Caso práctico para comprender el perceptrón

Ahora vamos a detallar de manera numérica un ejemplo que nos permitirá comprender bien los conceptos que hemos visto hasta el momento. Pero no se asuste, los cálculos no serán complicados y seguirán siendo accesibles en términos de comprensión.

Como hemos visto al principio de este capítulo, el objetivo del perceptrón es clasificar las observaciones. Nosotros le proponemos crear un modelo capaz de determinar si un estudiante, según criterios precisos, puede ser admitido en una universidad de prestigio: IA Academy.

La admisión en esta universidad depende de superar algunos exámenes de entrada. La tabla que aparece a continuación reagrupa los distintos casos de admisiones y rechazos en función de la superación de los exámenes de matemáticas e informática.

| **Superado el examen de matemáticas** | **Superado el examen de informática** | **Admitido** |
| --- | --- | --- |
| SÍ | NO | NO |
| SÍ | SÍ | SÍ |
| NO | SÍ | NO |
| NO | NO | NO |

Seguro que se ha dado cuenta de que la admisión en la universidad responde a la función lógica AND.

1. Inicialización del perceptrón

Vamos a inicializar nuestro perceptrón con una capa de 2 neuronas en la entrada, donde cada una de ellas corresponde a la superación de los exámenes, y una neurona de salida que permitirá clasificar al estudiante como admitido o rechazado en la universidad. Como complemento de las dos neuronas de entrada, vamos a añadir otra llamada sesgo (umbral), que tiene el objetivo de controlar la predisposición de la neurona a activarse o no y que siempre tomará el valor 1.

Ahora nuestro perceptrón está listo para aprender.

Observamos la notación particular de cada peso. W11, W21 y Wb. W significa Weight (Peso), seguida del número de la neurona y del número de la capa. Así, W11 se lee «peso de la primera neurona de la primera capa», W21, peso de la segunda neurona de la primera capa, y así sucesivamente. Wb significa peso (Weight) del sesgo (bias en inglés).

Gráfico, Diagrama, Gráfico de burbujas

Descripción generada automáticamente con confianza media

*Neurona formal*

2. Etapas de aprendizaje

**a. Etapa 1: Inicialización de los pesos**

La primera etapa consiste en inicializar los pesos. Esta inicialización se hace de manera aleatoria dentro de un intervalo comprendido entre -1 y 1, salvo por el sesgo, que toma el valor cero.

Existen diversos métodos de inicialización de los pesos (Nguyen-Window, Xavier). Hemos elegido utilizar el más habitual. El intervalo -1 y 1 se justifica por el hecho de que, cuando utilizamos un método de generación aleatoria de números en este intervalo, hemos hecho aproximadamente tantas generaciones por encima como por debajo de 0. Entonces tenemos una distribución regular.

Gráfico, Diagrama, Gráfico de burbujas

Descripción generada automáticamente

*Generación de los pesos*

W11 = -0.165955990594852

W21 = 0.4406489868843162

Wb = 0

En nuestro caso comprobamos que el peso W21 es más importante que los otros, eso significa que el valor contenido en la neurona X2 tiene más importancia que los otros en la predicción, algo que quizás es falso porque este valor solo se elige al azar. Esto tampoco significa que este valor se considerará importante al final del aprendizaje porque, como hemos visto, la neurona va a aprender de sus errores y este peso se ajustará a medida que avance el aprendizaje.

**b. Etapa 2: Carga de los datos de la primera observación**

La segunda etapa consiste en cargar la primera observación contenida en nuestro conjunto de observaciones, la cual alimentará las neuronas X1 y X2.

Por supuesto, los valores «Sí» y «No» de las distintas observaciones han sido sustituidos por 1 para Sí y 0 para No porque, como ya sabe desde hace tiempo, las cadenas de caracteres no se pueden utilizar en Machine Learning y veremos que trabajar con valores numéricos es importante para calcular el error.

Porque si nuestra neurona predice 0,30, ¿cuál es el error si el valor esperado es una cadena de caracteres?, calcular «Sí - 0,30» no tiene sentido. A continuación puede ver la nueva tabla de observaciones:

| **Superado el examen de matemáticas** | **Superado el examen de informática** | **Admitido** |
| --- | --- | --- |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Lo que nos da los valores:

X1 =1

X2 =0

**c. Etapa 3: Preactivación**

Ahora calculamos la suma ponderada de los distintos pesos, correspondiente a la fase de preactivación, procediendo de la siguiente manera:

sumaPonderada = valor del sesgo \*  wb + (w11\*X1 + w21 \* X2)

sumaPonderada = 1\*0 + (-0.165955990594852 \* 1 +

0.4406489868843162 \* 0)=

sumaPonderada = -0.165955990594852

Diagrama

Descripción generada automáticamente

*Cálculo de la preactivación de la neurona*

**d. Etapa 4: Uso de una función de activación**

La función de activación que vamos a utilizar es la sigmoide o también llamada curva S.

El papel de la función de activación es realizar la predicción (llamada y) al nivel de nuestra neurona, y su fórmula (exp significa exponencial) es:

y = 1 / (1 + exp(-suma\_ponderada))

Lo que para nuestro caso de estudio nos da:

Y = 1/(1+exp(-(-0.165955990594852)))

Y= 0,45860596

Diagrama

Descripción generada automáticamente

*Cálculo de la activación (predicción) de la neurona*

**e. Etapa 5: Cálculo del error lineal cometido durante el aprendizaje**

La predicción es de 0,363248 cuando esperábamos el valor 0. Vamos a calcular la diferencia entre el valor esperado y la predicción realizada para determinar el error lineal cometido durante el aprendizaje:

Error = 0 - 0.45860596

Error = - 0.45860596

**f. Etapa 6: Ajuste de los pesos sinápticos**

Ahora lo sabemos, el perceptrón va a aprender de sus errores ajustando los distintos pesos de cada entrada hasta alcanzar una convergencia. Estos ajustes se hacen en función del error antes calculado y efectuando la retropropagación del error.

Esta retropropagación se efectúa al principio calculando el gradiente y formulándose como podemos ver a continuación:

**Gradiente** = -1 \* ERROR \* PREDICCION \* (1-PREDICCION) \*

VALOR\_ENTRADA

En nuestro caso, si queremos actualizar el peso W11 que relaciona la entrada número 1 (X1) y la salida de la neurona, necesitamos los siguientes datos:

* El error = - 0.45860596
* La predicción = 0.45860596
* El valor de entrada X1 = 1

Gradiente = -1 \* - 0.45860596 \* 0.45860596 \* (1-0.45860596) \* 1

**Gradiente = 0,11386568**

Una vez determinado el gradiente, vamos a utilizar la tasa de aprendizaje que nos permitirá progresar un poco más en el descenso calculando el valor que deberá ajustarse el peso W11. Hemos elegido una tasa de aprendizaje arbitraria de 0,1.

Valor\_ajuste\_W11 = 0.11386568 \* 0,1

Valor\_ajuste\_W11 = 0,01138657

Solo nos queda actualizar el peso W11 procediendo de la siguiente manera:

Nuevo\_W11 = W11 - Valor\_ajuste\_W11

Nuevo\_W11 = -0.165955990594852 - 0,01138657

Nuevo\_W11 = -0.15456942

Entonces, debemos proceder de la misma manera para ajustar el peso W21 y el peso del sesgo.

|  | **X1** | **X2** | **SESGO** |
| --- | --- | --- | --- |
| VALOR | 1 | 0 | 1 |
| PESO | -0,16595599 | 0,44064899 | 0 |
| ERROR | -0,45860596 | -0,45860596 | -0,45860596 |
| PREDICCION | 0,45860596 | 0,45860596 | 0,45860596 |
| GRADIENTE | -0,11386568 | 0 | -0,11386568 |
| TASA DE APRENDIZAJE | 0,1 | 0,1 | 0,1 |
| VALOR\_AJUSTE | -0,01138657 | 0 | -0,01138657 |
| PESO NUEVO | -0,15456942 | 0,44064899 | 0,01138657 |

Una vez ajustados los pesos, tenemos que cargar los datos de la segunda observación y volver a empezar el proceso hasta que se haya minimizado la función error.

En lugar de hacer los diferentes cálculos manualmente, vamos a crear un proyecto nuevo de Python y programar nuestra primera neurona formal.