Informe Final

Desarrollo y Evaluación de una Red Neuronal

Profesor: Fernandez Agustin
Alumnos: Madruga Bautista, Alvarez Tomas

1. Introducción

El presente informe detalla el proceso de diseño, implementación y comparación de una red neuronal aplicada a un sistema de predicción de movimientos de un robot en función de entradas sensoriales. Se desarrollaron dos versiones de la red: una vectorizada (con uso de operaciones de la librería NumPy) y otra no vectorizada (basada en estructuras for, para comprender el comportamiento interno a bajo nivel). El trabajo se enfoca en comprender en gran detalle el funcionamiento / construcción de una red neuronal y sus tipos.

2. Objetivos

- Implementar desde cero una red neuronal MLP - Backpropagation.
- Comprender las ventajas / desventajas de la vectorización / no vectorización.
- Identificar problemas en redes neuronales: desbalance de datos de entrenamiento, underfitting / overfitting.
- Medir tiempos de entrenamiento para comparar rendimiento.

3. Arquitectura del Modelo

Ambas versiones comparten la misma arquitectura:

- Entradas: 4 (S1 a S4, representando sensores direccionales).
- Capa oculta: 1 capa oculta con 5 neuronas.
- Salidas: 2 (M1 y M2, representando motores).
- Función de activación: tanh.

Debido a que la función de activación tanh toma cualquier número real y lo transforma al rango [-1, 1], fue necesario establecer que los pesos iniciales (W y b) se generen aleatoriamente dentro de un rango acotado, en nuestro caso [-0.5, 0.5].

Si se hubiesen inicializado con valores más grandes, los productos Wx + b habrían caído en las zonas saturadas de la tanh, donde su derivada tiende a cero.

Como consecuencia, la retropropagación del error no habría sido efectiva, ya que la derivada tan baja no permite que el error se propague hacia atrás de forma significativa. Esto genera un entrenamiento lento o incluso nulo, dificultando el aprendizaje de la red.

4. Implementaciones

4.1. Red Neuronal Vectorizada (NeuralNetwork)

- Usa operaciones NumPy (np.dot, np.tanh).
- Entrenamiento rápido y eficiente.
- Maneja datasets por batch.
- Menos líneas de código.

4.2. Red Neuronal No Vectorizada (NeuralNetworkNoVectorizada)

- No se utiliza NumPy para la lógica principal del forward y backpropagation.
- Calcula cada activación, suma ponderada y actualización de pesos con ciclos for.
- Permite observar cómo se propaga el error y se ajustan pesos.
- Más lenta, especialmente en entrenamiento, leves diferencias sobre el tiempo de resolución.

Compatibilidad

Para permitir el intercambio de los modelos en el backend, ambas clases fueron creadas con los mismos nombres de métodos: train, train_step, forward, summary. De esta forma, el cambio de lógica no supone errores.

5. Entrenamiento y Pruebas

- El entrenamiento se realiza con 7 patrones, dejando 9 combinaciones como generalización (por consigna).

- El método de entrenamiento usa error cuadrático medio (MSE) como métrica.
- Se generaron CSVs con topologías y se analizaron las salidas de la relación entre neuronas - MSE promedio - tiempo de entrenamiento.
- Se midió el tiempo de entrenamiento en milisegundos.

6. Evaluación de Desempeño

Métrica	Vectorizado (np)	No Vectorizado
Tiempo de entrenamiento	Muy bajo (milisegundos)	Aumenta linealmente con dataset
Legibilidad de código	Alta	Baja
Propenso a errores	Baja	Media/Alta

7. Conclusion

El trabajo nos permitió no solo implementar una red neuronal desde cero, sino también comprender su funcionamiento más profundamente, además de poder ver y entender la eficiencia dada por los diferentes tipos de modelos (vectorizados,no vectorizados). También pudimos comprender diferentes conceptos sobre el entrenamiento de las redes neuronales, como métodos de activación, underfitting, overfitting, etc. siendo muy importantes para el entrenamiento de redes con una gran cantidad de datos de entrada.