Universidad Nacional de Colombia
Facultad de Ciencias - Departamento de Matemáticas
Álgebra Lineal Numérica - 2025 I

Mateo Sebastian Ortiz Higuera mortizhi@unal.edu.co

Reporte Proyecto
Estudio de Redes Neuronales Ortogonales

Resumen

En el presente trabajo se hace una revisión de una aproximación de las Redes Neuronales Ortogonales, en función de esto primero explicamos desde el punto de vista geométrico y desde la teoría del aprendizaje automático el funcionamiento de una Red Neuronal Profunda. Para esta explicación y para los experimentos posteriores seleccionamos el conocido problema de clasificación de dígitos utilizando el dataset de MNIST. Posteriormente implementamos la aproximación de una Red neuronal ortogonal utilizando el algoritmo de regularización ortogonal; analizamos los resultados de estabilidad numérica en el entrenamiento, tiempos de entrenamiento y de generalización entre cinco modelos entrenados con distintos valores de penalización. Los resultados arrojan que la selección del hiperparámetro del valor de regularización es vital para obtener un modelo con un balance adecuado entre generalización y estabilidad numérica en el entrenamiento.

1. Introducción

Las Redes Neuronales Profundas como agentes de entrenamiento automático han demostrado ser bastante útiles para modelar funciones y relaciones complejas. Sin embargo, el entrenamiento efectivo de estas arquitecturas, especialmente a medida que aumenta su profundidad, presenta desafíos numéricos y de optimización no triviales. En la literatura se discute ampliamente cómo problemas como la explosión y el desvanecimiento del gradiente pueden dificultar la convergencia del modelo, mientras que el sobreajuste (overfitting) limita su capacidad para generalizar ante datos no vistos[2].

En este contexto, la imposición de restricciones de ortogonalidad sobre las matrices de pesos surge como una solución elegante y geométricamente fundamentada. Las Redes Neuronales Profundas Ortogonales (OrthDNNs) buscan garantizar que las transformaciones lineales entre capas preserven la norma de los vectores (isometría) y los ángulos entre ellos. Como señalan Li et al. [1], esta propiedad no solo mitiga los problemas de estabilidad numérica asegurando un flujo de gradiente más estable durante la retropropagación, sino que también favorece la generalización al decorrelacionar las características aprendidas por el modelo.

En este reporte abordamos el estudio y la implementación de las OrthDNNs conectando herramientas del Álgebra Lineal Numérica, específicamente las transformaciones de Householder y la descomposición QR, con la optimización de redes neuronales. A diferencia de un enfoque puramente teórico, centramos

nuestro trabajo en la experimentación práctica aplicada al problema de clasificación de imágenes utilizando el dataset MNIST.

El objetivo principal es evaluar el impacto de la regularización ortogonal en el proceso de aprendizaje. Para ello, implementamos una arquitectura base y comparamos su desempeño bajo distintos coeficientes de penalización (λ). A través de estos experimentos, analizamos no solo la precisión (accuracy) y la función de pérdida, sino también métricas críticas como el error de ortogonalidad por capa y el número de condición de las matrices de pesos. De esta forma, buscamos determinar empíricamente si la teoría se traduce en una ventaja práctica y cuál es el balance costo-beneficio entre la estabilidad numérica y el tiempo de cómputo adicional que estas restricciones imponen.

2. Preliminares

Para entender las Redes Neuronales Profundas Ortogonales primero vamos a explicar las Redes Neuronales Superficiales/Profundas.

2.1 Redes Neuronales

2.1.1 Redes Neuronales Superficiales

Matemáticamente, una *Red Neuronal Superficial* es una familia de modelos no lineales parametrizados por unos pesos ϕ_i, $w_{ij} \in \mathbb{R}$ y un sesgo $b_i \in \mathbb{R}$, de la forma

$$F(\mathrm{x}) = \phi_0 + \sum_{i=1}^{n} \phi_i a \left[\sum_{j=1}^{d} w_{ij} + b_i \right] \tag{1}$$

en donde hay n neuronas o unidades ocultas, $\mathrm{x} \in \mathbb{R}^d$ y a es una función de activación que induce no linealidad. Por ejemplo, supongamos que la red neuronal tiene como entrada tres valores, tres unidades ocultas y predice dos valores, podemos denotar

$$h_i := a(w_{i1}x_1 + w_{i2}x_2 + w_{i3}x_3 + b_i)$$

Las salidas del modelo entonces es de la forma (2), (3).

$$y_1 = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3 \tag{2}$$
$$y_2 = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3 \tag{3}$$
$$\mathrm{y} = \Phi_0 + \Phi_1 h_1 + \Phi_2 h_2 + \Phi_3 h_3 \tag{4}$$

La forma de la ecuación(4) representa de forma matricial las mismas salidas del modelo, en donde $\Phi_i = \begin{bmatrix} \phi_{1,i} \\ \phi_{2,i} \end{bmatrix}$ y $i = 0, 1, 2, 3$.

Y, usualmente, podemos denotar este modelo a través de un grafo computacional. En este ejemplo el grafo computacional es como se ve en la Figura 1

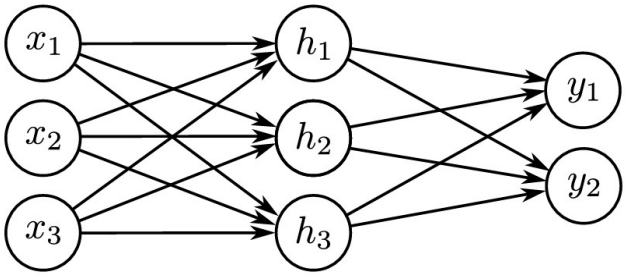

Figura 1: Grafo computacional del modelo correspondiente a la ecuación (4). Gráfica obtenida de [2].

2.1.2 Redes Neuronales Profundas

Podemos extender esta noción añadiendo más capas, así tenemos una *red neuronal profunda.* Esto se hace realizando una composición de redes neuronales superficiales. Los nuevos pesos serán matrices $W_i \in \mathbb{R}^{\ell \times d}$ y los sesgos serán vectores $b_i \in \mathbb{R}^d$, en donde i es la capa actual, ℓ es el número de neuronas de la capa anterior y d es el número de neuronales de la capa actual. Dicha familia se puede ver de la siguiente forma:

$$F(\mathrm{x}) = b_k + W_k a_{k-1}\left[b_{k-1} + W_{k-1} a_{k-2}\left[\cdots a_2\left[b_2 + W_2 a_1\left[b_1 + W_1 \mathrm{x}\right]\right]\cdots\right]\right] \tag{5}$$

o recursivamente, definiendo

$$h_i := \begin{cases} W_i a_{i-1}\left[b_{i-1} + W_{i-1} h_{i-1}\right] & \text{si } i > 1, \\ \mathrm{x} & \text{si } i = 1. \end{cases}$$

en donde a_i corresponde a una función de activación para cada i-ésima capa. Entonces la ecuación (5) podemos reescribirla de la forma

$$F(\mathrm{x}) = W_k h_k + b_k \tag{6}$$

Tomando el ejemplo anterior, supongamos ahora que tiene dos capas ocultas más, como se ve en la ecuación (7) y en la figura 2.

En donde se tiene que $W_4 \in \mathbb{R}^{2\times 4}$, $W_3 \in \mathbb{R}^{4\times 3}$, $W_2 \in \mathbb{R}^{3\times 5}$, $W_1 \in \mathbb{R}^{3\times 5}$, $b_4 \in \mathbb{R}^2$, $b_3 \in \mathbb{R}^4$, $b_2 \in \mathbb{R}^3$, $b_1 \in \mathbb{R}^5$ y su función sería

$$\mathrm{y} = b_4 + W_4 h_4 \tag{7}$$

En el aprendizaje automático, optimizamos una función de pérdida $\mathcal{L}$ para encontrar los valores de W_i y b_i para que el modelo pueda predecir con muestras no vistas anteriormente.

2.2 Problema de Clasificación de Dígitos

Para este trabajo tomamos el problema de la clasificación de dígitos en imágenes: supongamos que queremos crear una red neuronal que quiere clasificar imágenes del dataset MNIST (imágenes de

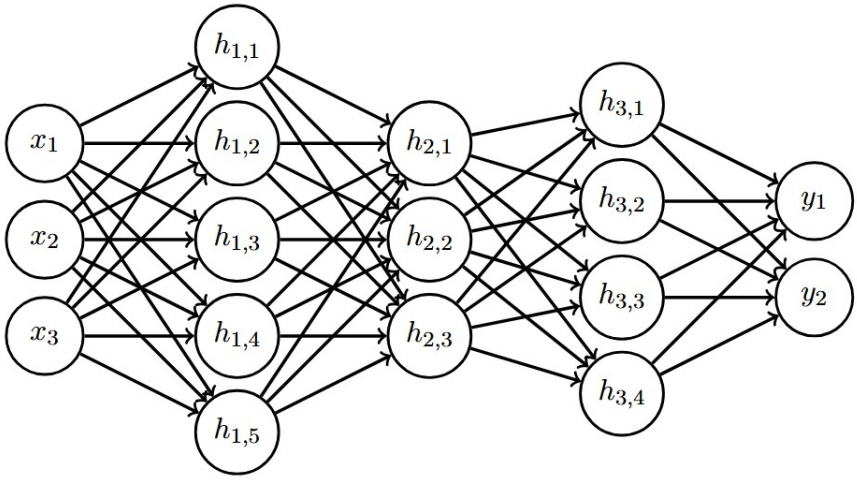

Figura 2: Grafo computacional correspondiente al modelo de la ecuación (7).

número escritos a mano). En donde cada imagen tiene un tamaño de 28×28 píxeles, que en total equivalen a 784 píxeles. Podemos proponer una primera red neuronal profunda que tenga como entrada los 784 píxeles, con dos capas ocultas y una capa final con 10 salidas, que representan las probabilidades para cada una de las clases (los dígitos del 0 al 9).

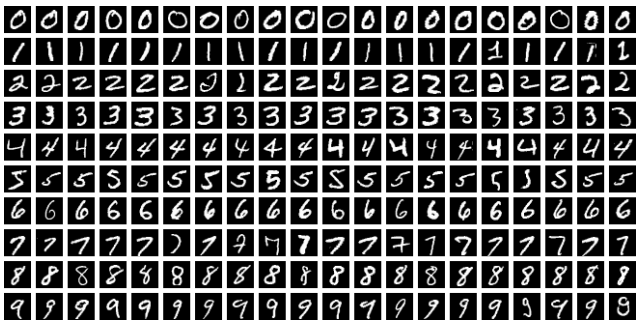

Figura 3: Dataset de MNIST para la clasificación de dígitos escritos a mano.

El grafo computacional referente a esta propuesta podría ser como se ve en la figura 4.

En este modelo, la primera capa representa cada píxel de la imagen, las dos capas ocultas extraen información de los píxeles (patrones de las imágenes) y la última funciona para predecir el dígito de la imagen. Entrenamos el modelo para ajustar los valores de las matrices de pesos W_1, W_2 y W_3 para que el modelo aprenda a representar y predecir cada dígito.[1]

Debe notarse que las imágenes de, por ejemplo, el dígito 2, van a encender las mismas neuronas a lo largo de las capas. Ya que los valores de la entrada serán muy similares, y así debería ser a lo largo del cómputo del grafo computacional. Esto da sentido a las redes neuronales profundas ortogonales,

[1]Puede ver este ejemplo en el siguiente enlace `https://adamharley.com/nn_vis/mlp/2d.html`.

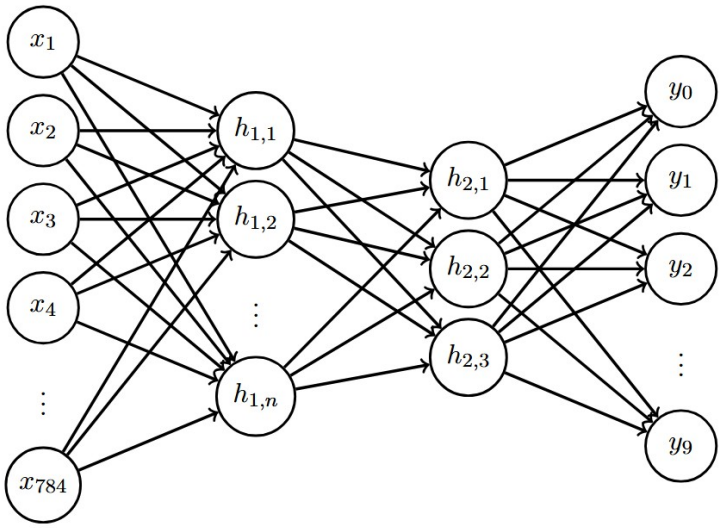

Figura 4: Grafo computacional del modelo propuesto para el problema de clasificación de imágenes.

como veremos a continuación; si conseguimos que los valores de las matrices de pesos W_i logren preservar las relaciones espaciales y los patrones detectados a través de las capas, entonces estos contribuyen a mejorar las predicciones de una red neuronal.

2.3 Redes Neuronales Profundas Ortogonales

Cuando nos referimos a las *redes neuronales profundas ortogonales* hablamos de redes neuronales profundas cuyas matrices de pesos W_i son ortogonales. Esta propiedad garantiza isometría, es decir, que cada transformación lineal entre capas preserve distancias y ángulos entre los vectores:

$$|W\mathrm{x}|_2^2 = (W\mathrm{x})^\top (W\mathrm{x}) = \mathrm{x}^\top W^\top W\mathrm{x} = \mathrm{x}^\top I\mathrm{x} = |\mathrm{x}|_2^2 \tag{8}$$

Retomando la ecuación (6) de las Redes Neuronales Profundas, la propagación de la señal a través de la red implica multiplicaciones matriciales sucesivas. Si no imponemos restricciones sobre los pesos W_i, la norma de las activaciones h_i puede crecer o disminuir exponencialmente con la profundidad de la red, dependiendo de los valores singulares de W_i.

Si logramos que las matrices de pesos W_i sean ortogonales (o próximas a serlo), la propiedad descrita en la ecuación (8) asegura que la energía de la señal se conserve a través de las capas. Esto es crucial no solo para el paso hacia adelante (*forward pass*), sino especialmente para la retropropagación (*backpropagation*).

2.4 Entrenamiento de los Modelos

Ahora bien, la mayoría de lo descrito anteriormente ocurre y está pensado para cuando ya tenemos un modelo con valores W_i y b_i fijos, es decir, ya entrenado. Sin embargo, el objetivo en el aprendizaje

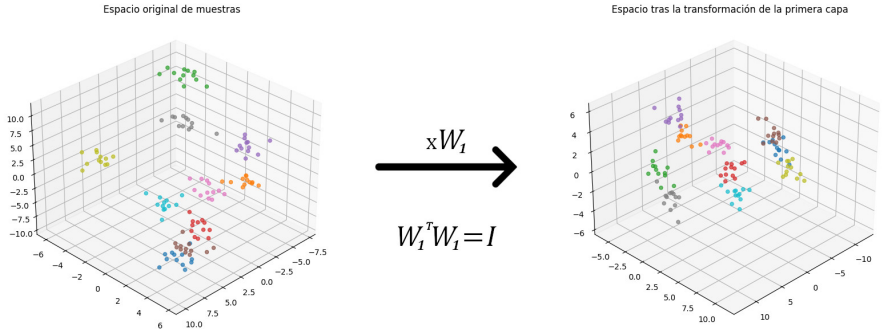

Figura 5: Ejemplo de preservación de distancias y ángulos entre muestras en un espacio al operarse con la primera matriz.

automático es hacer que la red neuronal "aprenda" los mejores valores que le permitan al modelo generalizar utilizando métodos de optimización. En particular, el método del gradiente en descenso - y sus variantes - es el método utilizado para entrenar redes neuronales, dad una función de pérdida $\mathcal{L} = L(F_\theta(x), y)$ seleccionada para nuestro problema, actualizamos nuestros valores $\theta := \{W_i, b_i : i = 0, 1, ..., n\}$ acercándonos a un mínimo local derivando dicha función de pérdida y actualizándolo en T pasos o épocas:

$$\theta_{t+1} \longrightarrow \theta_t - \alpha \frac{\partial \mathcal{L}}{\partial \theta}$$

en donde α es llamado tasa de aprendizaje y es un hiperparámetro que indica el peso con el que el gradiente se mueve. El algoritmo consiste en inicializar θ con una distribución de probabilidad, aplicar un paso hacia adelante (predecir) y retropropagar (derivar), y después actualizar θ.

Aquí podemos tener el problema mencionado anteriormente, de que a lo largo de la composición de las funciones de activación las derivadas exploten o se desvanezcan. Esto podemos controlarlo bastante bien forzando la ortogonalidad entre las matrices de pesos. Otro problema también común es el sobreajuste de los datos de entrenamiento, que tiene que ver con que el modelo se aprenda muy bien las predicciones con los datos de entrenamiento, pero que generalice muy mal y no haga buenas predicciones con datos nuevos. Podemos regularizar el modelo penalizando los pesos W_i.

3. Metodología: Regularización Ortogonal

Como se discutió en la sección anterior, imponer restricciones estrictas de ortogonalidad (Hard Orthogonality) mediante métodos como la descomposición QR o SVD en cada paso de entrenamiento resulta computacionalmente costoso e inviable para redes profundas. Por lo tanto, en este trabajo adoptamos una estrategia de *ortogonalidad suave* (Soft Orthogonality).

Esta aproximación consiste en relajar la restricción geométrica incorporándola directamente en la función de optimización. Modificamos la función de pérdida original $\mathcal{L}$ (en nuestro caso, la entropía cruzada para clasificación) añadiendo un término de penalización que castiga la desviación de las matrices de pesos respecto a la ortogonalidad.

Definimos el término de regularización ortogonal para una capa l con pesos W_l como:

$$\mathcal{R}(W_l) = |W_l^\top W_l - I|_F^2 \tag{9}$$

donde $\|\cdot\|_F$ denota la norma de Frobenius e I es la matriz identidad. De este modo, la función de costo global que optimizamos es:

$$\mathcal{L}_{total}(\theta) = \mathcal{L}(\theta) + \lambda \sum_{l=1}^{K} \mathcal{R}(W_l) \tag{10}$$

Aquí, $\lambda \in \mathbb{R}^+$ es un hiperparámetro de regularización que controla la fuerza con la que imponemos la ortogonalidad.

- Si $\lambda = 0$, el modelo se comporta como una red neuronal estándar sin restricciones.
- Si $\lambda \to \infty$, el modelo prioriza la ortogonalidad sobre la precisión de la clasificación, lo que puede llevar a un subajuste (underfitting).

El gradiente de este término de regularización respecto a los pesos es computacionalmente eficiente de calcular:

$$\frac{\partial \mathcal{R}}{\partial W} = 4W(W^\top W - I) \tag{11}$$

Esto permite actualizar los pesos utilizando descenso de gradiente estándar sin necesidad de proyecciones complejas sobre la variedad de Stiefel.

4. Diseño Experimental

Para evaluar el impacto de la regularización ortogonal en la estabilidad y el desempeño del modelo, diseñamos dos experimentos controlados utilizando el dataset MNIST. En el primero se implementó una arquitectura de Red Neuronal Profunda (DNN) con las siguientes características:

- **Entrada:** Vector de 784 dimensiones (imágenes aplanadas de 28×28).
- **Capas Ocultas:** Dos capas densas con activación ReLU.
- **Salida:** Capa de 10 neuronas (correspondientes a los dígitos 0-9) con activación Softmax.

El objetivo central del experimento es comparar el comportamiento del modelo bajo distintos valores del coeficiente de regularización λ. Se entrenaron cinco modelos independientes con los siguientes valores:

$$\lambda \in \{0, 0.01, 0.1, 0.5, 1.0\}$$

Todos los modelos fueron entrenados durante el mismo número de épocas, utilizando el mismo optimizador y tasa de aprendizaje para garantizar la comparabilidad de los resultados. Se monitorearon cuatro métricas principales: la función de pérdida (Loss), la precisión en validación (Accuracy), el error de ortogonalidad ($\|W^T W - I\|$) y el número de condición de las matrices de pesos.

En el segundo experimento tomamos el modelo que obtuvo mejor desempeño del experimento anterior y se comparó con un modelo no ortogonal, variando esta vez el número de capas ocultas:

$$n = \{6, 10, 14, 18, 22, 26\}$$

En este caso, todas las capas internas extras que se añaden por modelo tiene 128 neuronas. Se monitorean el tiempo de ejecución y el error de ortogonalización total de todas las capas de los modelos.

5. Resultados y Discusión

A continuación, presentamos los resultados obtenidos tras el proceso de entrenamiento.

5.1 Desempeño en Clasificación y Convergencia

En la Figura 6 observamos la evolución de la función de pérdida y la precisión durante el entrenamiento.

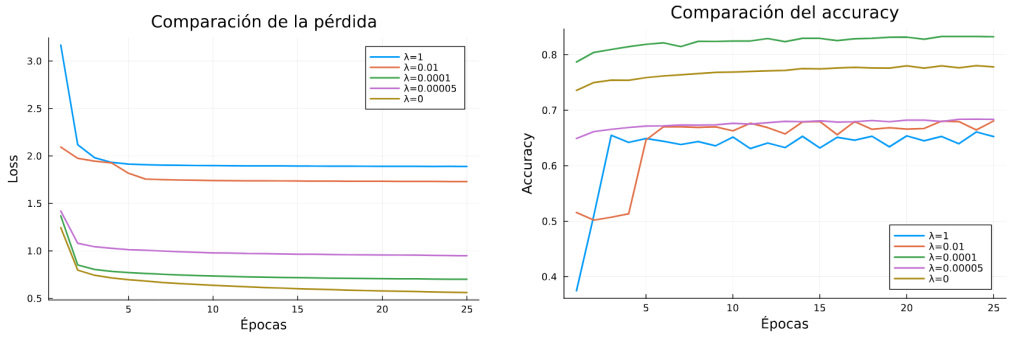

Figura 6: Comparación de la pérdida (izquierda) y la precisión (derecha) para distintos valores de λ

Se observa que los modelos con λ moderado (e.g., 0.01, 0.1) logran converger a una precisión comparable al modelo base ($\lambda = 0$). Sin embargo, cuando la penalización es muy alta ($\lambda = 1.0$), la restricción de ortogonalidad compite agresivamente con la tarea de clasificación, lo que puede ralentizar la convergencia o converger a una solución subóptima en términos de accuracy. Esto confirma que existe un compromiso entre imponer estructura geométrica y permitir libertad de representación.

5.2 Estabilidad Numérica y Ortogonalidad

La efectividad de la regularización se evidencia al analizar el error de ortogonalidad y el número de condición, mostrados en la Figura 7 y la Figura 9, respectivamente.

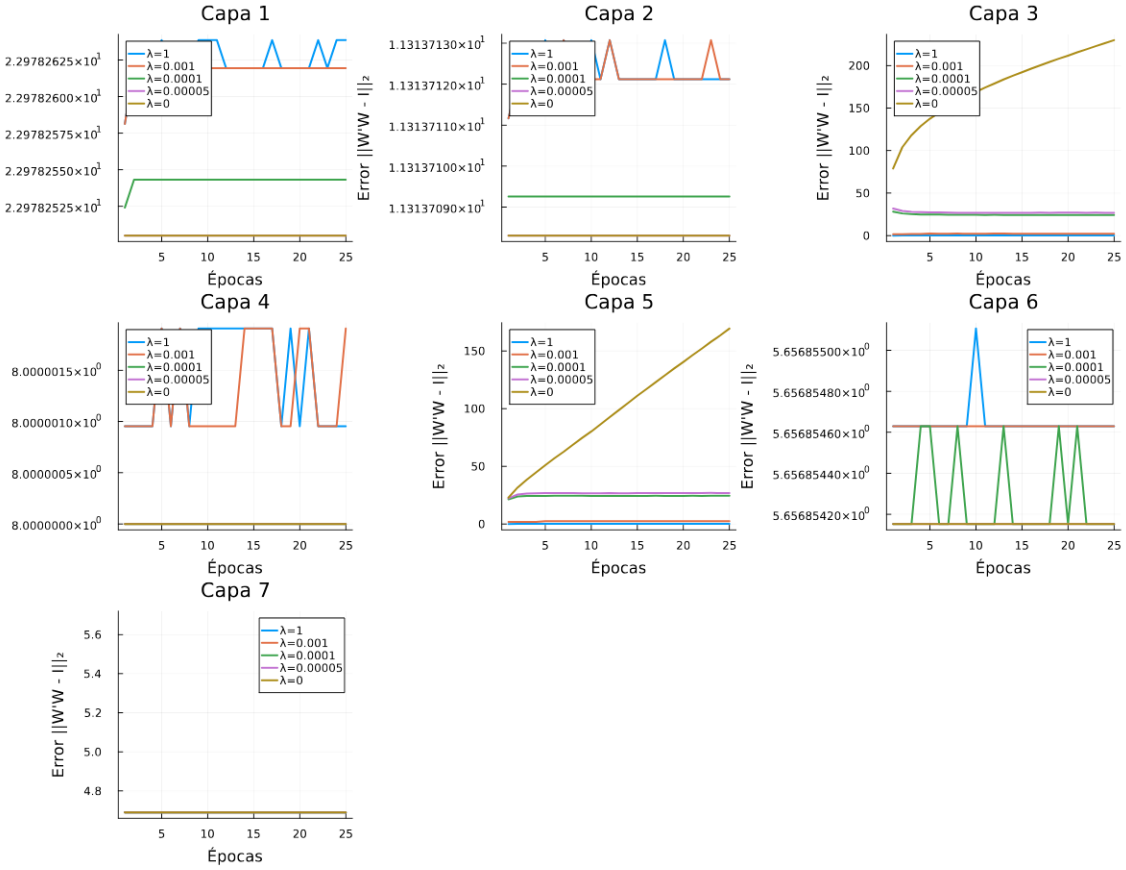

Figura 7: Error de ortogonalidad capa por capa en cada uno de los modelos.

El modelo sin regularización ($\lambda = 0$) muestra un crecimiento rápido en el error de ortogonalidad, lo que indica que sus matrices de pesos se alejan de ser isométricas. Por el contrario, los modelos regularizados mantienen este error acotado cerca de cero. Esto puede observarse mejor en los resultados obtenidos en la Figura 8 del segundo experimento, en donde se ve que mientras más grande es el modelo, en una DNN el error incrementa con un comportamiento lineal, mientras que el modelo ortogonal se mantiene constante.

Más importante aún es el análisis del *Número de Condición* ($\kappa(W)$). Un número de condición bajo (cercano a 1) indica que la matriz de pesos preserva bien la magnitud de los gradientes, evitando los problemas de desvanecimiento o explosión. Los resultados muestran que a medida que aumentamos λ, $\kappa(W)$ se mantiene significativamente más bajo y estable a lo largo de las épocas en comparación con el modelo base. Esto valida la hipótesis de que las OrthDNNs son numéricamente más estables.

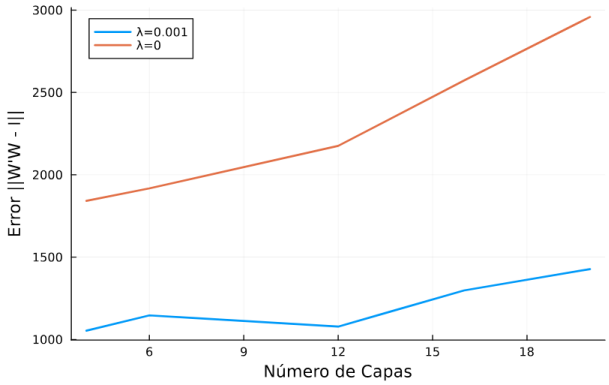

Figura 8: Comparativa del error total entre los modelos con $\lambda = 0$ y $\lambda = 0.0001$.

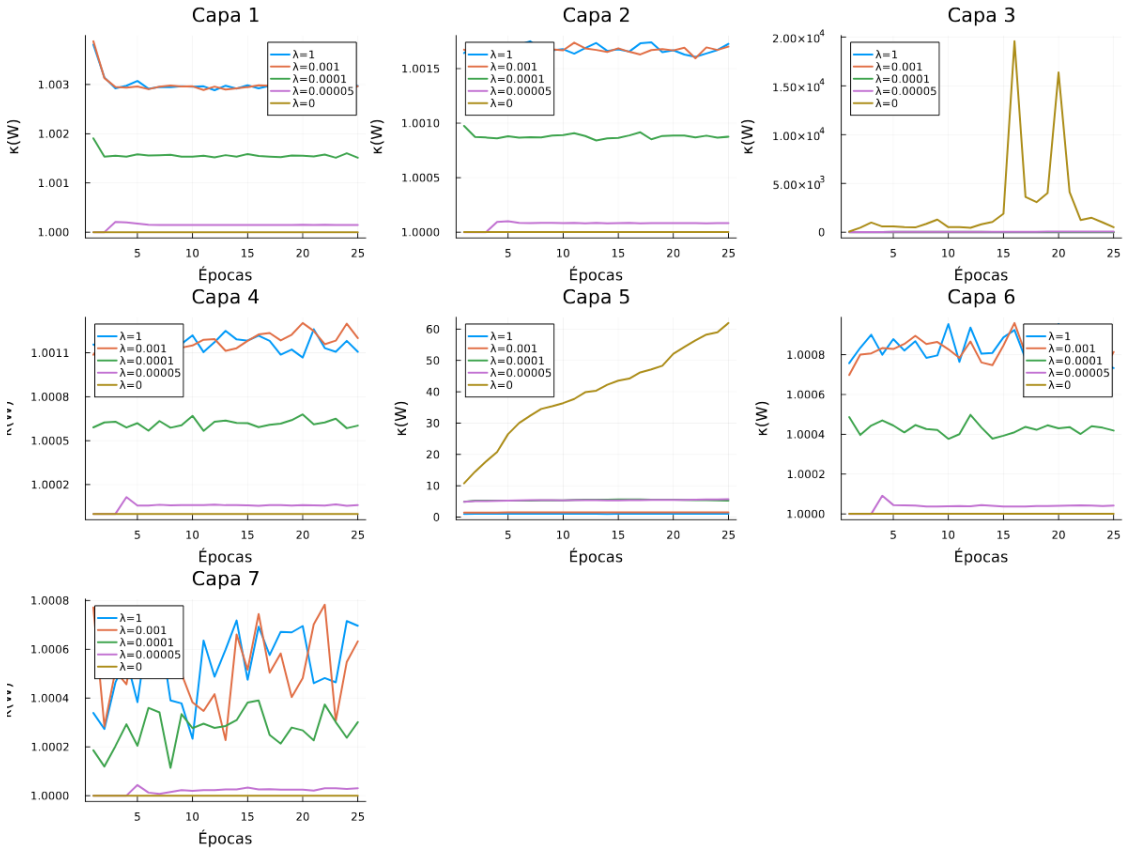

Figura 9: Número de condición capa por capa en cada uno de los modelos.

5.3 *Tiempo de Ejecución*

Se ha mostrado que los modelos ortogonales son numéricamente más estables, sin embargo, es bien sabido que el cálculo de ortogonalización de matrices es costoso computacionalmente hablando. Esto se muestra en la Figura 10 y la Figura 11.

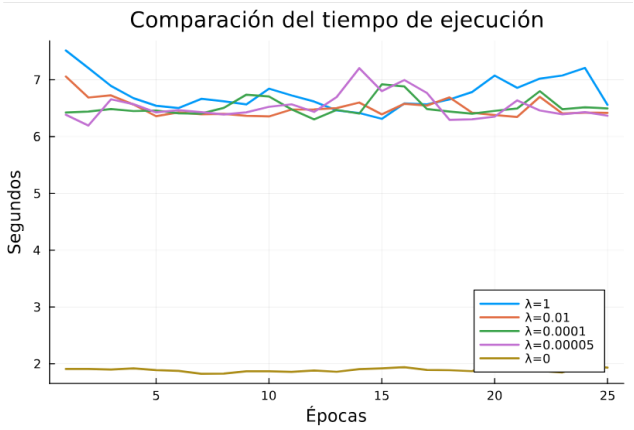

Figura 10: Tiempo de ejecución de cada época en el entrenamiento de los modelos en el primer experimento.

En los tiempos tomados el primer experimento se puede observar que, por época, los modelos ortogonales tardan un poco más del triple de lo que tarda un modelo sin restricción. Además, se

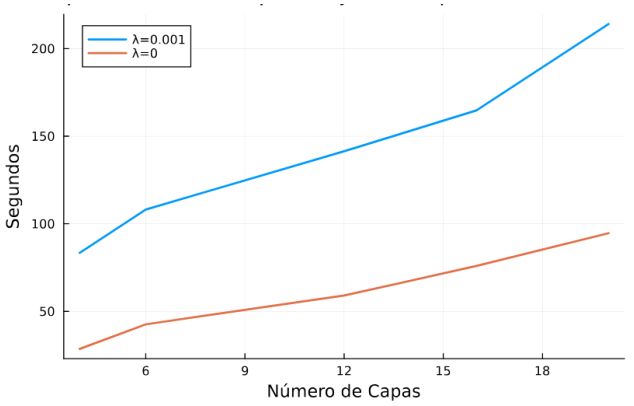

Figura 11: Tiempo de ejecución entre los modelos con $\lambda = 0$ y $\lambda = 0.0001$.

observa que el modelo ortogonal tarda aproximadamente un poco más de el doble en ejecución total del entrenamiento a medida que el número de capas aumenta. Sin embargo, no podemos concluir que escale o que el orden de magnitud aumente; para redes neuronales con pocas capas ambos modelos parecen tener un comportamiento asintótico lineal. De esta forma podemos concluir un claro compromiso entre tiempo de ejecución y estabilidad numérica al seleccionar un modelo ortogonal.

6. Conclusiones

En este trabajo hemos explorado la implementación y el impacto de las Redes Neuronales Ortogonales aplicadas al problema de clasificación de MNIST. A través de la experimentación con regularización suave, hemos llegado a las siguientes conclusiones:

1. **Estabilidad Numérica:** La regularización ortogonal mejora drásticamente el condicionamiento de las matrices de pesos. Esto sugiere que las OrthDNNs son menos propensas a sufrir inestabilidades numéricas en redes profundas, facilitando la propagación de gradientes limpios.
2. **Trade-off entre Restricción y Desempeño:** Existe un delicado equilibrio al seleccionar el hiperparámetro λ. Un valor demasiado alto fuerza la ortogonalidad a expensas de la capacidad del modelo para minimizar el error de clasificación, mientras que un valor muy bajo no aporta los beneficios de estabilidad deseados. La selección de λ es, por tanto, vital.
3. **Generalización:** Aunque en una red poco profunda y un dataset sencillo como MNIST las diferencias en generalización pueden ser sutiles, la estructura impuesta por la ortogonalidad actúa como un regularizador efectivo, reduciendo la varianza de los pesos y potencialmente decorrelacionando las características aprendidas.

Como trabajo futuro, sería relevante extender este análisis a arquitecturas más profundas (como ResNets o RNNs) donde los problemas de gradiente son más críticos, y comparar la regularización suave con métodos de optimización en variedades de Stiefel para evaluar el costo computacional frente a la exactitud de la ortogonalidad.

Declaración de uso de IA

Este reporte ha sido elaborado con la asistencia de herramientas de Inteligencia Artificial. Específicamente, se utilizaron modelos de lenguaje para la corrección de estilo, sugerencias de estructura en LaTeX y la implementación de algunos códigos. Sin embargo, la ejecución de los experimentos, la recopilación de datos y el análisis crítico de los resultados presentados son autoría original del estudiante.

Referencias

[1] S. Li, K. Jia, Y. Wen, T. Liu, and D. Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019.

[2] S. J. Prince. *Understanding Deep Learning*. The MIT Press, 2023.