Clase 101 — Regresión y clasificación con MLP

Parte: 2 — Deep Learning · Fuente: Géron, cap. 10 § Building an Image Classifier y § Building a Regression MLP. Duración estimada: 75 min.

Clase 101 — Regresión y clasificación con MLP

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 10 § Building an Image Classifier y § Building a Regression MLP. Duración estimada: 75 min.*

Objetivo

Saber construir y entrenar un MLP para los tres tipos de problemas tabulares estándar — regresión, clasificación binaria y clasificación multiclase — eligiendo correctamente la activación de salida y la loss para cada caso. Hacer un train/val/test split adecuado y leer las curvas de entrenamiento.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Mapear problema → activación de salida → loss: regresión → linear + MSE; binario → sigmoid + binary_crossentropy; multiclase → softmax + sparse_categorical_crossentropy.
- Hacer split train / validation / test con train_test_split y pasar validation_data a model.fit.
- Leer history.history['loss'] y ['val_loss'], identificar overfitting (val sube mientras train baja).
- Aplicar EarlyStopping y ModelCheckpoint callbacks como protección estándar.
- Diferenciar sparse_categorical_crossentropy (labels enteros) de categorical_crossentropy (labels one-hot).

Temas

- Mapeo problema → arquitectura de salida + loss.
- Activación de salida: linear, sigmoid, softmax.
- Train/val/test split — por qué hace falta los tres (val para selección, test para reporte final).
- Curvas de aprendizaje: lectura visual (subfitting / overfitting).
- Callbacks: EarlyStopping(patience=5, restore_best_weights=True), ModelCheckpoint.
- Normalización de inputs con Normalization() layer o StandardScaler.

Definiciones y características

- linear activation: sin transformación. Salida no acotada. Para regresión.
- sigmoid: $1/(1+e^x)$, sale en (0, 1). Una neurona = probabilidad binaria.
- softmax: $e^{x_i} / \Sigma e^{x_j}$. Convierte logits en distribución de probabilidad sobre K clases.
- MSE: mean((y - ŷ)²). Para regresión. Sensible a outliers (cuadrado).
- MAE: mean(|y - ŷ|). Robusto a outliers.
- Binary Cross-Entropy: -[y·log(p) + (1-y)·log(1-p)]. Para 2 clases.
- Sparse Categorical Cross-Entropy: como categorical pero con labels enteros ([0, 2, 1, ...]). Más eficiente y común.
- EarlyStopping: corta entrenamiento cuando val_loss deja de bajar por patience épocas.

Dataset / recursos

- Regresión: sklearn.datasets.fetch_california_housing().
- Clasificación binaria: sklearn.datasets.load_breast_cancer().
- Multiclase: keras.datasets.fashion_mnist.load_data() (10 clases).
- Librerías: tensorflow, keras, scikit-learn, matplotlib.

Ejercicios

1. MLP regresión: California Housing, MLP [64, 32] con Normalization(), salida linear, loss mse. Reportar MAE en test.
2. MLP binario: breast_cancer, MLP [32, 16], salida sigmoid, loss binary_crossentropy. Reportar accuracy y AUC.
3. MLP multiclase: Fashion-MNIST (aplastado a 784), MLP [256, 128], salida softmax(10), loss sparse_categorical_crossentropy. Reportar accuracy.
4. Curvas y overfitting: entrenar el modelo 3 por 50 épocas sin early stopping. Graficar loss y val_loss; identificar la época donde arranca overfitting.
5. EarlyStopping: repetir con EarlyStopping(patience=5, restore_best_weights=True). Verificar que cortó antes y los pesos guardados son los mejores.

Homework verificable

Fashion-MNIST end-to-end:

1. Cargar y normalizar (/ 255).
2. Split train (50 000) / val (10 000) / test (10 000).
3. MLP [300, 100], ReLU, salida softmax.
4. Compilar con optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'].
5. Entrenar con EarlyStopping(patience=5) y ModelCheckpoint('best.keras', save_best_only=True).
6. Reportar accuracy en test y matriz de confusión.

Criterio de aceptación: accuracy en test ≥ 0.87; matriz de confusión muestra que las prendas más confundidas son Shirt / T-shirt / Pullover.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
loss = nan desde la época 1	Inputs sin escalar + LR alta. Fix: Standar
softmax + binary_crossentropy	Mismatch. Fix: softmax va con (sparse_)cat
categorical_crossentropy con labels entero	Espera one-hot. Fix: usar sparse_categoric
accuracy = 0.10 en Fashion-MNIST sin entre	Predicción aleatoria. Fix: model.fit(...).
Reportar accuracy en val como "test final"	Validation está contaminada por la búsqued

Preguntas frecuentes

¿Cuántas neuronas por capa?

Primera capa entre n_features y 4·n_features; capas posteriores más angostas (pirámide invertida). Ej.: 784 → 256 → 128 → 10. Refinar con tuning (Clase 095).

¿Qué optimizador uso?

Adam con LR default (1e-3) es el caballito de batalla. Para producción, AdamW con LR ajustada (Clase 102).

¿Cuántas épocas?

Las que decidan tus callbacks. Setear epochs=100 + EarlyStopping(patience=10).

¿Hay que normalizar siempre?

Siempre para features con escalas distintas. Para imágenes basta / 255. Sin normalizar, el optimizador zigzaguea.

¿Por qué la última capa de Fashion-MNIST tiene 10 neuronas?

Una neurona por clase. Softmax convierte los 10 logits en probabilidades que suman 1. El argmax es la predicción.

Referencias

- Géron, cap. 10 — Building an Image Classifier Using the Sequential API y Building a Regression MLP.
- Keras docs — Training & evaluation.
- Keras docs — Callbacks API.
- Glorot & Bengio (2010), Understanding the difficulty of training deep feedforward neural networks.

Siguiente clase

Clase 102 — Keras Sequential API

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb