Guión de prácticas
P3 – Redes Neuronales

En esta práctica comprenderemos mejor las redes neuronales artificiales, que son el objeto del tema 8 de teoría. En particular, los objetivos son:

- Familiarizarnos con cada uno de los tipos de redes propuestas: los perceptrones multicapa (MLP) y las redes convolucionales (CNN).
- Ser capaces de comprender las ventajas y desventajas de cada red, así como para qué tipo de problema es más adecuada cada una de ellas.

Dividiremos la práctica en dos partes. En la primera (estudio guiado) trabajaremos con un conjunto de datos tabular (breast-cancer-train.csv y breast-cancer-test.csv) que ya hemos usado en las prácticas anteriores. En la segunda (caso práctico) abordaremos un problema que implica procesamiento de imágenes (fashion-mnist_train.csv y fashion-mnist_test.csv) y el objetivo será alcanzar el mejor rendimiento posible poniendo en práctica lo aprendido previamente en el estudio guiado. A lo largo de la práctica utilizaremos la implementación de MLP proporcionada por Scikit Learn:

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

También vais a trabajar con las implementaciones de MLP y CNN proporcionadas por Keras (https://www.tensorflow.org/guide/keras?hl=es), que es una API de alto nivel que nos permite trabajar con TensorFlow de forma sencilla. En el siguiente enlace podéis ver un ejemplo de cómo entrenar un perceptrón MLP en Keras:

https://keras.io/examples/structured_data/structured_data_classification_from_scratch/

1. Estudio guiado

Comenzaremos con el problema de clasificación binario BreastCancer. Seguid los siguientes pasos:

a. Cargad el conjunto de datos de entrenamiento (breast-cancer-train.csv) y realizad una inspección/visualización inicial del mismo. Podéis aplicar el mismo pre-procesamiento realizado en las prácticas anteriores a fin de tratar los *missing values*.
b. Para sacar el máximo rendimiento a las redes neuronales artificiales, necesitamos escalar los datos. Probad las siguientes opciones: MinMaxScaler, StandardScaler, RobustScaler, y Normalización logarítmica, para la cual podéis usar funciones
lamda: dt_train_log_X[i] = dt_train_red_X[i].apply(lambda x: np.log(x) if x != 0 else 0)
c. Entrenad un perceptrón multicapa, MLP, con sklearn para cada dataset escalado previamente (https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html), usando los parámetros por defecto. Visualizad gráficamente la arquitectura de las redes MLP construidas.
d. Ahora volved a entrenar los MLP pero usando GridSearchCV para ajustar los siguientes hiperparámetros: *hidden_layer_sizes*, *learning_rate_init*, *activation*, *solver* y *max_iter*
e. Comparad el mejor MLP con los mejores modelos construidos en las prácticas anteriores (P1 y P2) para este mismo conjunto de datos: KNN, Árbol de Decisión y SVM. Comparad el rendimiento de los modelos con los datos de entrenamiento (considerando 5 CV) y prueba (breast-cancer-test.csv).

Guión de prácticas
P3 – Redes Neuronales

2. Caso práctico

Vamos a trabajar con el conjunto de datos **FashionMNIST** (https://github.com/zalandoresearch/fashion-mnist): utilizad los ficheros para train (fashion-mnist_train.csv) y test (fashion-mnist_test.csv) dados en el aula virtual.

Fashion-MNIST es un conjunto de datos creado por la tienda de moda online Zalando. Consta de 70 000 imágenes en escala de grises de 28×28 píxeles. Sus autores lo publicaron con el fin de que sirviera como banco de pruebas para el desarrollo de nuevos algoritmos de aprendizaje automático. Puede considerarse el sucesor del famoso MNIST, un conjunto de datos de reconocimiento de dígitos manuscritos ampliamente utilizado por la comunidad. Fashion-MNIST mantiene muchas similitudes con su predecesor (resolución, escala de grises, número de ejemplos, número de clases, etc.). Sin embargo, como podemos ver en la siguiente imagen, en este caso las categorías son ropa y accesorios.

El conjunto de datos está completamente balanceado, con 7000 ejemplos de cada categoría. Utilizaremos 60 000 imágenes para el entrenamiento (fashion-mnist_train.csv) y reservaremos las 10 000 restantes para las pruebas (fashion-mnist_test.csv).

Seguid los siguientes pasos:

a. Cargad el conjunto de datos de entrenamiento (fashion-mnist_train.csv) y realizad una inspección/visualización inicial del mismo. Escalad los datos como consideréis más apropiado. Además de los operadores de escalado vistos en la parte guiada, al trabajar con píxeles podéis probar a escalar dividiendo por 255, dado que cada píxel va a tomar valores en [0, 255].
b. Entrenad un perceptrón MLP con sklearn para el conjunto de datos escalado previamente (https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html). Utilizad como variables de entrada todos los píxeles sin procesar (784). En esta etapa, no es

necesario optimizar los hiperparámetros de la red. Bastará con probar una arquitectura de red no demasiado compleja para que el entrenamiento no se prolongue demasiado. Es recomendable mostrar el progreso del aprendizaje en pantalla (parámetro "detallado"). Una vez obtenido un modelo prometedor, se mide el error en el conjunto de prueba.

c. Repetid el entrenamiento, pero esta vez usando la API de Keras con el framework TensorFlow en vez de Sklearn. De aquí en adelante, **utilizaremos GoogleColab para poder usar TensorFlow y GPUs en la nube.** La principal diferencia entre Sklearn y TensorFlow radica en que este último se centra en la optimización, buscando aprovechar al máximo los recursos disponibles (CPU y GPU). Por ello, junto con PyTorch, es uno de los frameworks más populares en aprendizaje profundo, donde la potencia de cómputo cobra mayor importancia. Compara los resultados obtenidos por Keras y Sklearn, en cuanto a rendimiento (precisión y tiempo de ejecución).

d. En el resto de la práctica nos vamos a centrar en el entrenamiento de una red CNN, que es un tipo de red neuronal profunda que se caracteriza por tener ciertas capas de neuronas inspiradas en el comportamiento de las neuronas de la corteza visual primaria de un cerebro biológico. Una arquitectura CNN típica consta de: capas de convolución, capas de agrupación, capas de normalización, capas de abandono y capas densas totalmente conectadas. Las primeras capas de la red (convoluciones, agrupación, etc.) realizan la extracción automática de características; es decir, a partir de las imágenes originales, aprenden por sí mismas qué propiedades son más relevantes para mejorar las predicciones del modelo. Aquí es donde suele destacarse la «profundidad» de la red. Las últimas capas suelen ser densas y se comportan de forma similar a un perceptrón multicapa. Entrenad una CNN que tenga (además de las capas de entrada y salida):
    - Una única capa de convolución 2D con 16 filtros y un tamaño de kernel de 3x3 (https://keras.io/api/layers/convolution_layers/convolution2d/). Los demás parámetros conservarán sus valores predeterminados.
    - Una capa de max pooling de 2x2 después de la capa de convolución (https://keras.io/api/layers/pooling_layers/max_pooling2d/).
    - Una capa densa de 20 neuronas y todos los demás parámetros por defecto (https://keras.io/api/layers/core_layers/dense/).
    - Utilizad el Descenso de Gradiente Estocástico (SGD) como optimizador con una tasa de aprendizaje fija de 0,1 y entrenad durante 10 épocas con un tamaño de lote (*batch*) de 128 imágenes.
    - Intentad mejorar la arquitectura de red anterior, con total libertad en el diseño, pudiendo aumentar la profundidad de la red, introducir otros tipos de capas, usar otros optimizadores, etc. Comparad las distintas arquitecturas de CNN construidas en 2d, en términos de complejidad y rendimiento respecto a los datos de entrenamiento y prueba.

    En el siguiente enlace podéis ver un ejemplo de una CNN simple que aprende a clasificar los dígitos de MNIST (Un problema de Procesamiento de Imágenes más simple que FashionMNIST pero que os puede servir de referencia para construir vuestra arquitectura de red CNN): https://keras.io/examples/vision/mnist_convnet/

Guión de prácticas
P3 – Redes Neuronales

Realización, entrega y evaluación de la práctica

- Realizaréis el trabajo en el mismo **grupo de personas** que el resto de prácticas, según las pautas siguientes:
  - La entrega será única, siendo responsables todas las personas integrantes del grupo.
  - Debéis explicar y justificar suficientemente las diferentes tareas realizadas en la práctica, demostrando dominio de los contenidos.
- Dedicaremos a la práctica cuatro sesiones interactivas.
- La fecha límite de entrega será la indicada en el aula virtual.
- La entrega consistirá en un único fichero comprimido en **formato .zip** que contenga una memoria (en PDF) describiendo el trabajo realizado, justificando cada decisión tomada, y todos los ficheros con la resolución de los ejercicios indicados en el guion:
  - El fichero fuente y el HTML de los notebooks Python (Docker y Colab) con la realización de la práctica, y todo el código debidamente documentado.
  - Todos los ficheros auxiliares necesarios.
- La entrega se calificará sobre un máximo de 10 puntos. En la evaluación tendremos en cuenta:
  - La claridad y calidad de las descripciones, explicaciones y justificaciones en el informe.
  - Calidad de los modelos construidos y claridad del código desarrollado.
  - La calificación de la entrega tiene dos componentes:
    - **C1 (70%):** Una componente común idéntica para todos los miembros del equipo, que se corresponde con la parte conjunta del trabajo, la memoria escrita y el código desarrollado. En esta parte se valorarán por separado el proceso seguido y los resultados obtenidos en cada parte de la práctica. El primer ejercicio (“Estudio Guiado”) pesará un 3/10, mientras que el segundo (“Caso Práctico”) supondrá el 7/10 restante.
    - **C2 (30%):** Una componente individual, que se corresponde con el resultado de las respuestas al test de evaluación de la práctica 3 (que incluirá cuestiones asociadas a los ejercicios realizados en las 4 sesiones interactivas correspondientes a la P3).