Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Implementación de una técnica de aprendizaje máquina sin el uso de un framework

Marisol S. Ramírez Herrera A01747396

Inteligencia artificial avanzada para la ciencia de datos I

Dr. Jorge Adolfo Ramírez Uresti

5 de septiembre de 2025

Introducción

El aprendizaje automático (ML, por sus siglas en inglés) constituye una rama de la inteligencia artificial dedicada al desarrollo de algoritmos capaces de identificar patrones en los datos y realizar predicciones. A diferencia de las técnicas clásicas de programación, no necesitan instrucciones explícitas para cada tarea específica (Leone, 2022). Estas técnicas permiten generalizar a partir de conjuntos de entrenamiento, generando modelos predictivos aplicables a diversos ámbitos. La clasificación y la regresión, particularmente, han demostrado un notable desempeño.

El presente trabajo tiene como objetivo implementar manualmente una red neuronal con una capa oculta de cuatro neuronas. Para ello, se utiliza el algoritmo de retropagación (*backpropagation*) y la función de pérdida de entropía cruzada para resolver un problema de clasificación binaria. A diferencia del uso de librerías especializadas, la construcción de este modelo permite comprender de manera más profunda los fundamentos matemáticos y computacionales del aprendizaje automático.

Preparación del conjunto de datos.

Con el objetivo de entrenar la red neuronal, se empleó la función *make_classification* de la biblioteca *scikit-learn*. Se generó un conjunto de datos (dataset) controlados compuesto por 7000 muestras, cada una con 2 características para la clasificación binaria. Esta estrategia permite trabajar con un entorno libre de ruido y distribución balanceada para facilitar la validación de la arquitectura implementada.

El dataset está dividido en tres subconjuntos independientes:

- ***Entrenamiento (60%)***: utilizado para el ajuste de hiperparámetros de la red.
- ***Validación (20%)***: empleado durante el entrenamiento para monitorear el desempeño y evitar *overfitting*.
- ***Prueba (20%)***: reservado para la evaluación final del modelo.

La partición estratificada garantizó que la proporción entre clases se mantuviera constante en cada subconjunto.

Arquitectura de la red neuronal

La implementación corresponde a una arquitectura simple diseñada para un problema de clasificación binaria. La arquitectura está conformada por una capa de entrada de dos nodos, correspondientes a la variable del dataset; una capa oculta con tres neuronas; y una capa de salida con una neurona.

La función de activación utilizada en ambas capas corresponde al sigmoide (1):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Los pesos (*W*) se inicializaron aleatoriamente a partir de una distribución normal estándar, y los sesgos (*b*) con valores unitarios.

Forward Propagation

La función de forward propagation transforma las entradas en una predicción para la capa de salida. El cálculo se realiza en dos etapas: de la entrada a la capa oculta (2) y de la capa oculta a la salida (3).

$$\begin{aligned} z^{[1]} &= XW^{[1]} + b^{[1]} \\ a^{[1]} &= \sigma\left(z^{[1]}\right) \end{aligned} \tag{2}$$

$$\begin{aligned} z^{[2]} &= a^{[1]}W^{[2]} + b^{[2]} \\ \hat{y} &= a^{[2]} = \sigma\left(z^{[2]}\right) \end{aligned} \tag{3}$$

donde X es la matriz de entrada, W^1 , W^2 son las matrices de peso, b^1, b^2 son los vectores de sego, $\hat{y}$ corresponde a la probabilidad predicha.

Función de pérdida

Para cuantificar la discrepancia entre las predicciones y las etiquetas reales se utilizó la entropía cruzada binaria (4):

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[\, y_i \log\left(\hat{y}_i\right) + (1 - y_i) \log\left(1 - \hat{y}_i\right) \right] \tag{4}$$

Backpropagation y entrenamiento

El proceso de entrenamiento se basó en gradiente descendente, actualizando pesos y sesgos mediante la retropropagación del error (*backpropagation*).

El gradiente se calculó con la regla de la cadena expresada en forma matricial (Geeks for Geeks, 2025). De este modo, se utilizan las propiedades del producto matricial y la transposición para propagar los errores de una capa a otra.

Para la capa de salida, el error se definió como:

$$\delta^{(2)} = \hat{y} - y \tag{5}$$

De manera análoga, el error reropropagado hacia la capa oculta se calcula como:

$$\delta^{(1)} = \left(\delta^{(2)}(W^{(2)})^T\right) \odot \left(a^{(1)}\left(1 - a^{(1)}\right)\right) \tag{6}$$

Para actualizar los pesos y sesgos:

$$W = W - \alpha \frac{\partial L}{\partial W}$$
$$b = b - \alpha \frac{\partial L}{\partial b} \tag{7}$$

donde α es el aprendizaje (*learning rate*) ajustado a 0.02.

Resultados del entrenamiento

Durante el proceso de entrenamiento se ejecutaron 100 épocas con una tasa de aprendizaje $\alpha = 0.02$. La red mostró una clara disminución de la función de pérdida tanto en el conjunto de entrenamiento como en el de validación. Correspondientemente, existe incremento progresivo de la exactitud.

Figura 1. Iteraciones del entrenamiento para el algoritmo de backpropagation.

```
Epoch 10/100 - Train Loss: 0.1544, Train Acc: 0.980, Val Loss: 0.1745, Val Acc: 0.966
Epoch 20/100 - Train Loss: 0.1202, Train Acc: 0.982, Val Loss: 0.1352, Val Acc: 0.973
Epoch 30/100 - Train Loss: 0.1162, Train Acc: 0.982, Val Loss: 0.1304, Val Acc: 0.974
Epoch 40/100 - Train Loss: 0.1128, Train Acc: 0.982, Val Loss: 0.1279, Val Acc: 0.974
Epoch 50/100 - Train Loss: 0.1094, Train Acc: 0.982, Val Loss: 0.1254, Val Acc: 0.974
Epoch 60/100 - Train Loss: 0.1072, Train Acc: 0.982, Val Loss: 0.1237, Val Acc: 0.974
Epoch 70/100 - Train Loss: 0.1051, Train Acc: 0.982, Val Loss: 0.1221, Val Acc: 0.974
Epoch 80/100 - Train Loss: 0.1028, Train Acc: 0.982, Val Loss: 0.1201, Val Acc: 0.974
Epoch 90/100 - Train Loss: 0.1005, Train Acc: 0.982, Val Loss: 0.1178, Val Acc: 0.974
Epoch 100/100 - Train Loss: 0.0988, Train Acc: 0.982, Val Loss: 0.1159, Val Acc: 0.974
```

Como se observa, la pérdida de entrenamiento (*train loss*) disminuyó progresivamente desde 0.1544 en la época 10 hasta 0.0988 en la época 100. De forma paralela, la pérdida de validación también descendió de 0.1745 a 0.1159. Esto demuestra que el modelo generaliza adecuadamente en los datos de validación.

Por su parte, la precisión de entrenamiento (*train accuracy*) se estabilizó en torno al 98.2 %, mientras que la de validación (*validation accuracy*) alcanzó un 97.4 %.

Figura 2. Gráfica de la precisión para entrenamiento y validación.

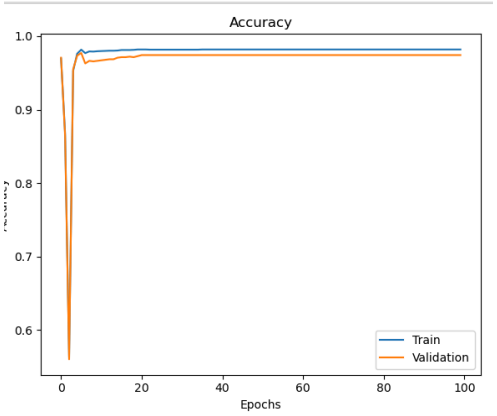
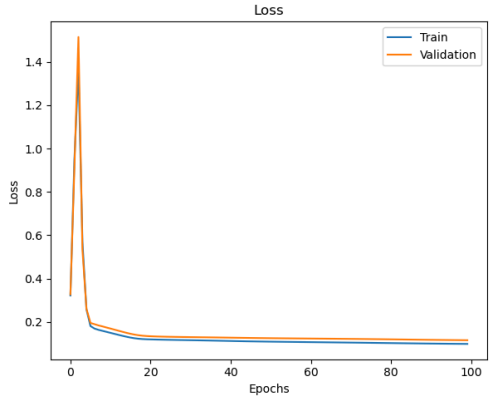


Figura 3. Gráfica de la pérdida para entrenamiento y validación.

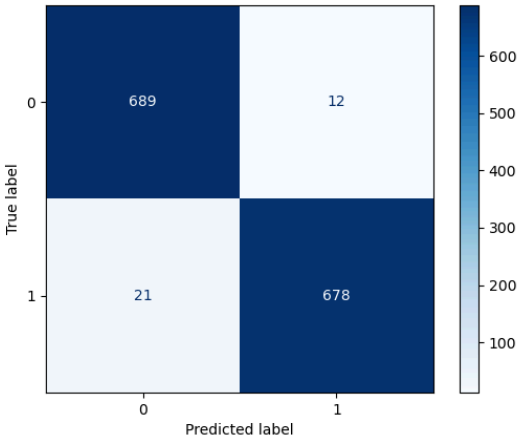


Los resultados sugieren que la red logró convergencia estable, manteniendo curvas de pérdida decrecientes y precisiones elevadas tanto en el conjunto de entrenamiento como en el de validación. La cercanía de los valores de *accuracy* entre el entrenamiento y validación es un indicador de que el modelo no sufrió *overfitting* marcado.

Resultados en el dataset de prueba

El desempeño de la red neuronal se evalúa sobre el dataset de prueba Este conjunto no fue utilizado en ninguna fase previa, por lo que constituye una medida imparcial de la capacidad de generalización del modelo.

Figura 5. Matriz de confusión



Este resultado indica que de un total de 1400 instancias de prueba, únicamente se cometieron 33 errores de clasificación. Éstos corresponden a 12 falsos positivos y 21 falsos negativos). Las clases restantes fueron clasificadas correctamente.

Figura 6. Reporte de clasificación en el conjunto de prueba.

```
=== Clasification Report en Test ===
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       701
           1       0.98      0.97      0.98       699

    accuracy                           0.98      1400
   macro avg       0.98      0.98      0.98      1400
weighted avg       0.98      0.98      0.98      1400
```

El modelo alcanzó una precisión global del 98 %, lo que evidencia un desempeño adecuado. En ambos casos se observa un balance adecuado entre precisión y *recall* (capacidad de evitar falsos negativos). Esto se traduce en un valor de f1-score elevado para ambas clases.

El desempeño en el conjunto de prueba confirma la estabilidad y capacidad de generalización de la red neuronal. La ausencia de discrepancias marcadas entre las métricas de entrenamiento, validación y prueba sugiere que el modelo no presenta *overfitting* significativo. Asimismo, la baja cantidad de errores en la matriz de confusión respalda la confiabilidad del clasificador en un problema de clasificación binaria sintética.

Evaluación con diferente inicialización

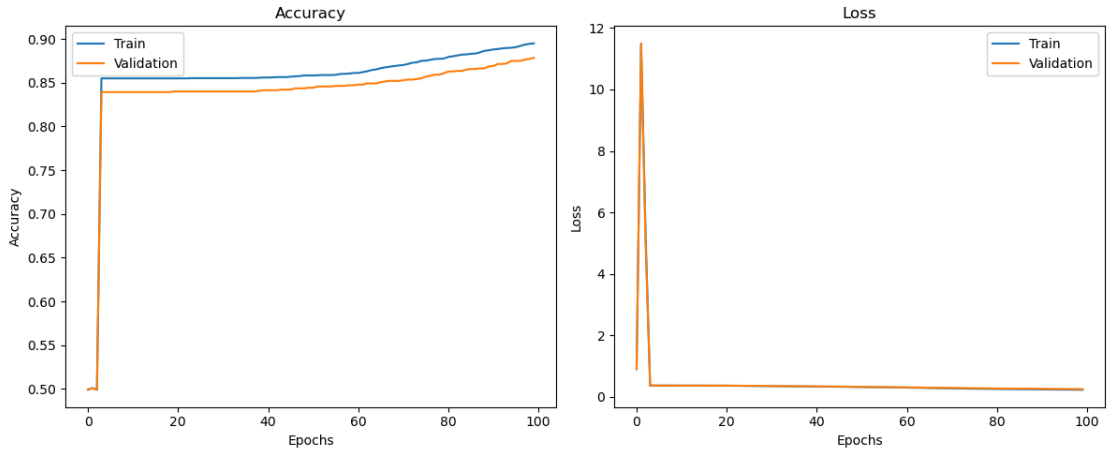
Con el fin de verificar la estabilidad del modelo ante la variabilidad estocástica de los parámetros iniciales, se repitió el entrenamiento estableciendo la semilla aleatoria en 0. Este cambio afecta la inicialización de los pesos y sesgos, lo que naturalmente genera diferencias en la evolución del entrenamiento y en las métricas obtenidas.

Figura 7. Iteraciones del entrenamiento para el algoritmo de backpropagation.

```
Epoch 10/100 - Train Loss: 0.3696, Train Acc: 0.855, Val Loss: 0.3731, Val Acc: 0.839
Epoch 20/100 - Train Loss: 0.3676, Train Acc: 0.855, Val Loss: 0.3704, Val Acc: 0.840
Epoch 30/100 - Train Loss: 0.3546, Train Acc: 0.855, Val Loss: 0.3590, Val Acc: 0.840
Epoch 40/100 - Train Loss: 0.3405, Train Acc: 0.856, Val Loss: 0.3462, Val Acc: 0.841
Epoch 50/100 - Train Loss: 0.3258, Train Acc: 0.858, Val Loss: 0.3311, Val Acc: 0.844
Epoch 60/100 - Train Loss: 0.3080, Train Acc: 0.861, Val Loss: 0.3146, Val Acc: 0.847
Epoch 70/100 - Train Loss: 0.2826, Train Acc: 0.870, Val Loss: 0.2965, Val Acc: 0.852
Epoch 80/100 - Train Loss: 0.2620, Train Acc: 0.878, Val Loss: 0.2784, Val Acc: 0.861
Epoch 90/100 - Train Loss: 0.2480, Train Acc: 0.887, Val Loss: 0.2681, Val Acc: 0.869
Epoch 100/100 - Train Loss: 0.2362, Train Acc: 0.895, Val Loss: 0.2563, Val Acc: 0.879
```

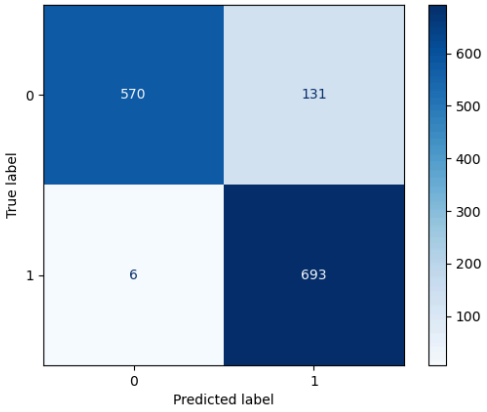
Durante las 100 épocas, la red mostró un comportamiento de disminución progresiva de la función de pérdida. En particular, el modelo alcanzó una precisión de entrenamiento del 89.5 % con pérdida de 0.2362. La precisión de validación del 87.9 %, con pérdida de 0.2563 respectivamente. Si bien presenta una precisión menor, la red logró converger hacia una solución adecuada, sin evidencias claras de *overfitting*.

Figura 8. Gráfica de métricas para entrenamiento y validación.



Como se observa en la gráfica, las métricas de entrenamiento y validación evolucionan de manera paralela.

Figura 9. Matriz de confusión.



De las 1400 instancias de prueba, se cometieron 140 errores. Éstos corresponden a 133 falsos negativos y 7 falsos positivos. Sin embargo, siguee representando una proporción baja en relación con el total.

Figura 10. Reporte de clasificación en el conjunto de prueba.

```
=== Clasification Report en Test ===
              precision    recall  f1-score   support

           0       0.99      0.81      0.89       701
           1       0.84      0.99      0.91       699

    accuracy                           0.90      1400
   macro avg       0.92      0.90      0.90      1400
weighted avg       0.92      0.90      0.90      1400
```

En el conjunto de prueba, el modelo alcanzó una precisión global del 90%. Esto es menor a la corrida previa, pero aún evidencía un rendimiento satisfactorio y consistente.

La comparación con la primera corrida muestra que, aunque los resultados varían dependiendo de la inicialización aleatoria de los parámetros, el modelo mantiene un desempeño sólido en términos de precisión, *recall* y f1-score.

Referencias

Geeks for Geeks. (2025). Implementation of neural network from scratch using NumPy. *Geeks for Geeks*. Recuperado de https://www.geeksforgeeks.org/numpy/implementation-of-neural-network-from-scratch-using-numpy/

Leone, E. (2022). Introducción artificial intelligence y machine learning para desarrolladores de aplicaciones. *AWS*. Recuperado de https://aws.amazon.com/es/blogs/aws-spanish/introduccion-artificial-intelligence-y-machine-learning-para-desarrollares-de-aplicaciones/

Nayar, S. (2021). *Backpropagation Algorithm | Neural Networks*. [Archivo de video]. Youtube. https://www.youtube.com/watch?v=sIX_9n-1UbM

Uresti, J. (2025). *Tema 6. Redes neuronales.* Inteligencia artificial para la ciencia de datos I. ITESM CEM.