Práctica Redes neuronales profundas (1ª parte)

El objetivo de esta práctica es entrenar una red neuronal que sea capaz de clasificar imágenes en 10 clases correspondientes a: 'frog', 'truck', 'deer', automobile', 'bird', 'horse', 'ship', 'cat', 'dog' and 'airplane'.

Disponéis de 3 datasets diferentes. La única diferencia entre ellos es el número de muestras de train y test:

- Dataset1: 1000 muestras de train + 100 de test por clase (total: 11000 imágenes)
- Dataset2: 2500 muestras de train + 250 de test por clase (total: 27500 imágenes)
- Dataset3: 4500 muestras de train + 500 de test por clase (total: 50000 imágenes)

El hecho de disponer de diferentes datasets puede ayudarte en el desarrollo de una versión operativa de la red neuronal. Comienza trabajando con un dataset pequeño hasta que hayas llegado a una versión definitiva. Entonces puedes porbar (si quieres) datasets más grandes y pesados.

El modelo base por el que debes comenzar es el que hemos trabajado en clase: una red neuronal con L capas (L elegido por usuario), cuyas neuronas de las capas 1 hasta L-1 utilizan una función de activación RELU y una capa de salida con una función sigmoid. Sin embargo, ese modelo solo permite clasificar 2 clases. Para ello, deberás realizar algunas modificaciones obligatorias:

- Capa de salida Softmax: deberás cambiar la capa de salida para que disponga de tantas neuronas como número de clases. Además, la función de activación de dichas neuronas será softmax.
- Derivada de la última capa: recuerda que, al utilizar una función de activación softmax, las derivadas del coste con respecto a $Z^{[L]}$ cambiará

Además de estos cambios obligatorios, podrás realizar cambios optativos de los que hemos hablado en clase. Evidentemente, estos cambios te ayudarán a encontrar una mejor solución: mayor rapidez en el algoritmo de aprendizaje, menos sobreentrenamiento,etc. Un listado no exhaustivo de posibles cambios son:

- Inicialización de pesos: modificar la función de inicialización de los pesos W según las fórmulas vistas en clase
- Regularización: puedes implementar la regularización L2 o bien la regularización Dropout. Si vas a implementar esta última te recomiendo:
    - No utilices dropout en la capa de salida
    - En las capas 1 hasta L-1, guarda en caché las matrices aleatorias D que utilizas para obtener $A^{[l]} = A^{[l]} * D^{[l]}$
    - Acuérdate de "normalizar" la salida $A^{[L]} = \frac{A^{[L]}}{prob\_perm}$
    - En la propagación hacia atrás, deberás volver a recalcular $dA^{[l]} = dA^{[l]} * D^{[l]}$ y normalizar $dA^{[l]} = \frac{dA^{[l]}}{keep\_prob}$
- Algoritmo de aprendizaje con inercias: a la hora de calcular las derivadas, deberás incorporar nuevas variables al diccionario gradientes para almacenar los términos V y S

- SGD: al comenzar el entrenamiento, genera una función que reciba todos los datos de entrenamiento (junto con las salidas esperadas) y el número de ejemplos de entrenamiento por lote. La función deberá dividir de manera aleatoria los datos de entrenamiento en lotes del tamaño determinado (excepto el último lote, que contendrá los restos). En la función final que representa al modelo, deberás iterar dentro de cada iteración por cada lote, y llamar a propagación adelante, propagación atrás y actualización de parámetros por cada lote.

Descarga de datasets:

https://www.transfernow.net/166bk8z58dkp

https://www.transfernow.net/80hjs7r5c4h3

https://transfernow.net/5140v8u6so6n

os pongo un nuevo enlace de descarga que estará habilitado durante 2 semanas. El arhcivo es mucho más pequeño q los anteriores (300MB) y contiene todos los datasets, pero en formato diferente.
Cuando vayáis a utilizarlos, en lugar de hacer np.loadtxt, tenéis que utiliza variable = np.load('nomber_de_archivo') , sin delimiter ni nada. Otra diferencia es que los datasets X de train y test están en formato hipermatriz (de dimensión mx32x32x3). Deberás utilizar la instrucción reshape para poder moverlo a dimensión 3072xm