Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba

Redes Neuronales
Red Feed-Forward Auto-Encoder

Gastón Briozzo
E-mail: gbriozzo@mi.unc.edu.ar
Fecha de entrega: lunes 16 de noviembre del 2020

Resumen
El objetivo de este trabajo es estudiar las redes neuronales Feed-Forward Auto-Encoder, que comprimen y reconstruyen el input para aprender donde reside la información más relevante.
Para esto se programó en Python a través de Google Colab una red con una única capa intermedia que, empleado la función error cuadrático medio para realizar un descenso por el gradiente estocástico, aprendiera la función identidad sobre el conjunto de datos MNIST de dígitos escritos a mano y digitalizados. Se estudió el comportamiento de la red para distintos valores de sus hiperparámetros, repitiendo el procedimiento para distintos números de neuronas en la capa intermedia, estudiando la dependencia de la función error con estas.
Se logró entrenar exitosamente a la red en la función identidad, encontrándose los valores de los hiperparámetros que optimicen su funcionamiento.

---

1. Introducción

Una red Feed-Forward es un modelo de red neuronal formada por una capa de entrada, por donde la red recibe la información, algún número de capas intermedias u ocultas y una capa de salida, donde la información se transfiere siempre de una capa a las siguientes, imposibilitando la creación de ciclos de retroalimentación

Un Auto-Encoder es generalmente una red neuronal Feed-Forward que tiene como objetivo aprender que parte de la información de un conjunto de datos es realmente relevante, generando nuevos datos primero comprimiendo la entrada en un espacio de variables latentes y luego reconstruyéndola en la salida en base a la información adquirida.

2. Procedimiento

Se programó en Python una red Feed-Forward Auto-Encoder mediante Google Colab, empleando las librerías de Google necesarias. La red consta de una capa de entrada de 784 neuronas, una capa oculta de 64 neuronas y una capa de salida de 784 neuronas, y emplea la función de error cuadrático medio para realizar un descenso por el gradiente estocástico.

Se importó la base de datos MNIST de números escritos a mano y digitalizados, que consta de 60000 imágenes en el conjunto de entrenamiento y 10000 en el conjunto de prueba, cada una de 28 por 28 píxeles de tamaño. Con estos datos se entrenó la red en la función identidad, implementando un dropout con p=0.01 y minibatch de tamaño 1000.

A fin de encontrar la mejor configuración, el entrenamiento se realizó para distintos valores de los hiperparámetros de la red, seleccionando aquellos que mejoraran su funcionamiento.

Una vez seleccionados los hiperparámetros, se repitió el procedimiento cambiando el número de neuronas en la capa oculta por 128, 256 y 512.

3. Resultados y discusión

3.1 Capa oculta de 64 neuronas

Se entrenó la red con una capa oculta de 64 neuronas sobre el conjunto de *train* de MNIST, para valores de learning rate (LR) de 0.1, 1 y 4, obteniéndose los valores para la función error cuadrático medio que se muestran en la *Figura 1*. Los valores de los demás hiperparámetros del modelo fueron momento de 0.5, 60 épocas de entrenamiento y batch-size-train de 64, según las recomendaciones de los docentes de la materia.

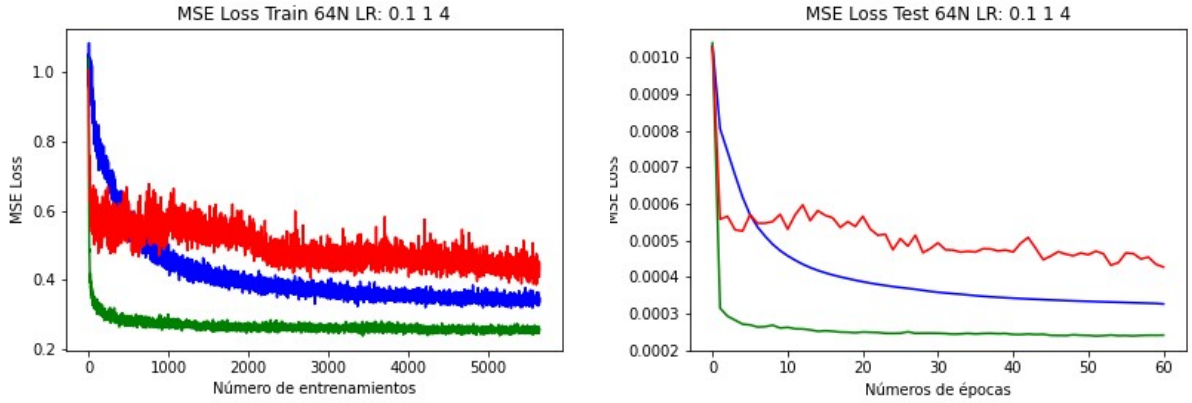

Figura 1: Función Error Cuadrático Medio (MSE) en función de las épocas. En azul se expresan los valores correspondientes a un LR de 0.1, en verde de 1 y en rojo de 4.

Podemos ver que para valores de LR del orden de 0.1 o menores, la curva de error es suave y decae constantemente, pero el aprendizaje es demasiado lento, haciendo que el coste computacional de entrenar a la red hasta un nivel adecuado sea excesivo.

Por su parte, vemos que para LR del orden de 1 la red aprende mucho más rápido, pero la curva de error no tarda en estancarse en torno a un valor fijo, presentando algunas "rugosidades" a medida que avanzan las épocas. Este estancamiento nos sugiere que conseguimos aproximarnos considerablemente al mínimo de la función error, mientras que las rugosidades podrían ser indicio de que la red está al límite de su capacidad y que aumentar el LR la desestabilizaría.

Efectivamente podemos ver que para LR mayor a 1, si bien la función error decae rápidamente al principio, no tarda en desestabilizarse y volverse sumamente rugosa, lo que indica que el tamaño de los pasos que estamos dando es mayor al de los mínimos que buscamos.

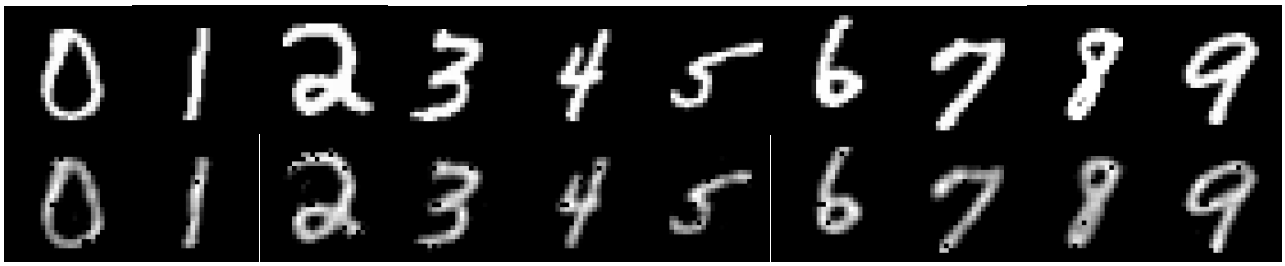

Figura 2: Entrada e imagen de la red. Las imágenes de arriba son las entradas originales mientras que las de abajo corresponden a la reconstrucción de la red. Los dígitos 0, 1, 5, 7 y 8 pertenecen al conjunto train, mientras que 2, 3, 4, 6 y 9 pertenecen al de test.

En vista de estos resultados, se continuó trabajando con un valor de LR de 1.

Se comprobó el funcionamiento de la red reconstruyendo un grupo de imágenes de entrada, tanto del conjunto train como del conjunto test. Los resultados se encuentran en la *Figura 2*, donde podemos ver que la red reconstruye correctamente las imágenes de entrada. Si bien existen algunas discrepancias entre los datos originales y su imagen, como la tonalidad del color, manchas o hasta algunos detalles eliminados, esto es de esperarse dado que al comprimir la información como lo hemos hecho se producen pérdidas que son imposibles de recuperar.

3.2 Capa oculta de 128, 256 y 512 neuronas

Manteniendo los valores de los hiperparámetros empleados en el punto anterior, y con un valor de LR de 1, se entrenó la red con 128, 256 y 512 neuronas en la capa oculta sobre el conjunto de *train* de MNIST, obteniéndose los valores para la función error cuadrático medio que se muestran en la *Figura 3*.

Se comprobó el funcionamiento de las redes reconstruyendo el conjunto de entradas que puede verse en la *Figura 4*.

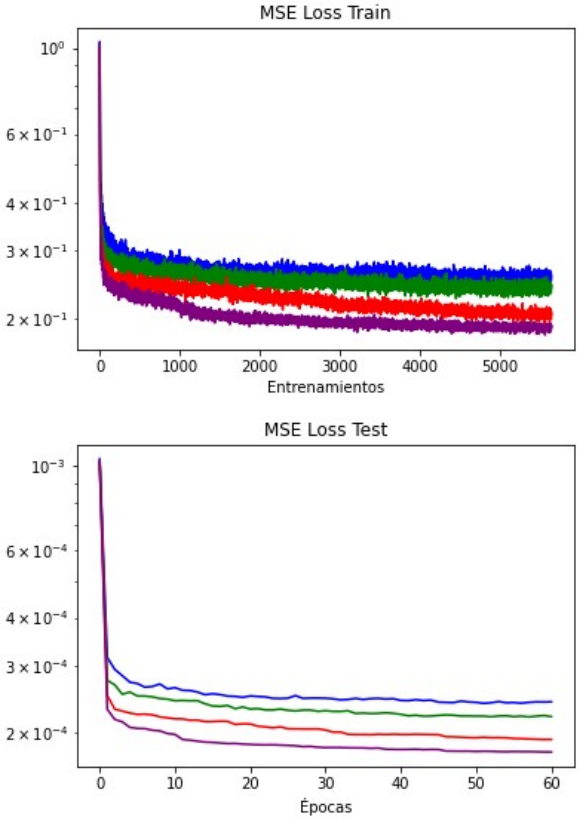

Figura 3: Función Error Cuadrático Medio (MSE) en función de las épocas. En azul se expresan los valores correspondientes a una capa oculta de 64 neuronas, en verde de 128, en rojo de 256 y en púrpura de 512.

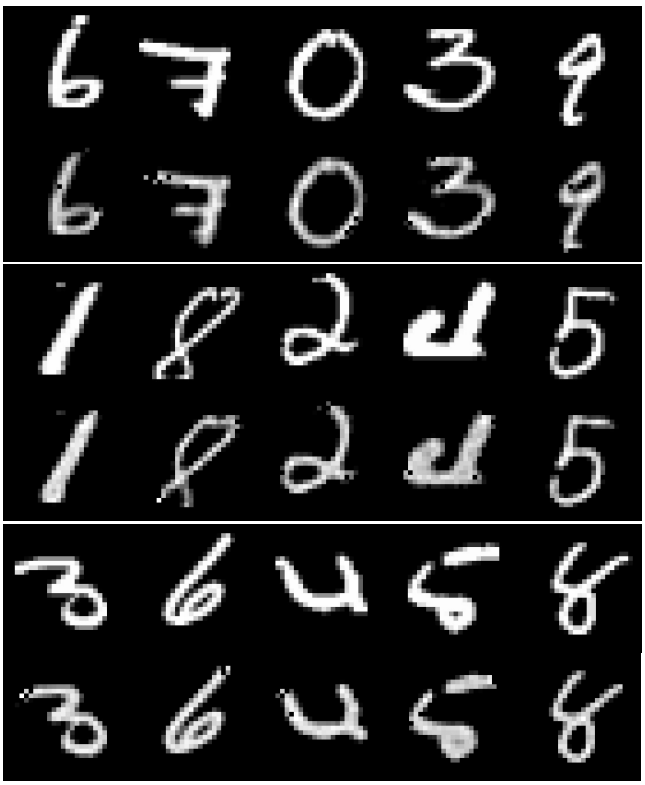

Figura 4: Entrada e imagen de las redes. Las imágenes de arriba corresponden a la red con capa oculta de 128 neuronas, las del medio de 256 y las de debajo de 512. Las dos primeras imágenes de cada grupo pertenecen a train mientras que las tres de la derecha son de test.

Como puede verse en la Figura 3, la curva de aprendizaje de la red es mejor mientras mas neuronas tenga en la capa oculta. Esto es de esperarse, ya que al no comprimir tanto la información lo lógico es que sobreviva una mayor parte de los datos iniciales, siendo más fácil reconstruirlos. Sin embargo, también podemos notar que estas mejoras pueden considerarse ínfimas tomando en cuenta que cada red duplica el número de neuronas, y por lo tanto el número de parámetros, de la capa oculta de la anterior. Lo mismo sugiere la Figura 4, donde podemos apreciar que, si bien hay mejoras entre una red y la siguiente, es discutible que justifique el duplicar los parámetros.

3.3 Aplicaciones

Antes de conseguir una buena configuración para nuestra red, se probó con otras menos afortunadas. Sin embargo, estas redes "imperfectas" dejan en evidencia una de las principales aplicaciones de las redes Auto-Encoder, la de filtrar el "ruido" de un conjunto de datos.

Figura 5: Entrada e imagen de la red "imperfecta".

Podemos ver que existe una gran pérdida de la información entre la entrada y su imagen. Nótese que las entradas del conjunto de entrenamiento 4, 9 y 7 producen imágenes casi idénticas, a pesar de ser claramente diferenciables. Lo mismo ocurre en menor medida con las entradas 3, 5 y 8, y con las entradas 2 y 6. Por su parte, las imágenes producidas para 0 y 1 son distinguibles. Esto se debe a que al comprimir tanto la información estamos forzando a la red a generar un pequeño grupo de soluciones genéricas, imágenes que se ajustan aproximadamente bien a una amplia variedad de entradas distintas.

En el caso 4, 7, 9, es evidente que la red esta distorsionando las entradas para hacer que encajen con una solución general. En los otros casos en que sucede esto, si bien las imágenes no son tan semejantes, podemos ver que parecen corresponder a ligeras variaciones de una misma solución general. También es interesante lo que sucede con la entrada 1, donde podemos ver que, a pesar de ser la entrada una línea casi vertical, la red tiende a inclinarla en el sentido en que, según la experiencia nos lo indica, tienden las personas a inclinar el 1 al escribirlo.

Esta red en particular no distingue correctamente los números, pero es posible que una red con mejores parámetros sea capaz de tomar una imagen contaminada y eliminar el ruido, devolviéndonos el número original

4. Conclusiones

Podemos notar la importancia de elegir correctamente los hiperparámetros de nuestra red para asegurarnos de que estamos trabajando en una configuración óptima.

También concluimos que, para el problema particular de reconstruir los datos del MNIST, 64 neuronas en la capa oculta son suficientes para aprender donde reside la información realmente relevante. Si bien con mayor número de neuronas en la capa oculta mejoran los resultados, estas mejoras no justifican los recursos que requieren, al mismo tiempo que sobredimensionar el problema agregando parámetros no es una práctica recomendable.

Por último, podemos ver que una red con bajos recursos (es decir, pocos parámetros) es útil para filtrar el ruido de una imagen, siempre y cuando no tenga carencias excesivas. Nuevamente, aumentar el número de parámetros podría arruinar esta propiedad, haciendo que nuestra red pase a ajustar el ruido en vez de los datos.

Bibliografía

[1] T.M. Mitchell, *Machine Learning,* McGraw-Hill (1997)
[2] https://www.famaf.unc.edu.ar/~ftamarit/redes2020/