Taller de Inteligencia Artificial

Julián Estévez. Gorka Garate

0.1. Introducción

La Inteligencia Artificial (IA) es una de las ciencias de moda. Y permite multitud de aplicaciones. Concretamente, gracias a la IA es posible:

- predecir qué artículo vas a comprar en Amazon
- aprender a salir de un laberinto
- jugar a ajedrez
- resolver problemas matemáticos
- clasificar perfiles de consumidores
- conducir vehículos autónomos

Ninguna de estas capacidades es por arte de birlibirloque. ¡Qué va! Detrás de cada algoritmo de Inteligencia Artificial no hay más que números y ecuaciones sencillas, que funcionan mejor cuantos más datos tengan.

En las siguientes páginas, veremos tres algoritmos matemáticos muy sencillos que esclarecen de una manera bastante intuitiva cómo la IA logra generar esa aparente 'magia'.

0.2. K-means

El objetivo genérico del algoritmo K-means es clasificar conjuntos de datos, según un criterio establecido de antemano.

En nuestro caso, el conjuntos de datos serán un conjunto de puntos del plano. Por ejemplo, como el que se ve en la Figura 1:

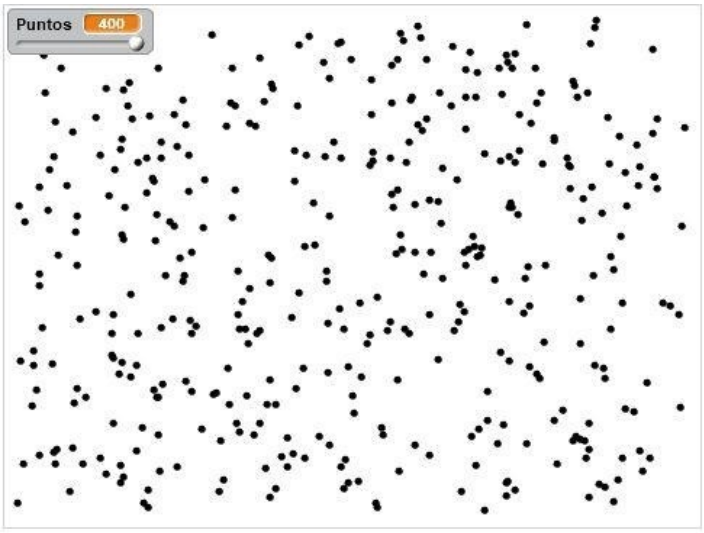

Figura 1: Puntos aleatorios creados inicialmente

Piensa, por ejemplo, que son planetas sueltos, y que queremos clasificar en sistemas estelares; para eso, crearemos otro conjunto de puntos—menos numerosos—que serán las propias estrellas.

Para determinar a qué sistema estelar pertenece cada planeta necesitaremos un criterio. En nuestro caso, será la distancia del planeta a la estrella más cercana; para efectuar la clasificación, tomaremos cada uno de los planetas, calcularemos las distancias hasta todas las estrellas y nos quedaremos con la menor de todas. Por último, para ponerlo bonito, colorearemos cada sistema estelar de un color distinto. El resultado final será el de la Figura 2:

Por explicarlo de manera sencilla: los puntos pintados en verde tienen como característica común que la estrella más cercana a ellos es la verde. Aunque la figura pueda llevar a alguna duda, todos los puntos morados

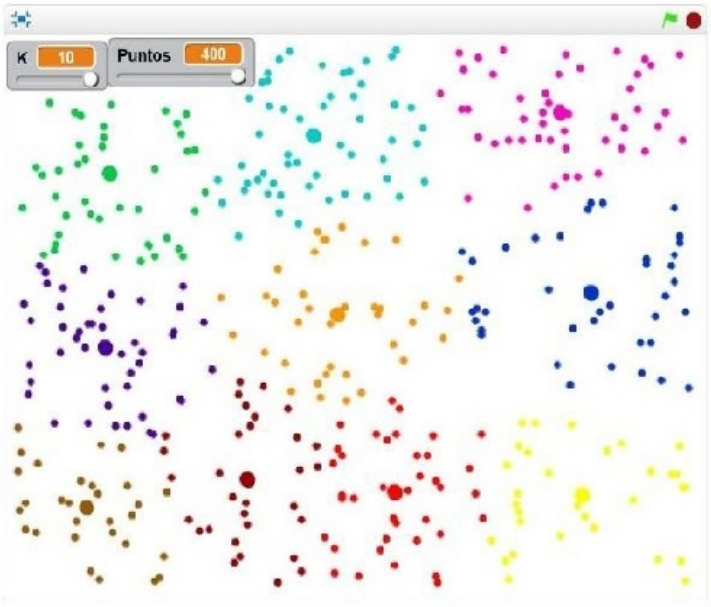

Figura 2: Puntos aleatorios clasificados en 10 clusters

tienen una distancia a la estrella verde superior a la distancia a la estrella morada.

Estos conjuntos de sistemas estelares no son del mismo tamaño. Eso conllevaría alguna operación más.

0.3. Redes neuronales

Las redes neuronales son una de las herramientas más usadas en inteligencia artificial. A partir de unas operaciones muy sencillas, permiten detectar patrones, reconocer imágenes, predecir valores futuros, aprender reglas sencillas o incluso clasificar datos. En el ejemplo actual, vamos a explicar cómo aprenden las redes neuronales.

Estas abstracciones se componen de *neuronas*, que habitualmente representamos por pequeños círculos. Esos círculos pueden estar en la entrada, en la salida, o en capas intermedias. Algunas de esas neuronas están unidas entre sí, y las líneas que las unen se denominan *pesos*.

Ocasionalmente, una o varias neuronas puede tener un sesgo o *bias* incorporado a ellas.

Veámoslo gráficamente:

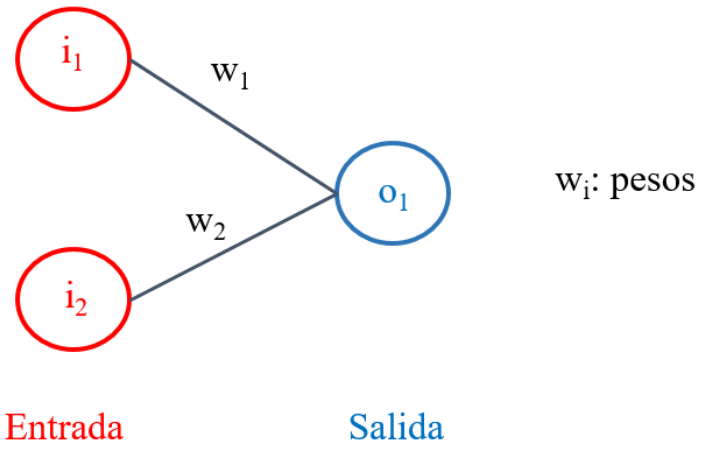

Figura 3: Red neuronal genérica

Y si la red neuronal tuviera un sesgo en alguna parte, sería así (Figura 4:

La red neuronal se entrena, y eso en lenguaje matemático significa lo

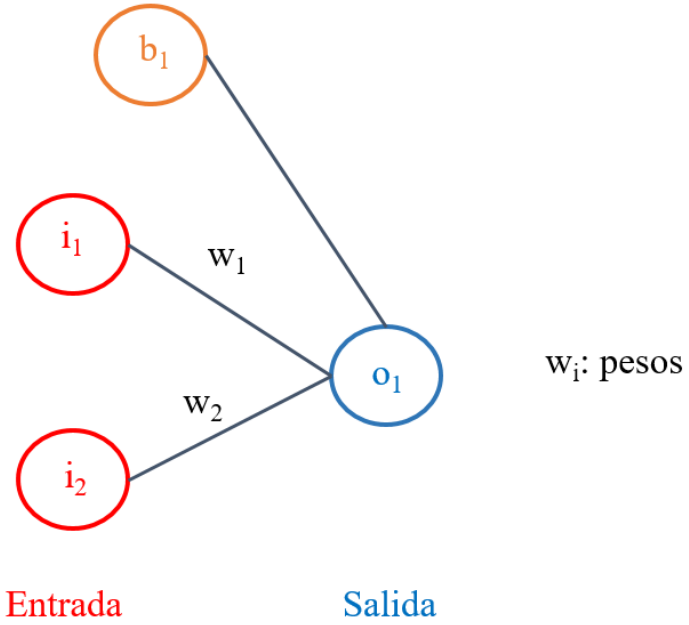

Figura 4: Red neuronal genérica con un sesgo

siguiente: el sistema recibirá unos valores de entrada y de salida. Para hacerlo fácil, fijémonos en la red neuronal de la Figura 3. Tenemos 2 entradas y 1 salida. Es decir, para entrenarla, introduciremos muchos conjuntos de datos para calcular los pesos.

Imaginemos que los datos con los que entrenamos son el día y el mes, y que el dato de salida que tenemos es la temperatura máxima de ese día. Podríamos tener una tabla de la siguiente manera:

Por ejemplo, podríamos tener 300 datos tríos de datos para entrenar a la red neuronal. ¿Cuál es la operación a llevar a cabo? (Cuadro 0.3)

$$O_1 = i_1 \cdot w_1 + i_2 \cdot w_2 \tag{1}$$

Si forzamos a que la neurona de la salida y las de las entradas tomen los valores de la tabla anterior, forzaremos a que los pesos sean unos números determinados. Hay que lograr que sean los mismos pesos para todo el año.

Dato de entrada 1	Dato de entrada 2	Dato de salida
Día 1	Mes	Temperatura máx.
1	1	7
2	1	8
3	1	7
1	2	3
2	2	2
1	4	10
5	5	15
10	7	25
15	8	28

Cuadro 1: Datos de entrada y salida de la red neuronal

Una vez obtengamos los pesos definitivos, podríamos intentar predecir la temperatura en otro año.

Para lograr entrenar una red neuronal y ajustar los pesos, se realizan dos pasos: la alimentación hacia adelante (*feedforward*) y la propagación hacia atrás (*backpropagation*).

Los pasos a seguir son los siguientes:

1. Se crean unos pesos aleatorios w_i
2. A partir de los datos de entrada de la tabla, se usan estos datos en las neuronas i_1 e i_2, y se calcula el valor de la neurona o_1 final
3. Comparamos el valor de o_1 con el valor de salida de la tabla. Lo más habitual, es que sea distinto. A esa diferencia le llamaremos ΔY
4. A continuación, actualizaremos el valor de los pesos usando esa diferencia y un coeficiente de aprendizaje (LR), que tendremos que fijar nosotros
$$\begin{cases} W_{1,nuevo} = W_1 + LR \cdot \Delta Y \cdot i_1 \\ W_{2,nuevo} = W_2 + LR \cdot \Delta Y \cdot i_2 \end{cases} \tag{2}$$
5. Se continúa el proceso tantas veces como sea posible hasta que la salida y la entrada de la red neuronal coincida según los datos de la tabla.

La red neuronal ya está entrenada, y ahora podemos probar valores de entrada nuevos y ver la su salida.

Vamos a empezar con una red neuronal muy simple: una puerta lógica AND sin sesgo y dos pesos, los cuales se almacenan en el vector Weigths. Para inicializarlo todo, haremos clic en la bandera verde.

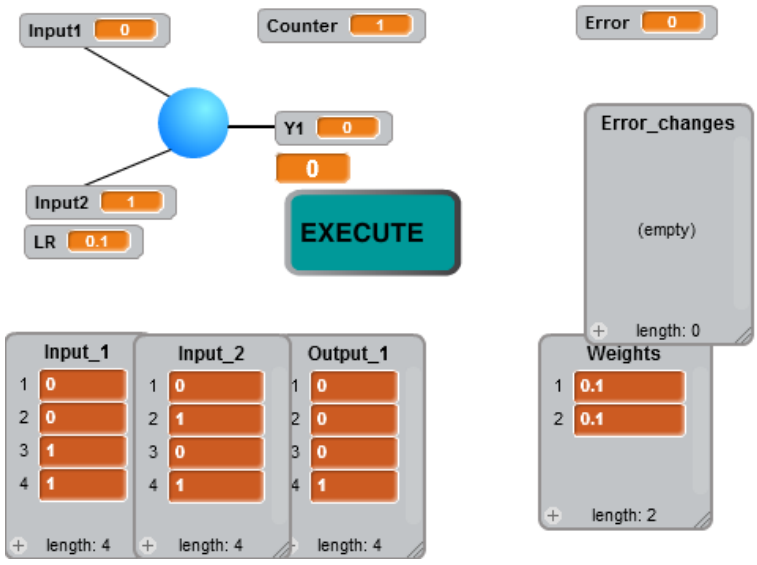

Figura 5: Situación inicial de la Red Neuronal AND

Una puerta lógica AND funciona de la siguiente manera: recibe dos inputs que pueden valer 0 (falso) ó 1 (verdadero). Si los dos inputs valen 1 (son verdaderos) tiene que devolver 1 (verdadero); si alguno de los dos inputs vale 0 (es falso) tiene que devolver 0 (falso). Es decir, tiene que devolver el $Output_1$ correspondiente a los $Input_1$ e $Input_2$ para las cuatro combinaciones que se muestran en la Figura 5 y en el Cuadro 2:

Ahora, para que la neurona aprenda la tenemos que entrenar. El entrenamiento consiste en calcular secuencialmente los outputs que la neurona propone para cada pareja de inputs; para ello programaremos la ecuación (1). Como conocemos el output que queremos obtener (en la lista $Output_1$), calcularemos el error y actualizaremos los dos pesos programando las fórmulas (2). Esta operación se realizará cada vez que cliquemos en el botón EXECUTE. Cuando en la columna de errores *Errorchanges* aparezcan sólo ceros la neurona habrá aprendido a funcionar como una puerta lógica AND.

$Input_1$	$Input_2$	$Output_1$
0	0	0
0	1	0
1	0	0
1	1	1

Cuadro 2: Datos de entrada y salida de la red neuronal AND

Si el alumno tiene tiempo, puede hacer lo mismo para una puerta OR, que debe devolver verdadero si alguno de los dos inputs es verdadero y falso si ambos son falsos.