Instituto Tecnológico y de Estudios Superiores de Monterrey

Inteligencia Artificial Avanzada para la Ciencia de Datos I (Grupo 101)

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Daniel Kaled Bernal Ayala A01750047

14 de Septiembre del 2025

Introducción

En el presente trabajo se presenta el análisis y la mejora de la implementación hecha a lo largo del curso. Esta implementación es el **Backpropagation** (Red multicapa implementada a mano), la cual cuenta con diferentes características que se pueden modificar para tener mejores resultados. Para el modelo se tomó en cuenta el mismo dataset con la clasificación de números representados en matrices 3x3, en donde se expandió para tener mejores desempeños en los modelos. El dataset fue ampliado con ruido para aumentar su tamaño de 6 muestras originales a unas 25 variaciones por dígito, lo que permite evaluar el comportamiento de los modelos en mejores condiciones.

La ventaja de la implementación del algoritmo de **Backpropagation** es la actualización de los parámetros como lo es el peso y los sesgos a partir del error calculado para así corregirlos y encontrar el mejor ajuste. No obstante, el uso de redes neuronales implica un gran consumo computacional. Lo que puede ser un factor decisivo entre implementarla o no hacerlo. El objetivo es comparar su desempeño, explicar su sesgo y varianza, diagnosticar el nivel de ajuste y, en su caso, aplicar técnicas de regularización y ajuste de parámetros para mejorar el rendimiento.

Base de Datos y División

Ahora bien, para la base de datos se define que se quiere predecir, es por ello que se realizó una base de datos para clasificar números definidos en una matriz de 3x3. De igual forma, se realiza una expansión para tener un cambio en el comportamiento de los modelos. Esa expansión se realiza agregando un dato más al database que es el número 5.

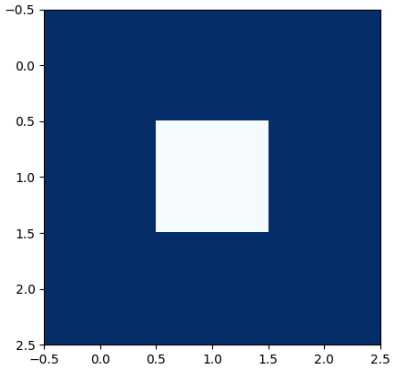

Figura 1. Matriz de 3x3 del número 0

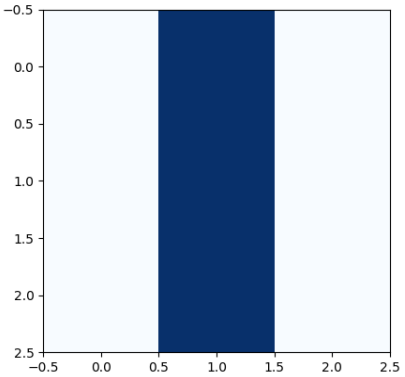

Figura 2. Matriz de 3x3 del número 1

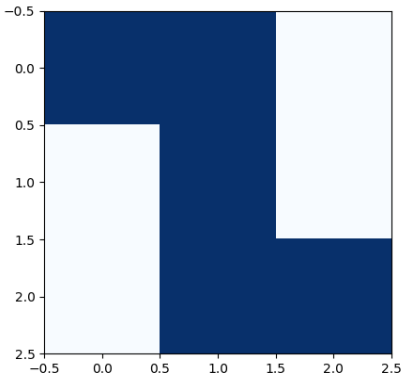

Figura 3. Matriz de 3x3 del número 2

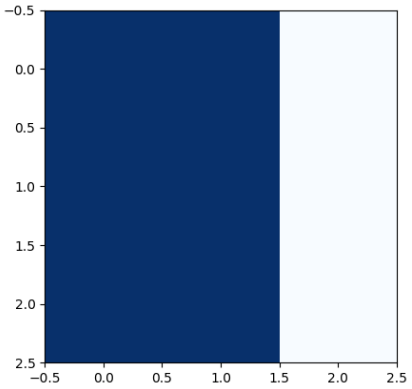

Figura 4. Matriz de 3x3 del número 3

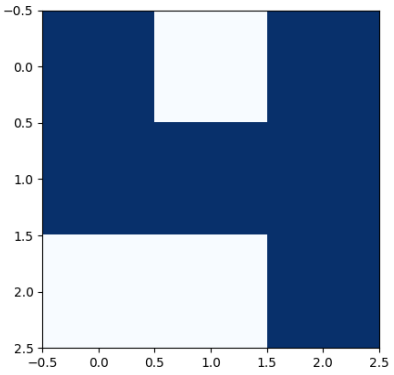

Figura 5. Matriz de 3x3 del número 4

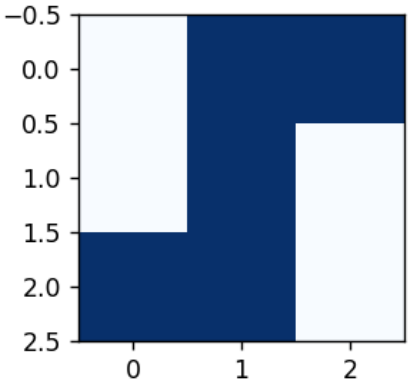

Figura 6. Matriz de 3x3 del nuevo número agregado 5

Con esto contemplado, como se mencionó anteriormente, se agrega ruido a los datos para tener una base de datos expandida. Teniendo un aumento de 25 figuras por cada dato base, lo que significa una database de 156 datos. De igual manera, para la salida de cada uno está dada en **One Hot Encoding** para una mejor clasificación de los valores. Siendo así las siguientes salidas:

- ➔ **0,0,0,0,0,1** es **0**
- ➔ **0,0,0,0,1,0** es **1**
- ➔ **0,0,0,1,0,0** es **2**
- ➔ **0,0,1,0,0,0** es **3**
- ➔ **0,1,0,0,0,0** es **4**
- ➔ **1,0,0,0,0,0** es **5**

Con esto establecido primero se dividen los datos correspondientes al modelo de **Backpropagation** con una repartición 70% entrenamiento, 20% validación y 10% testeo generando lo siguiente:

- **70% de los 156 datos = 109**
- **20% de los 156 datos = 31**
- **10% de los 156 datos = 16**

Entrenamiento

El procedimiento para el **Backpropagation** se organiza en **épocas**, que representan el número de veces que la red neuronal procesa el conjunto completo de datos de entrenamiento. Al final de la época, se calcula la pérdida del promedio para después graficar y analizar cómo disminuyó el error en el entrenamiento. De igual forma, se brinda el accuracy tanto en el entrenamiento como en la validación para el diagnóstico del bias y la varianza. Esto nos da una medida de qué tanto la red está memorizando o aprendiendo el patrón de los datos.

Con esto establecido, se realiza un entrenamiento con los datos apartados respectivamente. Se procede a hacer lo mismo con los datos de validación para obtener su accuracy. Y finalmente, se prueba el modelo con los datos de testeo para así tener el accuracy final y poder analizarlo más adelante. Este análisis se hará por medio de una matriz de confusión, que nos permite ver cuánto fue el error que tuvo. Además se despliega un reporte con las características correspondientes a la precisión, el recall, F1 - score y la división de los datos con los que fue entrenado.

Evaluación

Después de realizar el entrenamiento del primer modelo, se tienen los siguientes gráficos relacionados a su desempeño. El primero que se analiza es el modelo de **Backpropagation** evaluado con 35 épocas.

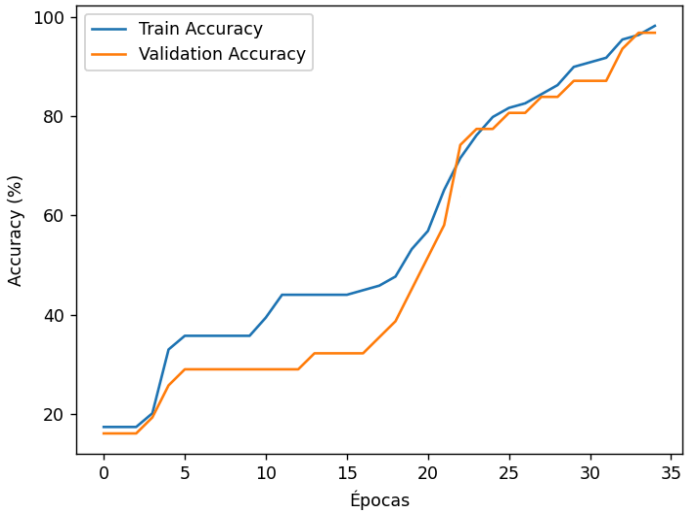

Figura 7. Porcentaje de accuracy del entrenamiento y la validación

Aquí tenemos un valor del **98.17%** de accuracy en el entrenamiento, contra **96.77%** en la validación. Sin embargo, al inicio del entrenamiento del modelo presenta un alto sesgo por la separación que existe entre el entrenamiento y la validación. Conforme avanzan las épocas el sesgo disminuye y se va aprendiendo las representaciones correctas. Posterior a ello indica que el modelo tiene baja varianza, ya que el sobreajuste no es grande. De igual manera, tenemos un sesgo bajo debido a que el modelo está generalizando correctamente.

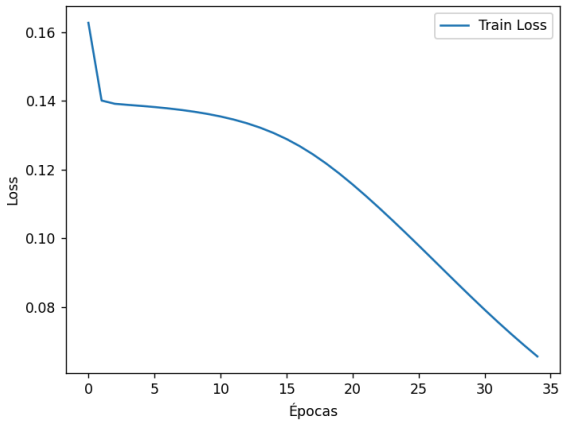

Figura 8. Gráfico del cambio del error entre época

De igual manera, en la Figura 8 podemos ver que el error final reportado es de 0.06, lo que refiere a casi un **6%** de error en predecir el dato. Teniendo en cuenta que el learning rate es de 0.1 lo que es un ritmo lento para las pocas épocas que entrenó, su desempeño es bueno ante el dataset.

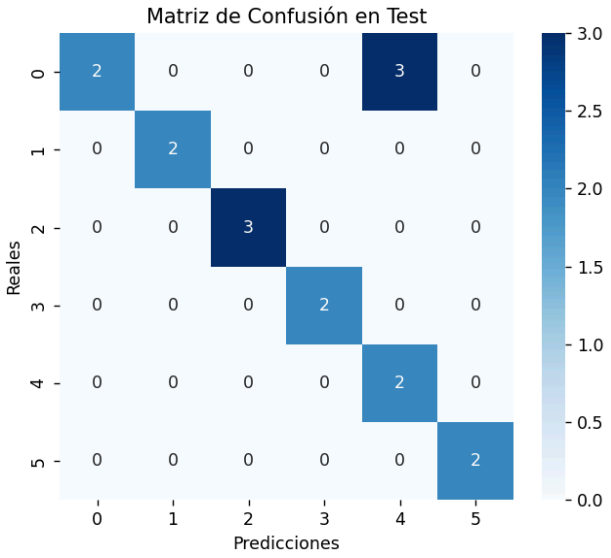

Figura 9. Matriz de confusión

Después de realizar el test la **Figura 9** muestra una confusión entre el 4 y el cero, dando un accuracy del **81.25%**. Esto quiere decir que 8 de cada 10 se clasificaron correctamente. Lo que refleja un modelo con bajo sesgo y varianza moderada, lo que sigue mostrando buenas generalizaciones sin llegar al overfitting.

Tabla I. Métricas del modelo con el test realizado

	Precisión	Recall	F1-score	support
Cero	1.00	0.80	0.89	2
Uno	1.00	1.00	1.00	2
Dos	1.00	1.00	1.00	3
Tres	1.00	1.00	1.00	2
Cuatro	0.50	1.00	0.67	5
Cinco	1.00	1.00	1.00	2

Como podemos ver, la confusión entre el cero y el 4 se marca en las métricas utilizadas. Además podemos observar que los datos no están bien repartidos y son pocos. Lo que puede ser indicios que se los aprendió o por esas razones el accuracy es bajo. Se necesitaría aumentar los datos para tener un mejor desempeño.

Mejoras

Una vez hecho el análisis de los resultados con la implementación hecha anteriormente, se tomaron dos mejoras para su desempeño. En este caso se cambió la función de sigmoide a **softmax**. La cual, consiste en transformar los valores entre 0 y 1 de modo que se puedan interpretar como probabilidades. Softmax resulta muy útil entre la capa oculta y la capa de salida, ya que convierte las puntuaciones en una distribución de probabilidad normalizada y así mantiene el orden en los datos. La función se representa de la siguiente forma.

$$softmax(z)_i = \frac{e^{zi}}{\sum_{j=1}^{N} e^{zi}}$$

Donde todos los valores **zi** son los elementos del vector de entrada y pueden tomar cualquier valor real. El término al final de la fórmula es el término de normalización, logrando así una distribución normalizada (Wood, 2025).

Además, se realiza el cambio del cálculo del error a **cross-entropy** que converge de mejor manera para la clasificación multiclase. Sirve mejor, ya que la entropía mide la diferencia entre la distribución de probabilidad descubierta de un modelo de clasificación y los valores predichos. Sin embargo, los resultados no mejoraron en la red implementada.

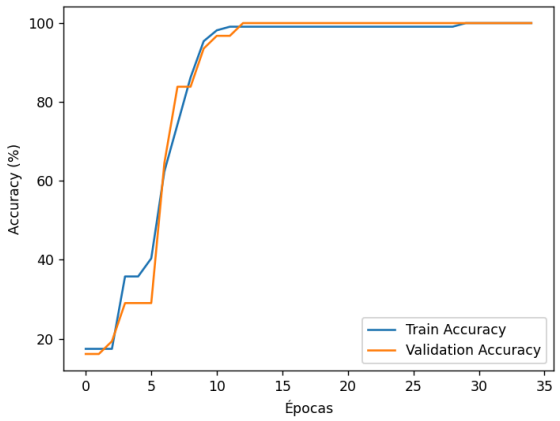

Figura 10. Porcentaje de accuracy del entrenamiento y la validación para la nueva implementación

Una vez aplicados los cambios podemos observar que sucedió overfitting inmediatamente. El accuracy del entrenamiento fue del **96%** mientras que los datos de validación se identificaron completamente con un **100%**. Esto debido a que las mejoras de cross-entropy y softmax hacen que los datos se normalicen muy rápido y se aprende los pesos, lo que ocasiona que rápido se aprende el dataset. Esto en parte de que el dataset solo cuenta con 156 datos, los cuales son muy pocos para el poder de la red neuronal.

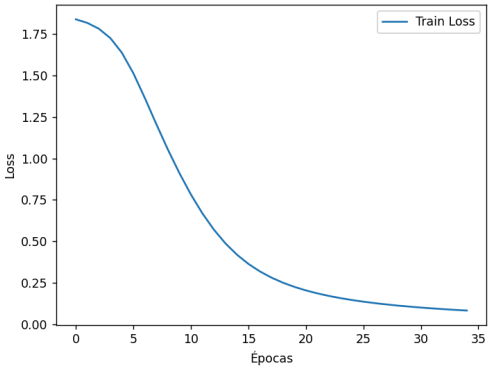

Figura 11. Gráfico del cambio del error entre época

Esto va de la mano con el error tan grande que presenta, este corresponde a 0.08, lo cual, no es alto pero sigue siendo extraño al analizar el accuracy tanto del entrenamiento como el de validación.

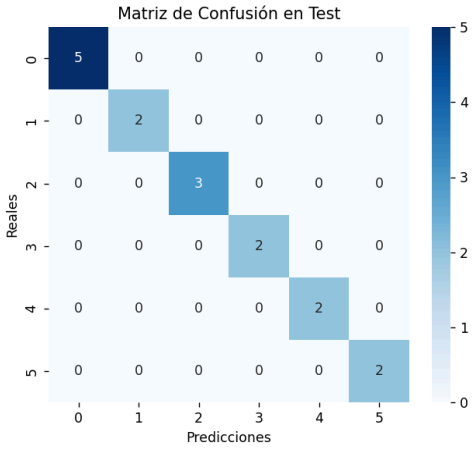

Figura 12. Matriz de confusión del modelo mejorado

Aquí se observa de manera más clara el overfitting que está sucediendo al momento de realizar el testeo. Como el ruido en los datos no es muy grande, es muy fácil para la red tener la comparación precisa en cada dado. Por lo que, estas mejoras no se pueden llevar a cabo para el dataset que se crea. Sin embargo, al solo cambiar la función **softmax** mejora mucho del modelo.

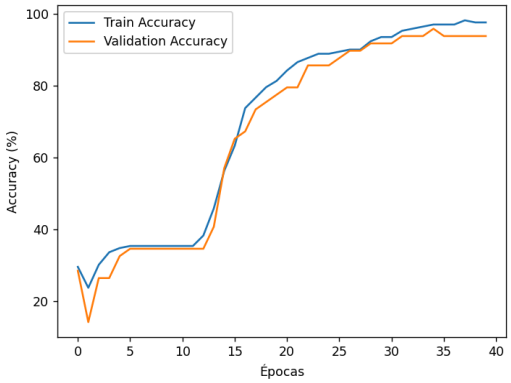

Figura 13. Porcentaje de accuracy del entrenamiento y la validación con cambio a softmax

En este modelo se tiene un accuracy del **97.67%** en el entrenamiento y **93.88%** en el de validación. De igual forma, se agregan más copias de los números para así tener mayor dataset y mejorar el desempeño en el test. Aumentando 10 números más para el test. Este modelo presenta un sesgo y varianza parecidos al primer modelo analizado. Teniendo un error de 0.04 que representa un **4%** en el error de predicción. Al momento de realizar el test se obtuvieron los siguientes resultados.

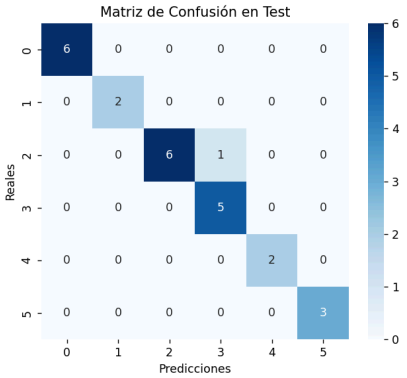

Figura 14. Matriz de confusión del modelo con softmax

Podemos observar un accuracy del **96.00%** lo que mejora al primer modelo descrito. Esto quiere decir que la mejora de cambiar la función **sigmoide** a **softmax** refleja un cambio positivo en el modelo. Además, el aumentar el dataset, ayuda a tener más claridad en el accuracy real del sistema. Teniendo una varianza y sesgo similar, siendo así una buena implementación y mejora del modelo.

Tabla II. Métricas del modelo con las mejoras planteadas

	Precisión	Recall	F1-score	support
Cero	1.00	1.00	1.00	6
Uno	1.00	1.00	1.00	2
Dos	1.00	0.86	0.92	7
Tres	0.83	1.00	0.91	5
Cuatro	1.00	1.00	1.00	2
Cinco	1.00	1.00	1.00	3

Con sus métricas podemos comprobar la mejoría ante el primer modelo presentado. Donde se aumenta el número de datos de prueba a 25. Sin embargo, estos se encuentran dispares, siendo pocos para el uno, cuatro y cinco. Se necesitaría balancear para observar mejoras en el modelo. No obstante, los desempeños son mejores, mostrando mejoras en el sesgo y la varianza, siendo poco en ambos casos y teniendo un buen modelo. Con ello, teniendo una sola confusión entre el número tres y dos.

Bias

El sesgo se refiere al error sistemático que comete un modelo cuando es demasiado simple o no logra captar las relaciones reales en los datos (underfitting). Se detecta comparando el rendimiento del conjunto de entrenamiento con el de validación: un modelo con alto sesgo tendrá bajas precisiones en ambos conjuntos. Además, el sesgo en una red neuronal es necesario para desplazar la función de activación a lo largo del plano, ya sea hacia la izquierda o hacia la derecha. Esto permite a la red neuronal ajustar el umbral de activación de la neurona, lo que le otorga mayor flexibilidad y capacidad para aprender funciones no lineales complejas (Turing, 2022).

En el **Modelo 1 (sigmoide)** se observa un accuracy del **98.17%** en entrenamiento y **96.77%** en validación. Esto indica un sesgo bajo, ya que el modelo aprende correctamente el patrón sin dejar demasiados errores sistemáticos. Al momento del test, el accuracy baja al **81.25%**, lo que refleja que, aunque el sesgo es bajo en entrenamiento/validación, en test hay ejemplos que no generaliza tan bien. En el Modelo 2 (softmax) el accuracy en entrenamiento fue de **97.67%** y en validación **93.88%**. Esto también corresponde a un sesgo bajo, con una ligera mejora respecto al primer modelo. El test alcanza **96.00%**, confirmando que la función softmax ayudó a disminuir errores sistemáticos y a mejorar la generalización.

Varianza

La **varianza** mide qué tan sensible es el modelo a los datos de entrenamiento; cuando es alta, significa que memoriza los datos y falla en generalizar (overfitting). Se observa cuando hay un desempeño muy alto en entrenamiento pero bajo en validación/test. De igual forma, cuando está bien balanceada al momento de generalizar esta no muestra valores tan alejados de los esperados en los datos de entrenamiento/validación.

En el **Modelo 1** la diferencia entre entrenamiento y validación es pequeña **(98.17% a 96.77%)**, lo que indica una **varianza baja**. Sin embargo, el descenso hasta **81.25%** en el test sugiere que el conjunto de prueba es más difícil y que se requiere un dataset más amplio para estabilizar la varianza. En el **Modelo 2** la diferencia entre entrenamiento y validación es mayor **(97.67 % a 93.88 %)**, pero el test logra **96.00 %**, mostrando una **varianza media-baja** y un mejor equilibrio entre aprendizaje y generalización. Este resultado confirma que aumentar datos y cambiar la función de activación ayuda a controlar la varianza.

Tabla III. Diagnóstico de modelos

Modelo	Acc. train	Acc. Validation	Acc. Test	Bias	Varianza
Sigmoide	**98.17%**	**96.77%**	**81.25%**	Bajo	media
Softmax	**97.67%**	**93.88%**	**96.00%**	Bajo	media-baja

Conclusiones

Para finalizar, se encontraron grandes mejoras ante el uso de la función sigmoide contra la función softmax. Los cuales, muestran un gran cambio en el accuracy en el test y mejorando el aprendizaje.

Tabla IV. Comparación de Modelos en el Test

	Función	Accuracy	Error
Modelo 1	Sigmoide	81.25%	0.06
Modelo 2	Softmax	96.00%	0.04

Observamos un mejor desempeño en el modelo 2 siendo una diferencia aproximada del **15%** entre el accuracy. De igual forma, hay una disminución del error sin llegar al overfitting. Al implementar **Backpropagation** en este set de datos es excesivo para el poder que supone una red neuronal. Para optimizar dicha implementación sería necesario agrandar el dataset y buscar tener un procesamiento de imagen para aprovechar las virtudes del modelo. Además se observó que para este dataset, considerar **cross validation** y un ajuste **L2** no es necesario. Esto haría que llegaramos al overfitting, debido a la cantidad de datos. Por esta razón es mejor considerar un modelo más simple como lo puede ser un árbol de decisiones que nos permitiría generalizar de buena forma sin llevarlo a un sobreajuste.

Referencias

Uresti, J. (2025). Presentación Tema 6 - Redes Neuronales. Recuperado del material visto en clase.

Uresti, J. (2025). Presentación Tema 7 - Aprendizaje Supervisado. Recuperado del material visto en clase.

Wood, T. (2025). Función Softmax. Recuperado de https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

TURING. (2022). ¿Cuál es la necesidad del sesgo en las redes neuronales? Recuperado de https://www.turing.com/kb/necessity-of-bias-in-neural-networks