Reporte Portafolio 2

Materia: Inteligencia Artificial Avanzada
Para la Ciencia de Datos I

Actividad: Portafolio 2

Axel Amos Hernández Cardenas - A00829837

1. Introducción

El objetivo de este reporte es evaluar el rendimiento de un perceptrón, una red neuronal no optimizada y una red neuronal optimizada para predecir si un paciente padece de diabetes en función de varías características.

2. Separación de Datos

El dataset contiene un total de 768 instancias caracterizadas por ocho features: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function (DPF) y Age. El label está definido por la columna Outcome, la cual toma un valor de 1 si el paciente padece diabetes y 0 si no la padece. Para construir el modelo, es necesario separar las features del target para que la red neuronal aprenda a predecir Outcome a partir de las otras 8 variables. Por lo anterior, se decidió dividir el dataset en dos partes: el conjunto de entrenamiento (train set) y el conjunto de prueba (test set).

El conjunto de entrenamiento corresponde al 70% de los datos y se utiliza para entrenar el modelo, permitiendo que los pesos se ajusten de tal manera que nos permita predecir el target correctamente. Por otra parte, el conjunto de prueba corresponde al 30% (*test_size = 0.3*) de los datos y se utiliza para evaluar el rendimiento final del modelo en datos no vistos durante el proceso de entrenamiento, lo cual es importante para verificar que el modelo generaliza bien en datos nuevos. Para realizar lo anterior, se utilizó la función de *train_test_split*, de la librería de *sklearn*, que realiza una división aleatoria, manteniendo la distribución de los datos como se mencionó anteriormente.

Además de los conjuntos mencionados, durante el proceso de entrenamiento también se utilizará un subconjunto del conjunto de entrenamiento del 15% para un conjunto de validación. Esto proporcionará una estimación temprana de la capacidad de generalización del modelo y ayudará a detectar varianza alta (overfitting) antes de probarlo con el conjunto de prueba. Para esto, se utilizó el parámetro *validation_split = 0.15* de la librería *Keras*.

Con esta separación de los datos, se asegura que el modelo no dependerá sólamente de los datos de entrenamiento, mejorando su capacidad de generalizar y evitando así la memorización de patrones (overfitting).

3. Evaluación de un Perceptrón

El perceptrón es un modelo simple de red neuronal diseñado para realizar problemas de clasificación binaria aplicando la función de activación sigmoide para asignar probabilidad entre dos clases. Para este caso, se utilizó la librería de *tensorflow.keras* para definir una configuración simple de un perceptrón.

Units	kernel_init	bias_init	Optimizer	α	Epochs	Batch Size
1	HeUniform	Ones	Adam	0.001	300	128

Tabla 1. Configuración del Perceptrón Evaluado

Durante el entrenamiento, se registraron las métricas de accuracy y loss tanto en el conjunto de entrenamiento como en el de validación a lo largo de los 300 epochs. En resumen:

- El accuracy en el conjunto de entrenamiento aumentó desde 35.21% hasta aproximadamente 46.56%. Por otra parte, el accuracy en el conjunto de validación se mantuvo en la mayor parte del entrenamiento alrededor del 34.57% hasta un máximo de 43.21%.
- El loss en el entrenamiento comenzó en un valor de 46.93 y disminuyó constantemente hasta valores aproximados a 2.39. Por otra parte, el loss de validación también disminuye hasta 2.45, pero sigue siendo más alto que el de entrenamiento.
- El accuracy final para el modelo del perceptrón al probarlo con el conjunto de pruebas fue de 48.91% y un loss de 2.34.

La figura 1 muestra los gráficos de accuracy y loss contra los epochs para el perceptrón.

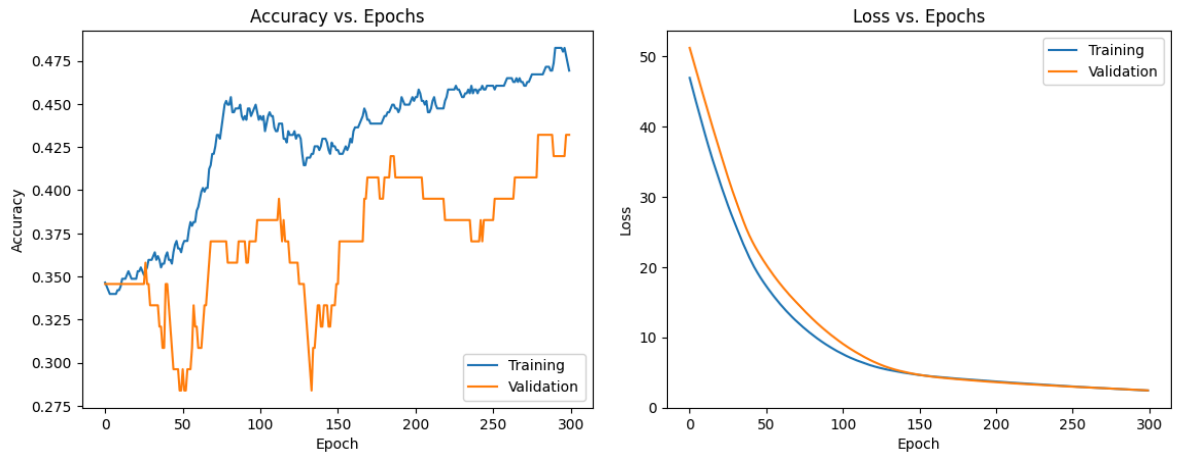

Figura 1. Gráficas de Accuracy vs Epochs y Loss vs Epochs del Perceptrón

Teniendo en cuenta lo anterior, se puede decir que el perceptrón muestra un alto bias, lo que implica que es demasiado simple para capturar las relaciones relevantes en los datos. La baja precisión observada tanto en el conjunto de entrenamiento (46.56%) como en el de validación (43.21%), como se observa en el gráfico de Accuracy vs Epochs, así como en el de prueba (48.91%), indica un rendimiento deficiente en los tres datasets. Sin embargo, este comportamiento es esperado al realizar un modelo simple como el perceptrón, el cual también puede no estar considerando alguna relación no lineal entre los datos.

Por otra parte, la varianza del modelo es baja. En primer lugar, el hecho de que el modelo no

tenga un performance significativamente mejor en el conjunto de entrenamiento (46.56% vs 42.21% y 48.91%) demuestra que no sufre de overfitting, si no que es incapaz de generalizar en los tres conjuntos, como ya se mencionó en el diagnóstico del bias. El gráfico de Loss vs Epochs respalda esta observación, ya que muestra que el error disminuye para los conjuntos de datos de entrenamiento (2.23) y validación (2.45), pero la precisión no mejora de manera significativa para el primero.

En conclusión, el perceptrón sufre de un alto bias debido a su simplicidad, lo que lo lleva a caer en underfitting. Esto se evidencia por su pobre desempeño tanto en los conjuntos de entrenamiento como de validación. Además, no existe una varianza alta, ya que el modelo tampoco presenta una alta precisión en el conjunto de entrenamiento.

4. Evaluación de una Red Neuronal

La creación de una red neuronal representa una optimización significativa respecto al perceptrón, ya que permite capturar relaciones no lineales entre los datos y manejar un mejor número de ajustes en los pesos y biases. Así mismo, una red neuronal puede aprovechar técnicas como el *backpropagation* para optimizar los anteriormente mencionados. Utilizando la librería de *tensorflow.keras* se definió una red neuronal secuencial que tiene predeterminadamente la técnica de backpropagation.

Capas	A. Function	Optimizer	α	Epochs	Batch_Size
[16, 16, 32, 32, 32, 16, 1]	[re, re, re, re, re, re, sig]	Adam	0.001	800	64

Tabla 2. Configuración de la Red Neuronal Evaluada. (Activation Function: re = ReLU, sig = Sigmoid)

Durante el entrenamiento, se registraron las métricas de accuracy y loss tanto en el conjunto de entrenamiento como en el de validación a lo largo de los 800 epochs. En resumen:

- El accuracy en el conjunto de entrenamiento aumentó desde 37.32% hasta aproximadamente 92.04%. Por otra parte, el accuracy en el conjunto de validación se mantuvo en la mayor parte del entrenamiento alrededor del 65.40% hasta un máximo de 66.67%.
- El loss en el entrenamiento comenzó en un valor de 4.35 y disminuyó rápidamente hasta valores cercanos a 0.19. Por otra parte, el loss de validación también disminuye hasta 0.58, pero después sube a 1.41.
- El accuracy final para el modelo de red neuronal al probarlo con el conjunto de pruebas fue de 69.69% y un loss de 1.40.

La figura 2 muestra los gráficos de accuracy y loss contra los epochs para la red neuronal.

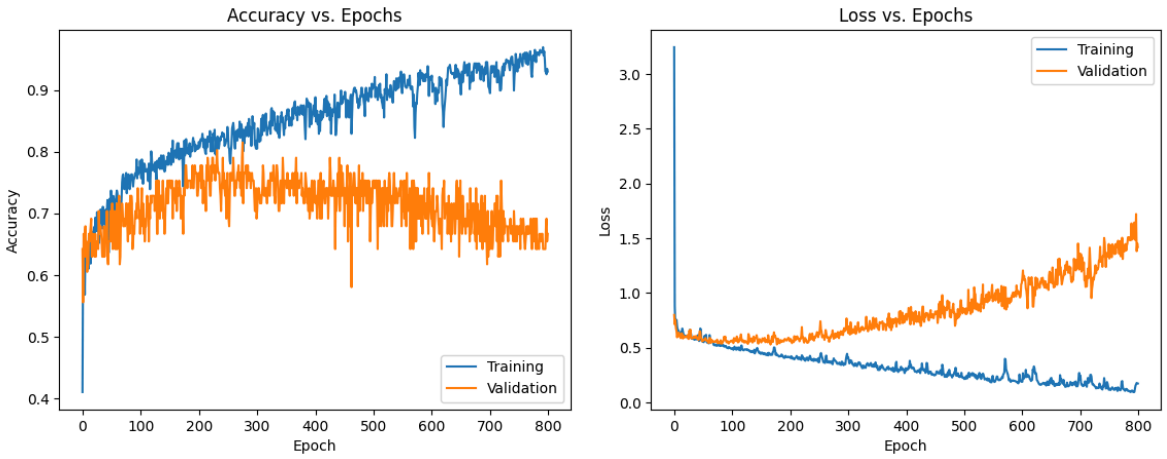

Figura 2. Gráficas de Accuracy vs Epochs y Loss vs Epochs de la red neuronal.

Teniendo en cuenta lo anterior, se puede decir que la red neuronal presenta un bajo bias, ya que muestra un rendimiento más aceptable que el perceptrón en los conjuntos de datos utilizados. En el gráfico de Accuracy vs Epochs, se observa que la precisión en el conjunto de entrenamiento se aproxima al 92% (con un valor final de 92.04%), mientras que en el conjunto de validación es cercana al 67% (con un valor final de 66.67%). Además, la precisión final del modelo con los datos de prueba resultó ser de 69.69%. No obstante, es notable una diferencia entre el desempeño en el conjunto de entrenamiento contra el de validación y prueba, lo que sugiere que el modelo podría estar cayendo en overfitting.

Por otra parte, la varianza del modelo es alta. En el gráfico de Loss vs Epochs, se puede observar como la pérdida en el conjunto de entrenamiento disminuye de manera constante (de 4.35 a 0.64, y finalmente a 0.19), mientras que en el conjunto de validación la pérdida comienza a aumentar después de los 300 epochs (de 0.79 a 0.58, y finalmente a 1.40). De igual manera, el loss para el conjunto de prueba fue de 1.40. Este comportamiento refuerza la hipótesis de que el modelo sufre de overfitting, ya que sigue mejorando en el entrenamiento, pero pierde performance en la validación y prueba.

En conclusión, este modelo de red neuronal sufre de alta varianza, lo que indica que es probable que sea demasiado complejo y lo hace caer en el overfitting. El performance desigual entre los conjuntos de entrenamiento frente a los de validación y prueba es un indicador de que la red se ajusta demasiado a los datos de entrenamiento, comprometiendo su capacidad para generalizar y predecir con datos nuevos.

5. Evaluación de Red Neuronal Optimizada (Técnicas de Regularización y Ajuste)

Finalmente, la última optimización realizada consistió en la modificación de la red neuronal, ajustes en algunos parámetros e hiperparametros y la adición de técnicas de regularización, como Dropout y Batch Normalization, así como callbacks como Learning Rate Reduction y Early Stopping. Para implementar lo anterior, se utilizó, una vez más, *tensorflow.keras*.

Capas	A. Func	Dropout	α	Patience	Epochs	Batch_Size
[8, 16, 16, D, 16, 16, BN, 16, 16, D, 28, 1]	[re, re, re, re, re, re, re, re sig]	[0.1, 0.3]	0.001	ES = 90 LR = 160	1200	114

Tabla 3. Configuración de la Red Neuronal Optimizada Evaluada. (Capas: D = Dropout, BN = Batch Normalization; Activation Function: re = ReLU, sig = Sigmoid; Optimizer: Adam; Patiende: LR = Learning Rate Reduction, ES = Early Stopping).

Durante el entrenamiento, se registraron las métricas de accuracy y loss tanto en el conjunto de entrenamiento como en el de validación a lo largo de los 1200 epochs (< 300 por callbacks). En resumen:

- El accuracy en el conjunto de entrenamiento aumentó desde 44.47% hasta aproximadamente 74.71%. Por otra parte, el accuracy en el conjunto de validación se logró una máxima precisión final de 77.78%.
- El loss en el entrenamiento comenzó en un valor de 0.85 y disminuyó rápidamente hasta estabilizarse en 0.50. Por otra parte, el loss de validación también disminuye de 1.38 a 0.48 a lo largo de los epochs.
- El accuracy final para el modelo de red neuronal optimizada al probarla con el conjunto de pruebas fue de 74.02% y un loss de 0.57.

La figura 3 muestra los gráficos de accuracy y loss contra los epochs para la red neuronal optimizada.

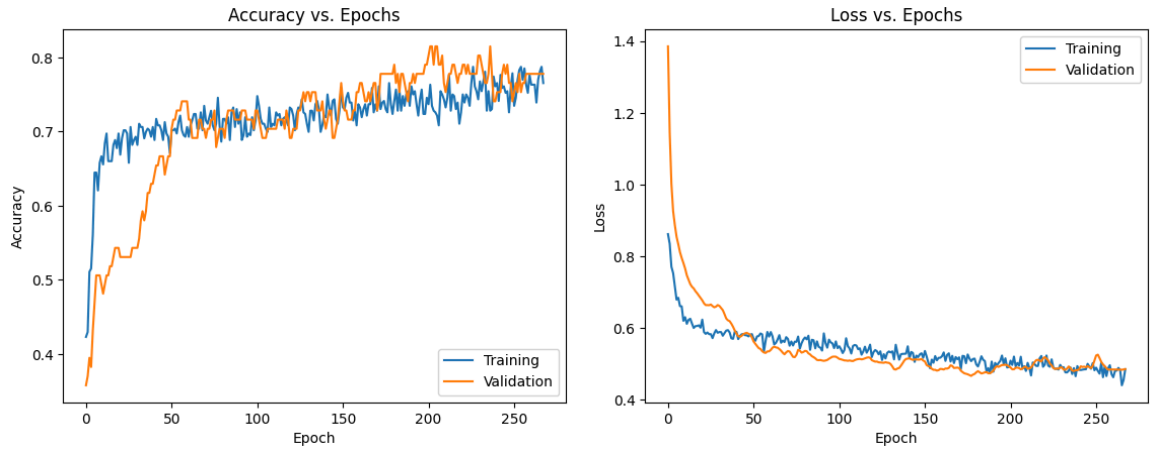

Figura 2. Gráficas de Accuracy vs Epochs y Loss vs Epochs de la red neuronal.

Teniendo en cuenta lo anterior, se puede decir que la red neuronal optimizada presenta un bias bajo, ya que muestra un rendimiento aceptable en los tres conjuntos de datos utilizados. En el gráfico de Accuracy vs Epochs, se observa que la precisión en el conjunto de entrenamiento se estabiliza cerca del 74% (con un valor final de 74.71%), mientras que el

conjunto de validación alcanza el 77.78%. Similarmente, el conjunto de prueba obtuvo un 74.02% de precisión. Los valores de loss para cada conjunto también se mantienen cercanos entre sí (0.50 - 0.48- 0.57), lo cual demuestra aún más la consistencia de los métricos obtenidos en cada conjunto de datos. Debido a lo anterior, el modelo no se encuentra underfitted.

Por otro lado, la varianza del modelo es baja. En el gráfico de loss vs Epochs, se puede observar cómo la pérdida en el conjunto de entrenamiento disminuye de manera constante (de 0.85 a 0.50) y ocurre lo mismo en el de validación (de 1.38 a 0.48) a lo largo de los epochs. Además, el loss para el conjunto de prueba fue de 0.57. Además, se podría decir lo mismo de los porcentajes de precisión de los datasets (74.71% - 77.78% - 74.02%). Lo anterior sugiere que el modelo no presenta overfitting, ya que mantiene un rendimiento consistente en los tres conjuntos de datos, lo que indica una suficiente capacidad de generalización.

En conclusión, este modelo de red neuronal muestra una buena capacidad de generalización sin signos significativos de overfitting ni underfitting debido a que el rendimiento entre los tres conjuntos de datos es balanceado. Sin embargo, es importante destacar que el modelo aún puede mejorarse, ya que su precisión actual de 75% puede elevarse con un mejor ajuste de hiperparámetros, un diferente split de datos o con técnicas de regularización diferentes.

6. Conclusión

Modelo	Precisión Final (%)	Loss Final	Fit
Perceptrón	48.91	2.33	Underfitted
Red Neuronal	69.69	1.40	Overfitted
Red Neuronal Optimizada	74.06	0.57	Balanceado

Tabla 4. Rendimientos finales de los modelos evaluados.

En conclusión, el modelo optimizado demostró una capacidad de generalización superior en comparación con los modelos más simples y sin optimizaciones, que sufrieron de underfitting y overfitting. Lo anterior es gracias a las técnicas de regularización y los ajustes aplicados a la red neuronal optimizada, los cuales permitieron un mejor desempeño al predecir nuevos datos.

7. Referencias

UCI Machine Learning. (2016). *Pima Indians Diabetes Database*. Kaggle. Retrieved September 4, 2024, from https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database