Clase 126 — Keras preprocessing layers

Parte: 2 — Deep Learning · Fuente: Géron, cap. 13 § The Keras Preprocessing Layers.
Duración estimada: 65 min.

Clase 126 — Keras preprocessing layers

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 13 § The Keras Preprocessing Layers. Duración estimada: 65 min.*

Objetivo

Hacer preprocesamiento dentro del modelo con las preprocessing layers de Keras (Normalization, StringLookup, IntegerLookup, Discretization, CategoryEncoding, Hashing, TextVectorization). Beneficio: el preprocesamiento viaja con el modelo (.keras), no como código separado — elimina el clásico "train-serve skew" en producción.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Usar Normalization() con .adapt(data) para escalar features tabulares.
- Usar StringLookup para encoding de categóricas.
- Aplicar TextVectorization para tokenización + indexing.
- Construir un modelo "todo en uno": preprocesamiento + red en el mismo keras.Model.
- Reconocer la ventaja: model.predict(raw_data) funciona, sin necesidad de scaler separado.

Temas

- Normalization: substrae mean, divide por std. .adapt(data) aprende los stats.
- StringLookup / IntegerLookup: mapea categorías a índices enteros.
- CategoryEncoding: one-hot, multi-hot, count.
- Discretization: convierte continua en bucket (bin_boundaries=...).
- Hashing: mapea categorías a buckets via hash (sin necesidad de vocabulario).
- TextVectorization: tokeniza + lookup en una capa.

Definiciones y características

- .adapt(data): aprende los parámetros (mean/std, vocabulario, etc.) desde un dataset. Reemplaza un fit-then-transform separado.
- Normalization: (x - mean) / std. Mean/std aprendidos con .adapt.
- StringLookup(vocabulary=..., output_mode='int'): dict {categoría: índice}. 'multi_hot' para one-hot.
- Hashing(num_bins=1024): para vocabulario gigante o variable. Hash mod num_bins.
- TextVectorization(max_tokens=10_000, output_mode='int', output_sequence_length=100): tokeniza + indexa.

Dataset / recursos

- California Housing (tabular).
- IMDB reviews (texto).
- Librerías: tensorflow, keras.

Ejercicios

1. Normalization tabular: norm = Normalization(); norm.adapt(X_train); X_norm = norm(X_test). Verificar que mean ≈ 0, std ≈ 1.
2. StringLookup: con un array de categorías, lookup = StringLookup(); lookup.adapt(categorias); lookup(['A', 'B', 'C']) → tensor de ints.
3. Modelo end-to-end tabular: inputs = Input((n,)); x = Normalization()(inputs); x = Dense(64, ...)(x); .... Después .adapt(...) la capa norm con X_train.
4. TextVectorization: sobre IMDB, tokenizar, entrenar un modelo de sentimiento.
5. Hashing: con un dataset que tiene 100 000 categorías únicas, Hashing(1024) lo mapea a 1024 buckets. Comparar accuracy vs StringLookup con vocabulario truncado.

Homework verificable

Modelo end-to-end sobre California Housing:

1. Pipeline tf.data con CSV.
2. Modelo con Normalization layer adaptada al train set.
3. Entrenar y guardar con model.save('m.keras').
4. Recargar y predecir sobre datos RAW (sin pre-escalar). Verificar que funciona.

Criterio de aceptación: predicción sobre raw data debe dar el mismo resultado que pre-escalar a mano y pasar al modelo "sin Normalization layer".

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Olvido .adapt() y los stats de Normalizati	Normalization no aprende. Fix: .adapt(trai
.adapt con val/test data	Leakage. Fix: solo train.
StringLookup no encuentra una categoría en	OOV (out-of-vocabulary). Por default se ma
TextVectorization muy lento	Suele ser por max_tokens enorme o output_s
Capa preprocessing fuera del modelo y al s	Anti-pattern. Fix: meter las capas DENTRO

Preguntas frecuentes

¿Normalization o StandardScaler de sklearn?

Si vas a deployar el modelo: Normalization de Keras (viaja con el modelo). Para análisis offline: StandardScaler es igual de bueno y más sklearn-idiomático.

¿Cuándo Hashing vs StringLookup?

Hashing para vocabularios gigantes (>100k) o dinámicos (categorías nuevas aparecen continuamente). StringLookup para vocabulario fijo y manejable.

¿Preprocessing en GPU?

Las preprocessing layers corren en CPU por default. Para mover a GPU: with tf.device('/GPU:0'). La normalización es barata; image augmentation puede ser bueno en GPU.

¿TextVectorization reemplaza a Hugging Face tokenizers?

Para tokenización word-level simple, sí. Para BERT/GPT necesitás los tokenizers específicos (subword/BPE) de Hugging Face — clase 127.

¿Y para LLMs?

LLMs llevan su propio tokenizer (BPE, SentencePiece, tiktoken). TextVectorization no aplica.

Referencias

- Géron, cap. 13 — Preprocessing Data with Keras Preprocessing Layers.
- Keras — Preprocessing layers.
- Keras — Working with preprocessing layers.

Siguiente clase

Clase 127 — TensorFlow Datasets (TFDS)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb