* 1. Describe el algoritmo de retropropagación implementado:

El algoritmo de retropropagación implementado se trata de una serie de pasos que emula los pasos del implementado en teoría. Este consiste de la iteración multiple de tres fases diferentes: propagación hacia delante, retropropagación del error y ajuste de pesos. Además, el modelo creado soporta múltiples capas ocultas puesto que la retropropagación tiene sintaxis parecido.

Otros cambios importantes que afecta al rendimiento de la ejecución:

1. Para ahorrar el tiempo de calcular los sigmoides que se tiene un componente exponencial, se ha creado una variable de la clase para almacenar los sigmoides que ya habían sido creados, para ello, se redondea a 5 decimales el valor de la entrada antes de aplicar el sigmoide. Se ha definido esta variable como de la clase porque hacemos múltiples simulaciones usando un script y se aprovecha todos los sigmoides calculados.
2. El modelo soporta múltiples capas ocultas, para ello se debe parar una lista de neuronas para capas ocultas, por ejemplo, si se pasa [2, 3], quiere decir que la primera capa oculta hay 2 neurona y la segunda 3. En el caso de pasar una lista vacía, ósea [], pues no se creará ninguna capa oculta.
3. Para lectura de ficheros, como se usa el sigmoide bipolar, se ha cambiado el código para que sustituya todos los 0 por -1, además, con el flag “norm”, se normalizará los valores de X con una desviación estándar del paquete Sklearn. Esto consiste en restar el valor con la media de todos los valores de esa columna y dividir entre la desviación estándar, es importante destacar que la media y la desviación estándar de cada columna sólo se obtiene a través de los datos de Train, y para los datos de Test, se usará los mismos valores de media y std para normalizar. Con el paquete Sklearn, es simplemente aplicar StandarScaler().fit(X\_train) y normalizar tanto X\_train como X\_test con scaler.tranform().

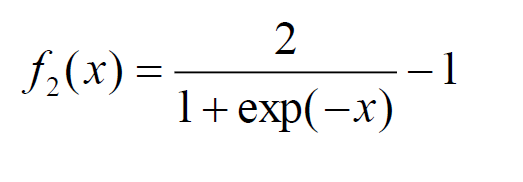
Propagación hacia delante

El algoritmo comienza inicializando la red, poniendo el valor de entrada de las diferentes neuronas y sesgos a valor 0, aunque en el sesgo la salida siempre valdrá 1 independientemente de la entrada. A continuación, mientras la condición de parada no se cumpla se ejecutan iterativamente los siguientes pasos que pasare a explicar, correspondiéndose este conjunto de pasos a una época. Para nuestra implementación existe 4 criterios de parada, el primero que es el más simple con el uso de épocas; el segundo se va a definir una tolerancia, y se para cuando el error cuadrático medio es inferior que esa tolerancia especificada; el tercero es validación con los ejemplos de test, esto realmente se debería trocear el dataset en 3 partes (train, test y validación) pero sólo vamos a usar el test como validación, entonces tras cada época, se evaluará el score en Test y si alguna época supera 5% de error respecto la mejor score, se parará; el cuarto y el último, que también se usa validación, se parará cuando se repite el score en Test durante 30 épocas de entrenamiento. Obviamente los criterios no son los mejores puesto que existe casos donde cumpla estas condiciones, pero el score en test sale muy bajo, un ejemplo sería cuando tenemos muchas capas ocultas y con una tasa de aprendizaje muy bajo, esto hará que la red no consiga variar su peso durante X épocas y el score en Test se repetirá y se parará. Para usar la validación, se debe pasar el conjunto de Train y Test de validación en la función train().

Se iterará sobre el conjunto de pares X e Y de entrenamiento a partir de los cuales primeramente se inicializa con los valores de X la entrada de las neuronas de la capa de entrada, cada neurona con uno de los valores, al bias por otra parte no se le modifica la entrada.

Posteriormente se calcula la respuesta de la última capa oculta realizando el cálculo de la respuesta de la capa de entrada y de las diferentes capas ocultas, a partir de iterar desde la primera capa hasta la anterior a la de salida, disparando, inicializando y propagando para cada una de ellas.

En cada capa, cuando se dispara, se obtiene el valor de salida en función del valor de entrada de cada neurona haciendo uso de la función de activación de la sigmoide bipolar para este caso, la cual tiene la forma que se muestra más adelante. Este valor de salida se pasa como valor a cada una de las conexiones.



Una vez se ha disparado la capa se inicializa, estableciendo para las neuronas el valor de entrada a 0.

Posteriormente se propaga en la capa, a partir de lo cual el valor de entrada de la neurona se recalcula a partir de multiplicar el valor de la conexión por el peso de la conexión, haciendo el sumatorio para su conjunto de conexiones.

Una vez realizados los pasos anteriores iterando sobre las diferentes capas se obtiene la salida de la capa anterior a la capa de salida, a partir de lo cual se actualiza el valor de la capa de las conexiones de la capa de salida. En este punto, para la capa de salida solo se dispara, en vez de inicializar y propagar ya que para el ajuste de sus pesos hace falta y\_in, es decir su salida habiendo aplicado la función de activación.

Retropropagación del error:

La retropropagación se puede definir en 2 pasos, el primero sería retropropagar el error desde capa de salida hacia última capa oculta y el segundo paso que se hará de forma común para X números de capas ocultas que se retropropagará el error hasta que encuentre la capa de entrada.

Estos pasos se consisten en minimizar el error cuadrático medio y, por lo tanto, se debe calcular y posteriormente obtener correspondiente. Para retropropagación de la capa de salida hacia última capa oculta consiste en y para iteración común de resto de capas ocultas hasta capa de entrada, consiste en , obviamente los subíndices de cada iteración se cambiarán, pero el cálculo es el mismo.

El cálculo de las derivadas no lo vamos a hacer así, puesto que calcular una exponencial es una tarea costosa para ordenador y aplicar derivada también. Para ello, como estamos usando un sigmoide bipolar, la derivada se puede expresar de siguiente forma, .

Además, sabemos que es donde es el valor tras realizar el sumatorio de los pesos, por lo tanto se puede expresar como

o equivalente a . Esto pasa también para donde y es el valor tras disparar la neurona aplicando el sigmoide bipolar. La única diferencia que existe respecto primer paso es que el valor de es aplicar un sumatorio de los errores de la capa posterior que acabamos de calcular para iteración anterior, la fórmula consiste en .

La variación del peso se calcula bastante fácil siguiendo la fórmula , esta fórmula es común para cualquier paso de retropropagación pero con diferentes subíndices. Otra cosa que se debe destacar que puesto que la salida de bias es siempre 1, en tema de código no es necesario sacar un bloque para calcular .

Ajuste de pesos:

Con la matriz que contiene los cambios de pesos de cada de cada neurona de cada capa, para el conjunto de conexiones con las neuronas de la capa anterior se itera y actualizan los pesos de estas conexiones.

Finalmente se comprueba la condición de parada y se inicializan las entradas de la última capa a 0.

2.1. Describid todos los experimentos llevados a cabo y los resultados obtenidos: