Inteligencia Artificial 2026

Lab 03

02.marzo.2026

En este laboratorio vamos a diseñar redes neuronales *fully connected* en la librería `tensorflow.keras` para un problema de clasificación y un problema de regresión.

Ejercicio 1: Clasificación de Dígitos (Scikit-Learn digits)

Este ejercicio se centra en entender cómo una red neuronal "ve" patrones espaciales aplanados.

Dataset: `sklearn.datasets.load_digits()` contiene 1,797 muestras, 64 atributos.

Objetivo: Configurar la capa de salida y la función de pérdida para clasificación multiclase (10 dígitos).

Instrucciones:

- Normalizar los datos dividiendo por 16 (el valor máximo de los píxeles). Debe explicar las transformaciones que se están haciendo tanto al conjunto de datos, como a las etiquetas.
- Diseñar una arquitectura con al menos 2 capas ocultas. En cada capa, utilizar las funciones de activación adecuadas. Justifique la elección del número de capas y de neuronas, así como de las funciones de activación.
- Elegir la función de pérdida apropiada, así como los hiper-parámetros que considere más adecuados. Justifique la elección de estos hiper-parámetros.
- Mostrar una Matriz de Confusión usando `seaborn` para identificar qué números confunde la red (comúnmente debería ser el 1 con el 7 o el 3 con el 8).

Muestre un conjunto de métricas de desempeño (tanto en entrenamiento como en test), para determinar la efectividad de su clasificador. Muestre 5 ejemplos de imágenes bien clasificadas y 5 ejemplos de imágenes mal clasificadas.

Ejercicio 2: Regresión de Precios (Scikit-Learn california$_housing$)

Este ejercicio se centra en construir un modelo de regresión mediante una red neuronal densa.

Dataset: `sklearn.datasets.fetch_california_housing()`.

Objetivo: Manejo de escalas heterogéneas y métricas de error continuo.

Instrucciones:

- Estandarizar los datos (Escalamiento Obligatorio). Usar `StandardScaler` en los datos de entrada. Sin esto, la red tardará demasiado en converger o los gradientes explotarán.

- Probar una red más profunda (ej. 3 capas: 64, 32, 16 o similar) dada la complejidad de los datos socioeconómicos. Justificar la elección de su arquitectura.

- Elegir la función de pérdida apropiada (ej. MSE o MAE) y usar un optimizador Adam. Justifique la elección de sus hiper-parámetros.

- Mostrar un Generar un Scatter-Plot de Predicciones vs. Valores Reales.

Muestre un conjunto de métricas de desempeño (tanto en entrenamiento como en test), para determinar la efectividad de su regresión. Elabore 3 ejemplos de predicción con su modelo de regresión para 3 observaciones nuevas (que no son parte del dataset).

Recomendaciones:

- Si no cuenta con `tensorflow`, se sugiere instalarlo en un nuevo ambiente virtual, para no interferir con otros ambientes de Python que ya tenga en funcionamiento.

- Recuerde partir el conjunto de datos en *train* y *test*, de forma que ambos conjuntos reflejen la distribución de los datos originales. Explique el procedimiento de partición y la elección de las proporciones train-test.

- No olvide elegir hiper-parámetros adecuados para:
  - número de capas
  - número de neuronas en cada capa
  - funciones de activación (principalmente en la última capa)
  - función de pérdida
  - algoritmo de optimización
  - *learning-rate* α
  - *batch-size* o tamaño del batch
  - porcentaje para conjunto de validación
  - número de *epochs*
  - métricas para evaluar el desempeño