Inteligencia Artificial II

LAB 02
Implementación de modelos neuronales supervisados

1. Objetivo del laboratorio

*Desarrollar de forma autónoma distintas implementaciones de **redes neuronales de aprendizaje supervisado** que permitan resolver distintos casos de uso. La práctica comenzará con la implementación más sencilla del funcionamiento de una neurona (percetrón) y terminará con la construcción de dos **MLPs** capaces de hacer predicciones para los casos concretos planteados.*

2. Elementos a utilizar:

1. Lenguaje Python
2. Librerías: numérica *NumPy*, estructuras de datos *pandas*, *scikit-learn*, gráfica *Matplotlib* (opcional si se quieren implementar gráficas) y entorno *PyTorch*.
3. Entorno Anaconda (opcional)
4. Editor Jupyter (opcional)
5. VSCode (opcional)
6. Archivos .csv proporcionados

3. Práctica 1 (Perceptrón para funciones lógicas)

Objetivo

Resolver funciones lógicas usando un modelo neuronal simple. Deberás codificar un modelo neuronal desde cero según lo visto en teoría y las instrucciones del apartado **Implementación**. Responde a las preguntas que se plantean en **Cuestiones**. Debes usar comentarios con profusión, incluyendo celdas específicas donde expliques los algoritmos que usas y el código que has programado.

Implementación

Crea el notebook *L2P1-Perceptron.ipynb*. Programa un PERCEPTRÓN cuyos pesos iniciales tendrán un valor al azar entre -1.0 y 1.0 ambos incluidos y que se entrenará usando la Ley de Hebb.

1. Resuelve la función **AND**. Prueba diferentes *learning rates* y umbrales para saber cuál es el óptimo. Sacar por pantalla las salidas de la red neuronal y el valor del error para cada iteración.
2. Repetir el ejercicio anterior para la función **XOR**.

Entregar los resultados de ambas pruebas en el archivo *L2P1-Perceptron.csv*

Cuestiones

Elabora una memoria de la práctica con una tabla para cada caso. Discútelas según lo visto en teoría. Escribe de forma explícita la ecuación del hiperplano que se ha generado en cada uno. Las tablas tendrán la estructura

Iteración	Entradas		Pesos iniciales		Yr	Yd	Error	Pesos finales	
	X1	X2	W1	W2				W1	W2

4. Práctica 2 (MLP para funciones lógicas)

Objetivo

Utiliza la librería *PyTorch* para construir y entrenar un MLP que resuelva problemas no lineales. En vez de la Regla Delta Generalizada, usaremos **Adam** como función de modificación de matriz de pesos (optimizer) de la forma que se indica en el apartado de **Implementación**. Responde a las preguntas que se plantean en **Cuestiones**.

Implementación

Crea el notebook *L2P2-MLP.ipynb*. Crea un modelo neuronal totalmente conectado (*fully connected*) y *secuencial*. El modelo tendrá que resolver la función **XOR**. Prueba con distinto número de neuronas en la capa oculta (estima el rango de variación según lo visto en teoría) y varios valores de *epoch*. Haz al menos 3 modelos distintos intentando optimizar los resultados. Usa como optimizador el método *Adam*, como funciones de activación *ReLU* y *sigmoide*, y como función de error *Mean Squared Error*.

Notas:

- El *learning rate* es un atributo de la clase *optimizer*.
- Usa arrays de *numpy* para los valores de entrenamiento y sus salidas de predicción.
- *Adam*: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Cuestiones

Continúa en la memoria creando varias tablas (como la siguiente) con los resultados del MLP que resuelve la función. Cada fila de la tabla será una de las variaciones que hayas probado para ese modelo. Haz una tabla distinta para los experimentos con cada una de las dos funciones de activación propuestas

Neuronas capa oculta	Épocas	Salida	Error

5. Práctica 3 (MLP para predecir muertes en “Juego de Tronos”)

Objetivo

Utiliza lo practicado en este laboratorio para resolver un caso práctico “real” usando la librería *PyTorch* y una arquitectura MLP para predecir la probabilidad de muerte de los personajes de **Juego de Tronos** (GoT)

Implementación

Crea el notebook *L2P3-MLP_GoT.ipynb*. El programa en *Python* deberá responder a los siguientes puntos:

1. Contendrá un MLP que has de **entrenar** y **validar** con la información del archivo ‘*got_train.csv*’. Los valores que se usarán para entrenar serán:
   - Los que indican el género del personaje
   - Si aparece en los libros 1, 2, 3, 4 y 5
   - Si está casado

- Si es noble o no
- El número de personas de su entorno que han muerto
- Si es un personaje popular.

El valor que nos indica la posibilidad de sobrevivir de un personaje es '**alive'**, con valores entre 0 y 1.

2. Una vez creada la red neuronal y viendo que puede predecir correctamente, usar los valores del archivo '*got_predict.csv*' y predice cual de ellos es el personaje con más posibilidades de morir.

Cuestiones

Continúa en la memoria respondiendo a las siguientes cuestiones

1. Dibuja la arquitectura de red y escribe el *optimizer*, *nº de neuronas de la capa oculta* y las *funciones de activación* y de *error* usadas. Indica cual es el valor de error final para el conjunto de entrenamiento y el de validación.
2. Predice el valor de *alive* de los personajes en *got_predict.csv*. ¿Quién es el personaje con más posibilidades de morir? ¿Cuál es el error cometido en esta predicción? ¿Es mayor o menor que el de validación? ¿Por qué?
3. Consigue que sea otro el personaje con más probabilidades de morir. ¿Qué cambio has realizado en el fichero *got_predict.csv*? Justifícalo desde el punto de vista de las redes neuronales. ¿Todas las entradas son igual de relevantes? ¿Cómo lo puedes demostrar?

6. Forma de entrega del laboratorio:

La entrega consistirá en un fichero comprimido RAR o ZIP con nombre LAB02-GRUPOxx.RAR subido a la tarea **LAB2** que **contenga únicamente**

1. **Por cada práctica** un notebook de Jupyter (archivos con extensión **.ipynb**).
2. Una **memoria del laboratorio** en Word.

Las entregas que no se ajusten exactamente a esta norma NO SERÁN EVALUADAS.

7. Rúbrica de la Práctica:

1. IMPLEMENTACIÓN: Multiplica la nota del trabajo por 0/1

Siendo una práctica de IA, todos los aspectos de programación se dan por supuesto. La implementación será:

- Original: Código fuente no copiado de internet. Grupos con igual código fuente serán suspendidos
- Correcta: Los algoritmos deben estar correctamente programados. El programa funciona y ejecuta correctamente todo lo planteado en el apartado "*Cuestiones*" de cada práctica.
- Comentada: Inclusión (**obligatoria**) de comentarios.

2. MEMORIA DEL LABORATORIO

Es obligatorio que la redacción de la memoria sea clara y correcta tanto ortográfica como gramaticalmente. Debe seguir la estructura recogida en la plantilla:

- *Portada con el nombre de los componentes del grupo y el número del grupo*
- *Índice*
- *Resultados de la Práctica 1*
- *Resultados de la Práctica 2*
- *Resultados de la Práctica 3*
- *Bibliografía*

Calificación de las cuestiones:

PRÁCTICA	CUESTIÓN	VALORACIÓN (sobre 10)
Práctica 1	Cuestión 1	3
Práctica 2	Cuestión 1	3
Práctica 3	Cuestión 1	1
	Cuestión 2	1
	Cuestión 3	2