Clase 107 — Vanishing/exploding gradients

Parte: 2 — Deep Learning · Fuente: Géron, cap. 11 § The Vanishing/Exploding Gradients Problems. Duración estimada: 70 min.

Clase 107 — Vanishing/exploding gradients

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 11 § The Vanishing/Exploding Gradients Problems. Duración estimada: 70 min.*

Objetivo

Entender el problema central que estancó el Deep Learning hasta 2010: cuando un gradiente atraviesa muchas capas, se desvanece (sigmoid/tanh saturadas → multiplicas números < 1 muchas veces → ≈ 0) o explota (pesos grandes → > 1 muchas veces → ∞). Identificar los 4 culpables (activación, inicialización, profundidad, LR) y conocer las soluciones que destrabaron el campo: Glorot/He init (097), ReLU y variantes (098), BatchNorm (099), Gradient clipping (100).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Diagnosticar vanishing gradient: gradients en capas tempranas con norma ~ 1e-8.
- Diagnosticar exploding: loss = nan o gradients con norma ~ 1e+10.
- Inspeccionar gradientes con tf.GradientTape + tf.norm.
- Mapear cada solución al problema: init para arrancar bien, ReLU para no saturar, BN para estabilizar, clipping para evitar explosión.
- Explicar por qué sigmoid en MLPs profundos no escala (derivada máx. 0.25 → gradiente decae rápido).

Temas

- Backprop como producto de derivadas a lo largo de capas.
- Sigmoid: σ(x) · (1 - σ(x)) máxima 0.25 en x=0; satura a 0 en colas.
- Tanh: derivada máx. 1, pero también satura.
- ReLU: derivada 0 o 1 — no decae al multiplicar, pero genera "dying ReLU".
- Inicialización mala: pesos N(0, 1) → activaciones explotan; pesos N(0, 0.01) → vanishing.
- BatchNorm: normaliza dentro del forward pass para que cada capa reciba inputs con varianza controlada.

Definiciones y características

- Vanishing gradient: las derivadas se acercan a 0 al propagar hacia atrás → capas tempranas no aprenden.
- Exploding gradient: las derivadas crecen sin límite → pesos divergen, loss = nan.
- Saturación: una activación está saturada cuando su derivada ≈ 0 (sigmoid en |x| > 5).
- Dying ReLU: una neurona ReLU queda con salida siempre 0 → gradiente 0 → no se actualiza más. Solución: Leaky ReLU, ELU, init He.
- Gradient norm: ||∂L/∂W||. Indicador visualizable durante el entrenamiento.

Dataset / recursos

- Fashion-MNIST.
- Librerías: tensorflow, keras, matplotlib.

Ejercicios

1. Diagnóstico vanishing: entrenar un MLP de 10 capas con sigmoid activations e init default. Inspeccionar gradientes de la primera capa vs la última. Verificar que los de la primera son órdenes de magnitud más chicos.
2. Mismo experimento con ReLU: comparar gradients. Mejor pero aún heterogéneo.
3. Exploding: forzar init RandomNormal(stddev=5). Observar loss = nan en pocas iteraciones.
4. Norma del gradiente por capa: con tf.GradientTape, calcular tf.norm(g) para cada peso y graficar a lo largo del entrenamiento.
5. Solución sencilla: cambiar a He init + ReLU + BatchNorm y mostrar que el problema desaparece (anticipa las siguientes 4 clases).

Homework verificable

Construir un MLP de 20 capas y reportar entrenamiento bajo 4 condiciones:

1. Sigmoid + Glorot init.
2. ReLU + Glorot init.
3. ReLU + He init.
4. ReLU + He init + BatchNorm.

Para cada uno: graficar loss durante 20 épocas + reportar val_accuracy final.

Criterio de aceptación: (1) prácticamente no aprende; (2) aprende lento; (3) aprende razonable; (4) aprende rápido y mejor. La diferencia visual debe ser dramática.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
loss = nan	Exploding. Fix: bajar LR, agregar gradient
loss se queda en valor constante alto	Vanishing. Fix: ReLU + He init.
Accuracy en train baja, val baja	El modelo no aprende. Diagnóstico: imprimi
50 % de las neuronas tienen salida 0 todo	Dying ReLU. Fix: Leaky ReLU o ELU.
Después de unas épocas la loss salta a nan	Exploding tardío. Fix: gradient clipping (

Preguntas frecuentes

¿Por qué sigmoid no funciona en MLPs profundos pero sí en la última capa?

Porque la última capa tiene solo 1 multiplicación; las capas profundas multiplican N veces. $(0.25)^{10} \approx 10^{6} \rightarrow$ desaparece. En la última capa, sigmoid sí sirve para probabilidades.

¿Cuánto es "profundo"?

A partir de 5-10 capas Dense (más con BN). En CNNs, con BN o residuals se llega a 100+. En Transformers, decenas pueden no usar residuals + LayerNorm = vanishing.

¿BatchNorm "soluciona" todo?

Resuelve el vanishing/exploding casi completamente al normalizar las activaciones. Pero tiene problemas propios (batch chico, dependencia entre muestras) — clase 099 entra en detalle.

¿GELU/Swish ayudan?

Sí. Son activaciones más suaves que ReLU (sin "muerte"), usadas en transformers modernos (BERT, GPT). Clase 098.

¿Esto sigue siendo relevante con LLMs?

Sí — los LLMs usan combinación de LayerNorm + residual + GELU + init cuidadoso (Xavier/Glorot custom) precisamente para esto. Sin estas piezas, no se podrían entrenar.

Referencias

- Géron, cap. 11 — Training Deep Neural Networks.
- Hochreiter (1991), thesis — primera descripción del vanishing gradient.
- Bengio, Simard & Frasconi (1994), Learning Long-Term Dependencies with Gradient Descent is Difficult.
- Glorot & Bengio (2010), Understanding the difficulty of training deep feedforward neural networks, AISTATS.

Siguiente clase

Clase 108 — Inicialización (Glorot, He)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb