Funciones de Pérdida en Redes Neuronales

Cesar Garcia

2025

Introducción

Objetivos de la sesión

- Comprender qué significa entrenar una red neuronal.
- Entender el rol de las funciones de pérdida.
- Explicar el concepto de descenso de gradiente de forma intuitiva.
- Introducir épocas, iteraciones y procesamiento por lotes (batches).

¿Qué significa entrenar una red?

Intuición general

Entrenar una red neuronal significa **ajustar los pesos** para mejorar las predicciones.

Proceso

1. La red hace una predicción con pesos iniciales aleatorios.
2. Se compara la predicción con el valor real.
3. Se calcula un error (pérdida).
4. Se ajustan los pesos para reducir ese error.

Funciones de Pérdida

¿Qué es una función de pérdida?

- En el entrenamiento de redes neuronales, una *función de pérdida* mide qué tan mal está prediciendo el modelo.
- El objetivo del entrenamiento es **minimizar esta pérdida** usando descenso de gradiente.
- Diferentes tareas requieren diferentes funciones de pérdida.

Funciones de pérdida para Regresión

Error Cuadrático Medio — MSE

Definición:

Mide la distancia cuadrática entre predicciones y valores reales.

Penaliza fuertemente errores grandes.

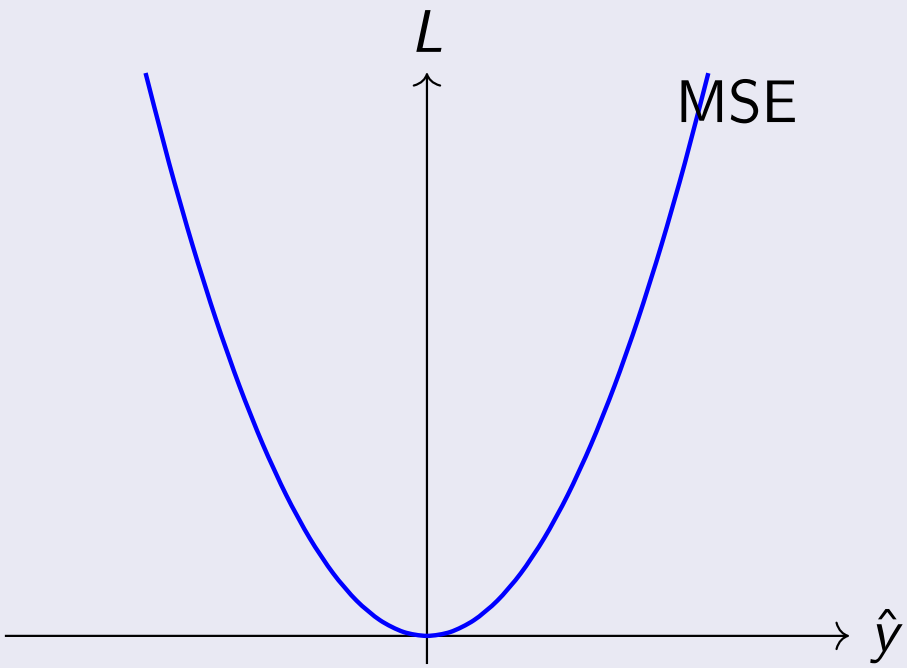
Fórmula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Intuición: - La curva es suave y diferenciable.

- Los errores grandes pesan más $\rightarrow$ puede ser sensible a outliers.

Visualizacion



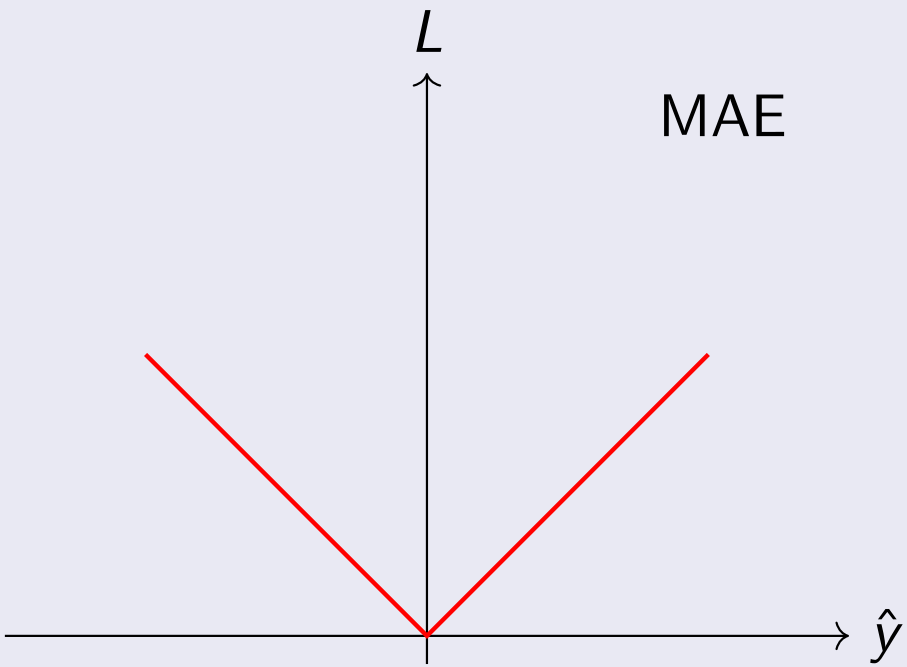
Error Absoluto Medio — MAE

Definición:
Suma las diferencias absolutas entre predicción y realidad.
Fórmula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Intuición: - Penaliza todos los errores de manera proporcional.
- Más robusto frente a *outliers*.
- No es diferenciable en 0 (pero frameworks lo manejan bien).

Visualizacion



Funciones de pérdida para Clasificación

Entropía Cruzada — Binary Cross Entropy (BCE)

Usada en:

- Clasificación binaria
- Redes con una salida sigmoide

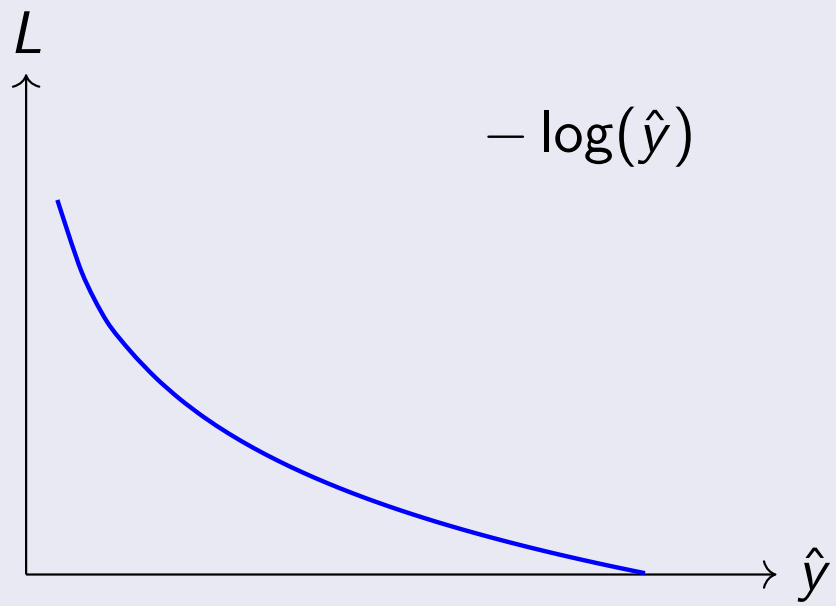
Fórmula:

$$\text{BCE} = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

Intuición: - Si la predicción es incorrecta y con mucha confianza, la pérdida es *muy grande*.

- Esto acelera el aprendizaje.
- Es equivalente a maximizar la verosimilitud de una distribución Bernoulli.

Visualizacion: (pérdida para y = 1)



Entropía Cruzada Categórica — CCE

Para:

- Problemas multiclase (Softmax)

Fórmula:

$$\text{CCE} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$$

Donde:

- y_c es 1 si la clase verdadera es c, 0 en otro caso (one-hot encoding).
- $\hat{y}_c$ es la probabilidad softmax de la clase c.

Intuición: - Obliga al modelo a asignar alta probabilidad a la clase correcta.

- Muy usada en visión por computadora y NLP.

Descenso de Gradiente (Gradient Descent)

Intuición

- Imagina una montaña donde el objetivo es llegar al punto más bajo.
- El gradiente indica en qué dirección sube más rápido la montaña.
- Para minimizar el error, tomamos pasos hacia la dirección opuesta.

Idea central

Actualizar los pesos con pequeños pasos que reduzcan la pérdida.
gradient descent

Importancia de las derivadas en redes neuronales

Idea clave

Las derivadas determinan **cómo cambia la salida** de la red cuando cambiamos ligeramente los pesos.

Sin derivadas no podríamos ajustar los parámetros $\rightarrow$ la red no podría aprender.

En cada capa

Dado: - $z = Wx + b$ - $a = f(z)$

Las derivadas que necesitamos son:

$$\frac{\partial a}{\partial z} = f'(z) \qquad \text{(derivada de la activación)}$$

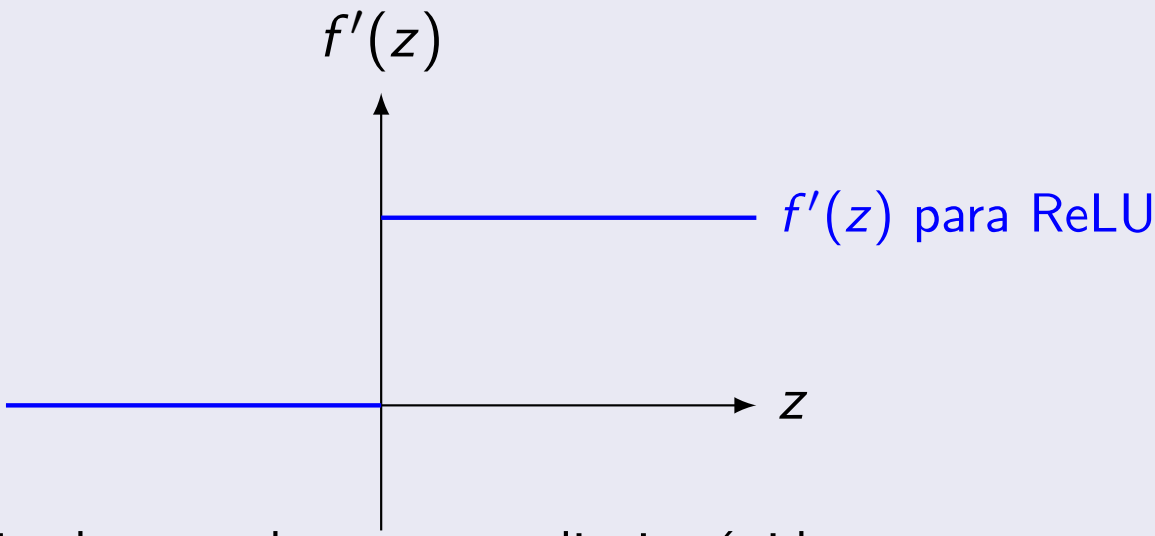
$$\frac{\partial z}{\partial W} = x \qquad \text{(cómo afecta el peso al valor interno)}$$

Estas derivadas permiten calcular el gradiente del error respecto a cada parámetro.

¿Por qué es crítico?

- Las activaciones determinan **cómo se propaga el gradiente** hacia atrás.
- Funciones como Sigmoid y Tanh pueden causar **gradientes muy pequeños** ("vanishing gradient").
- ReLU, GELU, Swish y ELU mantienen gradientes útiles en la mayor parte del dominio.
- Sin derivadas bien comportadas, el aprendizaje se vuelve lento o imposible.

Visualización



Derivadas grandes $\rightarrow$ aprendizaje rápido.

Derivadas cercanas a 0 $\rightarrow$ gradientes que desaparecen.

Variantes del Descenso de Gradiente

Batch Gradient Descent

Usa **todos los datos** en cada actualización.

Stochastic Gradient Descent (SGD)

Usa **un solo dato** por actualización.

Mini-Batch Gradient Descent

Usa pequeños grupos de datos:
- Más estable que SGD
- Más rápido que batch completo
- El método más utilizado

Learning Rate

Concepto

El learning rate define el tamaño del paso durante el descenso de gradiente.

Comportamiento

- Muy grande $\rightarrow$ el modelo salta el mínimo.
- Muy pequeño $\rightarrow$ entrenamiento lento.

Épocas, Iteraciones y Batches

Época

Una pasada completa por todos los datos.

Batch

Un subconjunto de los datos.

Iteración

Una actualización de pesos por cada batch.

Importancia

- Controlan la estabilidad y velocidad del aprendizaje.
- Permiten generalizar mejor sin sobreajustar.

Procesamiento por Lotes (Batch Processing)

Beneficios

- Estabiliza el cálculo del gradiente.
- Reduce ruido en la actualización.
- Optimiza el uso del hardware (GPU/CPU).

Actividad de Cierre

Comparar tres escenarios: 1. Batch completo
2. Mini-batch
3. SGD

Explicar cuál sería más estable, más ruidoso y más rápido.

Notas del Presentador

- Usar la analogía de la montaña para descenso de gradiente.
- Explicar por qué el batch afecta la estabilidad.
- Evitar matemáticas y enfocarse en intuición visual.
- Conectar esta sesión con la siguiente sobre generalización.