Redes Neuronales 2020 - Práctico 3

Universidad Nacional de Córdoba

Alumno: Simonian, Simón.

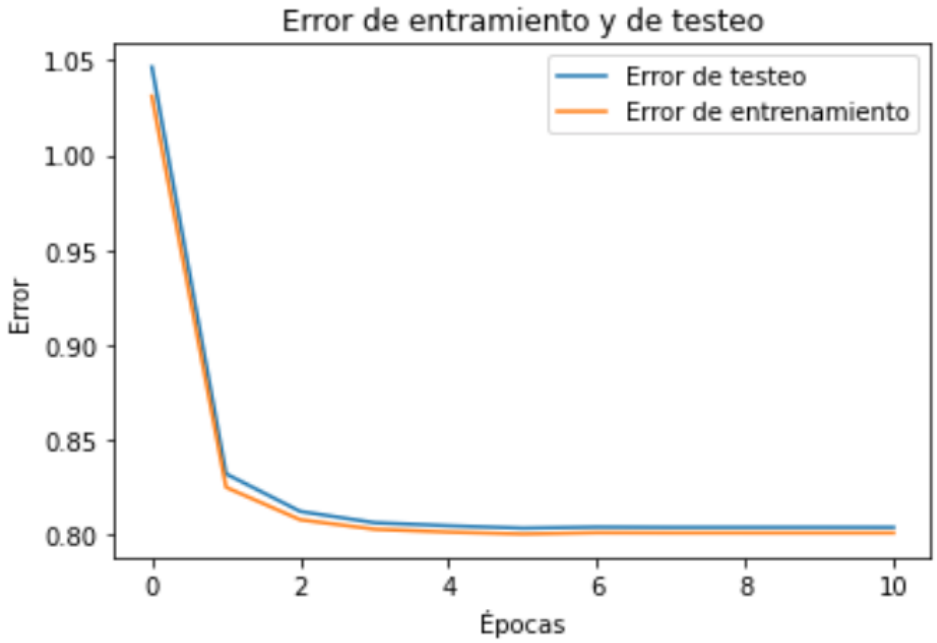
Profesor: Tamarit, A. Francisco

Noviembre de 2020

Respuestas

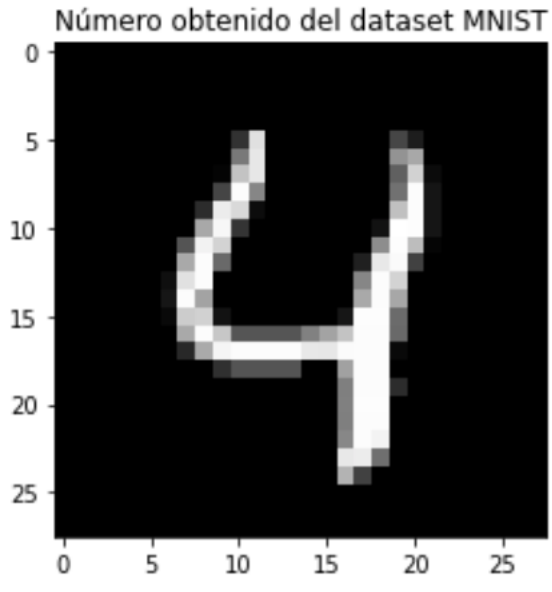
El auto-encoder es un algoritmo de aprendizaje que permite el preprocesamiento de los datos con el objetivo que la red aprenda la función identidad, es decir que la entrada sea igual a la salida, encontrandose que al realizar este preprocesamiento de los acoplamientos y partiendo de una red que ha sido autoaprendidas con un autoencoder el método del back propagation es más eficiente, eliminando el problema de la aleatoriedad de los pesos sinápticos iniciales, que hace tener una red mucho más lenta e ineficiente. En nuestro caso, implementamos una red feed-forward auto-encoder con una capa oculta, pero puramente con la finalidad que aprenda la identidad, utilizando la base de datos MNIST de dígitos de número escritos a mano. En la primera instancia, nuestra red tiene 784 unidades de entrada, una capa oculta de 64 neuronas y obviamente una capa de salida de 784 neuronas. Utilizando como algoritmo de optimización al descenso por gradiente estocástico (SGD), trabajando con un valor del learning rate de 10 (se decide trabajar con este valor del parámetro de aprendizaje porque cuando se usa uno de los valores dado por el estado del arte de entre 10^{-1} y 10^{-4}, la red es muy lenta para aprender), dándole un valor de 0,1 al parámetro de dropout, minitbatch de tamaño 1000, y entrenando a la red durante 10 épocas, obtenemos los siguientes resultados:

- Se adjunta en un primer gráfico los valores de la función error para el conjunto de entrenamiento como también para el conjunto de testeo en función del número de épocas (10):



Podemos notar que hasta la época 4 la caída de ambos errores es importante, pero a partir de esta su valor oscila cercano a los 0.8 y 0.82 respectivamente, para ambas curvas, que es un valor relativamente alto, y arroja problemas a la hora de replicar la imagen de entrada. Además se puede observar que a grandes razgos no se presenta el problema del overfitting, el cual ocurriría si a medida que pasan las épocas, el error de entrenamiento continúa con una tendencia decreciente pero el error de testeo cambia su tendencia y comenzara a aumentar, que implicaría que la red predice con alta precisión dentro del conjunto de entrenamiento pero es cada vez peor para predecir fuera de este, imposibilitando la generalización de forma correcta. (Hay que tener en cuenta que el número de épocas con el que se está trabajando es muy bajo, es posible que al aumentar el mismo si pudiera derivarse en un problema de overfitting).

- En los siguientes archivos se toma una imágen del conjunto de testeo y se la replica a través del autoencoder creado para ver que tan bien logra aprender la identidad.

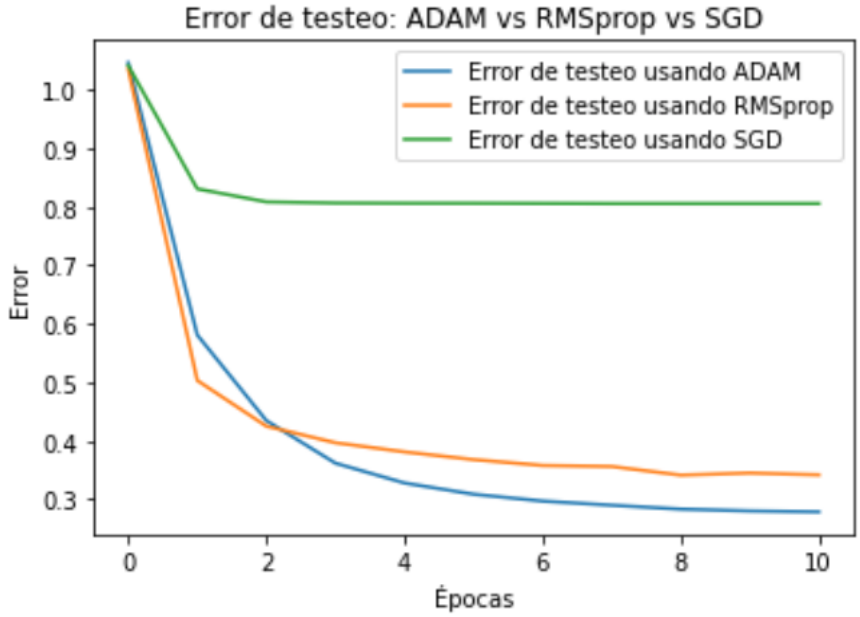


Se puede notar que la red no termina de aprender los valores de los píxeles de la imagen de entrada de la forma esperada y forma una especie de nube de puntos de distintos tonos pero que no se parece al 4 de la primera imagen.

Aumentando el número de neuronas de la capa oculta la red mejora en su proceso de aprendizaje de la identidad, pero todavía no logra los resultados esperados, por lo menos cuando se trabaja con 10 épocas. Una solución podría ser aumentar el número de épocas considerablemente y ver si se logran mejoras, pero dado el alto costo

computacional y de eficiencia, para solucionar este problema, se cambiará el algoritmo de optimización por RMSprop y por ADAM, viendo cuál de los 2 logra mejores resultados tomando como medida de comparación el error del conjunto de testeo. Los resultados obtenidos son los siguientes:

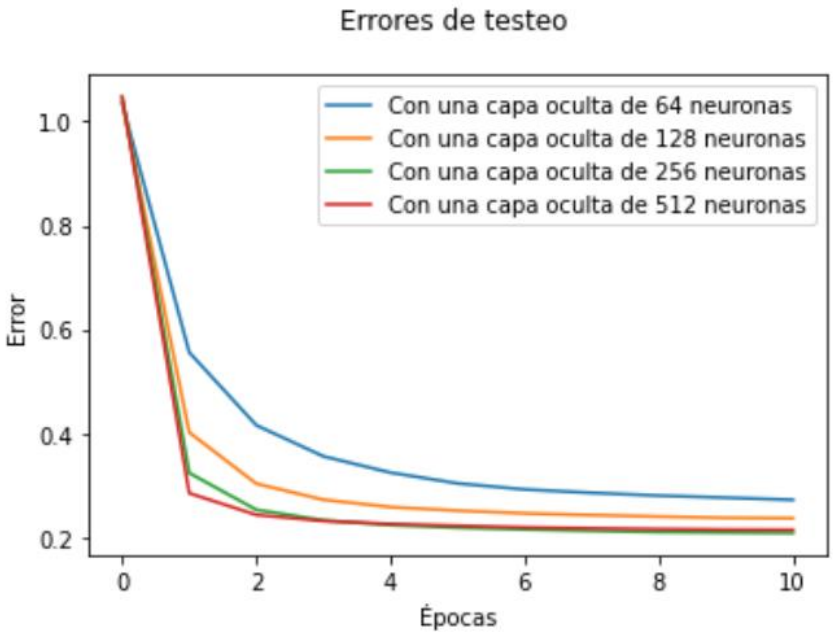
- Cuando se utiliza como algoritmo de optimización ADAM, se usa un learning rate de 10^{-3}, betas de (0.9, 0.999), y un épsilon de 10^{-8}, y cuando se usa RMSprop también se utiliza un learning rate de 10^{-3}, beta de 0.9 y un épsilon de 10^{-8}. Además también se incluirá en el gráfico el error de testeo utilizando SGD con los mismo valores de los parámetros antes indicados.
  Comparando los errores de testeo de todas las funciones de optimización se obtienen los siguientes resultados:

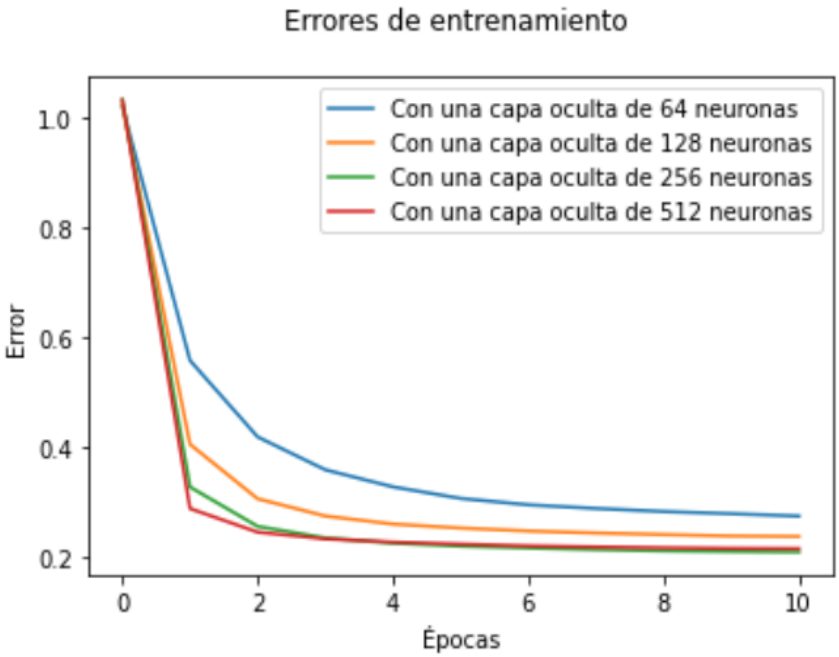


Puede observarse que con el algoritmo de optimización ADAM se logra el mínimo error de testeo cuando se trabaja con 10 épocas. Por esta razón se continuará trabajando con esta función de optimización en lo que resta del trabajo.

Así al replicar la identidad por medio del autoencoder y usando ADAM podemos ver una fuerte mejora en la red, como se nota en la siguiente imagen que se adjunta, donde se utiliza un autoencoder de 784 entradas, 64 neuronas en la capa oculta y 784 neuronas en la capa de salida, trabajando con learning rate de 10^{-3}, minitbatch de tamaño 1000, y entrenando a la red durante 10 épocas:

b) En este inciso lo que se hace es ir modificando la cantidad de neuronas de la capa oculta. Para esto se usarán capas de 64,128,256 y 512 neuronas respectivamente. Las funciones error tanto para el conjunto de entrenamiento, como para el conjunto de testeo arrojan los siguientes resultados:





Podemos notar que la red tanto para su proceso de testeo como de entranemiento desimuye el error a medida que se agregan neuronas a la capa intermedia, pero esto es muy notorio cuando la cantidad de neuronas pasan de 64 a 128, y a medida que van duplicandose las neuronas en la capa oculta se hace cada vez menos notorio este proceso, tal es así que para el caso de la red con una capa intermedia de 256 neuronas y de 512 neuronas los valores del error obtenidos son casi iguales.