Clase 115 — Learning rate scheduling

Parte: 2 — Deep Learning · Fuente: Géron, cap. 11 § Learning Rate Scheduling. Duración estimada: 60 min.

Clase 115 — Learning rate scheduling

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 11 § Learning Rate Scheduling. Duración estimada: 60 min.*

Objetivo

Saber variar el LR durante el entrenamiento —no dejarlo fijo— porque ningún LR es óptimo en todas las fases. Aplicar las 4 estrategias estándar: step decay, exponential decay, cosine annealing (default moderno), y warmup + decay (estándar en Transformers).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Configurar keras.optimizers.schedules.CosineDecay y pasarlo como learning_rate= al optimizer.
- Diferenciar ExponentialDecay, PiecewiseConstantDecay y CosineDecayRestarts.
- Implementar warmup lineal + cosine — receta default en BERT/GPT.
- Usar ReduceLROnPlateau (reactivo) vs schedule (proactivo).
- Graficar la curva de LR a lo largo del entrenamiento para verificar.

Temas

- LR fijo: arrancás bien, terminás demasiado alto para refinar.
- Step decay: cortar LR cada N épocas. Simple, anticuado.
- Exponential decay: lr = lr_0 · γ^t.
- Cosine annealing (Loshchilov & Hutter 2017): lr = 0.5·lr_0·(1 + cos(πt/T)).
- Warmup: empezar bajo y subir linealmente las primeras X steps. Esencial en Transformers.
- One-cycle policy (Smith 2018): warmup + cosine descent + tail decay.

Definiciones y características

- Warmup: subir LR linealmente desde 0 (o muy bajo) hasta el lr_max en los primeros warmup_steps. Estabiliza el inicio cuando los gradientes son ruidosos.
- Cosine annealing: bajar LR siguiendo una media curva coseno desde lr_max hasta lr_min. Suave, sin saltos.
- Restarts (SGDR): cada vez que llega al mínimo, reinicia con lr_max — exploración periódica.
- ReduceLROnPlateau: callback reactivo; baja LR cuando una métrica deja de mejorar.
- OneCycle: variante moderna que sube hasta lr_max en la primera mitad y baja en la segunda, terminando 100× por debajo del inicial.

Dataset / recursos

- Fashion-MNIST con un MLP medio.
- Librerías: tensorflow, keras, matplotlib.

Ejercicios

1. Schedule básica: lr = CosineDecay(initial_learning_rate=1e-3, decay_steps=10_000); Adam(learning_rate=lr). Entrenar y graficar val_loss.
2. Visualizar el LR: para una schedule, evaluarla en steps 0, 100, 1000, 5000, 10000 y graficar.
3. Warmup + Cosine: implementar custom callback (o usar CosineDecay(warmup_steps=...) en Keras 3) y entrenar un Transformer chico (anticipo). Comparar contra sin warmup.
4. ReduceLROnPlateau: alternativa reactiva — ReduceLROnPlateau(factor=0.5, patience=3). Comparar con cosine.
5. One-cycle: implementar con LearningRateScheduler callback. Probar y comparar.

Homework verificable

Sobre Fashion-MNIST:

1. Entrenar 3 modelos con la misma arquitectura: (a) LR fijo 1e-3, (b) ExponentialDecay, (c) CosineDecay(warmup_steps=100).
2. Reportar val_accuracy y graficar las curvas de loss + LR.
3. Concluir qué schedule produjo mejor accuracy.

Criterio de aceptación: cosine con warmup debe ganar o empatar contra fijo. La curva de val_loss del schedule debe ser visualmente más suave hacia el final.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Schedule decae demasiado rápido	decay_steps está mal calibrado vs total st
Combinar schedule + ReduceLROnPlateau	Conflicto: ambos modifican LR. Fix: elegí
LearningRateScheduler callback no funciona	Conflicto. Fix: usar uno u otro.
Sin warmup, transformer diverge en las pri	Gradientes iniciales son ruidosos. Fix: wa
Reiniciar entrenamiento desde checkpoint p	optimizer.iterations cuenta steps; al reca

Preguntas frecuentes

¿Cuál schedule por default?

Cosine con warmup. Es el estándar 2022+ en visión y NLP.

¿Cuántos warmup steps?

5-10 % del total. Para 100 épocas de 500 steps cada una = 50 000 steps → warmup 2 500-5 000.

¿Schedule en steps o en épocas?

Steps casi siempre. Cuanto más granular, más suave.

¿CosineDecayRestarts cuándo?

Cuando entrenás muy largo y querés exploraciones periódicas. Útil en redes muy profundas y datasets grandes; raro en proyectos chicos.

¿Schedule depende del optimizer?

Casi no — Adam(lr=schedule) funciona igual que SGD(lr=schedule). La diferencia es en los valores absolutos.

Referencias

- Géron, cap. 11 — Learning Rate Scheduling.
- Loshchilov & Hutter (2017), SGDR: Stochastic Gradient Descent with Warm Restarts, ICLR.
- Smith (2018), Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates.
- Keras LR schedules.

Siguiente clase

Clase 116 — Regularización: L1/L2, dropout, max-norm, MC dropout (+ Stochastic Depth, DropPath)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb