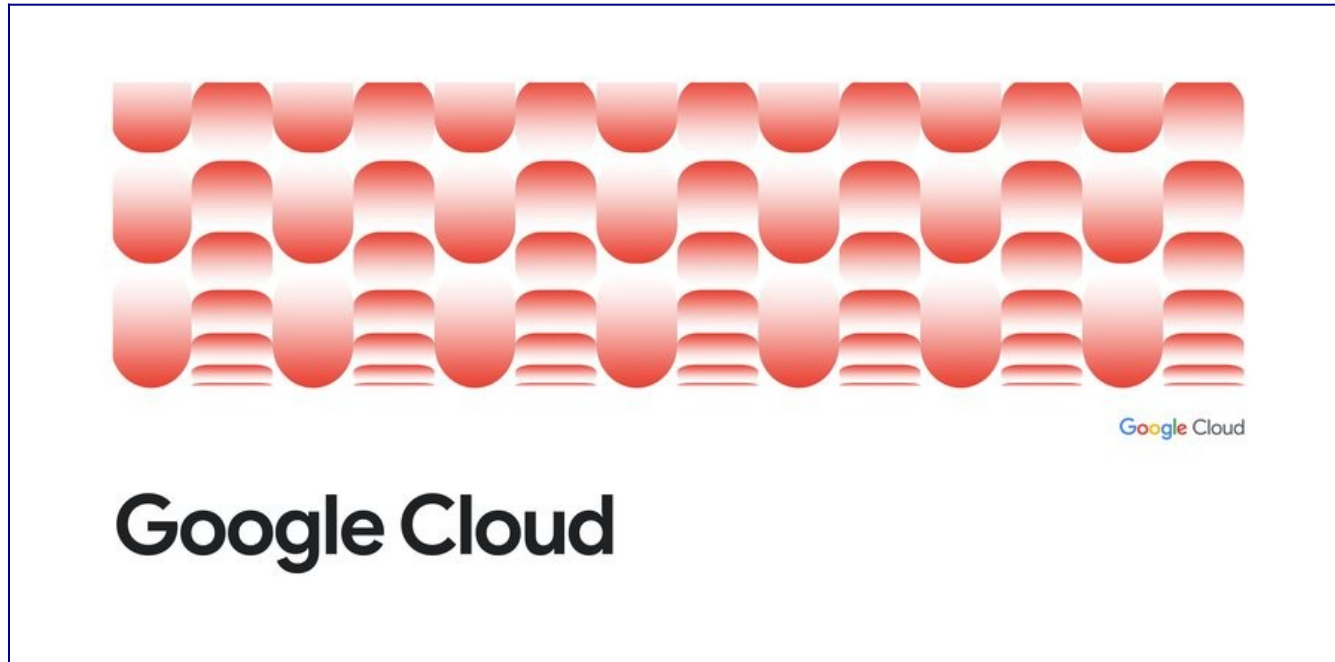
Muchísimas gracias a David Ha, Etsuji Nakai, Christopher Olah y Alexandra Barrett por revisar la publicación y aportar valiosos comentarios, además de refinar el texto. Y un agradecimiento especial a los autores de TensorFlow Playground, Daniel Smilkov, Shan Carter y D. Sculley, por su magnífico trabajo.

[1] Fuente: [MNIST para principiantes de ML](#)

Cómo un agricultor japonés de pepinos utiliza el aprendizaje profundo y TensorFlow

Los usos del aprendizaje automático y el aprendizaje profundo solo están limitados por nuestra imaginación. Un agricultor de pepinos puede usar el aprendizaje profundo para clasificarlos. Vea cómo.

Por [□□□□](#) • Lectura de 5 minutos



Publicado en

- [IA y aprendizaje automático](#)
- [Google Cloud](#)