Clase 135 — RNNs: neuronas recurrentes, BPTT

Parte: 2 — Deep Learning · Fuente: Géron, cap. 15 § Recurrent Neurons and Layers.
Duración estimada: 70 min.

Clase 135 — RNNs: neuronas recurrentes, BPTT

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 15 § Recurrent Neurons and Layers. Duración estimada: 70 min.*

Objetivo

Entender las redes recurrentes (RNN) — la primera arquitectura para secuencias: misma celda aplicada en cada timestep, estado oculto h_t que acumula contexto, y BPTT (Backpropagation Through Time) que entrena estos modelos. Reconocer sus limitaciones (vanishing en secuencias largas) que motivaron LSTM (clase 120) y eventualmente Transformers.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Explicar la ecuación h_t = tanh(W_h · h_{t-1} + W_x · x_t + b).
- Usar keras.layers.SimpleRNN(units=N, return_sequences=False|True).
- Diferenciar return_sequences=False (un único output al final) vs True (output por timestep).
- Implementar BPTT truncado (longitud máxima de window).
- Reconocer cuándo una RNN simple es suficiente (secuencias cortas, < 20 pasos) y cuándo necesitás LSTM/Transformer.

Temas

- Estado oculto h_t como memoria.
- Unfolding: la RNN como red feedforward muy profunda en el tiempo.
- BPTT: gradientes a través de cada paso temporal.
- Vanishing/exploding en secuencias largas (compounding multiplicativo).
- BPTT truncado.

Definiciones y características

- RNN cell: función (h_{t-1}, x_t) → h_t.
- return_sequences=True: output (batch, T, units) — para apilar otra RNN o aplicar Dense a cada timestep.
- return_state=True: devuelve también el estado final (útil para encoder-decoder).
- BPTT: backprop estándar aplicado al grafo unfolded.
- Truncated BPTT: cortar el gradiente cada K pasos para evitar costo + vanishing.

Dataset / recursos

- Serie temporal sintética (sumas, sine).
- Librerías: tensorflow, keras, numpy, matplotlib.

Ejercicios

1. SimpleRNN básico: predecir el siguiente valor de sin(t). Modelo: SimpleRNN(20) → Dense(1). Entrenar y graficar predicciones.
2. return_sequences=True: apilar 2 RNN, primera con return_sequences=True, segunda sin. Verificar shapes.
3. Predicción de N pasos adelante: iterar prediciendo, alimentando la predicción anterior como nuevo input.
4. BPTT truncado: con secuencia de 200 pasos, comparar BPTT completo vs truncado a 20.
5. Vanishing demo: usar SimpleRNN con secuencias de 100 pasos. Las primeras observaciones casi no influyen en la última predicción.

Homework verificable

Forecasting con SimpleRNN sobre sin(t) + ruido:

1. Generar 5 000 pasos de sin(0.01·t) + N(0, 0.05).
2. Construir samples de longitud 50 → predecir el paso 51.
3. SimpleRNN [20] + Dense(1).
4. Evaluar MAE en test y graficar predicción vs realidad.

Criterio de aceptación: MAE < 0.1; gráfico muestra la predicción siguiendo la sinusoide.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Input shape error en RNN	RNN espera (batch, timesteps, features). F
Olvido return_sequences=True al apilar	Capa intermedia debe devolver toda la secu
Loss explota con secuencia larga	Exploding gradients. Fix: clipnorm=1.0 + L
Predicción autoregresiva diverge	Error acumulado. Fix: usar teacher forcing
mask_zero no soportado en SimpleRNN	Es para Embedding + LSTM/GRU. Fix: usar LS

Preguntas frecuentes

¿RNN vs LSTM vs Transformer en 2026?

Para secuencias cortas y simples: SimpleRNN o LSTM. Para >50 pasos o NLP serio: Transformer. SimpleRNN ya casi no se usa en producción salvo casos muy específicos.

¿return_state para qué?

Para encoder-decoder: el estado final del encoder inicializa el decoder.

¿Bidireccional?

Bidirectional(SimpleRNN(...)) corre la RNN forward y backward, concatenando. Mejor para tareas non-causales (clasificación, NER).

¿Cuántas units?

Empezá con 32-128. Más complejidad de secuencia → más units. RNNs no son tan parameter-hungry como

Transformers.

¿Predecir 1 paso o varios?

Modelos de "1 paso" suelen ser más precisos. Para "varios", entrenar con return_sequences=True y target shifted, o seq2seq (clase 124).

Referencias

- Géron, cap. 15 — Recurrent Neural Networks.
- Pascanu et al. (2013), On the difficulty of training RNNs.
- Keras RNN guide.

Siguiente clase

Clase 136 — Forecasting de series con RNN

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb