Proyecto final

Métodos numéricos II

OPTIMIZACIÓN EN REDES NEURONALES

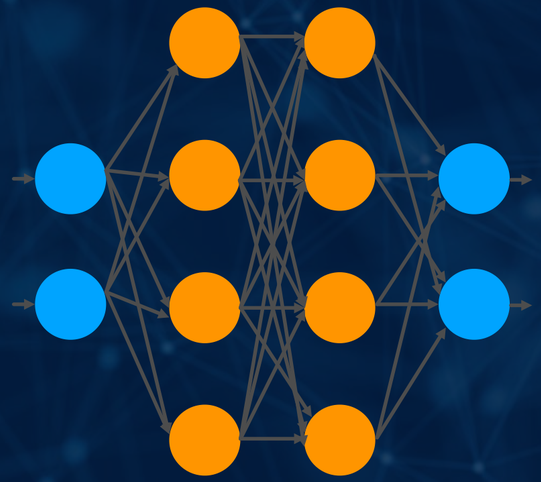
Lourdes Saavedra
Gabriela Archila

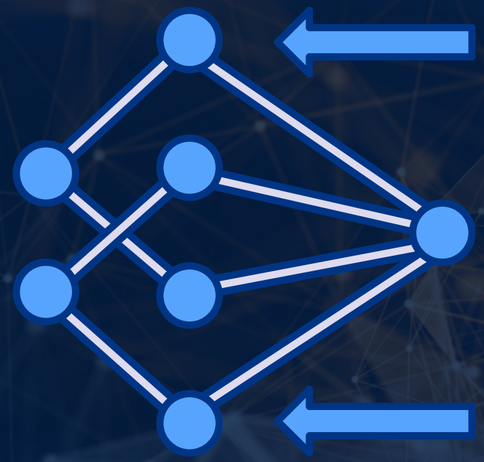
CONTENIDOS

MOTIVACIÓN

Redes neuronales

- Modelos computacionales.
- Reconocer patrones
- Resuelve problemas complejos
- Inteligencia artificial y aprendizaje automatico.



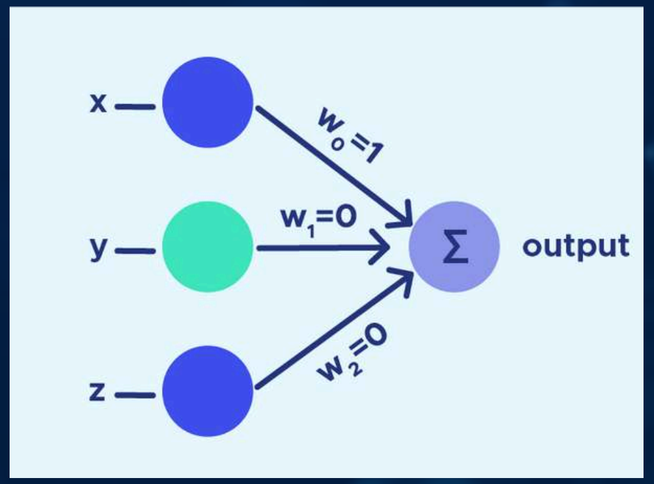


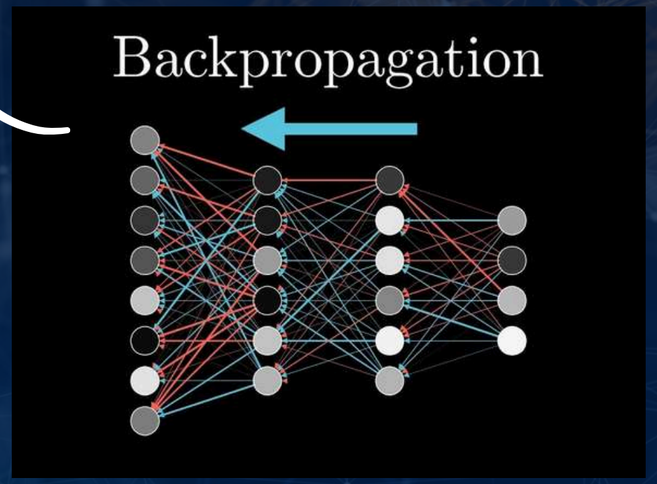
Importancia

- Optimizar la red neuronal.
- ***Backpropagation.***
  - Aprende de los errores.
  - Ajusta pesos.
  - Minimiza el error entre predicción y resultado real.

UN POCO DE HISTORIA....

Rumelhart y Hinton ganadores del **nobel de física** por incoporar ideas de potenciales y movimiento de partículas en la ciencia de la computación.





PERCEPTRÓN

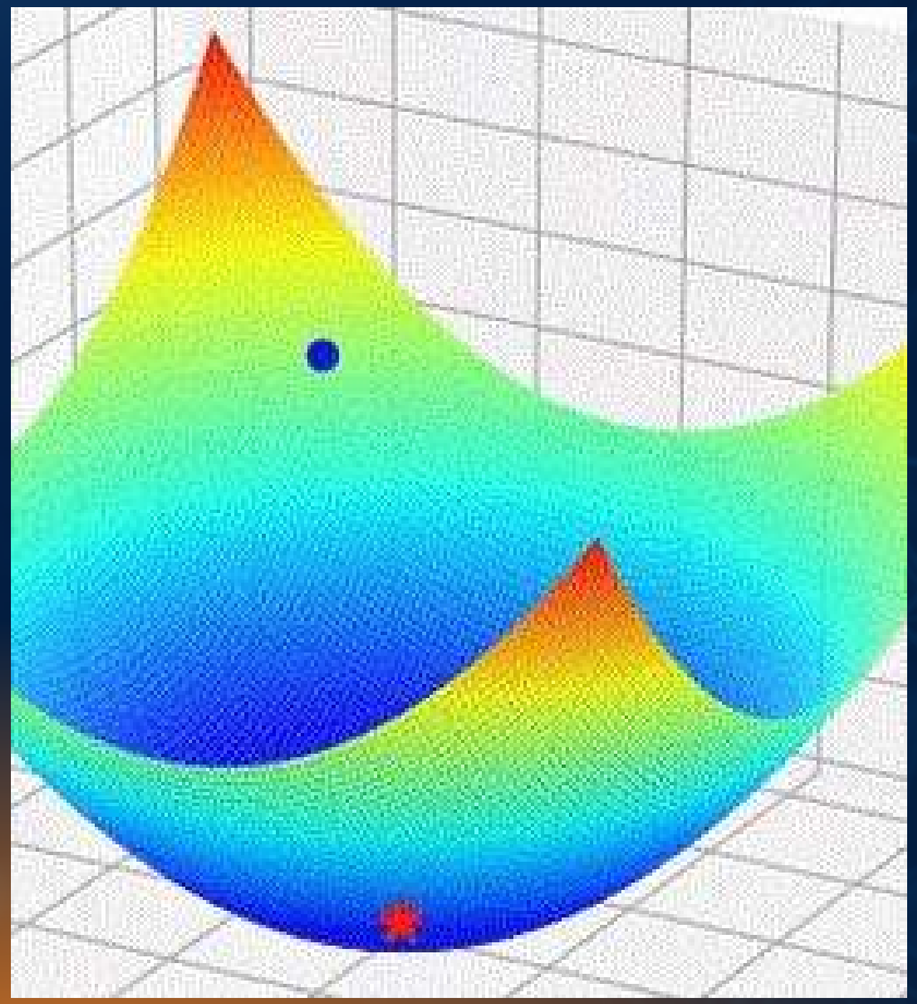
- **Modelo simple** de dos capas.
- En los años 50.

INVIERNO DE LA AI

- Marvin Minsky y Seymour Papert. (1969).
- Perceptrón únicamente podían realizar las **funciones más básicas.**

BACKPROPAGATION

- David Rumelhart, Geoffrey Hinton y Ronald Williams (1986).
- "*Learning representations by back-propagating errors*",.
- **Entrenar la red neuronal profunda**.

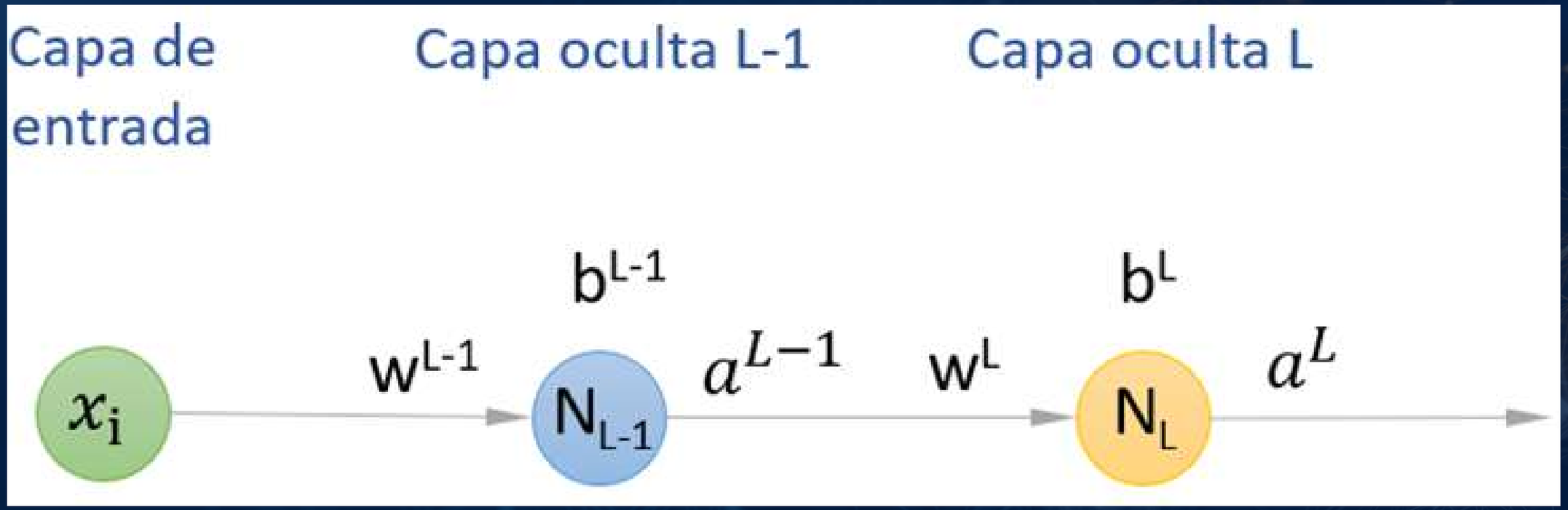


FUNCIÓN DE PÉRDIDA

Indica qué tan **lejos** está la salida de la red neuronal del resultado deseado

Son **más que métricas de evaluación**; sirven como entrada para los algoritmos de optimización, ayudando a **ajustar los parámetros para minimizar la pérdida**

BACKPROPAGATION:
FUNCIONAMIENTO



APRENDIZAJE DE UNA RED NEUROAL

Algoritmo *backpropagation*: se basa en el principio de la optimización por **descenso de gradiente**, ya que dichos gradientes indican la dirección y la magnitud en la que deben llevarse a cabo los **ajustes para minimizar los errores.**

Representación **simplificada** de una red neuronal.

- 1 capa de entrada, 2 capas ocultas y la propagación hacia adelante de los datos.
- Neuronas, pesos, sesgos y el valor devuelto por la neurona al aplicarle una función de activación

LA MATEMÁTICA POR DETRÁS

1

Derivada parcial de la función de coste con respecto los pesos y sesgos

$$\frac{\partial C}{\partial w^L}, \frac{\partial C}{\partial b^L}$$

donde C es la función de coste.

2

Para la i-ésima derivada

$$\frac{\partial C_i}{\partial w^L}, \frac{\partial C_i}{\partial b^L}$$

Suponga que la función de coste viene dada por el error cuadrático medio

$$c_i = (y_i - p_i)^2$$

donde y_i es el valor real y p_i el valor predicho.

3

Recordar que,

$$p_i = \sigma(w^L a^{L-1} + b^L)$$

donde σ es la función de activación. Ahora, llamemos

$$z^L = w^L a^{L-1} + b^L$$

4

Entonces al reemplazarlo en lo anterior,

$$p_i = \sigma(z^L) \rightarrow c_i = (y - \sigma(z^L))^2$$

5

Por lo tanto, por regla de la cadena se tiene que,

$$\frac{\partial C_i}{\partial w^L} = \frac{\partial C_i}{\partial \sigma} \frac{\partial \sigma}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

LA MATEMÁTICA POR DETRÁS

5

$$\frac{\partial C_i}{\partial w^L} = \frac{\partial C_i}{\partial \sigma}\frac{\partial \sigma}{\partial z^L}\frac{\partial z^L}{\partial w^L}$$

¿Qué nos dice?

- $\frac{\partial C_i}{\partial \sigma}$ para la función de pérdida escogida es $2(y-\sigma)$.
- $\frac{\partial \sigma}{\partial z^L}$ es la derivada de la función de activación.
- $\frac{\partial z^L}{\partial w^L}$ es a^{L-1}

6

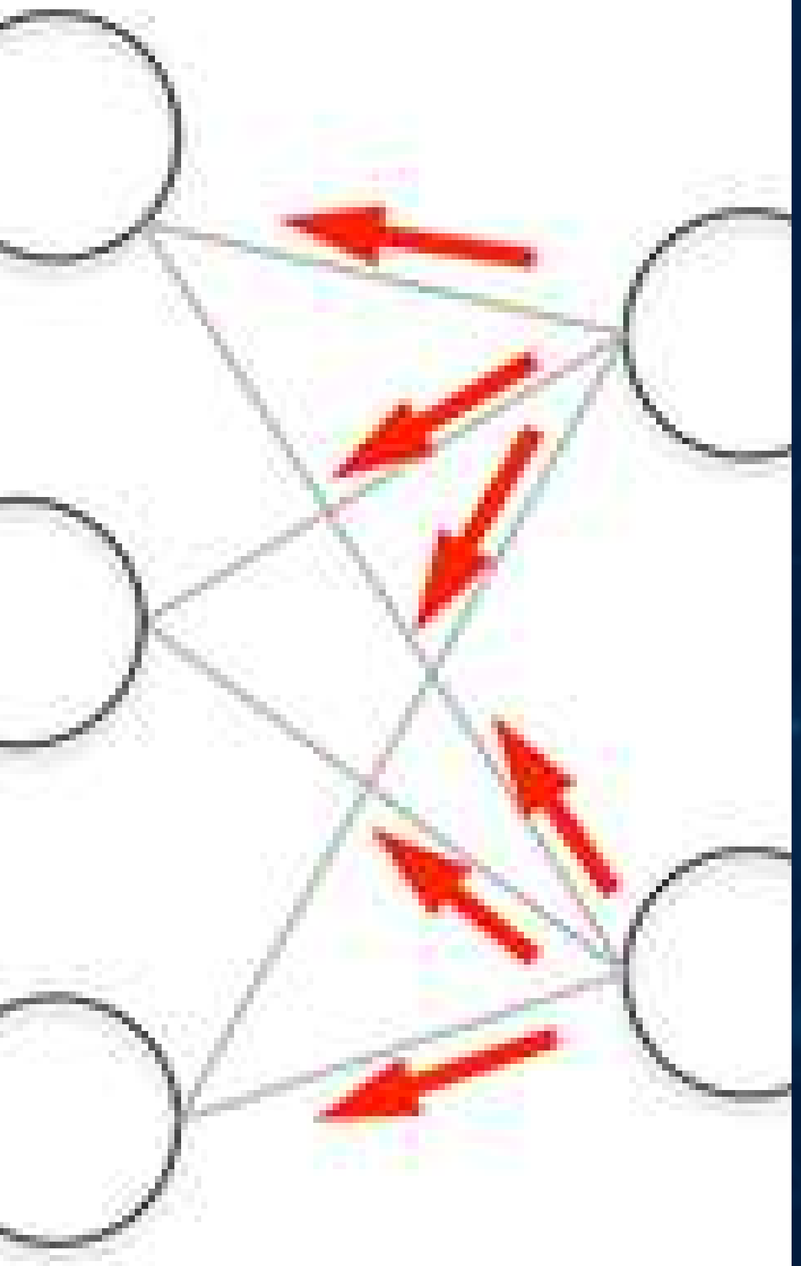
Se aplica lo mismo para el sesgo,

$$\frac{\partial C_i}{\partial b^L} = \frac{\partial C_i}{\partial \sigma}\frac{\partial \sigma}{\partial z^L}\frac{\partial z^L}{\partial b^L}$$

donde $\frac{\partial z^L}{\partial b^L}$ es 1.

Resumen

Aplicando la **regla de la cadena**, podemos ir desde **atrás hacia delante** (desde la función de coste expresada en función de los valores devueltos por la capa de salida hacia las primeras capas) calculando las derivadas parciales de la función de coste con respecto a todos los pesos y los sesgos.



ETAPAS

Backpropagation se divide en dos fases principales: **propagación hacia adelante** y **propagación hacia atrás**.

1

Propagación hacia adelante

- Se introducen los datos de prueba en la red neuronal.
- En cada neurona se aplica la función de activación.
- Genera una señal que se transmite

2

Cálculo del error

- Con la salida de la red neuronal, se compara con la deseada mediante una función de pérdida.

3

Retropropagación del error

- Se calculan los gradientes del error respecto a cada peso y sesgo en la red neuronal.
- El proceso es desde la salida a la entrada

4

Actualización de parámetros

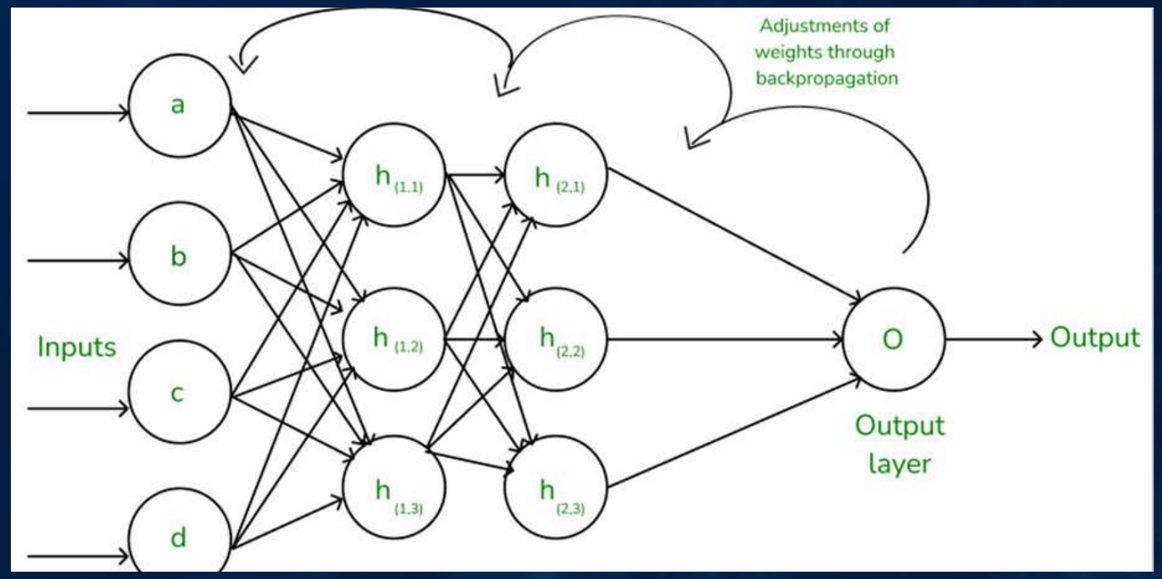
- Se actualizan los pesos y sesgos.
- Hay ajustes en la red para minimizar los errores en la predicción de manera gradual con cada iteración del entrenamiento.

5

Modelado de la predicción

- Tras todas las optimizaciones, el método de cálculo vuelve a revisar las entradas de prueba.

VISUALIZACIÓN



Backpropagation

USOS

- **Reconocimiento de patrones.**
- **Procesamiento de lenguaje natural.**
- **Sistemas de recomendación.**
- **Predicción y análisis de datos**
- **Control y robótica.**

DIFERENCIAS ENTRE

DESCENSO GRADIENTE CLÁSICO (DG) Y DECESO GRADIENTE ESTOCÁSTICO (SGD)

DIFERENCIAS

Descenso Gradiente (GD)

- **Calcula el gradiente** de la función de pérdida utilizando **todo** el conjunto de **datos de entrenamiento.**
- Actualiza los **parámetros** del modelo dando **pasos proporcionales al negativo del gradiente** de todo el conjunto de datos.
- Garantiza la **convergencia al mínimo** (si se cumplen ciertas condiciones).
- **Costoso computacionalmente**.

Descenso Gradiente Estocástico (SGD)

- Actualiza los **parámetros** del modelo utilizando el **gradiente de la función de pérdida para cada ejemplo de entrenamiento individual.**
- Realiza **actualizaciones frecuentes** basadas en **lotes** únicos o pequeños de ejemplos de **entrenamiento.**
- **Más rápido** que el descenso de gradiente para **grandes conjuntos de datos.**
- Por sus actualizaciones ruidosas, tiene **más fluctuaciones** y **no converge necesariamente al mínimo absoluto.**

MOMENTUM Y RMSPROP

MOMENTUM

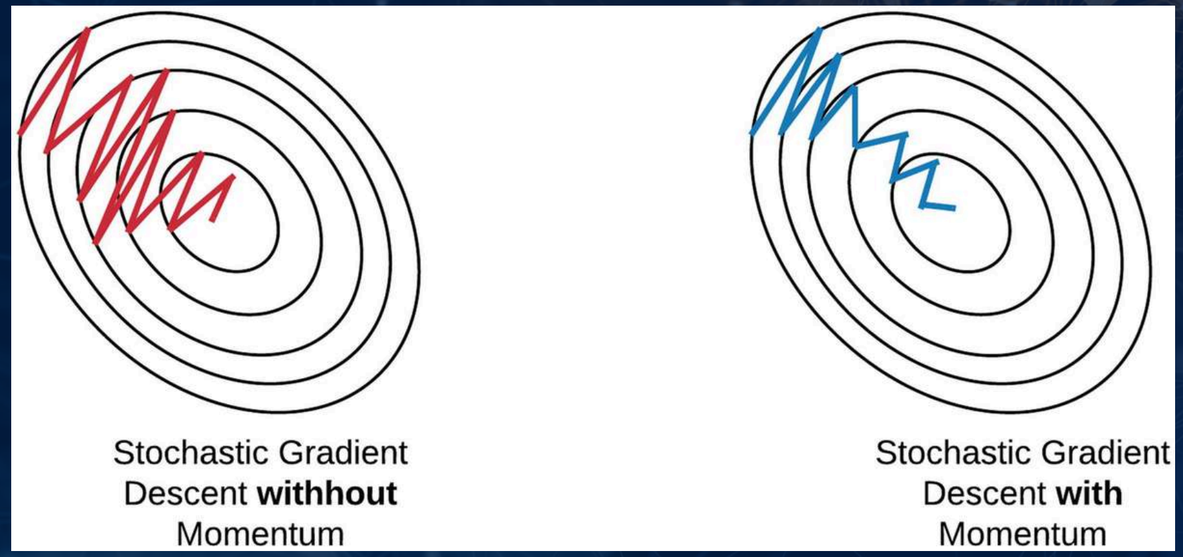
- **Acelera la convergencia** y ayuda a **superar mínimos locales.**

- Considera el **gradiente de la iteración anterior** para **actualizar los pesos** con el gradiente actual.

- Similar a una esfera descendiendo por una rampa; el momentum permite avanzar más rápido en pendientes pronunciadas y suaviza el avance en zonas planas.

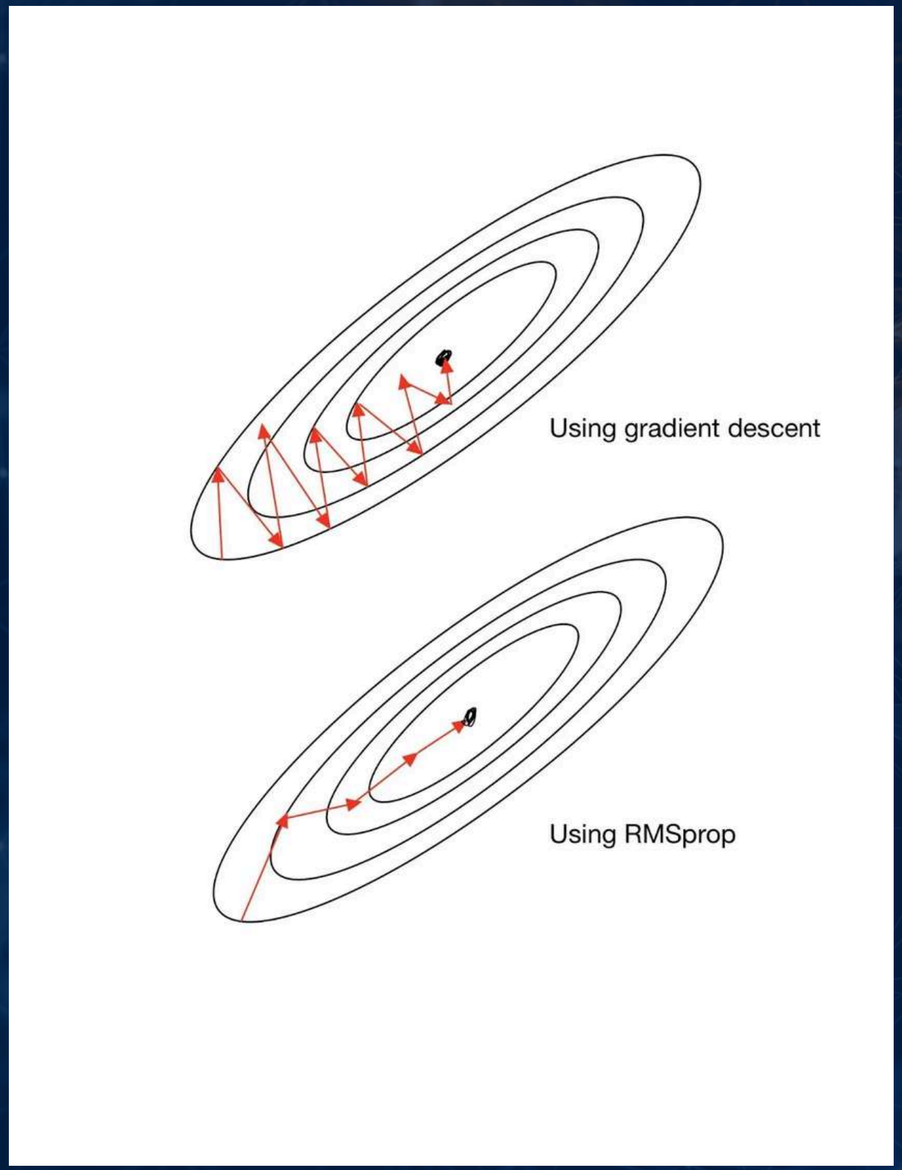
- Esfera: learning rate - pendiente: gradiente almacenado

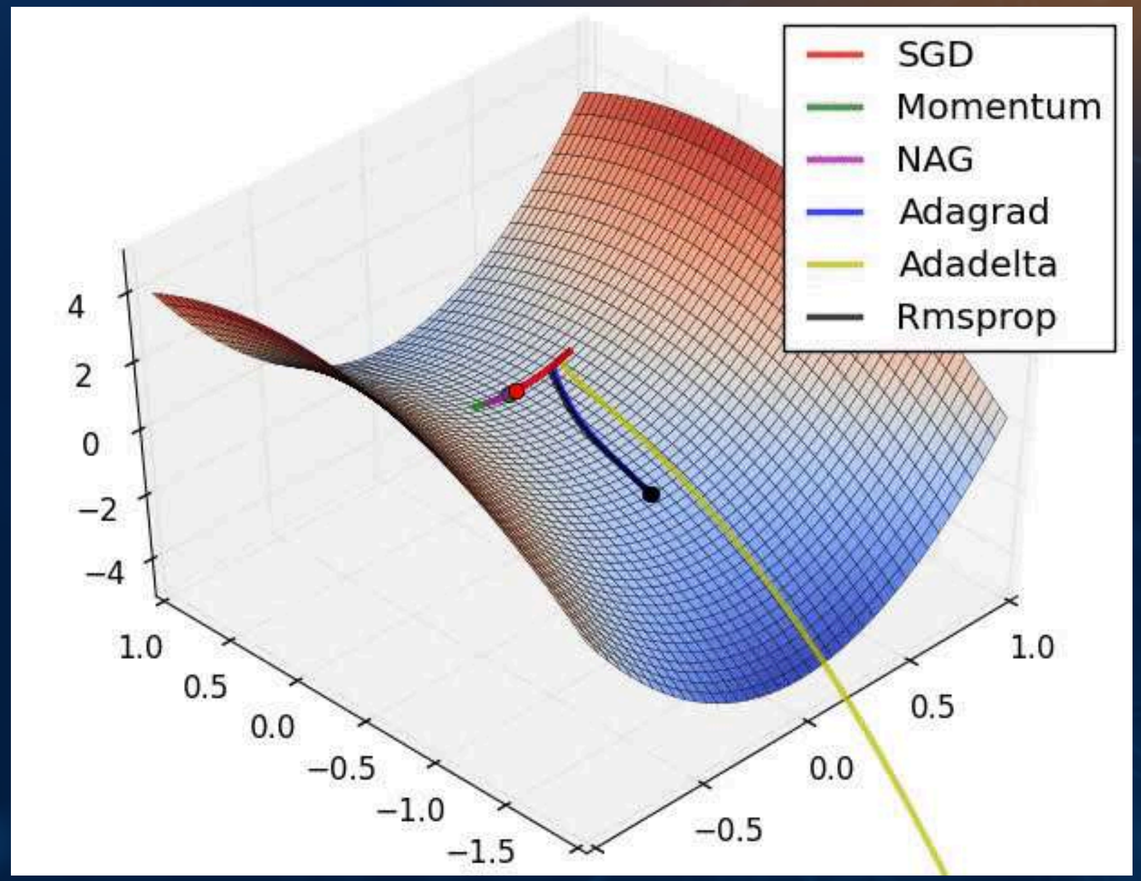
- Cuando el gradiente anterior sea grande, las actualizaciones de los pesos serán mayores y por lo tanto el gradiente nuevo modificará la actualización de los pesos con la información nueva, pero sin perder la dirección del gradiente anterior.

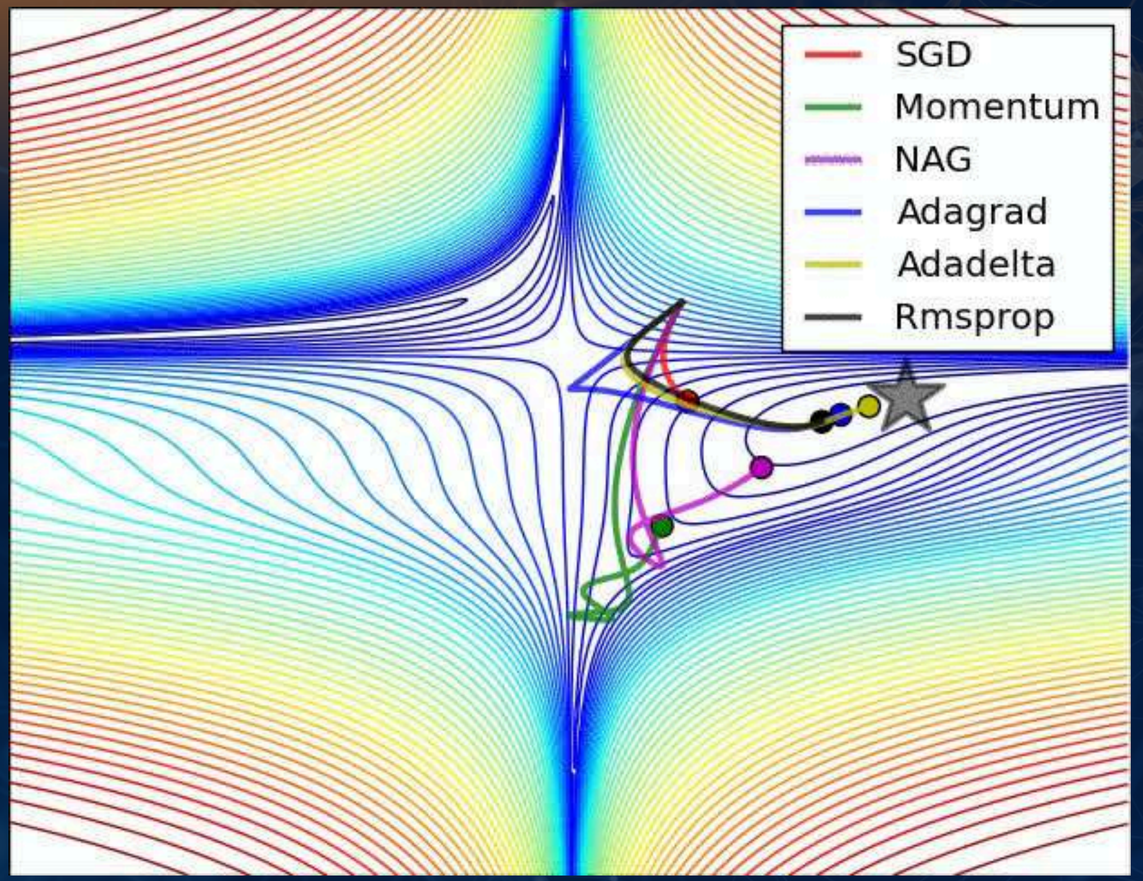


RMSprop

- **Ajusta la tasa de aprendizaje** en cada iteración
- Calcula el **promedio exponencial de los cuadrados** de los gradientes para **reducir el tamaño de los pasos** cuando los gradientes son grandes y **permitir pasos más amplios** cuando son pequeños.
- **Suaviza** los gradientes y **evita oscilaciones brusca**
- Se **adapta** bien en problemas con gradientes de distinta escala




SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop

IMPLEMENTACIÓN EN PYTHON

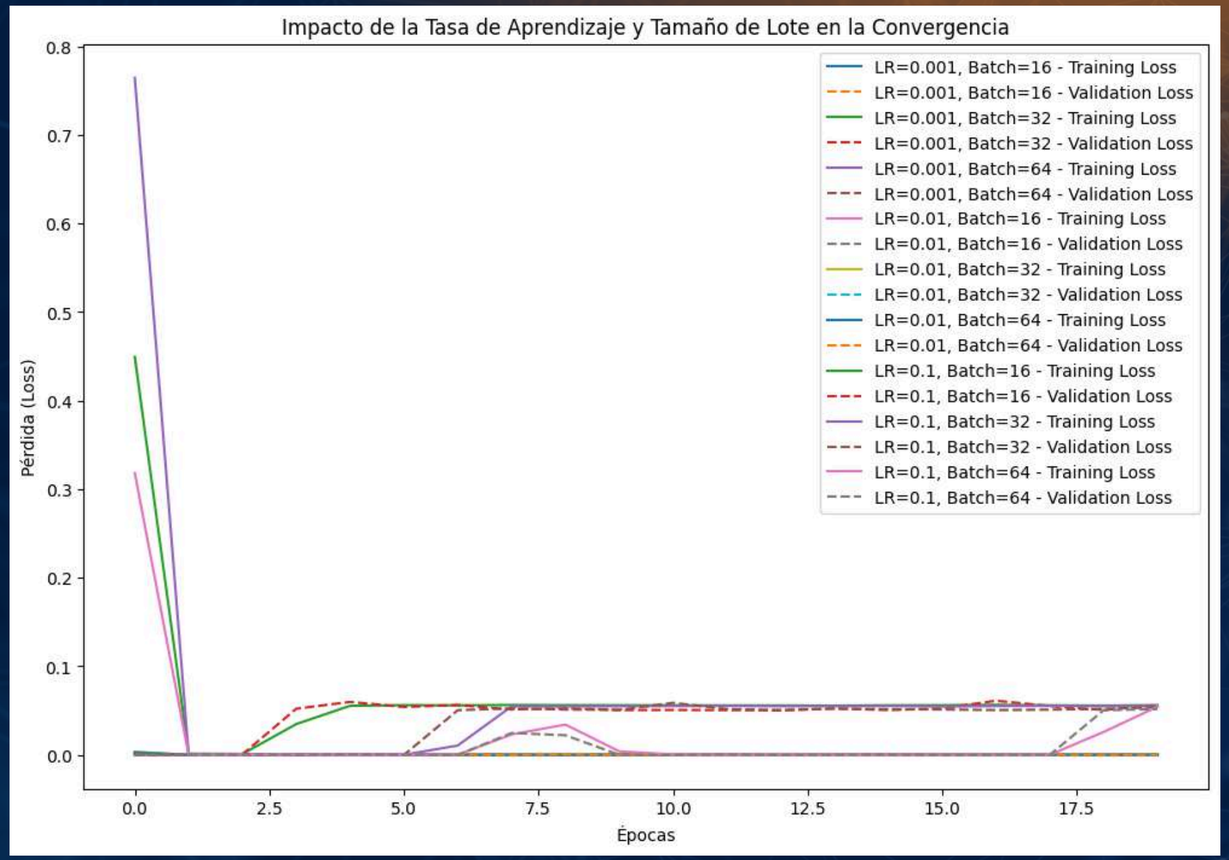
HIPERPARÁMETROS

- Ajustar la tasa de aprendizaje, el tamaño de lote y el número de iteraciones **minimiza la función de pérdida**.

- **Equilibrio entre Sesgo y Varianza**:
  - Sesgo: Divergencia entre predicciones y realidad.
  - Varianza: Sensibilidad del modelo a nuevos datos.

- El ajuste adecuado encuentra un **balance** para obtener **predicciones precisas y consistentes**.

- Permite **optimizar el rendimiento y uso de recursos** durante el entrenamiento del modelo.



REFERENCIAS

Gavilan, I. (s. f.). Catálogo de componentes de redes neuronales III: Funciones de pérdida. Recuperado de https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-iii-funciones-de-perdida/#:~:text=%C2%BFQu%C3%A9%20es%20una%20funci%C3%B3n%20de,es%20el%20correcto%20o%20deseado.

IBM. (s. f.). Loss function. IBM. Recuperado de https://www.ibm.com/es-es/think/topics/loss-function

IBM. (s. f.). Gradient descent. IBM. Recuperado de https://www.ibm.com/mx-es/topics/gradient-descent

Rohan, A. (2019, 11 junio). Batch, mini-batch & stochastic gradient descent. Towards Data Science. Recuperado de https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

Campos Soberanis, M. (2020, 4 junio). Introducción al deep learning III: Métodos de optimización II. Medium. Recuperado de https://medium.com/@mario.campos.soberanis/introducci%C3%B3n-al-deep-learning-iii-m%C3%A9todos-de-optimizaci%C3%B3n-ii-2458ede34bd4

Gupta, D. (2020, 8 junio). Momentum: A simple yet efficient optimizing technique. Analytics Vidhya. Recuperado de https://medium.com/analytics-vidhya/momentum-a-simple-yet-efficient-optimizing-technique-ef76834e4423

Naukri. (s. f.). Nesterov accelerated gradient. Naukri. Recuperado de https://www.naukri.com/code360/library/nesterov-accelerated-gradient

Saha, A. (2018, 2 octubre). Understanding RMSprop: Faster neural network learning. Towards Data Science. Recuperado de https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

Brownlee, J. (2020, 29 abril). Adam optimization algorithm for deep learning. Machine Learning Mastery. Recuperado de https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

IBM. (s. f.). Hyperparameter tuning. IBM. Recuperado de https://www.ibm.com/es-es/think/topics/hyperparameter-tuning

Por su atención

MUCHAS GRACIAS