Red feedforward sobre el dataset Fashion-MNIST

Martín Trucco[1]
Facultad de Matemática, Astronomía y Física - Universidad Nacional de Córdoba
Av. Medina Allende Nº2144, Ciudad Universitaria X5016LAE, Córdoba, Argentina
martintrucco@mi.unc.edu.ar[1]

1 de Noviembre, 2022

En el presente trabajo se empleó un Autoencoder de una sola capa oculta sobre el conjunto de datos Fashion-MNIST. Para esta capa se seleccionaron diferentes números de neuronas con el fin de observar la respuesta del modelo ante estos cambios y se graficaron los resultados obtenidos para el error promedio de entrenamiento y testeo en función de las épocas iteradas.

Descriptores: Autoencoder, red feedforward, redes neuronales.

In the present work, a single hidden layer Autoencoder was used on the Fashion-MNIST dataset. For this layer, different numbers of neurons were selected in order to observe the response of the model to these variations and the train and test average error were plotted as a function of the epochs.

Keywords: Autoencoder, feedforward network, neural networks.

Introducción

Un Autoencoder es una red neuronal artificial utilizada para la codificación de un conjunto de datos de entrada a fin de reducir su dimensionalidad. La forma de un Autoencoder es una red *feedforward*[1] con una capa de entrada -*inputs*-, una (o varias) capa oculta y una capa de salida -*outputs*- (ver figura 1). Esta última tiene el mismo número de neuronas que la capa de entrada con el propósito de reconstruir sus propios inputs, haciendo del Autoencoder una forma de aprendizaje no supervisado [1].

Son muy usados por ejemplo para clasificación, reducción de ruido en señales de audio e imágenes o en el reconocimiento facial [2].

En este trabajo se implementará una red feedforward Autoencoder con una capa oculta para aprender la función identidad con la base de datos Fashion-MNIST.

1 Marco Teórico

Los Autoencoders consisten en un codificador -*encoder*-, una capa oculta (menor que la de entrada para forzar al modelo a crear una representación comprimida de los inputs y no a usar la función identidad) y un decodificador -*decoder*-.

La transición desde la entrada a la capa oculta se llama *encoding* y se puede describir matemáticamente como un mapeo: sea $\phi : X \to Z$ una función definida entre los conjuntos X y Z respectivamente, si x es un elemento del dominio de ϕ (el vector de entrada), $x \mapsto \phi(x) = \sigma(Wx + b) := z$ donde W es una matriz de pesos, b un término de bias y σ una operación no lineal (sigmoide, tanh o ReLU).

Similarmente, la transición desde la capa oculta a la salida (*decoding*) puede identificarse con una función $\varphi : Z \to X$ tal que $z \mapsto \varphi(z) = \sigma(\tilde{W}x + \tilde{b}) := x'$.

El error e se calcula como la diferencia entre el vector de entrada original x y la señal reconstruida x', $e = x - x'$. De esta manera el Autoencoder aprende al reducir el Error Cuadrático Medio (MSE) [1][4].

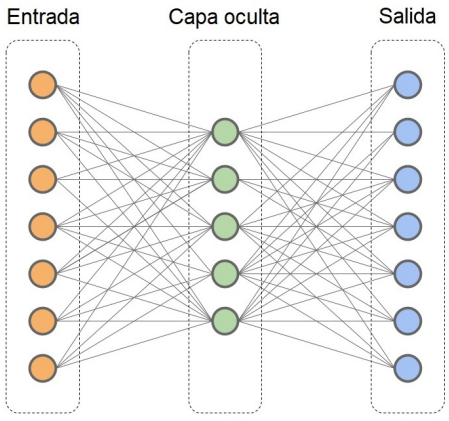

Figura 1: Arquitectura típica de un Autoencoder. Ilustración adaptada de [1].

[1] La información solo se mueve hacia adelante: de los nodos de entrada, a los nodos ocultos y hacia los nodos de salida [3].

2 Procedimiento

Se crea un Autoencoder con $28 \times 28 = 784$ unidades de entrada (y de salida) y una capa oculta de $L = 64$ neuronas. Luego se emplea el dataset Fashion-MNIST con los conjuntos de entrenamiento y testeo predeterminados y se utiliza el MSE como función de pérdida y el descenso por el gradiente estocástico (SGD) como optimizador.

Se grafica el error en función de las épocas para ambos conjuntos de datos de entrenamiento y testeo y se exponen imágenes del dataset utilizando el Autoencoder con las muestras pre y post entrenamiento para visualizar los cambios durante el procedimiento.

Finalmente se repiten los resultados variando el tamaño de la capa oculta para $L = 128, 256, 512$ y manteniendo fijos los demás parámetros.

3 Resultados y Discusión

En primer lugar se elige para el Autoencoder un dropout = 10% y una ReLU como función de activación. Además se fijan un minibatch de tamaño 1000, un learning rate = 1 y un número de épocas $n_E = 50$. Estos dos últimos surgen de probar repetidamente el modelo hasta encontrar aquellos valores más convenientes. Por ejemplo en el caso de n_E, de las figuras 2 y 3 se puede ver que el error disminuye muy poco a partir de las 50 épocas pero el costo computacional se duplica (ver tabla 1).

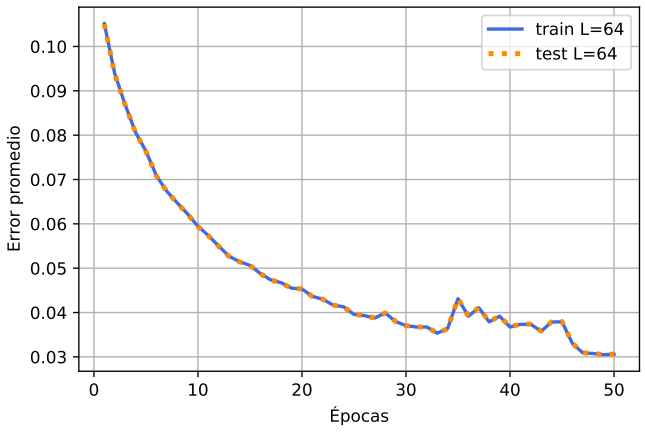

Figura 2: Error promedio de entrenamiento y testeo para 50 épocas en una red con $L = 64$ neuronas.

Número de épocas n_E	Tiempo [s]
50	709,99
100	1443,31

Tabla 1: Tiempo de ejecución del Autoencoder por n_E.

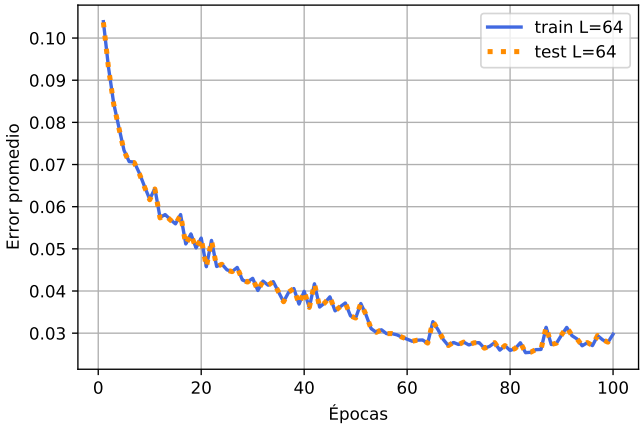

Figura 3: Error promedio de entrenamiento y testeo para 100 épocas en una red con $L = 64$ neuronas.

A continuación se grafican los errores en función de las épocas para los diferentes números de neuronas de la capa oculta: $L = 64, 128, 256, 512$ (ver figura 4). Seguidamente se presenta el valor de los errores para la primera y última época ejecutada y el tiempo computacional para cada L (ver tabla 2).

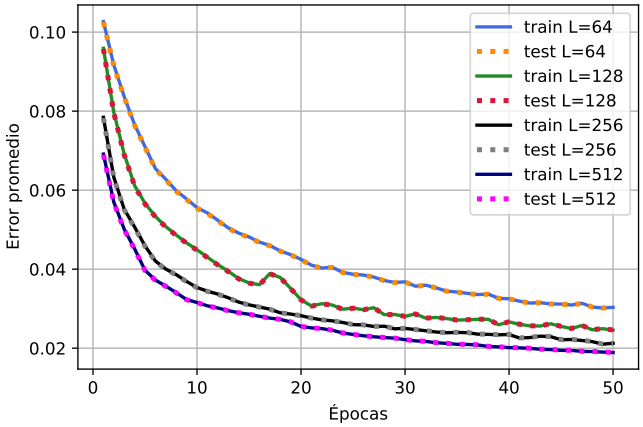

Figura 4: Error promedio de entrenamiento y testeo para 50 épocas en una red con $L = 64, 128, 256, 512$.

L	Error 1	Error 50	Tiempo [s]
64	0,103	0,030	683,99
128	0,096	0,025	761,28
256	0,078	0,021	919,42
512	0,069	0,019	1230,39

Tabla 2: Error de las épocas 1 y 50 y tiempo de ejecución del Autoencoder según L (para las 50 épocas).

Notar que el error se reduce al aumentar n_E y L a cambio de un costo en el tiempo de cálculo. Además, de la tabla 2 se puede advertir que los errores decaen en proporciones similares: un 30% para $L = 64$, 26%

(a) (b)

Figura 5: (a) Izq.: imagen original del dataset. Der.: predicción del Autoencoder pre-entrenamiento. (b) Izq.: imagen original del dataset. Der.: predicción del Autoencoder post-entrenamiento.

para $L = 128$, 26,9% para $L = 256$ y 27,5% para $L = 512$. Con todo esto, se puede pensar en los L intermedios (128 y 256) como aquellos con el mejor desempeño en relación al costo/error del modelo.

Por otra parte, es interesante observar que para cada uno de los L las curvas de error son muy similares, es decir, no existe sobreajuste. Esto se debe a que el Autoencoder tiene pocos parámetros de modo que su capacidad de ajuste se ve restringida.

Por último, en las figuras 5(a) y 5(b) se exponen imágenes originales del dataset junto a las predicciones del Autoencoder previo a su entrenamiento y después del mismo donde se contempla el significativo cambio luego del procedimiento.

4 Conclusión

Se realizó un análisis comparativo para esta red con diferentes tamaños de capa oculta en función de distintas variables tales como el número de épocas, el tiempo de ejecución o el error promedio de entrenamiento y testeo.

También se comprobó que las predicciones mejoran sustancialmente luego de que el modelo sea entrenado.

De esta forma, se puede concluir que la red feedforward Autoencoder se empleó de manera satisfactoria, reflejando la efectividad de la misma para aproximar la función identidad como era requerido. No obstante, se podría experimentar con otros optimizadores del tipo Adam o Adadelta, o en un ordenador con mayor capacidad de cálculo que la utilizada y comparar con los resultados obtenidos.

Bibliografía

[1] Oppermann, Artem. "Deep Autoencoders for Collaborative Filtering — Towards Data Science." *Medium*, 5 July 2021, https://towardsdatascience.com/deep-autoencoders-for-collaborative-filtering-6cf8d25bbf1d.

[2] Wikipedia contributors. "Autoencoder." *Wikipedia*, 5 Nov. 2022, https://en.wikipedia.org/wiki/Autoencoder.

[3] Colaboradores de Wikipedia. "Red Neuronal Prealimentada." *Wikipedia, La Enciclopedia Libre*, 24 Oct. 2022, https://es.wikipedia.org/wiki/Red_neuronal_prealimentada.

[4] K, Prashanth. "Autoencoders - Prashanth K." *Medium*, 27 Mar. 2018, medium.com/@prashanth9962/autoencoders-308f14bbbcf6.