Estudio de red neuronal *feed-forward auto-encoder* de una única capa oculta

Redes Neuronales - Trabajo Práctico 3

Matías Ezequiel González, Franco Nicolás Nieto

Estudiantes de grado de Cs. Físicas en la Facultad de Ciencias Exactas y Naturales (FCEN) - UBA

Segundo Cuatrimestre 2023, Cátedra Francisco A. Tamarit

Facultad de Matemática, Astronomía, Física y Computación (FAMAF) - UNC

Resumen

En este trabajo se estudia el proceso de entrenamiento de variaciones de una red neuronal *feed-forward auto-encoder* de una única capa oculta de n neuronas, con $n \in \{64, 128, 256, 512\}$. Como conjunto de entrenamiento se utilizó el *dataset* de imágenes de *Fashion-MNIST* [2]. El análisis se centró entorno a los efectos que produce aumentar el número de neuronas n como por ejemplo el sobre-ajuste observado en la red con $n = 512$ neuronas en la capa oculta. Esto se llevó a cabo estudiando el valor del Error Cuadrático Medio (*MSE*) calculado sobre los conjuntos de entrenamiento, validación y testeo. En un principio se utilizó el optimizador *SGD* pero fue contrastado con el algoritmo *Adam* y se observó que este último reducía el error *MSE* para todo n y por lo tanto se lo utilizó para gran parte del estudio.

1. Introducción

1.1. Motivación

La motivación de este trabajo es la generar una red neuronal *feed-forward auto-enconder* con la cual poder comprimir de manera eficiente las imágenes de la base de datos *Fashion-MNIST* como demostración de aprendizaje.

Este proceso fue llevado a cabo con la intención de familiarizarse con la estructura de la red neuronal ya previamente mencionada debido a su utilidad en el filtrado de imágenes para su subsiguiente clasificación.

Por último, la elección de esta base de datos de entrenamiento y validación no fue arbitraria sino que fue hecha para poner a prueba esta base de datos reemplazando a la usual *MNIST* en la tarea de evaluación de rendimiento de la red.

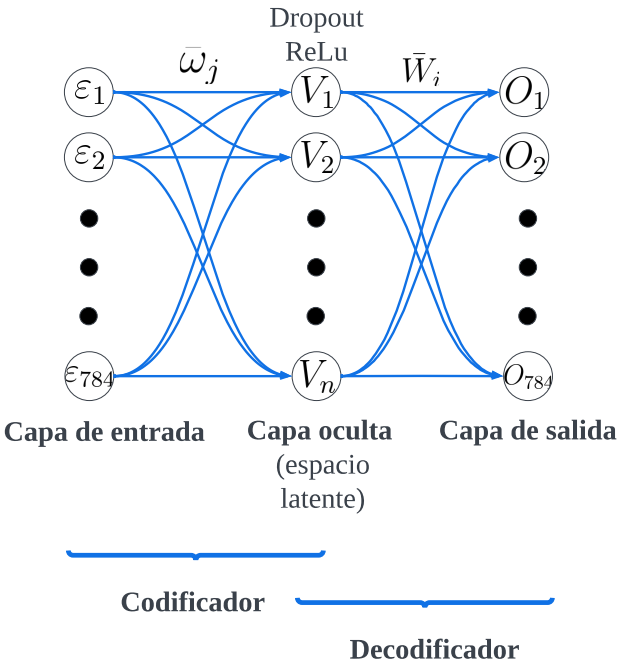

Figura 1: Esquema representativo de una red neuronal *feed-forward auto-encoder*. Se tiene una capa de entrada representada por ε, una capa oculta V y una capa de salida O conectadas entre sí por sinapsis con sus pesos ω y W. Se tiene un tamaño a la salida y entrada de 784 neuronas.

1.2. Marco teórico

1.3. Red neuronal *feed-forward*

Una red neuronal *feed-forward* (fig. 1) es un algoritmo dedicado al aprendizaje y resolución de tareas mediante el entrenamiento del mismo. La característica de *feed-forward* crea una discriminación en su estructura que hace referencia a una red neuronal cuya información fluye de entrada a salida consecutivamente a través de sus distintas capas siendo imposible omitir una, retroceder o ir hacia los costados de la estructura.

1.3.1. *Backpropagation*

Al momento de abordar el ajuste de pesos sinápticos en la red desarrollada en este trabajo, se utilizará el método de *backpropagation*. Es decir que la asignación de peso sináptico empezará desde los resultados

obtenidos, el último eslabón de nuestra red, para terminar en la primer sinapsis.

Para formalizar la idea de *backpropation* tenemos que cuantificar ciertas partes de la red como la función error (ec. 1) y salida de la capa salida (ec. 2).

$$E[\bar{\omega}] = \frac{1}{2}\sum_{\mu=1}^{P}\sum_{i=1}^{N}[J_i^\mu - O_i^\mu]^2 \tag{1}$$

$$O_i^\mu = g_2(\sum_{j=1}^{L} W_{ij} \cdot g_1(\sum_{k=1}^{N} \omega_{jk} \cdot \varepsilon_k^\mu)) \tag{2}$$

Teniendo estas dos últimas expresiones podemos notar que la función error, y por lo tanto el proceso de minimizarla, dependen de los pesos sinápticos (W_{ij} y ω_{ij}) por lo tanto tendremos que aplicarles variaciones con la intención de disminuir la función error:

$$\Delta W_{ij} = n \cdot \sum_{\mu=1}^{P}[J_i^\mu - O_i^\mu] \cdot g_2(h_i^\mu) \cdot V_j^\mu \tag{3}$$

$$\Delta\omega_{ij} = n \cdot \sum_{\mu=1}^{P}[J_i^\mu - O_i^\mu] \cdot g_2(h_i^\mu) \cdot W_{ij} \cdot g_1(h_i^\mu) \cdot \varepsilon_k^\mu \tag{4}$$

Como podemos apreciar, el ajuste del último peso sináptico (ec. 3) depende de los resultados de la última capa y el ajuste del primer peso sináptico (ec. 4) depende del último ajuste dando esta estructura de función compuesta de atrás hacia delante.

1.4. Red neuronal *autoencoder*

El tipo de red neuronal *autoencoder* es una clasificación de la ya mencionada red *feed-forward* la cual tiene la particularidad de que el espacio vectorial de inicio (*input*) y el espacio vectorial de salida (*output*) tienen las mismas dimensiones. Además en los procesos alrededor de las capas ocultas se realiza la compresión y descompresión de la información (actividad la cual le da el nombre a la estructura).

Esta última característica se debe a que el modelo entrena y aprende detectando patrones en el espacio latente (espacio de información comprimida) con el fin de encontrar patrones y reducir el error.

1.5. Función error: *MSE*

La función error ya fue mencionada en el apartado relacionado a *backpropagation*, sin embargo es necesario desarrollar algunos de sus componentes y/o ideas conceptuales.

La función de Error Cuadrático Medio, o *MSE* por sus siglas en inglés, se utilizó en este trabajo para calcular el error cometido por lo predicho por el modelo (O_i^μ) calculando la diferencia con el valor esperado (J_i^μ).

Cabe aclarar que en el trabajo se tendrán en cuenta dos errores distintos que difieren en el *dataset* utilizado para calcularlos: el error de entrenamiento (E_{in}) y el error de validación (E_{out}). El primero cuantifica que tan lejos están los resultados predichos de los esperados cuando como entrada se pasan elementos del *dataset* de entrenamiento y el otro el error que comete el modelo al enfrentarse a nuevos datos del *dataset* de validación. Ambos errores serán útiles para estudiar la adaptabilidad del modelo a nuevos datos y la posibilidad de sobre-ajustar los datos de entrenamiento lo que significa que la red comete un error considerablemente menor en E_{in} que en E_{out}. Esto quiere decir que la red comienza a "memorizar"los datos de entrenamiento mientras que lo que se desea es que sea capaz de reconocer patrones generales frente a muestras nuevas.

1.6. Estructura y capas de la red

En el presente trabajo se utilizó una red *feed-forward auto-encoder* de una única capa oculta cuyo número de neuronas n se varió entre los valores $\{64, 128, 256, 512\}$ y se analizó su efecto. Las entradas de esta red fueron imágenes en escala de grises de 28×28 píxeles convertidas a vectores unidimensionales de 784 elementos cuyo valor varía entre 0 y 255. Estas imágenes fueron extraídas de la base de datos *Fashion-MNIST*, 50.000 de estas se utilizaron en el entrenamiento, 10.000 en la validación y otras 10.000 en el testeo para contrastar el desempeño de las redes. El entrenamiento se llevó a cabo en 30 épocas para todas las redes analizadas en el presente trabajo. La red fue compuesta utilizando las siguientes funciones de la librería *PyTorch*:

Flatten: función encargada es convertir la entrada de una matriz de 28×28 en un array unidimensional de 784 elementos más apropiado para el procesamiento.

Linear: función encargada de la generación y conexión completa entre las 784 neuronas de la capa inicial con las n de la capa oculta y estas con las 784 neuronas en la capa final.

ReLu: función encargada de introducir una no linealidad al aprendizaje y procesamiento de datos.

Dropout: función con el fin de deshabilitar neuronas durante el aprendizaje para introducir estocasticidad al entrenamiento. La probabilidad de dropout fue ajustada al 10 % y deshabilitada fuera del entrenamiento.

1.7. Optimizador: *SGD* y *Adam*

El optimizador utilizado inicialmente en este trabajo es el llamado *SGD* o Descenso por Gradiente Estocástico. La principal función de este optimizador es la de reducir el error (ec. 1) mediante la variación del peso sináptico de las conexiones entre capas. Esto último es conseguido mediante la iteración de un *dataset* de entrenamiento estocásticamente.

La actualización de estos pesos sinápticos se da entre épocas de entrenamiento y se basa en la dirección del gradiente calculado en *minibatches* de 1000 imágenes.

Los parámetros utilizados en la *SGD* fueron: *learning rate = 1; momentum = 0,9 y dampening = 0,3*. Por otro lado, también se experimentó cambiando al optimizador *Adam* [1] ya que, como se discutirá más adelante, con este se lograba un menor error.

2. Desarrollo y Resultados

2.1. Red con $n = 64$ neuronas ocultas

Para analizar el progreso del entrenamiento de la red con $n = 64$ neuronas en la capa oculta se muestran en la figura 4 las curvas de *MSE* calculadas sobre el conjunto de entrenamiento y de validación en función del número de época. Se puede observar que la curva es decreciente y que no se alcanza un mínimo absoluto donde la *MSE* comience a crecer nuevamente.

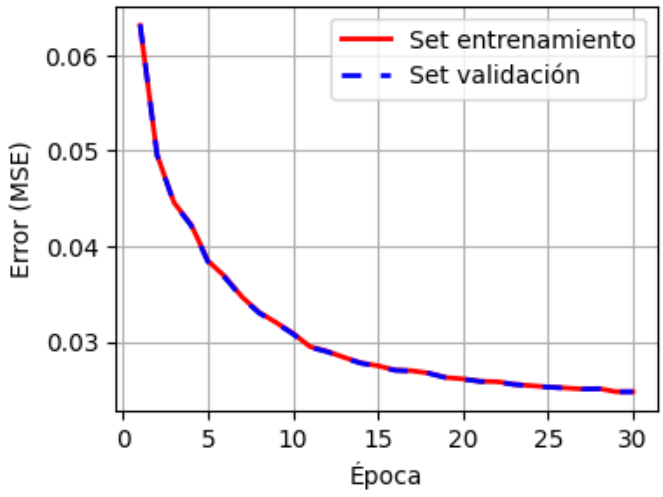

Figura 2: Error Cuadrático Medio de una red de $n = 64$ neuronas en la capa oculta entrenada con el algoritmo de optimización *SGD*.

Por otra parte, el error calculado sobre el conjunto de entrenamiento es idéntico al error sobre el conjunto de validación ya que ambas curvas están superpuestas en todo momento. Esto indica que en ningún momento se produce un sobre-ajuste.

Estas dos razones indican que 30 épocas no fueron suficientes para alcanzar el máximo potencial de la red.

2.2. SGD vs Adam

Para poder decidir qué algoritmo de optimización utilizar para lo que sigue del análisis se entrenó el modelo para cada uno de los n con ambos algoritmos por separado. En esta sección analizaremos cuál dio mejores resultados a partir su evaluación sobre el conjunto de testeo y calculando el error cometido por cada uno. Los valores de *MSE* se pueden observar en la figura 3.

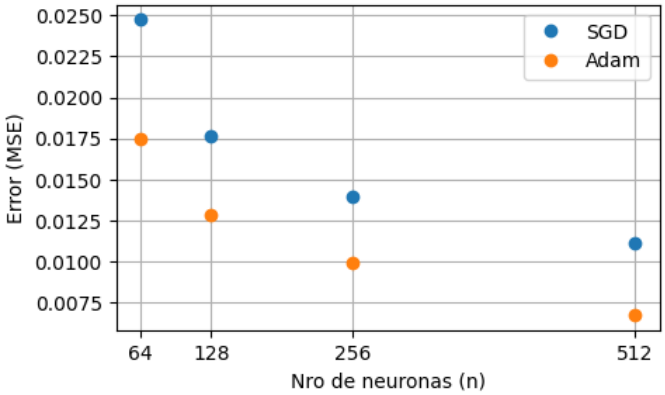

Figura 3: Error Cuadrático Medio calculado con el conjunto de testeo para redes de diferente cantidad de neuronas n en la capa oculta, entrenadas tanto con el algoritmo de optimización *SGD* como *Adam*.

Como se puede observar, para cualquier n el *MSE* es siempre menor para la red que utiliza el algoritmo *Adam* que para aquella que utiliza el *SGD* y por esta razón de aquí en adelante se utilizará el primer algoritmo. Como se comentará luego, esto permitirá observar un caso de sobre-ajuste en la red de $n = 512$ entrenada con *Adam* que no se da en las redes que resultaron de usar *SGD*. Sin embargo, a pesar de este criterio vale la pena destacar que ambos algoritmos resultaron en implementaciones que funcionan satisfactoriamente como se puede observar con ejemplos en el Apéndice.

En la figura 3 también se puede notar que a medida que se agregan neuronas en la capa oculta la red reduce su error. Esto último es debido a que logra transmitir mayor detalle de la entrada hacia la salida debido a que no necesita comprimir tantas dimensiones al pasar por el cuello de botella que genera la capa oculta.

2.3. Red con $n \geq 64$ neuronas ocultas

Para analizar el progreso del entrenamiento de cada red se muestran en la figura 4 las curvas de *MSE*. Al igual que para $n = 64$, en ninguna se alcanza un

mínimo absoluto donde la *MSE* comience a crecer nuevamente, lo que indica nuevamente que 30 épocas no fueron suficientes para alcanzar el máximo potencial de ninguna de las redes.

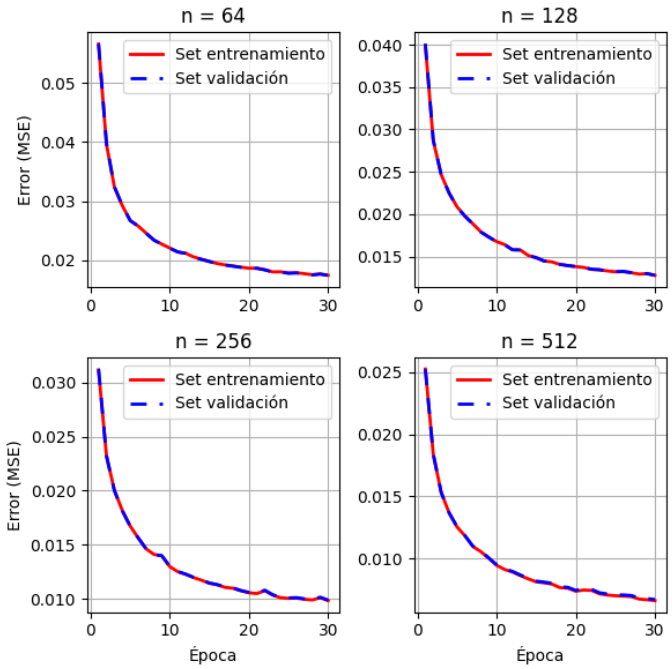

Figura 4: Error Cuadrático Medio, calculado sobre el conjunto de entrenamiento y el de validación en función de la época para diferentes n. Todas fueron entrenadas con el algoritmo de optimización *Adam*.

Se ve también cómo en ningún caso se produce un sobre-ajuste considerable, esto es debido a que la red no tiene la complejidad necesaria para lograr aprender o "memorizar"los detalles del conjunto de entrenamiento y sobre-ajustar estos datos. Para respaldar esta hipótesis se compara en la figura 5 la red más pequeña de $n = 64$ con la más compleja de $n = 512$, donde se aumenta la escala del eje de *MSE* en ambos casos.

En esta figura, con esta escala, se observa que a partir de un $MSE = 0,008$, la red de $n = 512$ presenta una separación entre las curvas de validación lo que indica un ligero sobre-ajuste, mientras tanto, esta separación en la red con $n = 64$ resulta imperceptible. Esto verifica que a medida que aumenta el número de neuronas en la capa oculta o su complejidad, se da lugar al sobre-ajuste. Notar que este fenómeno no se apreciaría con la red de $n = 512$ entrenada a partir de *SGD* ya que esta alcanza un $MSE > 0,01$ como se observa en la figura 3.

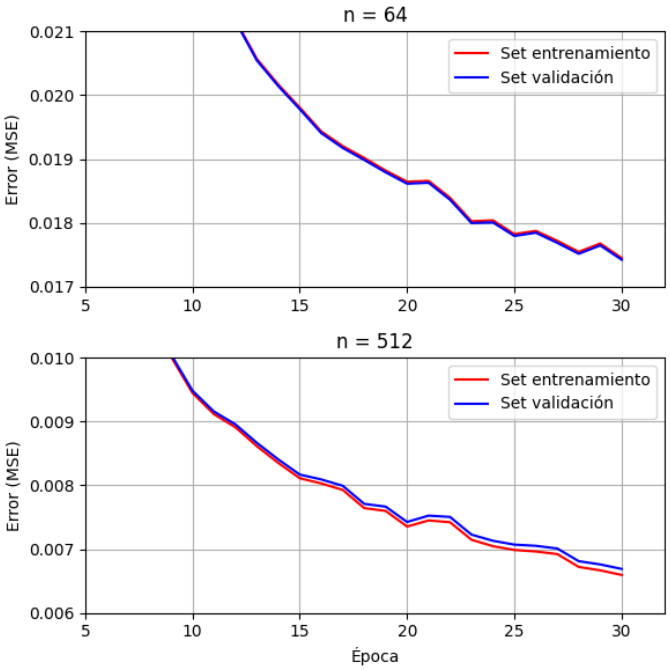

Figura 5: Error Cuadrático Medio, calculado sobre el conjunto de entrenamiento y el de validación en función de la época para $n = \{64, 512\}$. Todas fueron entrenadas con el algoritmo de optimización *Adam*.

esto se utilizaron dos algoritmos optimizadores por separado para poder compararlos, el algoritmo *SGD* y el *Adam* , llegando a la conclusión de continuar el análisis con *Adam* ya que presentaba un menor Error Cuadrático Medio durante el testeo en todos los casos.

Para analizar el entrenamiento de las redes se observó la curva de *MSE* en función de las épocas. Esta en todos los casos fue monótonamente decreciente y no llegaba a un mínimo absoluto por lo que se concluye que al menos con 30 épocas no fue suficiente para alcanzar el máximo desempeño del modelo. En ninguno de estos gráficos se observó un sobre-ajuste apreciable con la excepción de la red de $n = 512$ entrenada con *Adam*, donde al aumentar la escala de *MSE* se observó que a partir de $MSE = 0,008$ la curva de error calculada con el conjunto de entrenamiento comenzaba estar por debajo de aquella calculada con el de validación indicando un leve sobre-ajuste.

3. Conclusiones

En el presente trabajo se lograron entrenar diversas redes neuronales *feed-forward autoencoder* de una única capa oculta con $n \in \{64, 128, 256, 512\}$ neuronas utilizando la base de datos *Fashion-MNIST*. Para

Referencias

[1] Cornell University, *Adam*, 2003, https://optimization.cbe.cornell.edu/index.php?title=Adam

[2] Fashion-MNIST, 17/2/2024. https://github.com/zalandoresearch/fashion-mnist

Apéndice

3.1. *SGD* vs *Adam*: ejemplos

Como se comentó en la sección 2.2 se entrenaron redes de distintos numeros de neuronas. Para esto se usaron tanto el algoritmo *SGD* como el *Adam* y, si bien se decidió continuar el análisis con el segundo optimizador, ambas alternativas dieron como resultado implementaciones útiles como se puede observar en la figura 6.

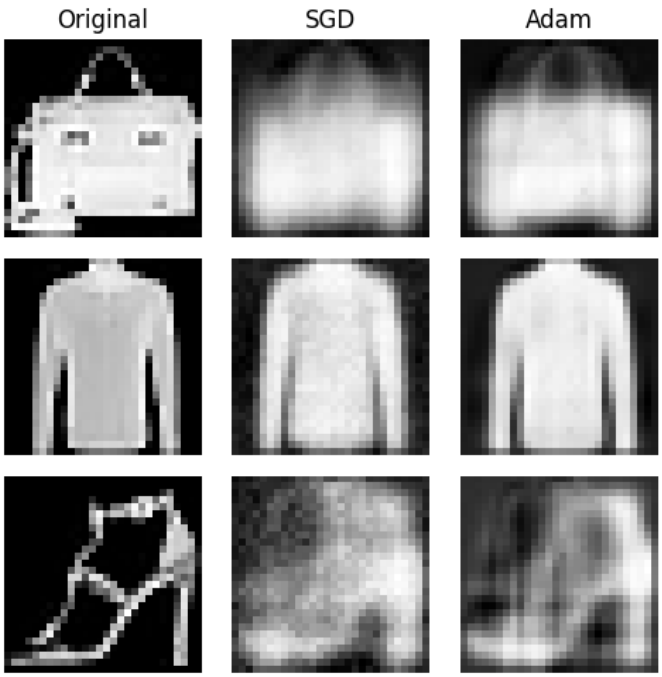

Figura 6: Comparación del filtrado de imágenes generado por dos redes de una capa oculta de $n = 64$ neuronas, una entrenada usando el algoritmo *SGD* y la otra con *Adam*.

En estas imágenes se compara el resultado del filtrado realizado por dos redes de $n = 64$ neuronas ocultas entrenadas cada una con un optimizador distinto. Se logra ver que luego de comprimir la imagen original la salida de ambas redes guarda similitud con la entrada. Adicionalmente, se hace evidente que a pesar de tener la misma cantidad de neuronas y de haber sido entrenadas la misma cantidad de épocas el resultado obtenido a partir de *Adam* logra preservar un mayor nivel de detalle respecto de la entrada reflejando un menor *MSE* .