Pseudo código de retro propagación (algoritmo de entrenamiento):

Para cada época

Para cada patrón (etapa de entrenamiento)

-Paso hacia adelante

-Calculo el error (si tengo una salida es un valor y si tengo más de una salida es un vector de errores)

-Paso hacia atrás

-Actualizo los pesos

Para cada patrón (etapa de validación)

-Paso hacia adelante

-Calculo el error

-Evalúo el desempeño

Evalúo si sigo en el bucle o corto.

Luego el código para el algoritmo de pruebas sería casi lo mismo que la etapa de validación.

**RECORDAR QUE HAY DOS ERRORES:**

Uno es el que yo mido al final de la red, que la red lo va a tratar de ir minimizando (yd - y)2, y otro error de clasificación (en la etapa de validación).

Sirve graficar para ir viendo si se va reduciendo el error interno de la red para saber si va aprendiendo.

(Esta explicado en las anotaciones del ejercicio del 2022).

El gradiente siempre tiene que ir dando una matriz del mismo tamaño que la matriz de pesos que estoy utilizando... ir teniendo cuidado con esos detalles cuando hacemos producto matriz vector.

REVISAR LAS DIMENSIONES QUE DEBEN TENER LOS DELTA, ENTRADAS Y PESOS, PARA SABER EXPLICAR ESO.

**Importante**:

Algo que se hace al retro propagar, es a la matriz de pesos quitarle la columna de sesgo y transponerla, y esa es la que se va a usar en la fórmula cuando retro propagamos.

(Abajo explico más)

La estructura de una matriz de pesos será algo como:

[w10 w11 w12 w13

w20 w21 w22 w23]

Donde la primera fila son los pesos que van a la neurona 1 (primer sub índice) provenientes de las entradas 0, 1, 2, 3 (segundo sub índice). Y lo mismo para la segundo fila de la matriz que son los pesos que van hacia la neurona 2.

Entonces, cuando retro propagamos el delta a la matriz de pesos se le quita la primer columna que es del sesgo porque vamos a calcular un delta para cada neurona (calculamos todos los delta de una capa a la vez) y ese sesgo no es una neurona. Sin embargo, ese peso del sesgo también se entrena, y como está incluido en las entradas (X) y salidas de las capas, también se actualiza luego cuando se multiplica la velocidad de aprendizaje por el delta por la entrada para obtener el incremento de peso que vamos a sumar al peso actual.

Entonces, al hacer la retro propagación transponemos esa sub matriz de pesos que nos queda y la multiplicamos por los deltas, y esa es la parte clave de la retro propagación.

Y lo mismo para las capas anteriores.

Otro detalle es que si para el error usamos (yd - y)2 luego en las derivadas que aparecen en la regla de la cadera, nos queda un -1 cuando se hace la derivada parcial de “e” con respecto a “y”, ya que e = (yd - y). Y ese -1 es el que cancela el “-“ que hay en el gradiente al sumarlo para actualizar los pesos.

Pero un posible error es que en vez de usar (yd - y) usemos (y - yd), entonces ahí cuando hagamos la derivada parcial de “e” con respecto a “y” nos da +1 y no se cancela el “-“, entonces es como que vayamos en la dirección del gradiente en vez de en la dirección opuesta.

Si tenemos algún error recordar revisar detalles como que hayamos añadido el sesgo, luego quitado el sesgo y transponer, etc. Y como última opción probar paso por paso las cosas, poniendo a mano los pesos y en papel y lápiz ver que nos daría la salida y así encontrar donde está el error. Por eso el ejercicio 1 es con un XOR que es más fácil de resolver.