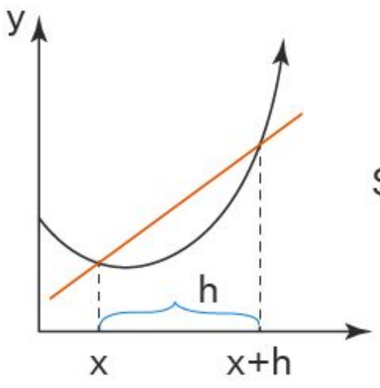
CSE4261: Neural Network and Deep Learning

Lecture: 18.06.2025

Sangeeta Biswas, Ph.D.
Associate Professor,
University of Rajshahi, Rajshahi-6205, Bangladesh

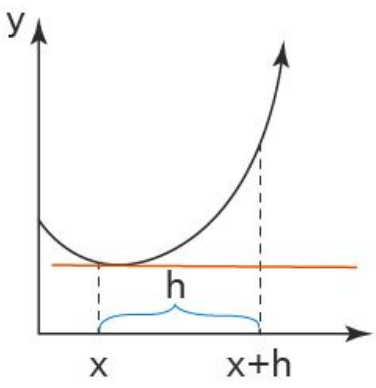
Derivative

- It is a way to measure the rate of change of a function with respect to an independent variable.
- It tells us how quickly a function is changing at a specific point.



Slope of Secant $= \dfrac{f(x+h) - f(x)}{h}$

("Difference quotient")



$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

(if limit exists)

$y = f(x)$

***x*:** independent variable

***y*:** dependent variable

Partial Derivative

- It measures the rate of change of a multivariable function
- It is done with respect to one of its variables, while holding all other variables constant.

$$f(x,y,z) = 2x+3y+4z$$

$$\frac{\partial f(x,y,z)}{\partial x} = 2$$

$$\frac{\partial f(x,y,z)}{\partial y} = 3$$

$$\frac{\partial f(x,y,z)}{\partial z} = 4$$

Gradient

- It is a vector with partial derivatives of a multivariate function.

For a function of two variables, i.e., $f(x_1, x_2)$

Gradient: $\nabla_x f = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}]$

Chain Rule

- It is a formula for finding the derivative of a composite function.

$y = (3x+3)^3$, making $t = 3x+3$

then $y = t^3$

$$\frac{dt}{dx} = 3, \qquad \frac{dy}{dt} = 3t^2$$

using the Chain Rule: $\dfrac{dy}{dx} = \dfrac{dy}{dt} \cdot \dfrac{dt}{dx}$

$$\therefore \quad \frac{dy}{dx} = (3t^2).(3) = 9t^2$$

$$= 9(3x+3)^2 = 9(3)(x+1)(3)(x+1)$$

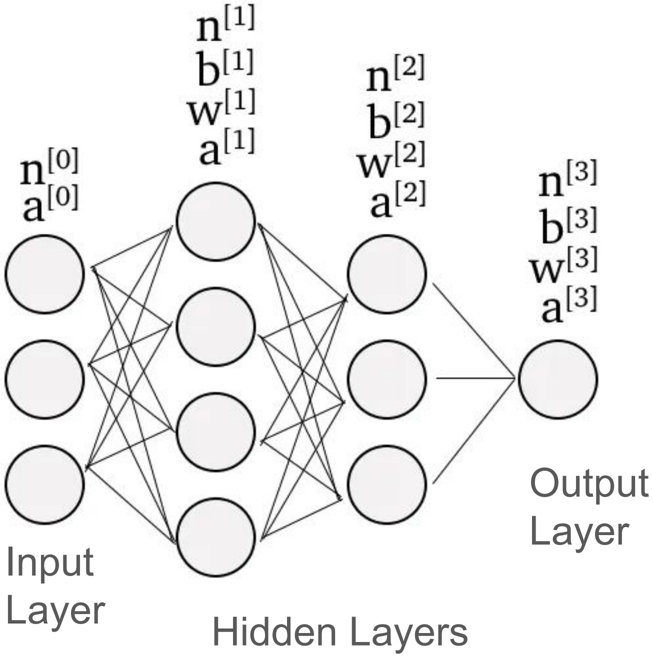
$$= \underline{81(x+1)^2}$$

Backpropagation Algorithm

- It is used for efficient weight adjustments to minimize the error between predicted and actual outputs.
- It works by:
  - calculating the gradient of the loss function with respect to the network's weights.
  - Propagating error information backward through the network layers

A Simple Reference Neural Network

- n: number of neurons
- a: vector of activation function
- w: matrix of weights
- b: vector of biases



Formulas

- Weighted input to the neurons in layer ℓ, $z^{[l]} = w^{[l]}.a^{[l-1]} + b^{[l]}$

- Activation of layer ℓ, $\mathrm{a}^{[l]} = g^{[l]}(z^{[l]})$

- Say, one neuron's output y can be 0 or 1.

  Then cross-entropy cost function is: $\mathrm{C} = \text{-}(\mathrm{ylog}\,\hat{y} + (1 - y)\mathrm{log}(1 - \hat{y}))$

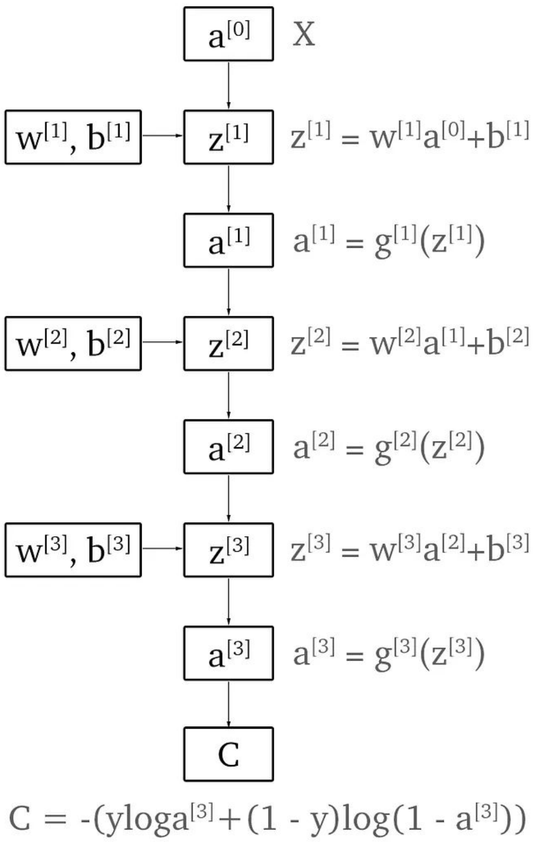
Update Rule

Alpha is the learning rate, which decides how big the updates will happen.

$$w^{[l]} = w^{[l]} - \alpha \frac{\partial C}{\partial w^{[l]}}$$

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial C}{\partial b^{[l]}}$$

Forward Propagation Computation Graph



Partial Derivative of C

- Partial Derivative of C with respect to parameters of any layer, ℓ

$$\frac{\partial C}{\partial w^{[l]}} = \frac{\partial C}{\partial z^{[l]}} \cdot \frac{\partial z^{[l]}}{\partial w^{[l]}}$$

$$\frac{\partial C}{\partial b^{[l]}} = \frac{\partial C}{\partial z^{[l]}} \cdot \frac{\partial z^{[l]}}{\partial b^{[l]}}$$

Partial Derivative of C

- Partial Derivative of C with respect to parameters of Layer-3

$$\frac{\partial C}{\partial w^{[3]}} = \frac{\partial C}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial w^{[3]}}$$

$$\frac{\partial C}{\partial b^{[3]}} = \frac{\partial C}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial b^{[3]}}$$

Partial Derivative of C

- Partial Derivative of C with respect to parameters of Layer-2

$$\frac{\partial C}{\partial w^{[2]}} = \frac{\partial C}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w^{[2]}}$$

$$\frac{\partial C}{\partial b^{[2]}} = \frac{\partial C}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial b^{[2]}}$$

Partial Derivative of C

- Partial Derivative of C with respect to parameters of Layer-1

$$\frac{\partial C}{\partial w^{[1]}} = \frac{\partial C}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial w^{[1]}}$$

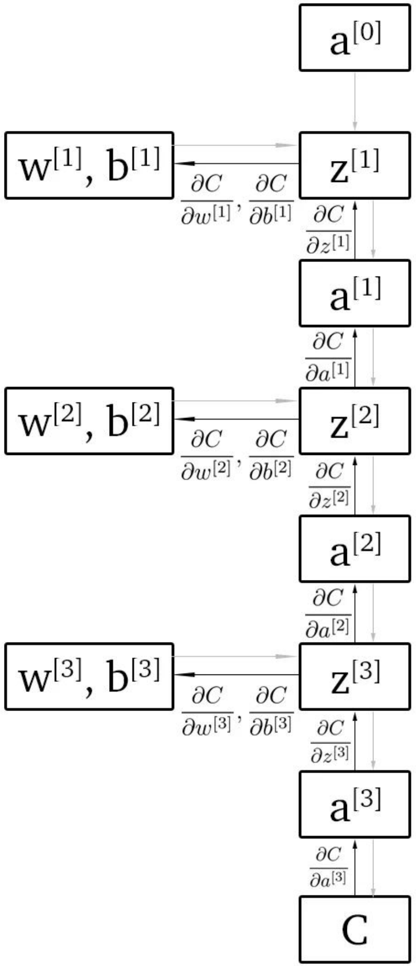
$$\frac{\partial C}{\partial b^{[1]}} = \frac{\partial C}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial b^{[1]}}$$

Backpropagation Computation Graph

- for calculating the partial derivatives of C with respect to $w[l]$, $b[l]$, we need to calculate:

$$\frac{\partial C}{\partial z^{[L]}}, \frac{\partial C}{\partial z^{[l]}}, \frac{\partial z^{[l]}}{\partial w^{[l]}}, \frac{\partial z^{[l]}}{\partial b^{[l]}}$$

- error information is propagated backward through the network layers for updating the weights



tf.GradientTape()

- It records operations for automatic differentiation
- https://www.tensorflow.org/api_docs/python/tf/GradientTape
- Example:

```
x = tf.constant(3.0)

with tf.GradientTape() as g:

    g.watch(x)

    y = x * x

dy_dx = g.gradient(y, x)
```