3ª Emisión

Módulo 8

Algoritmo Backpropagation

Morales Palafox Edgar

DGTIC
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC
Educación Continua
1971 - 2021

Objetivo

El participante identificará cómo se entrenan las redes neuronales, usando descenso por gradiente o variantes y el algoritmo de retropropagación de errores.

Contenido

- Algoritmo de descenso por gradiente y variantes
- Cálculo de los gradientes en redes neuronales con múltiples capas
- Algoritmo de retropropagación de errores
- Diferenciación automática

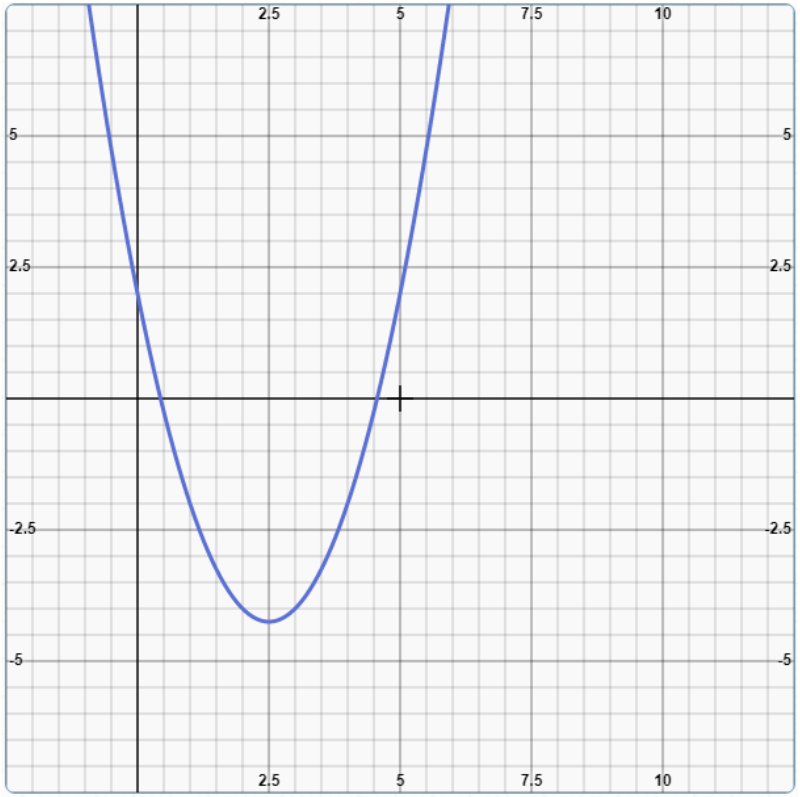
Algoritmo de descenso por gradiente y variantes

Derivada de una función matemática

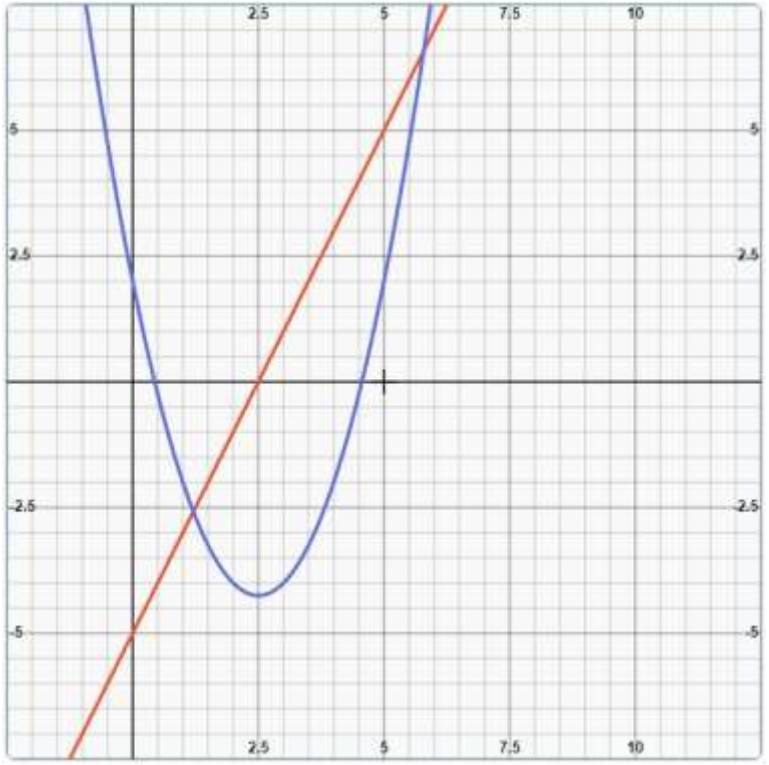
Es la razón o velocidad de cambio de una función en un determinado punto. Es decir, qué tan rápido se está produciendo una variación. Desde una perspectiva geométrica, la **derivada** de una función es la pendiente de la recta tangente al punto donde se ubica x.

Supongamos que se tiene la siguiente función:

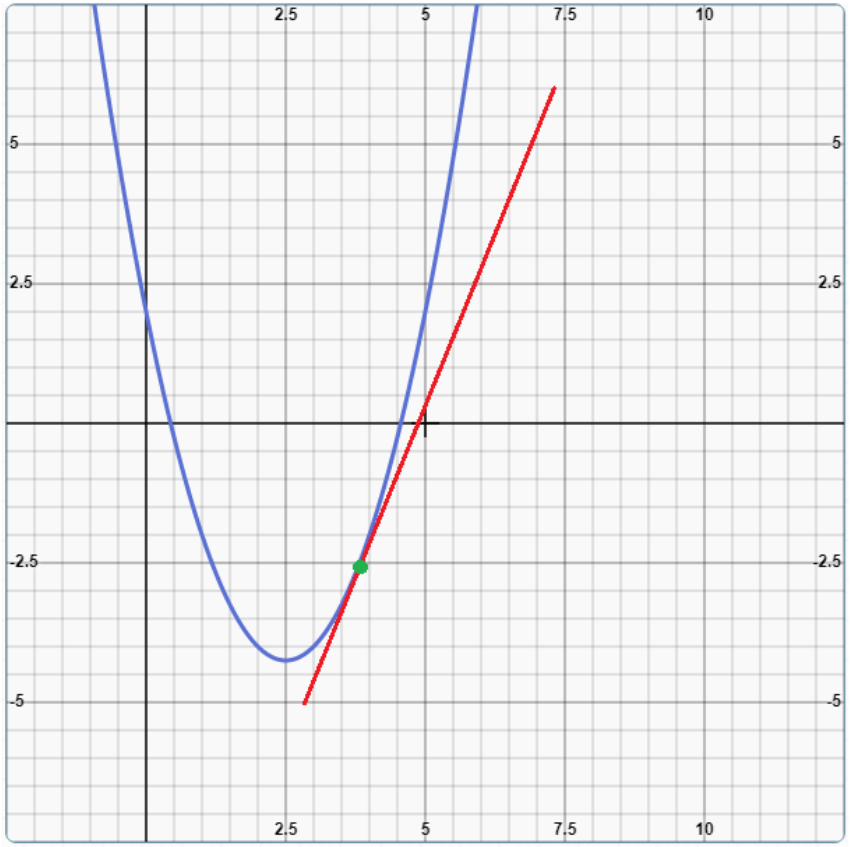
$F(x)= X^2-5x+2$



La derivada de la función, es:
F’(x)=2x-5

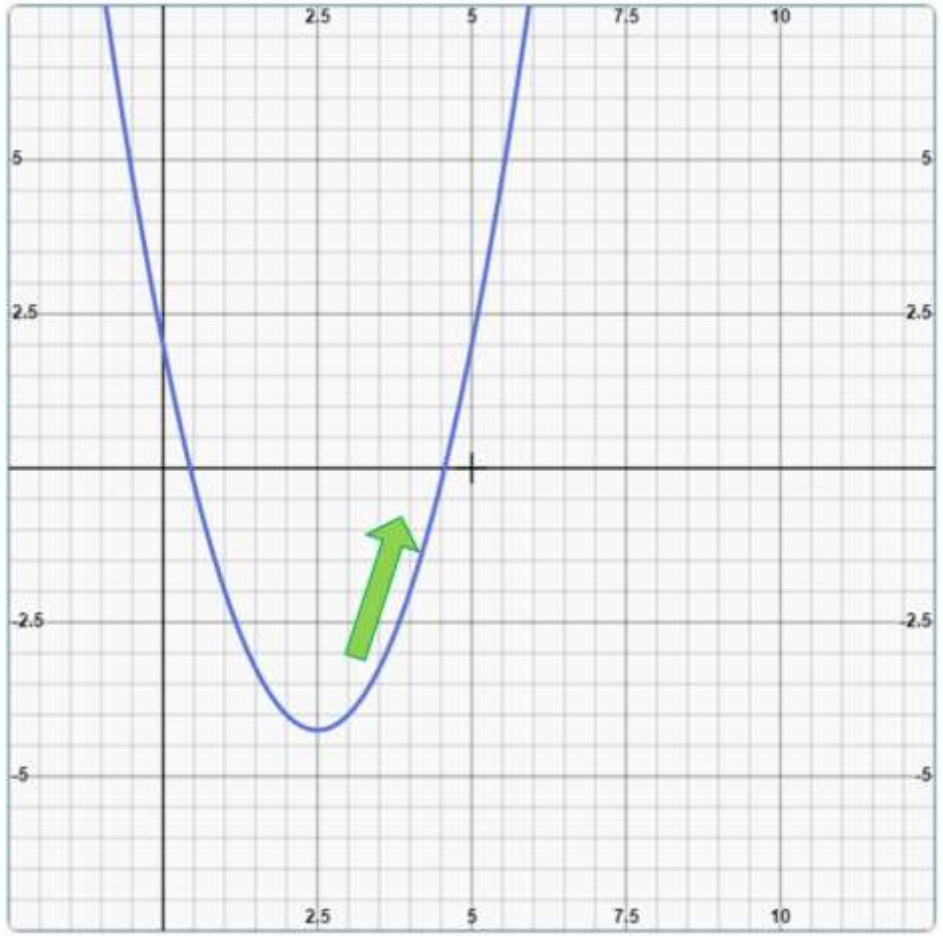


Con la ecuación de la recta aproximamos los dos puntos, hasta que convergen en un punto, y esto es la recta tangente a la función en un punto, que es la derivada.



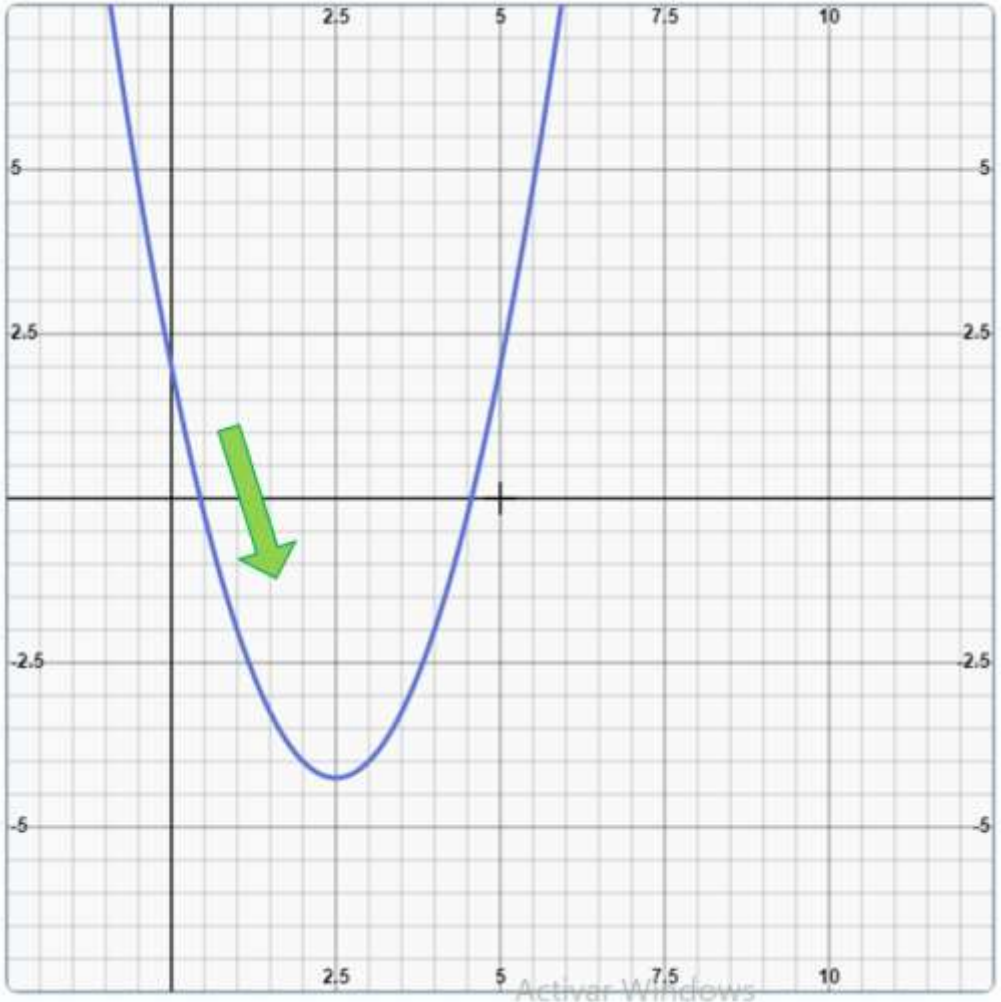
Si sustituimos valores en la x, se obtienen dos datos relevantes:

1. Si es cuesta arriba, cuesta abajo o llano
2. Pendiente (dureza de la subida)
   - Positivo: cuesta arriba.
   - Negativo: cuesta abajo.
   - Cero: llano (sin pendiente)



Cuesta arriba

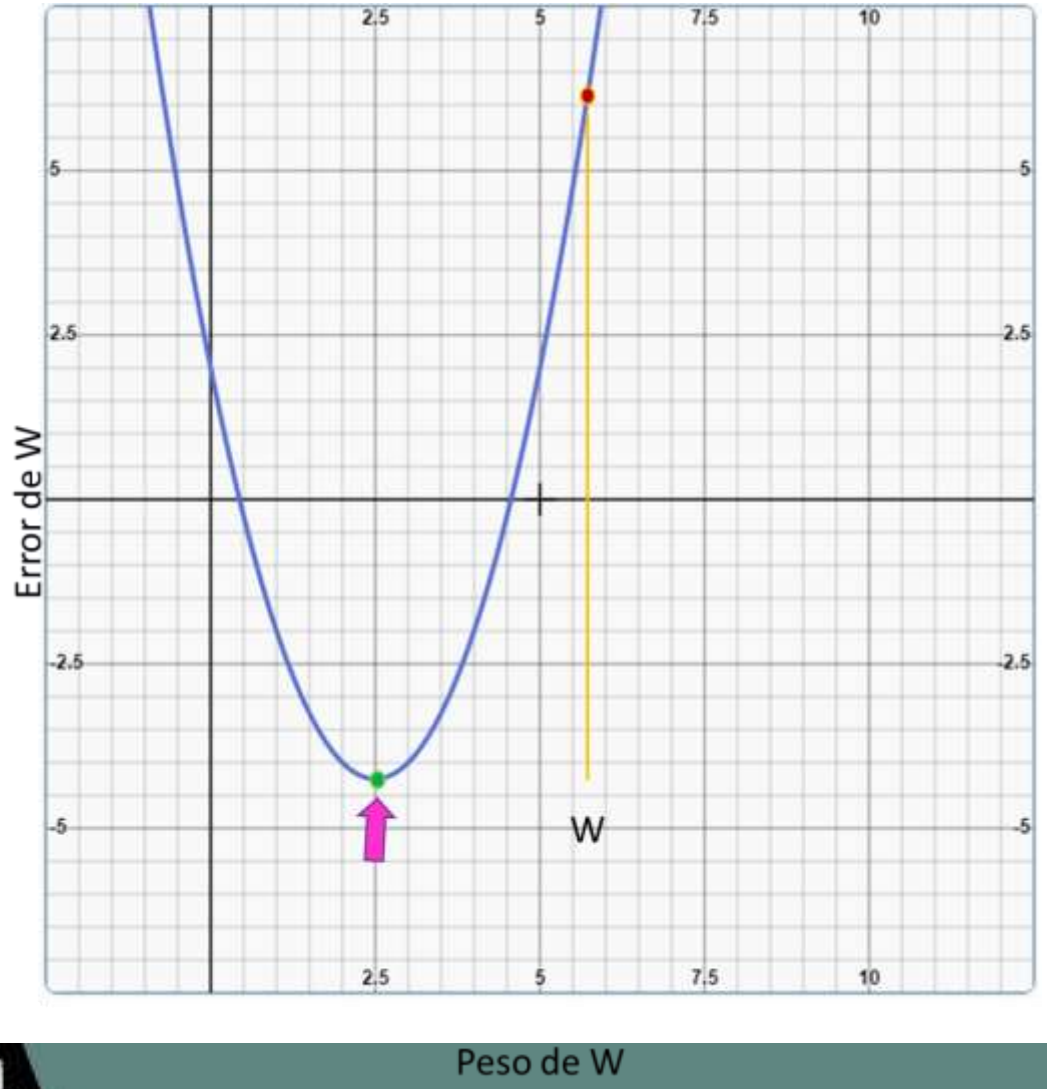
Entre más alto es el valor, más dura es la subida.



Cuesta abajo

Entre más bajo sea el número, más pronunciada es la bajada.

Descenso de gradiente



Se tiene que modificar el w, aproximándolo al punto cero para que el error sea mínimo.

Cálculo de los gradientes en redes neuronales

https://colab.research.google.com/drive/1xIT-m6pRQh0hx7wQt5bFxGfKbN607pp4#scrollTo=ggrM8WhAXBtr

Algoritmo de retropropagación

Adelante

Descripción:

- Tras inicializar los pesos de forma aleatoria y con valores pequeños, seleccionamos el primer par de entrenamiento.
- Calculamos la salida de la red.
- Calculamos la diferencia entre la salida real de la red y la salida deseada, con lo que obtenemos el vector de error.

Atras

- Ajustamos los pesos de la red de forma que se minimice el error.
- Repetimos los tres pasos anteriores para cada par de entrenamiento, hasta que el error para todos los conjuntos de entrenamiento sea aceptable.

Descenso por la superficie del error.
Cálculo de derivadas del error, respecto a los pesos y las bias.

Algoritmo backpropagation

Consideraciones:

- SSE: $E=\Sigma E_p=\Sigma(y_{pk}-o_{pk})$
- $\Delta w_{ij}=-\eta\ \partial E/\ \partial w_{ij}$

Pasos:

- Inicialización:
  - Construcción de la red. Inicialización aleatoria de pesos y umbrales (-0.5, 0.5)
  - Criterio de terminación (número máximo de iteraciones,…).
  - Contador de iteraciones n=0.
- Fase hacia delante:
  - Calcular la salida de la red para cada patrón de entrada.
  - Calcular sumatoria de los errores cuadráticos (Summed Squared Error “SSE”)
  - Si la condición de terminación se satisface, parar.

- Fase hacia atrás:
  - Incrementar el contador n=n+1.
  - Para cada neurona de salida calcular: $\delta_k=(O_k-Y_k)f'(net_k)$ donde $net_j= \Sigma_i w_{ij}x_i+b_j$
  - Para cada unidad oculta, calcular: $\delta_j=f'(net_j)\ \Sigma_k \delta_k w_{jk}$
  - Actualizar pesos: $\Delta w_{ij}(n+1)= \eta\ \delta_j o_i+ \alpha\ \Delta\ w_{ij}(n)$
  - Volver a la fase hacia adelante

Que unen a la última capa oculta con la de salida

Que unen la capa de entrada con la oculta

Inconvenientes del algoritmo backpropagation:

- Tiempo de entrenamiento no acotado
- Dependiente de las condiciones iniciales:
  - Parálisis de la red
  - Mínimos locales
  - Underfitting
  - Memorización o sobreaprendizaje.

Diferenciación automática

Busca el valor de la derivada en un punto, mediante la aproximación.

Para la diferenciación automática es necesaria la descomposición de diferenciales, proporcionada por la regla de la cadena. Por la composición simple:

$$y = f(g(h(x))) = f(g(h(w_0))) = f(g(w_1)) = f(w_2) = w_3$$

$$w_0 = x$$

$$w_1 = h(w_0)$$

$$w_2 = g(w_1)$$

$$w_3 = f(w_2) = y$$

Regla de la cadena de la diferenciación automática

$$\frac{dy}{dx} = \frac{dy}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{dx} = \frac{df(w_2)}{dw_2}\frac{dg(w_1)}{dw_1}\frac{dh(w_0)}{dx}$$

Aplicaciones

- Redes neuronales y backpropagation: son modelos que dependen de numerosas entradas, que requieren por sí mismos recorrer el grafo hacia adelante y suelen definirse por partes en el código. Todo esto hace la DA hacia atrás idónea para esta clase de problemas.
- Aquellas tareas de visión por computador que requieren de matrices jacobianas o matrices hessianas: reducción de ruido, segmentación, registro, etcétera, pueden beneficiarse del uso de DA y suelen mostrar mejor rendimiento.
- Los modelos estadísticos dentro del procesado del lenguaje natural, que se entrenan usando descenso por gradiente, son también candidatos a emplear la DA para mejorar el coste del entrenamiento y simplificar su implementación.

Referencias

- D. Michie, D.J. Spiegelhalter, C.C. Taylor (eds). Machine Learning, Neural and Statistical Classification, 1994.
- R. Rojas. Neural Networks: A Systematic Introduction, Springer, 1996.ISBN 3-540-60505-3.
- José R Hilera Martínez, “Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones”, Alfa Omega, México, (2000)
- Simon Haykin, “Neural Networks”, Prentice - Hall, New Jersey, (1999)
- Sadaoki Furui, “Digital Speech Processing, Synthesis, and Recognition, Cambridge University Press, (2001)
- Laurene Fausett. “Fundamentals Neuronal Network, architectures, algorithms, and applications”, Prentice – Hall, New Jersey, (1995).

Contacto

Edgar Morales Palafox

Doctor en Ciencias de la Computación

edgar_morales_p@yahoo.com.mx

Tels: (opcional)

Redes sociales: