

Funciones de Pérdida en Redes Neuronales

Cesar Garcia

2025

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.

¿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.

Error Cuadrático Medio — MSE

Definición:

Mide la distancia cuadrática entre predicciones y valores reales.
Penaliza fuertemente errores grandes.

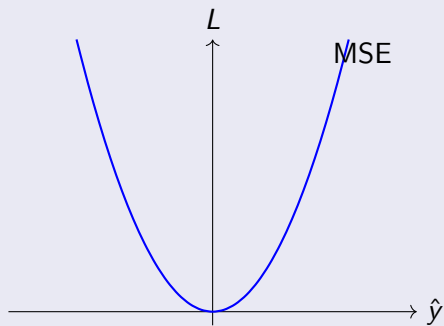
Fórmula:

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

Intuición: - La curva es suave y diferenciable.

- Los errores grandes pesan más → puede ser sensible a outliers.

Visualizacion



Error Absoluto Medio — MAE

Definición:

Suma las diferencias absolutas entre predicción y realidad.

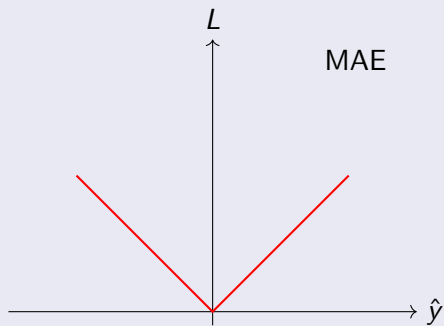
Fórmula:

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

Intuición: - Penaliza todos los errores de manera proporcional.

- Más robusto frente a *outliers*.
- No es diferenciable en 0 (pero frameworks lo manejan bien).

Visualización



Entropía Cruzada — Binary Cross Entropy (BCE)

Usada en:

- Clasificación binaria
- Redes con una salida sigmoide

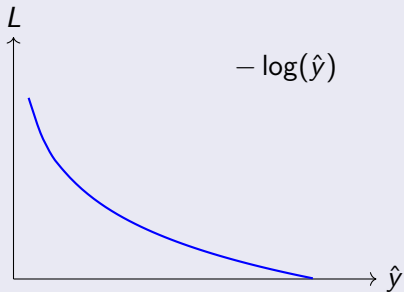
Fórmula:

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

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.

Visualización

- gradient descent

Idea clave

Las derivadas nos indican **qué tan sensible es el error del modelo** a pequeños cambios en sus parámetros (pesos y sesgos).

En otras palabras, responden a la pregunta:

“Si cambio un peso un poco, ¿el error mejora o empeora, y cuánto?”

Sin derivadas no hay forma sistemática de saber **cómo ajustar los pesos**, por lo tanto **no hay aprendizaje**.

¿Dónde aparecen las derivadas?

En cada neurona ocurren dos pasos fundamentales:

① Combinación lineal

$$z = Wx + b$$

② Función de activación

$$a = f(z)$$

Para poder entrenar la red necesitamos saber:

- Cómo cambia la activación cuando cambia z
- Cómo cambia z cuando cambian los pesos

Esto se expresa mediante derivadas locales:

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

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

Estas derivadas **no optimizan nada por sí solas**, pero son los bloques básicos que permiten calcular el gradiente del error.

Derivadas y aprendizaje

El entrenamiento de una red neuronal funciona porque:

- La función de pérdida mide el error.
- Las derivadas indican **en qué dirección cambia ese error**.
- El descenso de gradiente ajusta los pesos en la dirección que **reduce la pérdida**.

Conceptualmente:

Derivadas grandes → *el aprendizaje avanza rápido*

Derivadas cercanas a cero → *el aprendizaje se vuelve lento o se detiene*

Importancia de las funciones de activación

Las funciones de activación no solo introducen no linealidad, también determinan **cómo se propaga el gradiente hacia atrás**.

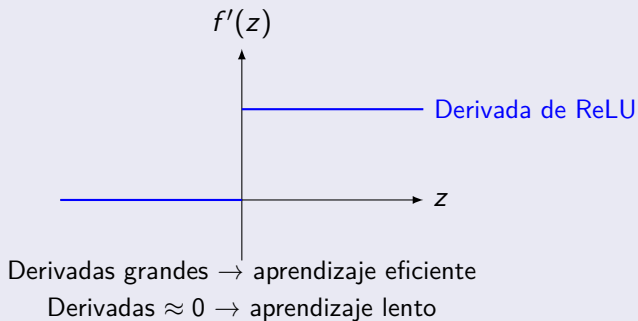
- **Sigmoid y Tanh**

- Se “aplanan” para valores grandes o pequeños.
- Sus derivadas se vuelven muy pequeñas.
- Pueden causar el problema de **vanishing gradient**.

- **ReLU, GELU, ELU, Swish**

- Mantienen derivadas útiles en gran parte del dominio.
- Permiten entrenar redes profundas de forma estable.
- Son preferidas en arquitecturas modernas.

Visualización conceptual



Qué **no** necesitamos calcular a mano

- No calculamos derivadas completas de la función de pérdida a mano.
- Frameworks modernos (TensorFlow, PyTorch) lo hacen automáticamente.
- Lo importante es entender:
 - **Qué papel juegan las derivadas**
 - **Por qué ciertas activaciones funcionan mejor que otras**

Más adelante veremos cómo estas ideas se organizan en el algoritmo de **propagación hacia atrás (backpropagation)**.

Mensaje final

Las derivadas son el **mecanismo interno del aprendizaje** en redes neuronales.

No son un fin matemático en sí mismas, sino la herramienta que permite:
convertir el error en información útil para mejorar el modelo.

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).