Tutorial 3

Fei Gao & Yuliia Frund

Overview

- Exercise overview
- Computation graphs and Forward pass
- Backward pass
- Gradient descent
- Practice: Chain rule + Product rule
- Practice: Chain rule and Product rule in computational graphs
- Application of Partial Derivatives in Machine Learning
    - Revisiting the update rule
- Practice: Derivatives (Optional)

Exercise overview

Part 1

1 Vectors & Matrices

1.1 Dot Product

$$\mathbf{x} = \begin{bmatrix} 1 \\ 4 \\ 5 \\ 8 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 9 \\ 2 \\ 7 \\ 8 \end{bmatrix}$$

Calculate the dot products:

1. $\mathbf{x} \cdot \mathbf{w}$
2. $\mathbf{w} \cdot \mathbf{y}$
3. $\mathbf{x} \cdot \mathbf{y}$
4. $\mathbf{y} \cdot \mathbf{x}$

Part 1

1.2 Matrix Multiplication

$$\mathbf{W} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 8 & 9 \\ 9 & 3 & 2 \\ 8 & 3 & 1 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 4 & 8 \\ 7 & 2 \\ 6 & 9 \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} 5 & 2 \\ 6 & 2 \\ 8 & 1 \\ 9 & 1 \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$$

Calculate if possible:

1. $\mathbf{W} \cdot \mathbf{X}$
2. $\mathbf{W} \cdot \mathbf{a}$
3. $\mathbf{X} \cdot \mathbf{Y}$
4. $\mathbf{X} \cdot \mathbf{Y}^{\mathbf{T}}$

Part 2

2 Computational Graphs

2.1 Drawing a Computational Graph

Draw the computational graph for

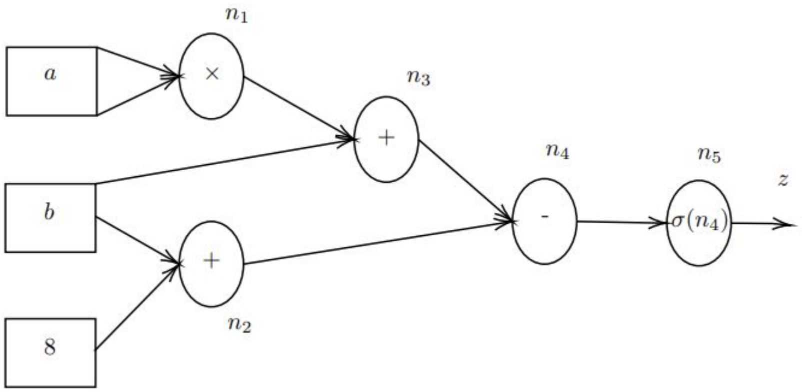
$$z = \sigma(3 + (y + 5x))$$

where σ = sigmoid activation function applied as a single step (i.e. a single node in the graph).

Part 2

2.2 Analyzing Computational Graphs and Performing Calculations

1. Observe the computational graph below. Write its function expression z.



Part 2

2. Given that $a = 3$ and $b = 5$, perform the forward pass (hint: calculate from n_1 to n_5, and then z).

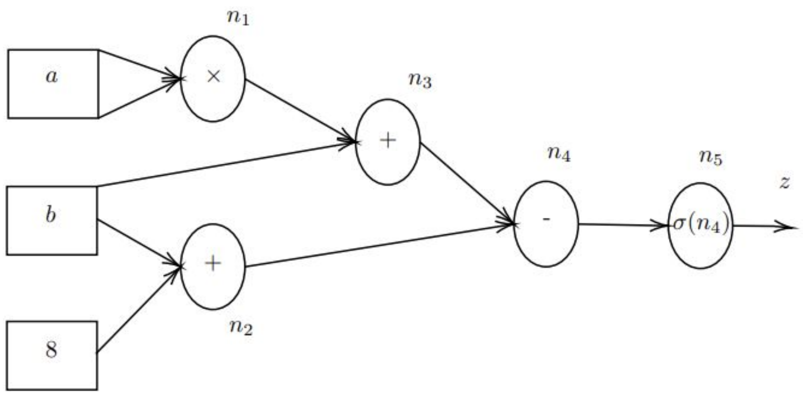
3. Given that $a = 3$ and $b = 5$, calculate the following derivatives:

   (a) $\frac{\partial z}{\partial n_4}$

   (b) $\frac{\partial z}{\partial n_3}$

   (c) $\frac{\partial z}{\partial a}$

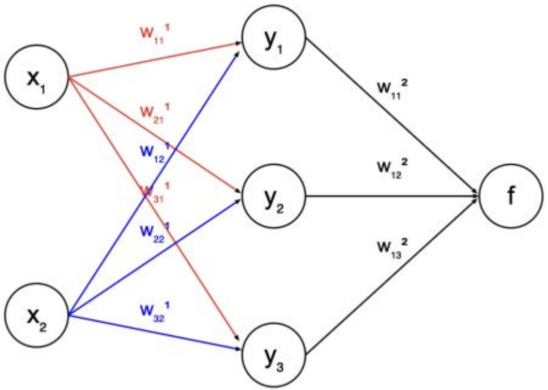
   (d) $\frac{\partial z}{\partial b}$



Part 3

3 Feed Forward Networks

Consider the following neural network. The numbers above the lines correspond to the weights of the connections. **In the hidden layer**, the sigmoid activation function $\sigma(y_i)$ is applied. The network does not have any biases ($b = 0$).



Part 3

Make sure that you get a dimensionality that WORKS!
Dimensionality: m x n
m: number of rows
n: number of columns

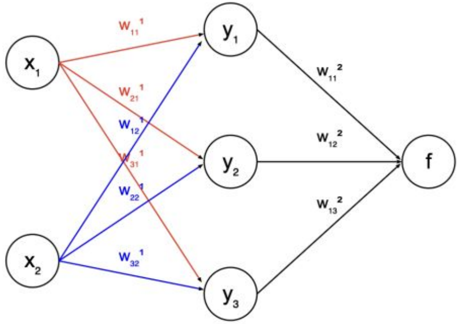
1. Given input $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, what is the dimensionality of the input x?

2. When the input x is a column vector, the formula to use is $y = \mathbf{W}\mathbf{x}$. What is the dimensionality of weight matrix $\mathbf{W}^{[1]}$?

3. Given that the output is a scalar value, and $f = \mathbf{W}^{[2]}\mathbf{x}$, what is the dimensionality of $\mathbf{W}^{[2]}$?

3 Feed Forward Networks

Consider the following neural network. The numbers above the lines correspond to the weights of the connections. **In the hidden layer**, the sigmoid activation function $\sigma(y_i)$ is applied. The network does not have any biases ($b = 0$).



Part 3

Pay attention to the shape for Q1-Q3
Incorrect shape = wrong
If you have questions, ask us
(contact via olat/forum)

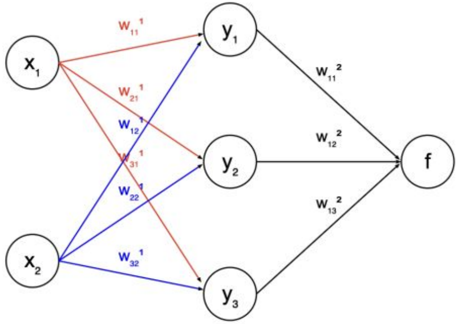
1. Given input $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, what is the dimensionality of the input x?

2. When the input x is a column vector, the formula to use is $\boxed{y = \mathbf{W}\mathbf{x}}$. What is the dimensionality of weight matrix $\mathbf{W}^{[1]}$?

3. Given that the output is a scalar value, and $f = \mathbf{W}^{[2]}\mathbf{x}$, what is the dimensionality of $\mathbf{W}^{[2]}$?

3 Feed Forward Networks

Consider the following neural network. The numbers above the lines correspond to the weights of the connections. **In the hidden layer**, the sigmoid activation function $\sigma(y_i)$ is applied. The network does not have any biases ($b = 0$).



Part 3

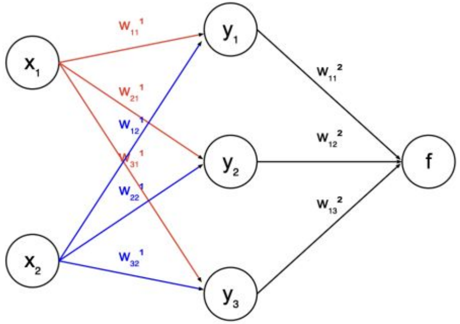
You get the same graph for the current page on top just for your convenience

If the colors are difficult to read (you printed the exercises in B/W, etc.), read these:

4. Express the neural network's output as a function f, using the individual weights $w_{ij}^{[l]}$ and input components x_i.
$W_{11}^{[1]}$, $W_{21}^{[1]}$, and $W_{31}^{[1]}$ are the weights from x_1 to y_1, y_2, and y_3 respectively. Similarly, $W_{12}^{[1]}$, $W_{22}^{[1]}$, and $W_{32}^{[1]}$ are the weights from x_2 to y_1, y_2, and y_3.
Your expression should show how the output is computed through each network layer.

3 Feed Forward Networks

Consider the following neural network. The numbers above the lines correspond to the weights of the connections. **In the hidden layer**, the sigmoid activation function $\sigma(y_i)$ is applied. The network does not have any biases ($b = 0$).



Part 3

$x_1 = 1 \quad x_2 = -1$

$w_{11}^{[1]} = 1 \quad w_{12}^{[1]} = 0$

$w_{21}^{[1]} = 0 \quad w_{22}^{[1]} = 1$

$w_{31}^{[1]} = 1 \quad w_{32}^{[1]} = 1$

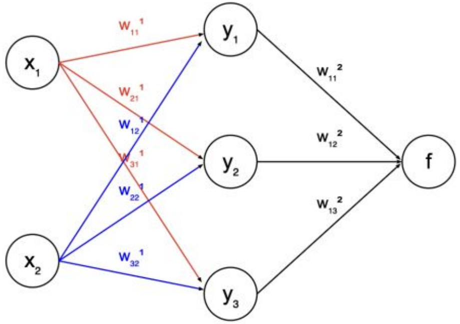
$w_{11}^{[2]} = \frac{1}{2}$

$w_{12}^{[2]} = \frac{1}{2}$

$w_{13}^{[2]} = \frac{5}{2}$

3 Feed Forward Networks

Consider the following neural network. The numbers above the lines correspond to the weights of the connections. **In the hidden layer**, the sigmoid activation function $\sigma(y_i)$ is applied. The network does not have any biases ($b = 0$).



5. Using values above, express $\mathbf{x}$, $\mathbf{W}^{[1]}$, and $\mathbf{W}^{[2]}$ in matrix or vector form.

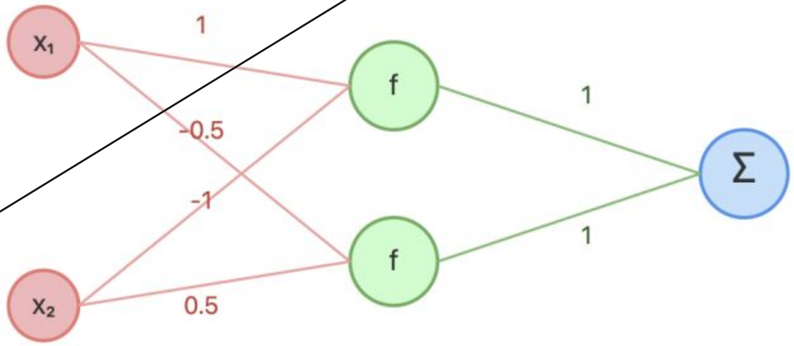
6. Compute the forward pass.

   You can use the same approach as in Q2.2, or other efficient methods, but ensure you detail how you arrive at the final output value.

7. Compute $\dfrac{\partial f}{\partial w_{32}^{[1]}}$. Note that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

   Use insights from your expression in 4 to guide your calculation. Show your work, including any application of the chain rule or other relevant calculus concepts.

Here is a 1-layer neural network with a **column vector** input $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.



Part 4

4.1 Linear Activation Function

For the above neural network, We include a bias $b = 1$ for each neuron in the hidden layer, and we define the activation function as $f(z) = 2x$.

1. Express the two weight matrices $\mathbf{W_1}$, $\mathbf{W_2}$.

Same, pay attention to the shape!!!

2. Write all possible inputs x_1, x_2, and the output y of the XOR problem.
   You can find the truth table in the lecture or tutorial slides.

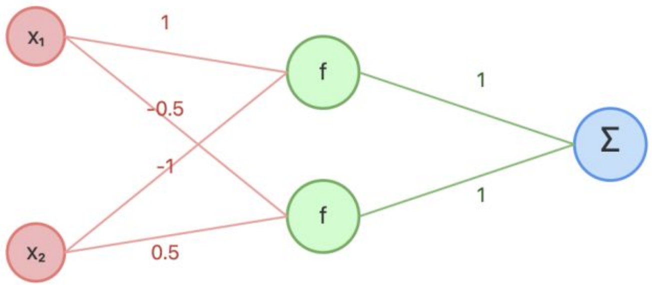
3. Perform the forward pass for **just one** pair of XOR inputs.
   Show your work, include all steps to arrive at the final output.

4. Perform the forward pass for the rest of the pairs on your own and answer: Does the current neural network solve the problem?
   ☐ Yes ☐ No

5. If you remove the bias, would this solve the problem?
   ☐ Yes ☐ No

Part 4

Here is a 1-layer neural network with a **column vector** **input** $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.



4.2 Nonlinear Activation Function

Now we remove the bias and the previous activation function. Instead, we use the Heaviside step activation function (also called the unit step function):

$$f(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

1. Perform the forward pass again for **all** XOR input pairs.
   For each pair, show the values for both neurons in the hidden layer, and show how these values lead to the final output.

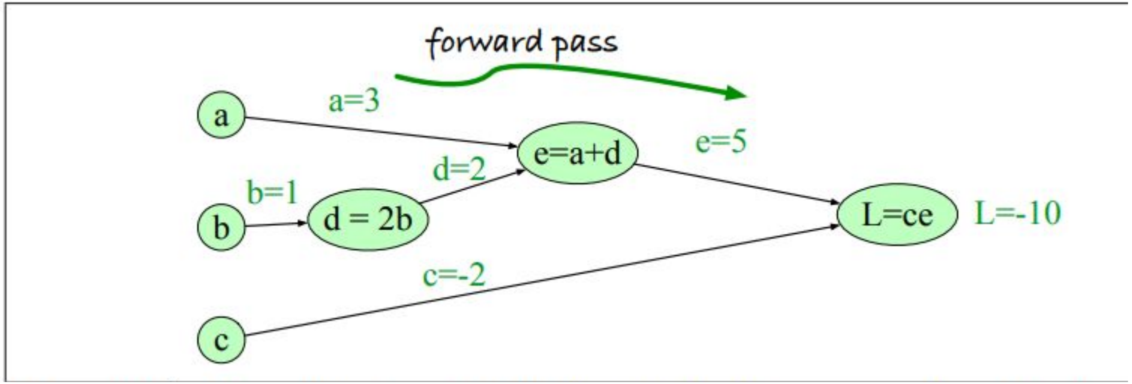
2. Does the new activation function solve the problem? If yes, what makes the difference? If no, why does it fail? Explain in your own words (maximum 2 sentences).

Questions?

- Start as early as possible
- If you have questions, please ask as early as possible!
- Team up with someone
  - Need a teammate? There is a forum thread for exercises!
  - Or you can come to us after class, give us your name and we can try to assign you in pairs.

Computation Graphs & Forward Pass

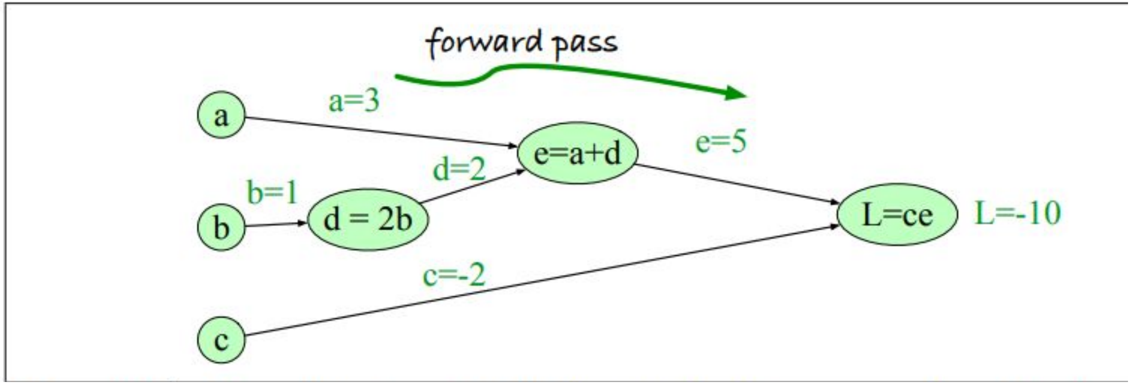
Computation Graphs



https://web.stanford.edu/~jurafsky/slp3/7.pdf

Computation Graphs

Can you write the function expression of this graph?



https://web.stanford.edu/~jurafsky/slp3/7.pdf

Computation Graphs

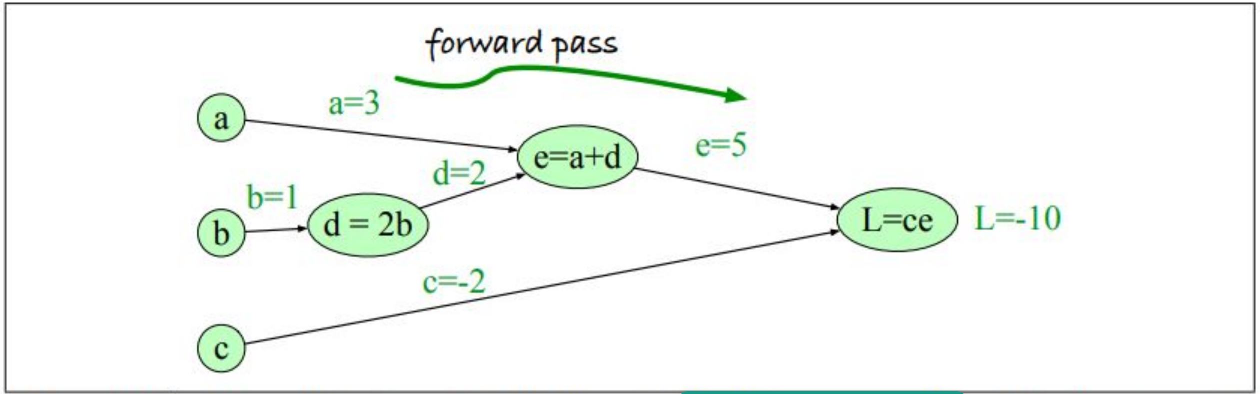


Figure 7.12 Computation graph for the function $\boxed{L(a,b,c) = c(a+2b)}$, with values for input nodes $a = 3$, $b = 1$, $c = -2$, showing the forward pass computation of L.

https://web.stanford.edu/~jurafsky/slp3/7.pdf

Computation Graphs

- **Forward pass**: the path your network goes through to make a prediction.
- How do we “tell” it whether it’s right or wrong? How do we “teach” the network?

Backward Pass

Backward Pass

Loss function: how right / wrong is the network?

Backward Pass

Loss function: how right / wrong is the network?

$$L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y$$

https://web.stanford.edu/~jurafsky/slp3/5.pdf

Backward Pass

Loss function: how right / wrong is the network?

$$L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y$$

E.g.: **Cross Entropy**:

- **Measures the difference between predicted probabilities and correct classes**
- **The lower — the better**

https://web.stanford.edu/~jurafsky/slp3/5.pdf

Gradient Descent

Gradient Descent

You need to **descend**, but you can't see too far because of the fog. What do you do?

Gradient Descent

You need to **descend**, but you can't see too far because of the fog. What do you do?

Choose the **steepest descent** you can see. How?

Gradient Descent

You need to **descend**, but you can't see too far because of the fog. What do you do?

Choose the **steepest descent** you can see. How? - Measurements! (**Gradient**)

Gradient Descent

You need to **descend**, but you can't see too far because of the fog. What do you do?

Choose the **steepest descent** you can see. How? - Measurements! (**Gradient**)

How big a step do you make?

Gradient Descent

You need to **descend**, but you can’t see too far because of the fog. What do you do?

Choose the **steepest descent** you can see. How? - Measurements! (**Gradient**)

How big a step do you make? - Controlled by the **learning rate**.

Gradient Descent

The **gradient** computes which direction you should move in, and the **learning rate** helps to control how fast (by how much) you move.

Gradient Descent

In other words:

Find the **direction** you need to move in to reach the **minimum of the loss function**.

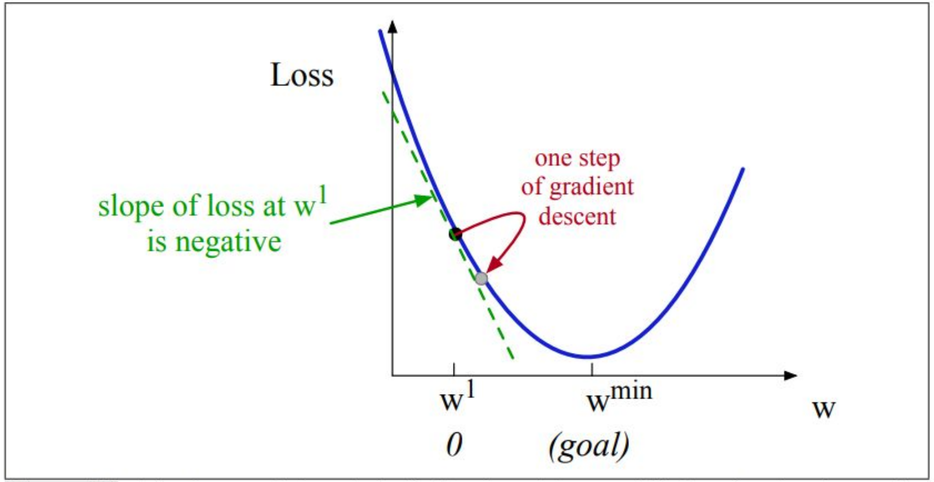


Figure 5.4 The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function. Since the slope is negative, we need to move w in a positive direction, to the right. Here superscripts are used for learning steps, so w^1 means the initial value of w (which is 0), w^2 the value at the second step, and so on.

https://web.stanford.edu/~jurafsky/slp3/5.pdf

Gradient Descent

In other words:

Find the **direction** you need to move in to reach the **minimum of the loss function**.

$$w_{new} = w_{old} - \gamma \frac{\partial L_{CE}}{\partial w_{old}}$$

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w),y)$$

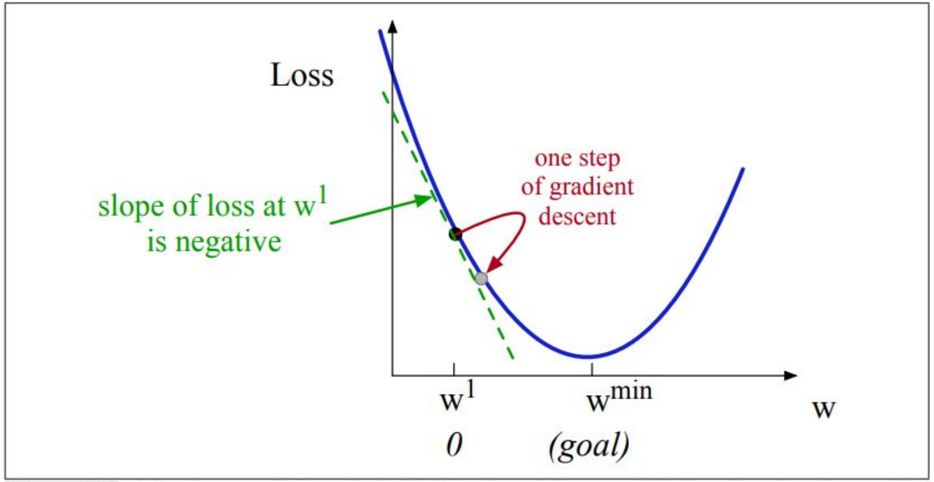


Figure 5.4 The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function. Since the slope is negative, we need to move w in a positive direction, to the right. Here superscripts are used for learning steps, so w^1 means the initial value of w (which is 0), w^2 the value at the second step, and so on.

Formulae from lecture slides, https://web.stanford.edu/~jurafsky/slp3/5.pdf

Learning Rate

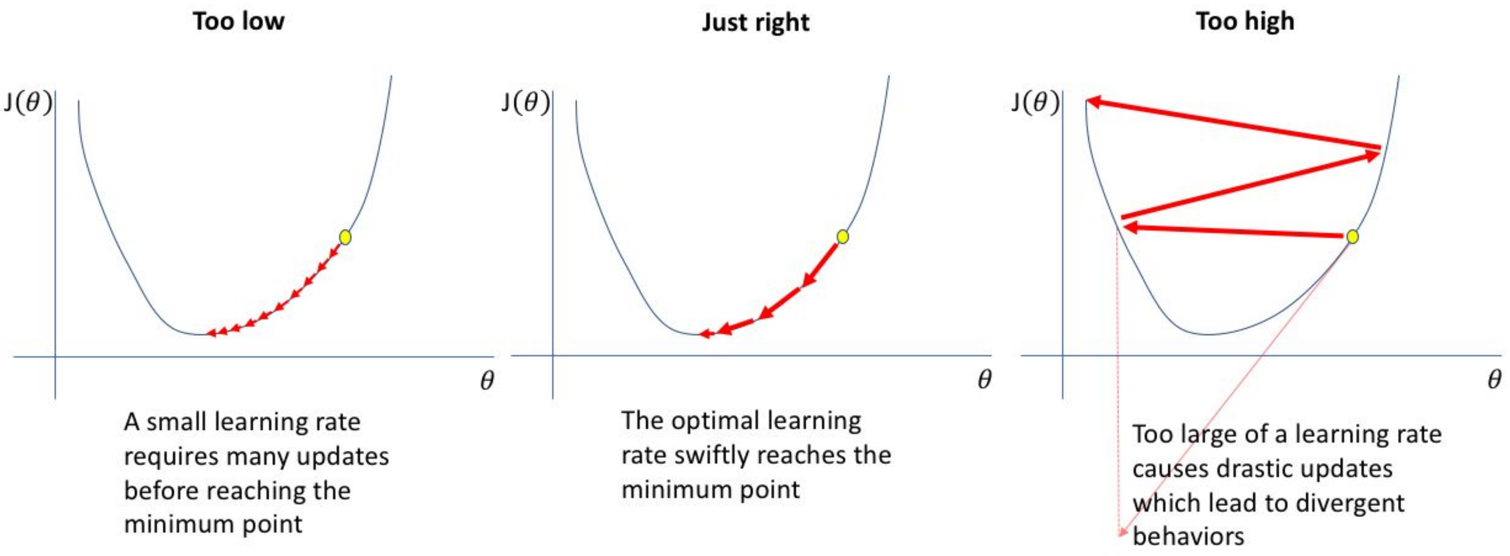


Image source: https://www.jeremyjordan.me/nn-learning-rate/

Updating Parameters — When?

After looking at **all training data**: Gradient Descent

Updating Parameters — When?

After looking at **all training data**: Gradient Descent

Can be slow

Need to load all data -> uses a lot of memory

Updating Parameters — When?

After looking at **all training data**: Gradient Descent

Can be slow

Need to load all data -> uses a lot of memory

After **every example**: Stochastic Gradient Descent

Updating Parameters — When?

After looking at **all training data**: Gradient Descent

Can be slow

Need to load all data -> uses a lot of memory

After **every example**: Stochastic Gradient Descent

Fast and uses less memory

Updates can be noisy

Updating Parameters — When?

After looking at **all training data**: Gradient Descent

- Can be slow
- Need to load all data -> uses a lot of memory

After **every example**: Stochastic Gradient Descent

- Fast and uses less memory
- Updates can be noisy

After a **batch of examples**: Mini-Batch Gradient Descent

Updating Parameters — When?

After looking at **all training data**: Gradient Descent

- Can be slow
- Need to load all data -> uses a lot of memory

After **every example**: Stochastic Gradient Descent

- Fast and uses less memory
- Updates can be noisy

After a **batch of examples**: Mini-Batch Gradient Descent

- Faster than GD but not as fast as SGD
- Uses less memory than GD but more than SGD
- Good for parallel computing (speed)
- Requires tuning of batch size

Derivatives: theory and practices

Recap: Basic Rules

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	$2x$
Square Root	$\sqrt{x}$	$(\frac{1}{2})x^{-\frac{1}{2}}$
Exponential	e^x	e^x
	a^x	$\ln(a)\, a^x$
Logarithms	$\ln(x)$	$1/x$
	$\log_a(x)$	$1 / (x \ln(a))$

Rules	Function	Derivative
Multiplication by constant	cf	cf'
Power Rule	x^n	nx^{n-1}

Image source: see week 1

Recap: Basic Rules

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	$2x$
Square Root	$\sqrt{x}$	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\, a^x$
Logarithms	$\ln(x)$	$1/x$
	$\log_a(x)$	$1 / (x \ln(a))$

Rules	Function	Derivative
Multiplication by constant	cf	cf'
Power Rule	x^n	nx^{n-1}

Image source: see week 1

Recap: More Rules

Rules	Function	Derivative
Multiplication by constant	cf	cf'
Power Rule	x^n	nx^{n-1}
Sum Rule	$f + g$	$f' + g'$
Difference Rule	$f - g$	$f' - g'$
Product Rule	fg	$f\,g' + f'\,g$
Chain Rule (as "Composition of Functions")	$f \circ g$	$(f' \circ g) \times g'$
Chain Rule (using ′)	$f(g(x))$	$f'(g(x))g'(x)$
Chain Rule (using $\frac{d}{dx}$)	$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$	

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	$2x$
Square Root	$\sqrt{x}$	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\, a^x$
Logarithms	$\ln(x)$	$1/x$
	$\log_a(x)$	$1 / (x \ln(a))$

Recap: Chain Rule

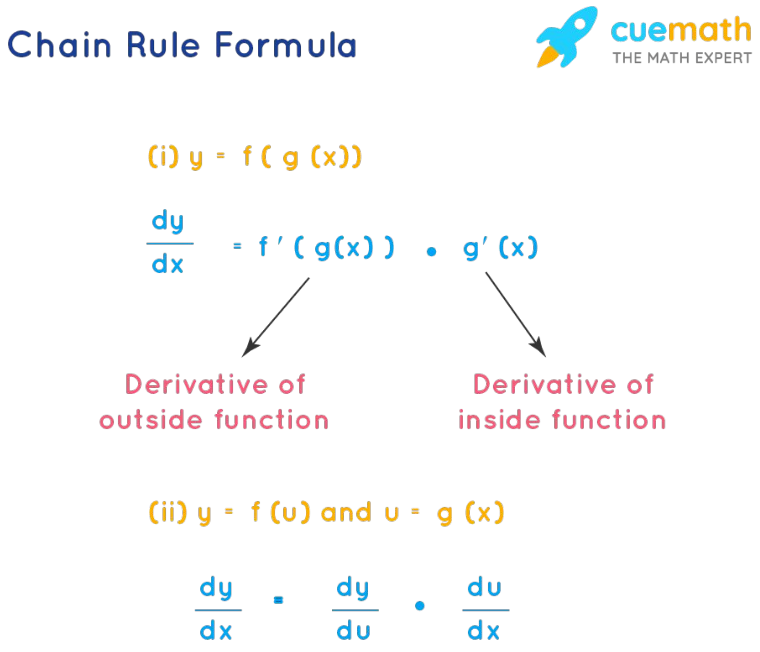


Image source: https://www.geeksforgeeks.org/chain-rule-formula/

Derivative Exercise: 1/4

$$y = e^{3x^4+2x+6}$$

Chain Rule Formula

(i) $y = f(g(x))$

$$\frac{dy}{dx} = f'(g(x)) \cdot g'(x)$$

Derivative of outside function

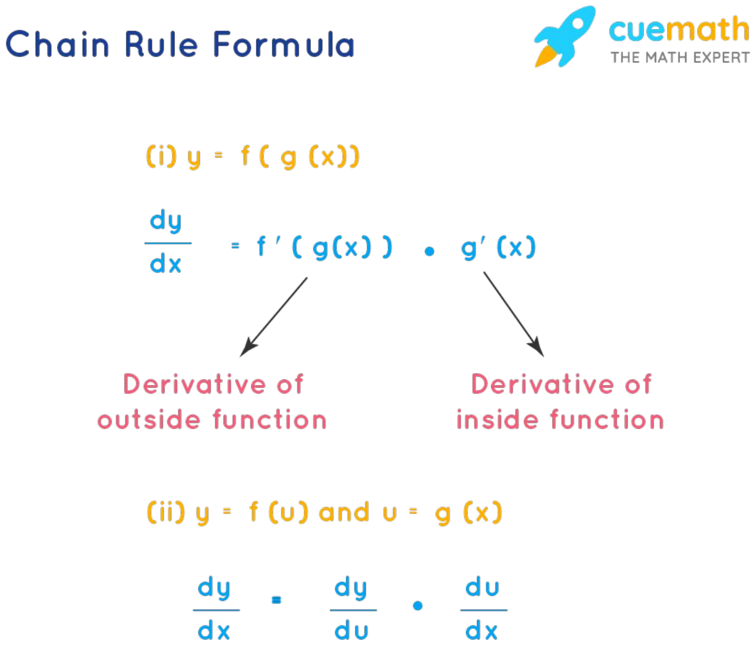
Derivative of inside function

(ii) $y = f(u)$ and $u = g(x)$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Derivative Exercise: 2/4

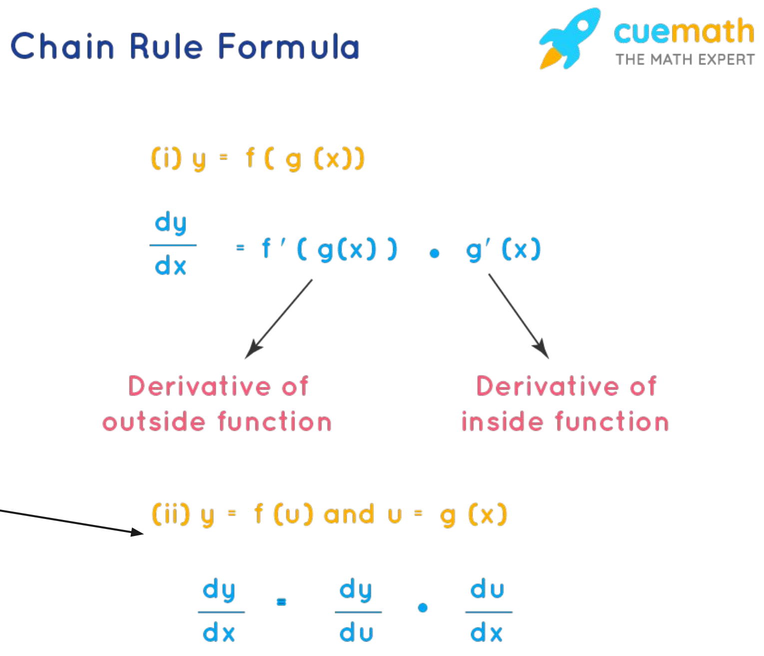
$$y = (3x^2 + 2x + 1)^5$$



Derivative Exercise: 3/4

How to apply (ii) to the following formula?

$$y = (3x - 2)^2 + 1$$



Derivative Exercise: 3/4

$$y = u^2 + 1; u = 3x - 2$$

Chain Rule Formula

(i) $y = f(g(x))$

$$\frac{dy}{dx} = f'(g(x)) \cdot g'(x)$$

Derivative of outside function

Derivative of inside function

(ii) $y = f(u)$ and $u = g(x)$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Recap: Product Rule

$$\frac{d}{dx}[f(x)g(x)] = f(x)g'(x) + f'(x)g(x)$$

Derivative Exercise: 4/4

$$f(x) = (x^3 + 3)^3 \cdot (2x - 2)^5$$

Application in Computational Graphs

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

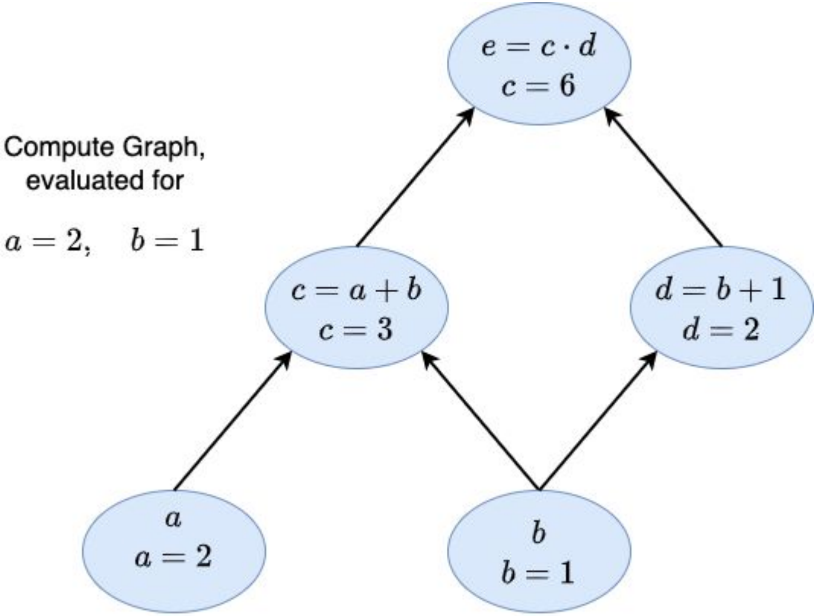


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial a} =$$

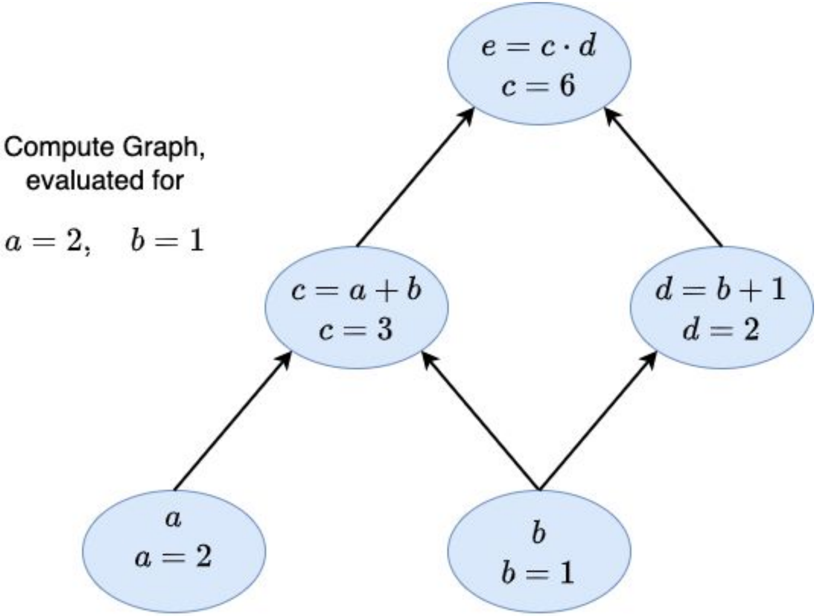


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let’s practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c} \times \frac{\partial c}{\partial a} =$$

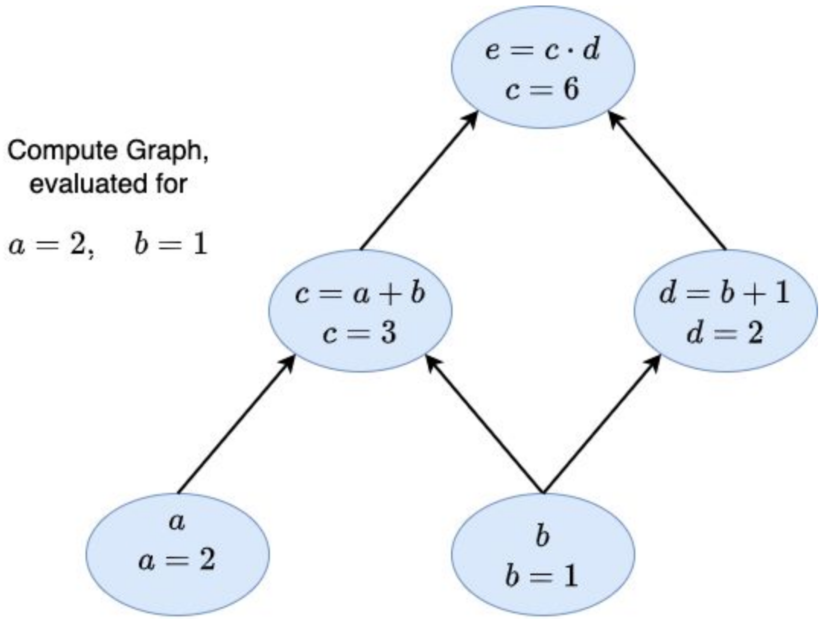


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c} \times \frac{\partial c}{\partial a} =$$

CHAIN RULE

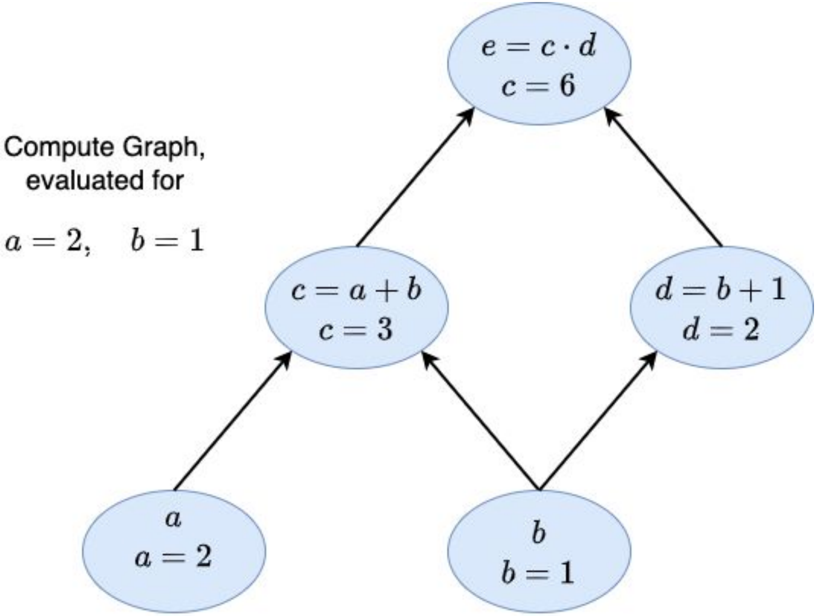


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c} \times \frac{\partial c}{\partial a} = 2 \times 1 = 2$$

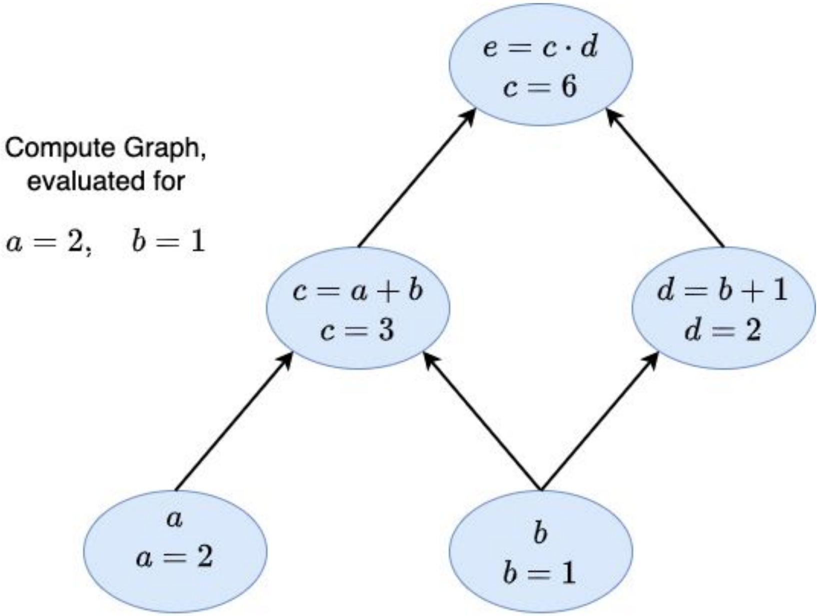


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial b} =$$

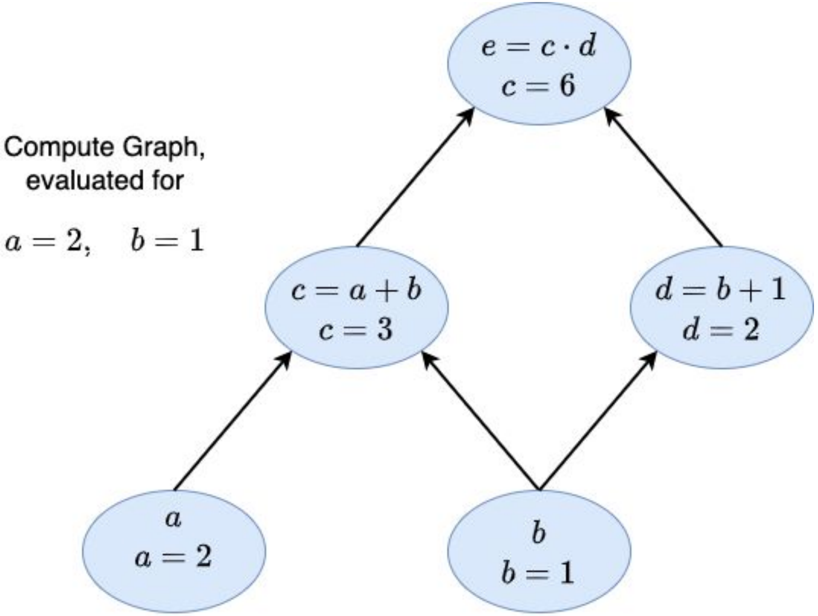


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \times \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \times \frac{\partial d}{\partial b} =$$

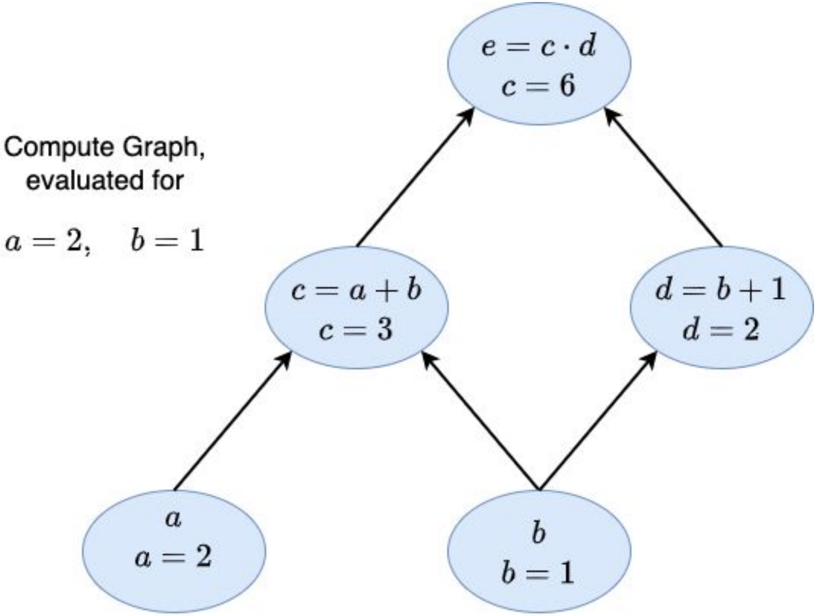


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \times \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \times \frac{\partial d}{\partial b} =$$

PRODUCT RULE

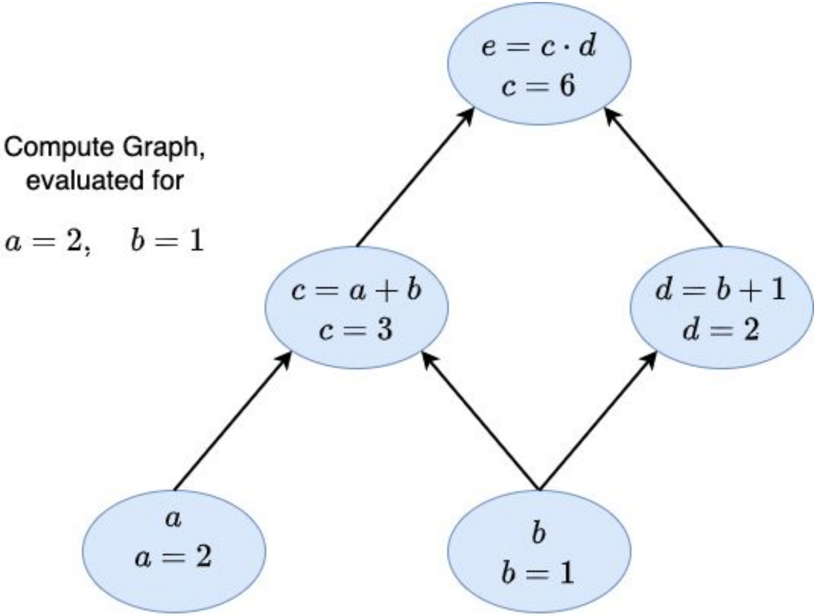


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Derivatives and Computational Graphs

Let's practise backpropagation by looking at a graph.

Find:

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c}\times\frac{\partial c}{\partial b}+\frac{\partial e}{\partial d}\times\frac{\partial d}{\partial b} = 2\times1+3\times1 = 2+3 = 5$$

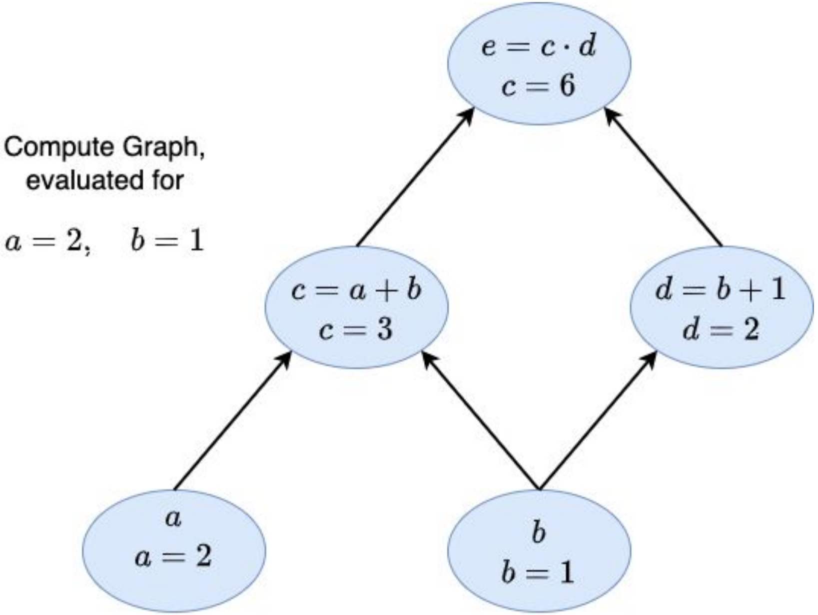


Image: https://maucher.pages.mi.hdm-stuttgart.de/artificial-intelligence/00_Computational_Graphs.html

Application of partial derivatives in ML

Partial Derivatives Exercise 1/1

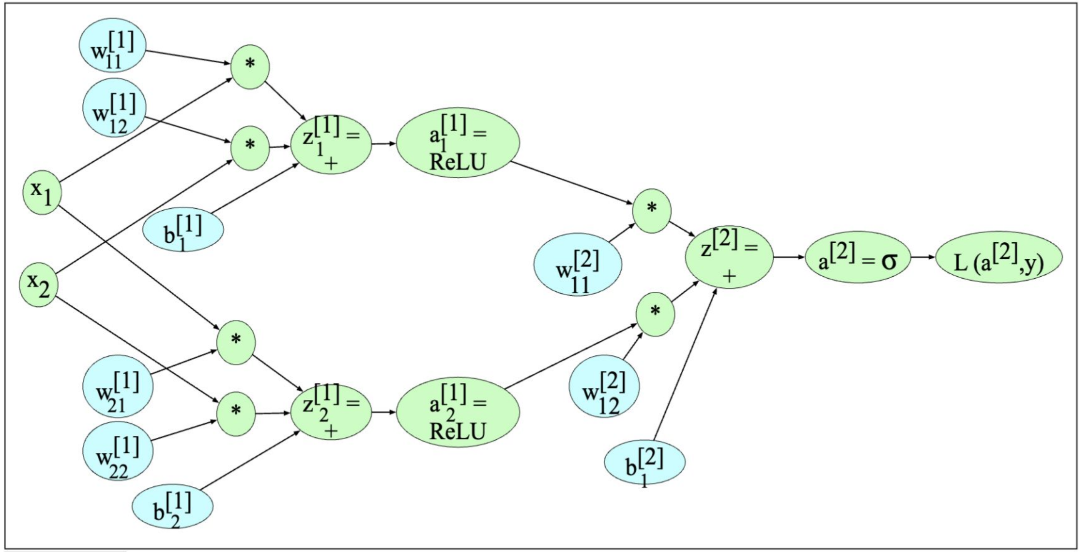
$$z = x^2 - 6y^2 + 3xy - x + 2y + 6$$

Partial Derivatives: from Math to ML

$$z = x^2 - 6y^2 + 3xy - x + 2y + 6$$

In our calculation examples, we are calculating the derivatives with regard to the variables x, y

Partial Derivatives: from Math to ML

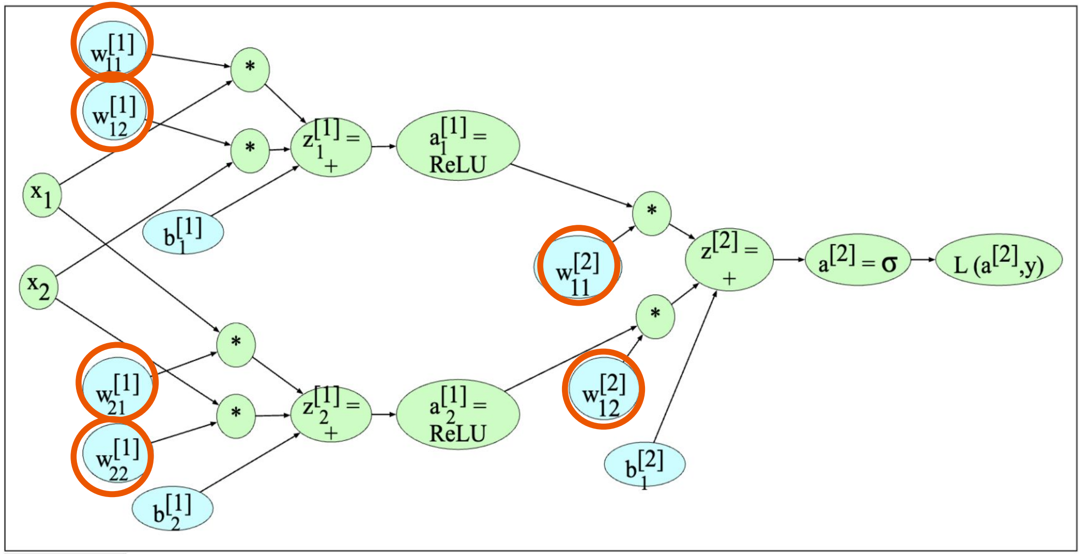


but in machine learning, we are calculating the derivatives with regard to the weights

$$z = x^2 - 6y^2 + 3xy - x + 2y + 6$$

Note: there is no correlation between this formula and the graph above. It's just for illustration purpose!

Partial Derivatives: from Math to ML

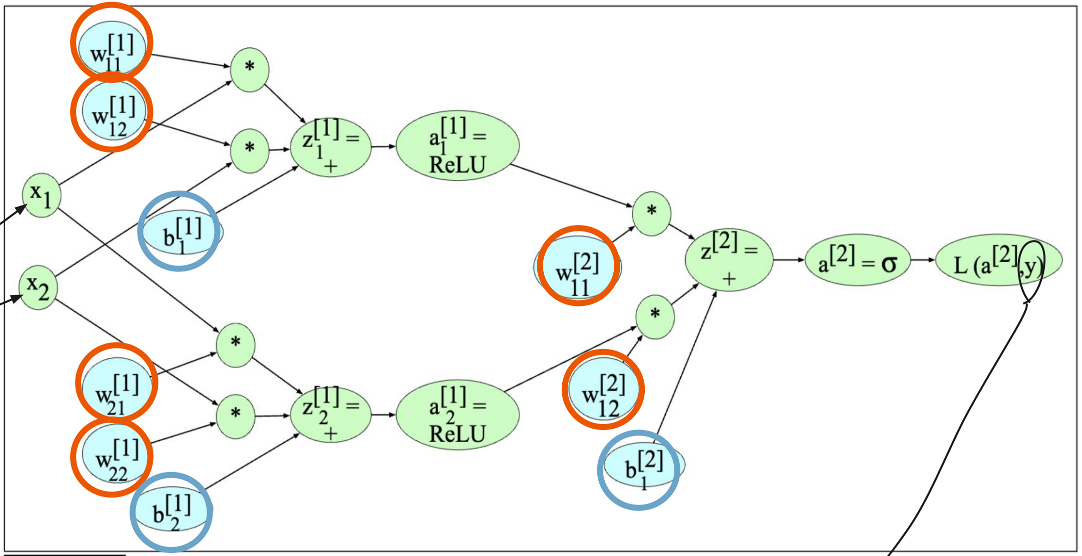


but in machine learning, we are calculating the derivatives with regard to the weights

$$\hat{y} = x_1^2 - 6x_2^2 + 3x_1x_2 - x_1 + 2x_2 + 6$$

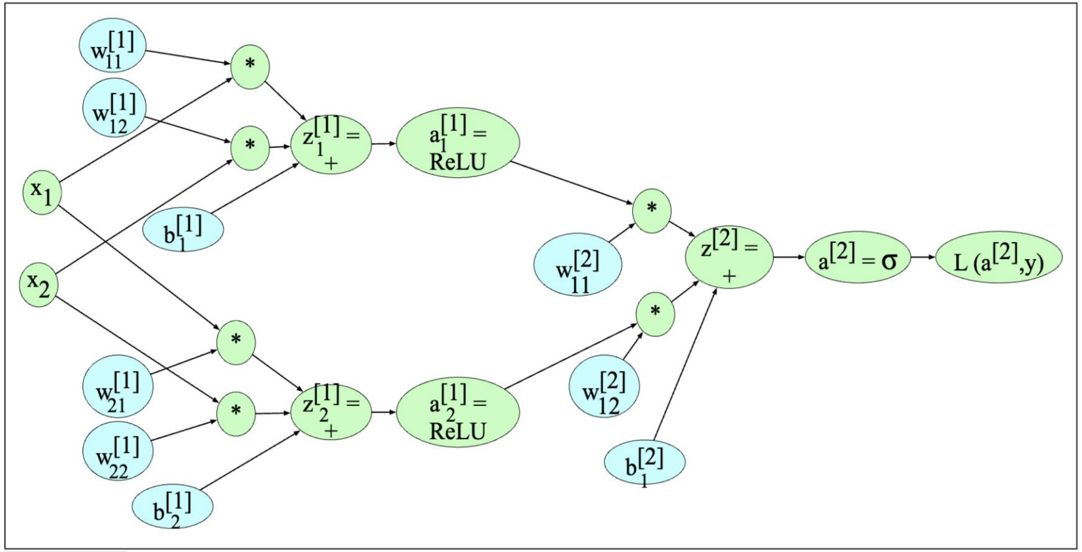
Partial Derivatives: from Math to ML

Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



Partial Derivatives: from Math to ML

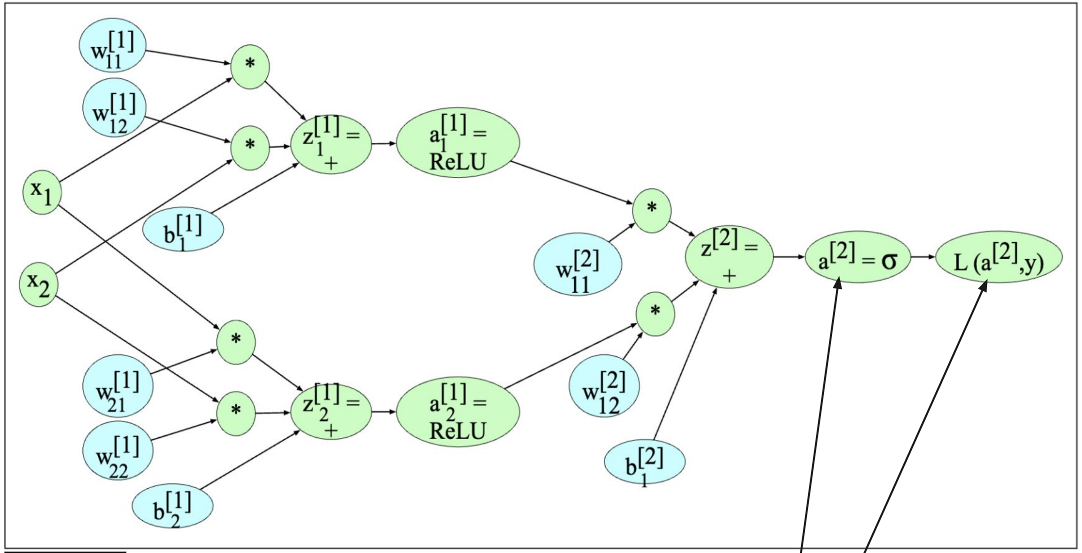
Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform FORWARD PASS
    - What do you get as a result

Partial Derivatives: from Math to ML

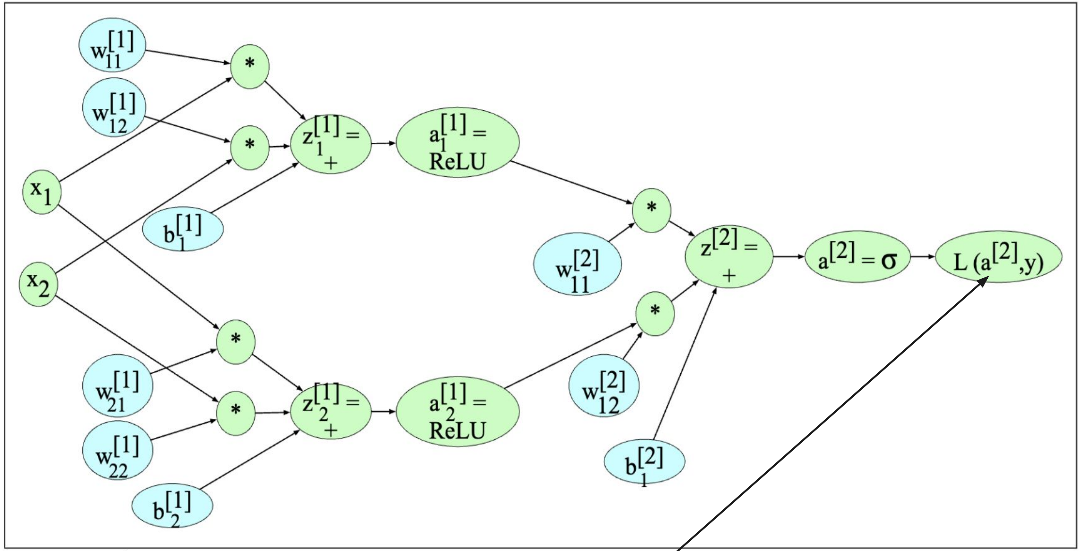
Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform FORWARD PASS
    - As as result, you get the predicted value

This $a^{[2]}$ can also be written as the predicted value $\hat{y}$

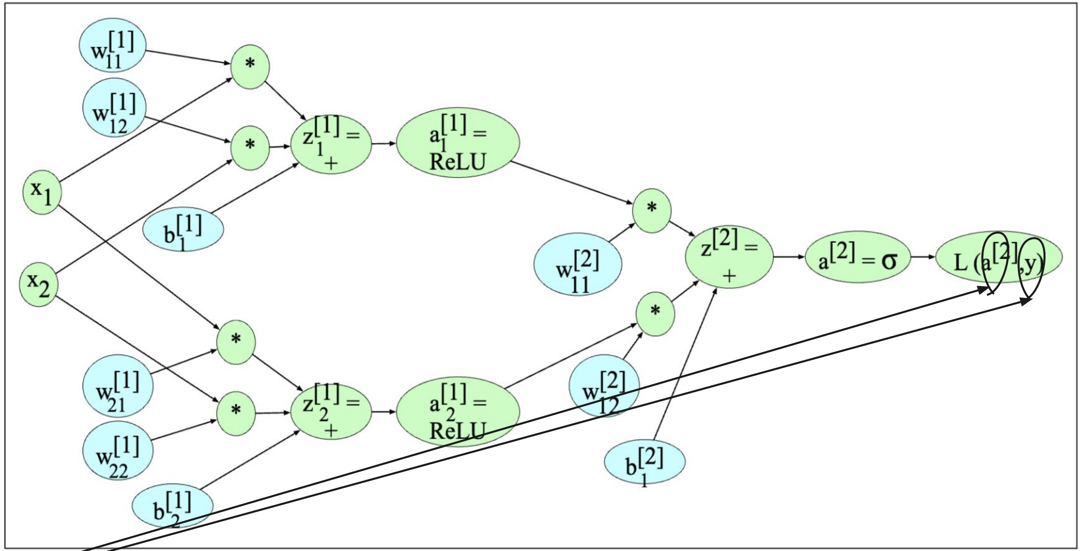
Partial Derivatives: from Math to ML



Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?

- Using inputs and weights and biases, perform FORWARD PASS, obtain ŷ
- What next?

Partial Derivatives: from Math to ML

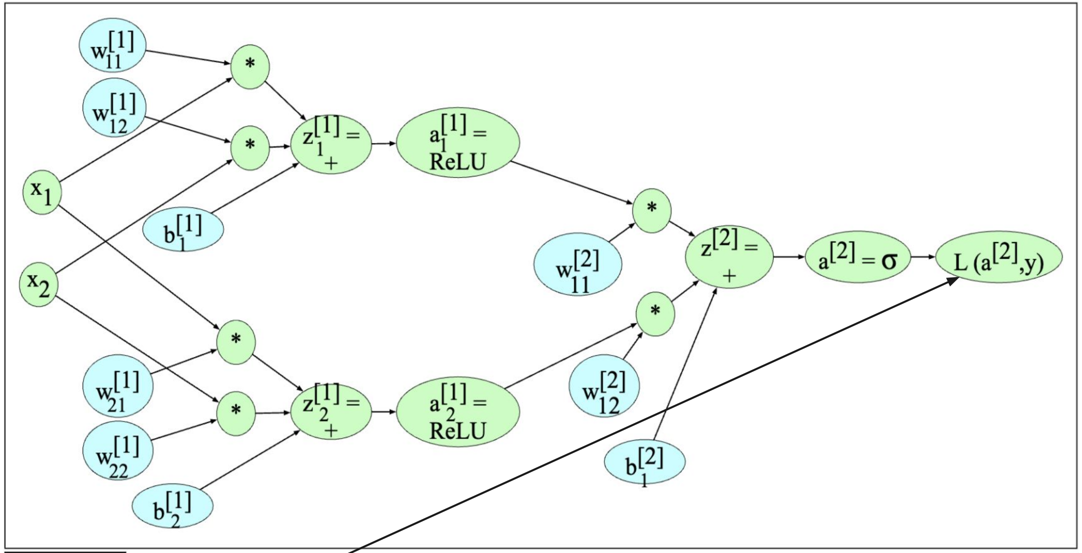


Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result
    - How to compare?

Partial Derivatives: from Math to ML

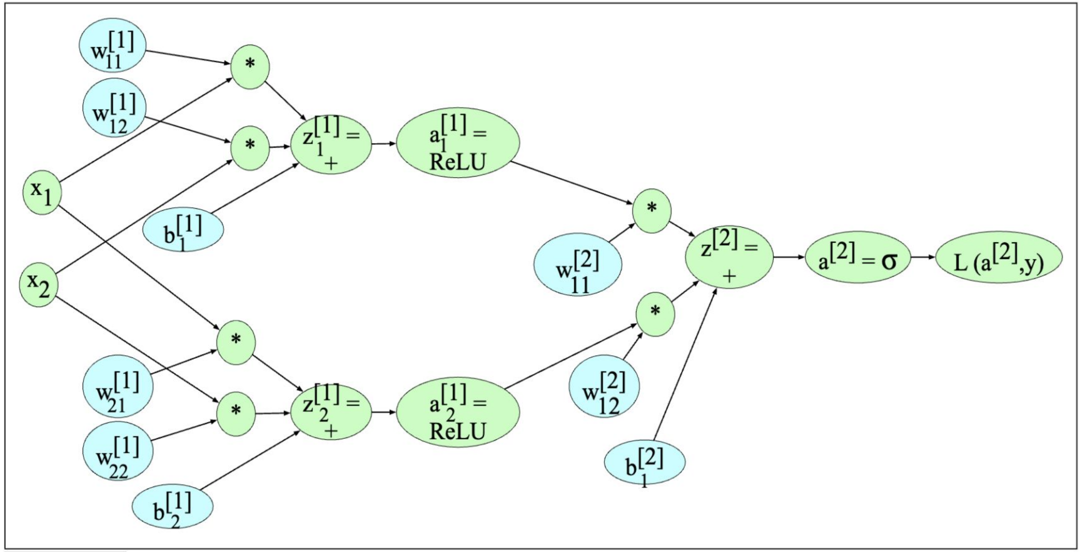
Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result
    - How to compare?
        - Loss function!
            - Suppose your ŷ (or $a^{[2]}$) is 0.886, your y is 0
            - You use a binary cross entropy loss
            - You get some number (2.2)
            - This is your loss!

$$L_{CE}(a^{[2]}, y) = -\left[y \log a^{[2]} + (1-y)\log(1-a^{[2]})\right]$$

Partial Derivatives: from Math to ML



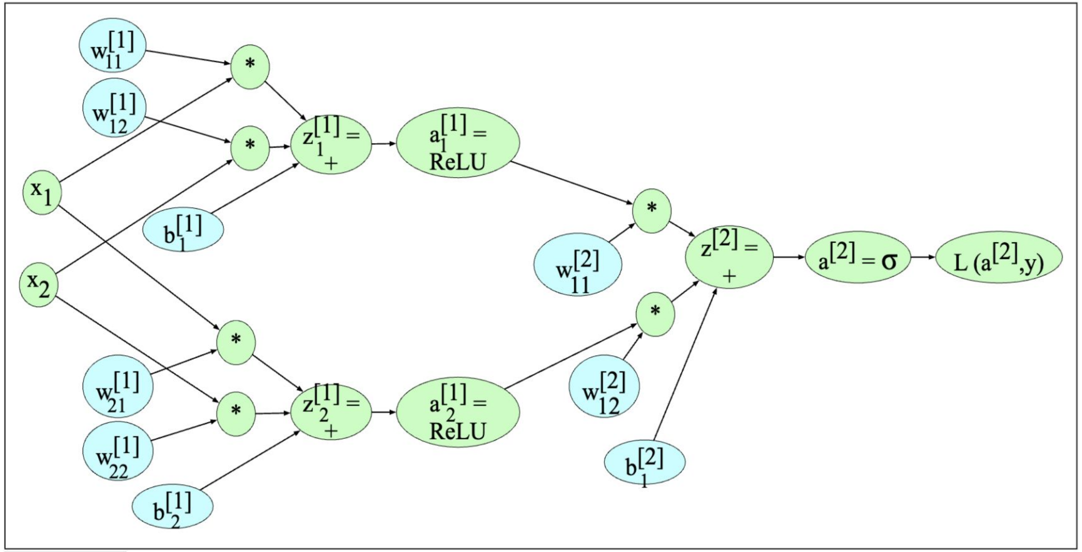
Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
  - Recall how is it written?

$$L_{CE}(a^{[2]}, y)$$

More generally (not just cross entropy loss, but just a L for loss)
More specific (expand $a^{[2]}$ as a function of ?)
Q: how to expand

Partial Derivatives: from Math to ML



Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?

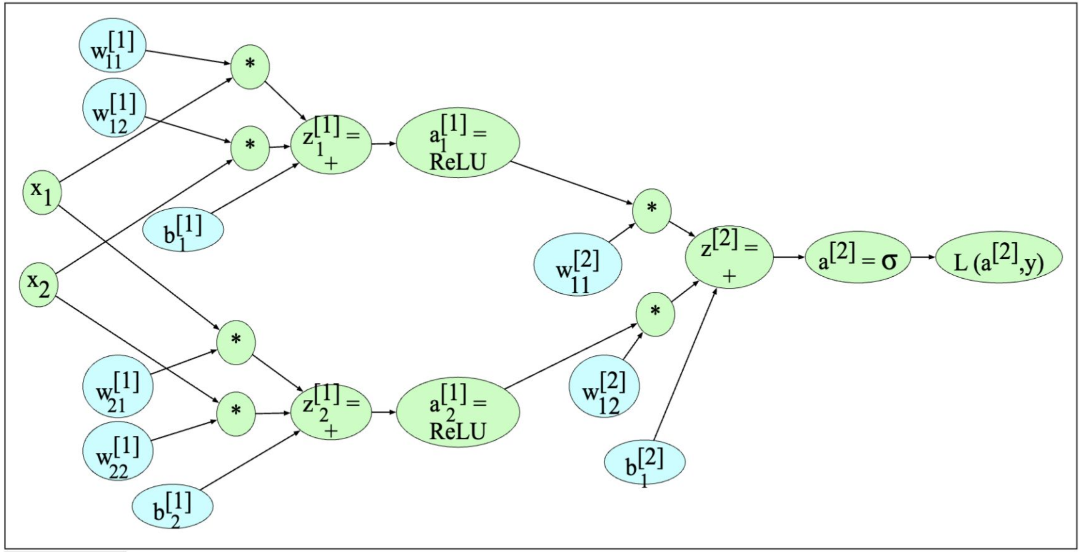
- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
    - Recall how is it written?

$$L_{CE}(a^{[2]}, y)$$

More generally (as L)
More specific (expand $a^{[2]}$ as a function of x, w and b)

Partial Derivatives: from Math to ML

Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
  - Recall how is it written?

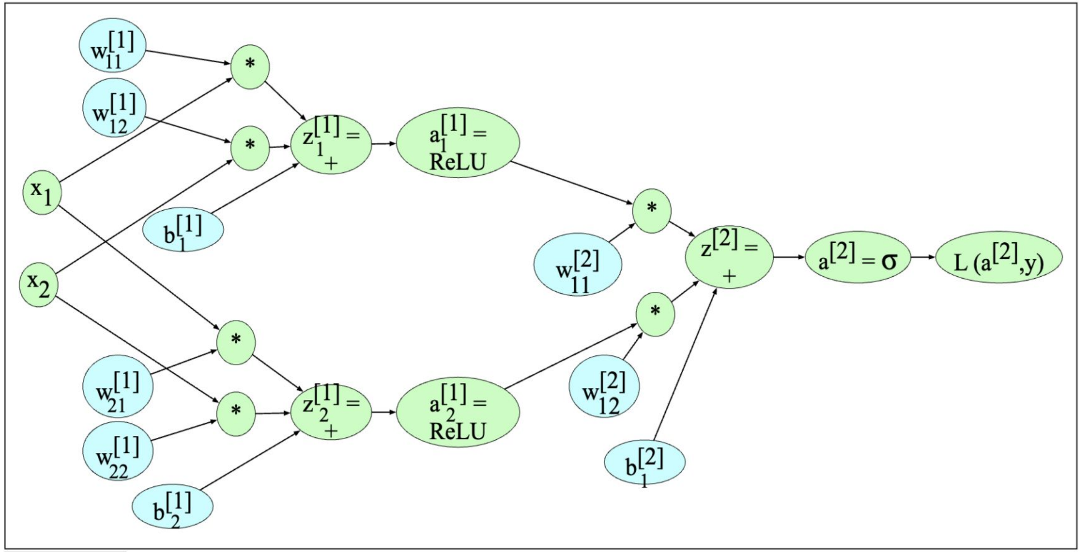
$$L_{CE}(a^{[2]}, y)$$

More generally (as L)
More specific (expand $a^{[2]}$ as a function of x, w and b)

$$\theta = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

Partial Derivatives: from Math to ML

Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
    - Recall how is it written?

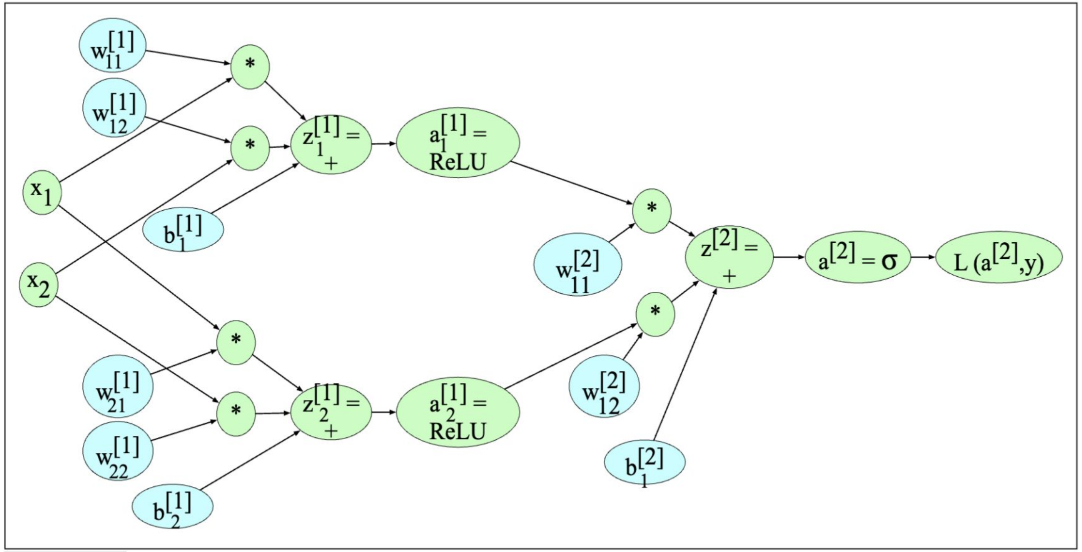
$$L_{CE}(a^{[2]}, y)$$

More generally (as L)
More specific (expand $a^{[2]}$ as a function f of x and θ)

$$\theta = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

Partial Derivatives: from Math to ML

Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



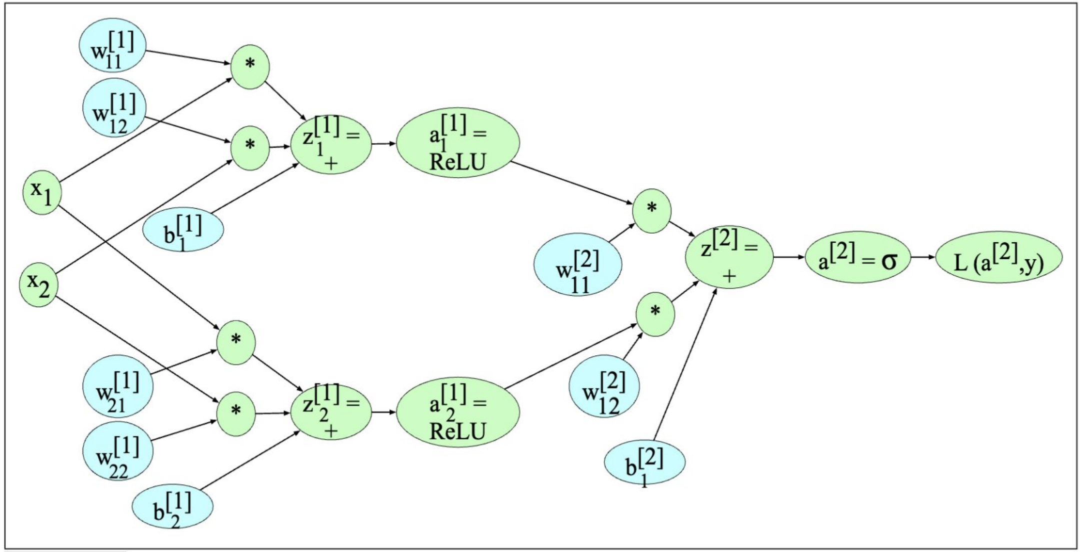
- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
    - Recall how is it written?

$$L_{CE}(a^{[2]}, y)$$

More generally (as L)
More specific (expand $a^{[2]}$ as $f(x,\theta)$)

Partial Derivatives: from Math to ML

Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
    - Recall how is it written?

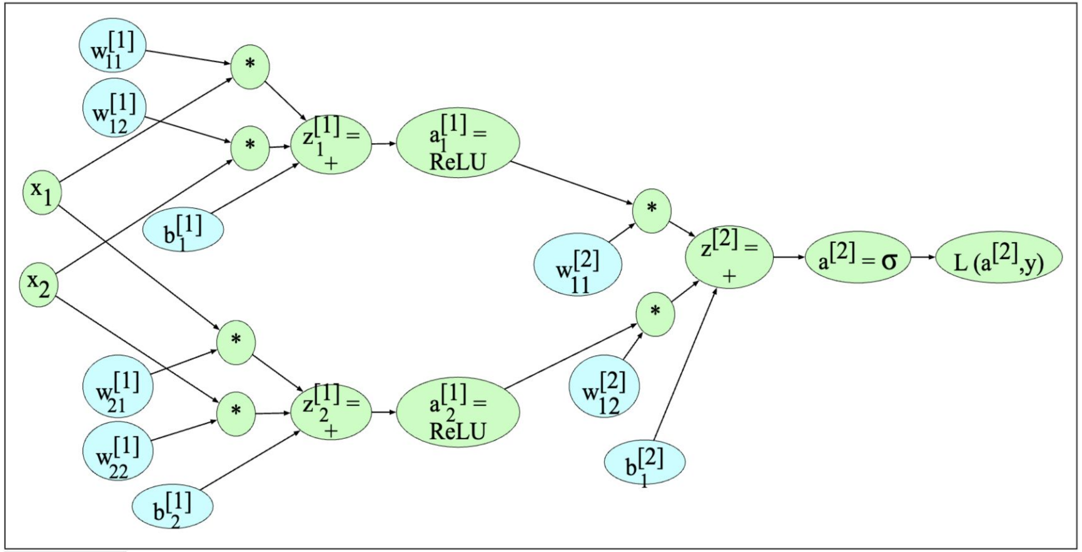
$$L_{CE}(a^{[2]}, y)$$

More generally (as L)
More specific (expand $a^{[2]}$ as $f(x;\theta)$)

A semicolon is equivalent to a comma, but a semicolon is used to make some differentiation, and it means that things belong to different types. In this case we have two types: input x and parameters θ.

Partial Derivatives: from Math to ML

Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
  - Recall how is it written?

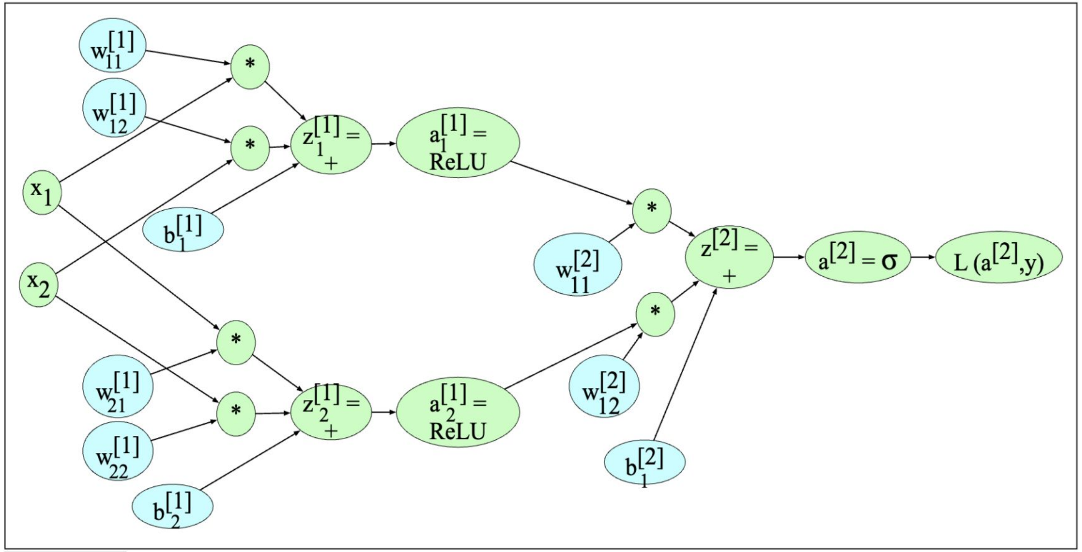
$$L_{CE}(a^{[2]}, y)$$

f(x1, x2, ~~x3, x4, w1, w2, w3~~, b1, b2, b3)

More generally (as L)
More specific (expand $a^{[2]}$ as $f(x;\theta)$)

A semicolon is equivalent to a comma, but a semicolon is used to make some differentiation, and it means that things belong to different types. In this case we have two types: input x and parameters θ.

Partial Derivatives: from Math to ML



Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a Loss Function to compute the loss
  - Recall how is it written?

$$L_{CE}(a^{[2]}, y)$$

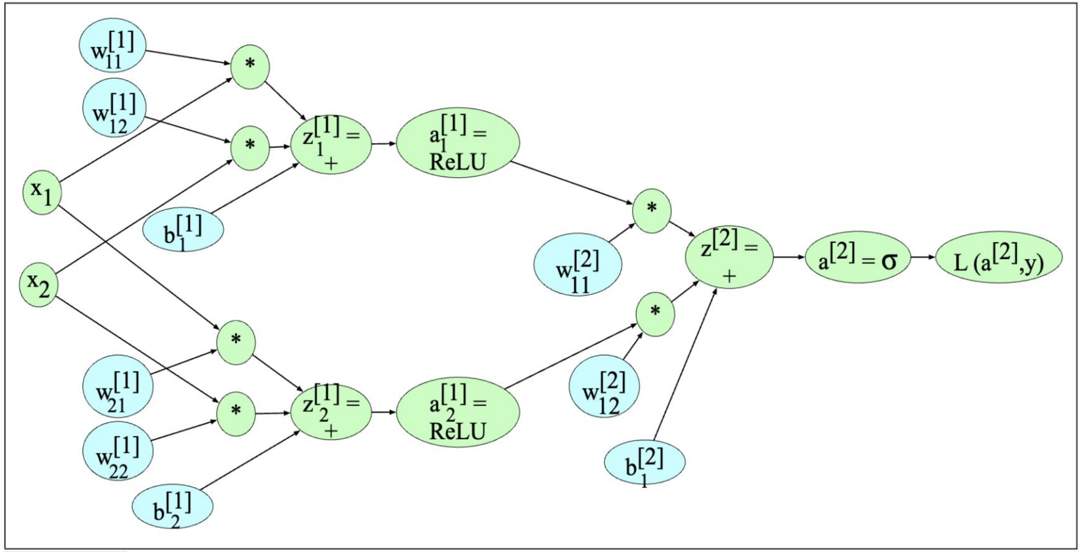
More generally (as L)
More specific (expand $a^{[2]}$ as $f(x;\theta)$)

$$L(f(x;\theta), y)$$

A semicolon is equivalent to a comma, but a semicolon is used to make some differentiation, and it means that things belong to different types. In this case we have two types: input x and parameters θ.

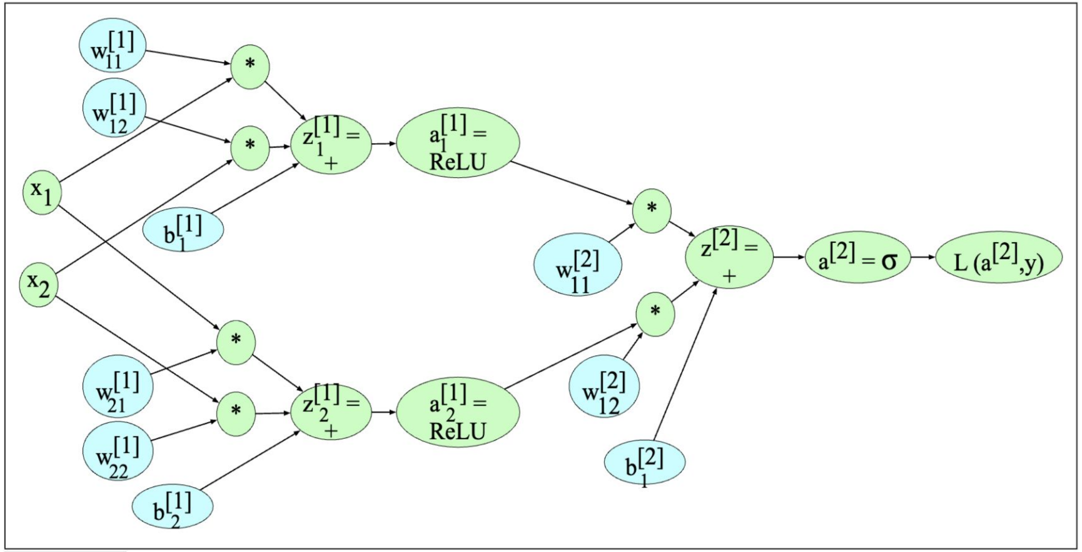
Partial Derivatives: from Math to ML

Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Then what..?

Partial Derivatives: from Math to ML



Suppose you have inputs
You have initialized weights and biases
You have the target result
What you should do to train this model?

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES!
    - This is where the partial derivatives comes in
    - Also the nabla and what have talked about last week

$$L(f(x;\theta),y)$$

Image source: SpongeBob episode "Sir Urchin and Snail Fail" in Season 13

Partial derivatives

How to write the partial derivatives for: $f(x, y) = 3x^2y$

The partial derivative with respect to x

The partial derivative with respect to y

Partial derivatives

$f(x, y) = 3x^2y$

The partial derivative with respect to x is written $\frac{\partial}{\partial x}3x^2y$

The partial derivative with respect to y is written $\frac{\partial}{\partial y}3x^2y$

Partial derivatives walkthrough

$f(x, y) = 3x^2y$

~~The partial derivative with respect to x is~~ 6xy

~~The partial derivative with respect to y is~~ $3x^2$

How about no text, just symbols...

$$\boxed{\begin{aligned}\frac{\partial}{\partial x}3x^2y &= 6xy\\ \frac{\partial}{\partial y}3x^2y &= 3x^2\end{aligned}}$$

$$\left[\frac{\partial}{\partial x}3x^2y \quad \frac{\partial}{\partial y}3x^2y\right]$$

Partial derivatives walkthrough

$$\begin{bmatrix} \frac{\partial}{\partial x} 3x^2 y & \frac{\partial}{\partial y} 3x^2 y \end{bmatrix} = \begin{bmatrix} 6xy & 3x^2 \end{bmatrix}$$

Partial derivatives walkthrough

$$\left[\frac{\partial}{\partial x}\boxed{3x^2y} \quad \frac{\partial}{\partial y}\boxed{3x^2y}\right] = \left[6xy \quad 3x^2\right]$$

Now let's write more generally

Partial derivatives walkthrough

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x}\boxed{f(x,y)} & \frac{\partial}{\partial y}\boxed{f(x,y)} \end{bmatrix} = \begin{bmatrix} 6xy & 3x^2 \end{bmatrix}$$

Done! We represented it by f(x,y)

Partial derivatives walkthrough

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) & \frac{\partial}{\partial y} f(x,y) \end{bmatrix} = \begin{bmatrix} 6xy & 3x^2 \end{bmatrix}$$

nabla symbol

Partial derivatives walkthrough

$$\boxed{\nabla f(x, y)} = \left[\frac{\partial}{\partial x} f(x, y) \quad \frac{\partial}{\partial y} f(x, y)\right] = \left[6xy \quad 3x^2\right]$$

Denotes the gradient
(the gradient in “gradient descent”)

2.4 A Brief Note on Numerator Layout vs Denominator Layout

There are two different layouts to express vector/matrix derivatives, namely the numerator and the denominator layout. In this course, we use the **denominator layout**. These layouts are mostly the same and can easily be switched using transpose operations. To demonstrate this better, some examples are shown below:

	Numerator Layout	**Denominator Layout**
$\frac{\partial y}{\partial \mathbf{x}}$	1-D row vector	1-D column vector
$\frac{\partial \mathbf{y}}{\partial x}$	1-D column vector	1-D row vector
$\frac{\partial \mathbf{a}^T \mathbf{z}}{\partial \mathbf{z}}$	$\mathbf{a}^T$	$\mathbf{a}$
$\frac{\partial \mathbf{M}\mathbf{z}}{\partial \mathbf{z}}$	$\mathbf{M}$	$\mathbf{M}^T$

A handy way to distinguish numerator vs denominator layout is to remember that **the layout type corresponds the number of rows in the output matrix**. In a numerator layout, the output matrix has number of rows equal to the size of the numerator, while in a denominator layout, the output matrix has number of rows equal to the size of the denominator.

https://www.cs.cmu.edu/~aarti/Class/10315_Spring22/315S22_Rec4.pdf

Partial derivatives walkthrough

To align with course materials: we write gradients in the format of column vectors

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$$

Partial derivatives walkthrough

To align with course materials: we write gradients in the format of column vectors

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$$

Note: for the dimensionalities in the exercises / exams - there is no such flexibility.
There is only one single answer because the dimensionalities must be matched.

Partial derivatives walkthrough

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$$

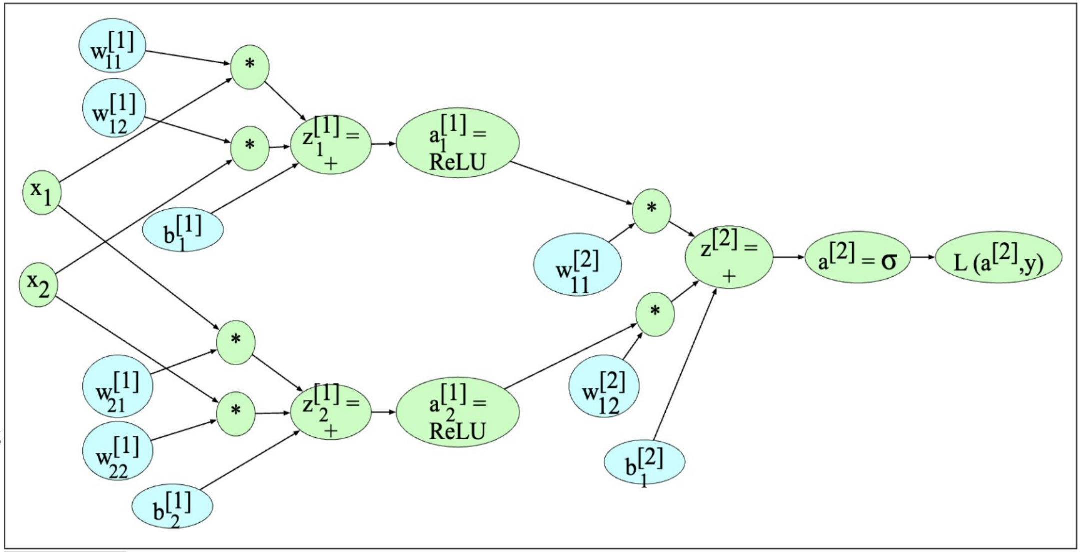
What we did?

- compute the partial derivatives
- put them all together

Now
come back
to today's slides

Partial Derivatives: from Math to ML

We also want to compute the partial derivatives
We also want to put them all together

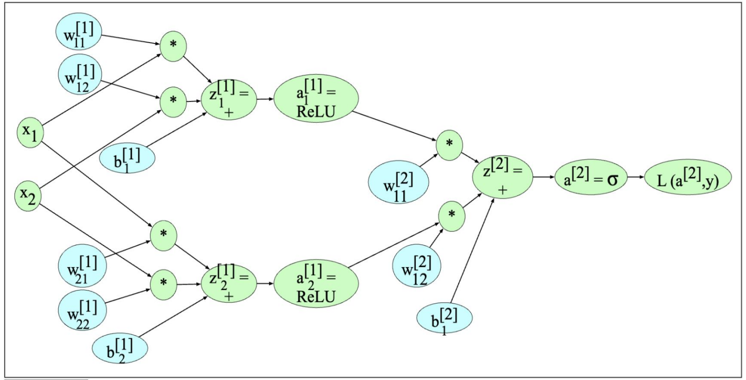


- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss

$$L(f(x;\theta),y)$$

- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES!
    - Combine them in a column vector

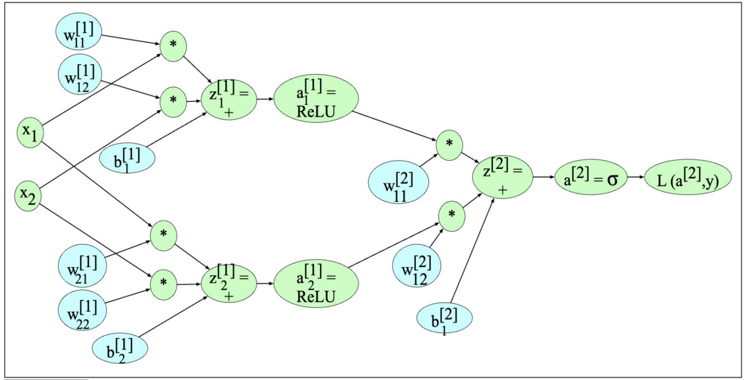
Partial Derivatives: from Math to ML



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES!
    - Combine them in a column vector

$$\begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \\ \frac{\partial}{\partial b} L(f(x;\theta),y) \end{bmatrix}$$

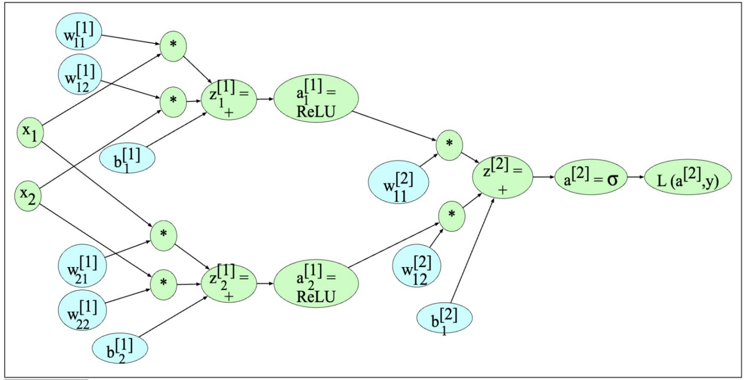
Partial Derivatives: from Math to ML



- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES!
    - Combine them in a column vector
    - Recall: this is also called a gradient
        - Use our nabla symbol

$$\begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \\ \frac{\partial}{\partial b} L(f(x;\theta),y) \end{bmatrix}$$

Partial Derivatives: from Math to ML

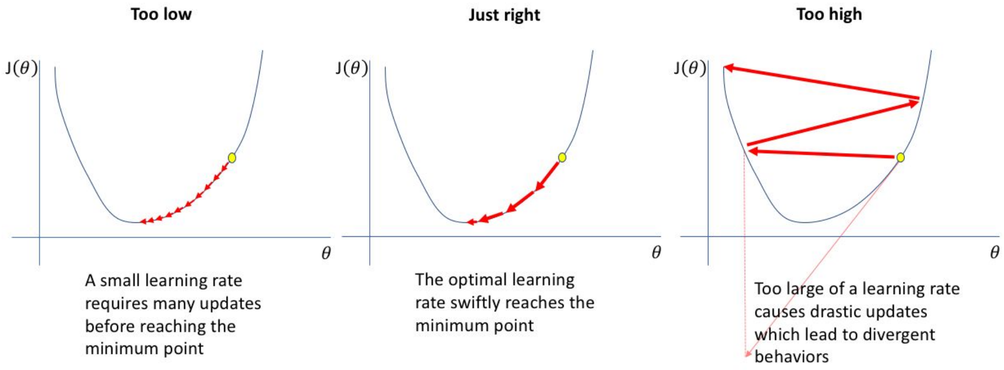


- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES!
  - Combine them in a column vector
  - Recall: this is also called a gradient
    - Use our nabla symbol

$$\nabla L(f(x;\theta),y) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \\ \frac{\partial}{\partial b} L(f(x;\theta),y) \end{bmatrix}$$

Partial Derivatives: from Math to ML

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES! $\nabla L(f(x;\theta),y)$
- Use a learning rate η to control 'how much to learn/go down'
    - How to update the previous parameter θ^t? (t: timestep)



Partial Derivatives: from Math to ML

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES! $\nabla L(f(x;\theta),y)$

Partial Derivatives: from Math to ML

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES! $\nabla L(f(x;\theta),y)$
- Then what…?

Partial Derivatives: from Math to ML

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES! $\nabla L(f(x;\theta),y)$
- Use a learning rate η to control 'how much to learn/go down'
    - How to update the previous parameter θ^t?

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta),y)$$

Partial Derivatives: from Math to ML

- Using inputs and weights and biases, perform forward pass, obtain ŷ
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES! $\nabla L(f(x;\theta),y)$
- Use a learning rate η to control 'how much to learn/go down' (update the previous parameter θ^t)

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta),y)$$

Partial Derivatives: from Math to ML

- Plug in all the data, perform forward pass
- Compare it to y, the real target result, use a loss function to compute the loss $L(f(x;\theta),y)$
- Compute the partial derivatives with regard to ALL WEIGHTS and BIASES! $\nabla L(f(x;\theta),y)$
- Use a learning rate η to control 'how much to learn/go down' (update the previous parameter θ^t)

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta),y)$$

Partial Derivatives: from Math to ML

- Plug in all the data, perform forward pass
- Compute the loss $L(f(x;\theta),y)$
- Compute the partial de $\nabla L(f(x;\theta),y)$ regard to ALL WEIGHTS and BIASES!
- Use a learning rate η to control 'how much to learn/go down' (update the previous parameter θ^t)

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta),y)$$

Partial Derivatives: from Math to ML

- Plug in all the data, perform forward pass
- Compute the loss $L(f(x;\theta),y)$
- Compute the gradients $\nabla L(f(x;\theta),y)$
- Use a learning rate η to control 'how much to learn/go down' (update the previous parameter θ^t) $\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta),y)$

Partial Derivatives: from Math to ML

- Plug in all the data, perform forward pass
- Compute the loss $L(f(x;\theta),y)$
- Compute the gradients $\nabla L(f(x;\theta),y)$
- Update the parameters $\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta),y)$

🎉We have just revisited slide 19 🎉

Revisit the update rule

$$\theta = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \qquad \boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \eta \nabla L(f(x;\boldsymbol{\theta}), y)$$

$$\nabla L(f(x;\boldsymbol{\theta}), y) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\boldsymbol{\theta}), y) \\ \frac{\partial}{\partial w_2} L(f(x;\boldsymbol{\theta}), y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\boldsymbol{\theta}), y) \\ \frac{\partial}{\partial b} L(f(x;\boldsymbol{\theta}), y) \end{bmatrix}$$

$$\frac{\partial L_{CE}(\hat{y}, y)}{\partial w_j} = \left(\sigma(\mathbf{w}^T\mathbf{x} + b) - y\right) x_j = -(\hat{y} - y)x_j$$

- Plug **all** the data
- Compute the loss
- Compute the gradients
- Update parameters. Repeat until loss can no longer be further minimized

Optional derivative exercises with answers

Partial derivatives(1)

- $f(x,y) = 2x - 5y + 3$
- $f(x,y) = x^2 - 2y^2 + 4$
- $f(x,y) = x^2y^3$
- $f(x,y) = 4x^3y^{-2}$
- $z = x\sqrt{y}$
- $z = 2y^2\sqrt{x}$
- $z = x^2 - 4xy + 3y^2$
- $z = y^3 - 2xy^2 - 1$

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	2x
Square Root	√x	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\, a^x$
Logarithms	ln(x)	1/x
	$\log_a(x)$	1 / (x ln(a))

Rules	Function	Derivative
Multiplication by constant	cf	cf′
Power Rule	x^n	nx^{n-1}

Image source: see week 1

Partial derivatives(1)

- $f(x,y) = 2x - 5y + 3$
- $f(x,y) = x^2 - 2y^2 + 4$
- $f(x,y) = x^2y^3$
- $f(x,y) = 4x^3y^{-2}$
- $z = x\sqrt{y}$
- $z = 2y^2\sqrt{x}$
- $z = x^2 - 4xy + 3y^2$
- $z = y^3 - 2xy^2 - 1$

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	2x
Square Root	√x	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\ a^x$
Logarithms	ln(x)	1/x
	$\log_a(x)$	1 / (x ln(a))

Rules	Function	Derivative
Multiplication by constant	cf	cf′
Power Rule	x^n	nx^{n-1}

Image source: see week 1

Solutions(1)

- $f(x,y) = 2x - 5y + 3$
- $f(x,y) = x^2 - 2y^2 + 4$
- $f(x,y) = x^2y^3$
- $f(x,y) = 4x^3y^{-2}$
- $z = x\sqrt{y}$
- $z = 2y^2\sqrt{x}$
- $z = x^2 - 4xy + 3y^2$
- $z = y^3 - 2xy^2 - 1$

1. $\frac{\partial f}{\partial x} = 2 \; \frac{\partial f}{\partial y} = -5$
2. $\frac{\partial f}{\partial x} = 2x \; \frac{\partial f}{\partial y} = -4y$
3. $\frac{\partial f}{\partial x} = 2xy^3 \; \frac{\partial f}{\partial y} = 3x^2y^2$
4. $\frac{\partial f}{\partial x} = 12x^2y^{-2} \; \frac{\partial f}{\partial y} = -8x^3y^{-3}$
5. $\frac{\partial z}{\partial x} = \sqrt{y} \; \frac{\partial z}{\partial y} = \frac{x}{2\sqrt{y}}$
6. $\frac{\partial z}{\partial x} = \frac{y^2}{\sqrt{x}} \; \frac{\partial z}{\partial y} = 4y\sqrt{x}$
7. $\frac{\partial z}{\partial x} = 2x - 4y \; \frac{\partial z}{\partial y} = -4x + 6y$
8. $\frac{\partial z}{\partial x} = -2y^2 \; \frac{\partial z}{\partial y} = 3y^2 - 4xy$

Partial derivatives with chain/product rules(2)

- $z = e^{xy}$
- $z = e^{x/y}$
- $z = x^2 e^{2y}$
- $z = y e^{y/x}$

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	2x
Square Root	√x	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\, a^x$
Logarithms	ln(x)	1/x
	$\log_a(x)$	1 / (x ln(a))

Rules	Function	Derivative
Multiplication by constant	cf	cf’
Power Rule	x^n	nx^{n-1}
Sum Rule	f + g	f’ + g’
Difference Rule	f - g	f’ − g’
Product Rule	fg	f g’ + f’ g
Chain Rule (as "Composition of Functions")	f º g	(f’ º g) × g’
Chain Rule (using ’)	f(g(x))	f’(g(x))g’(x)
Chain Rule (using $\frac{d}{dx}$)	$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$	

Solutions(2)

• $z = e^{xy}$	9. $\frac{\partial z}{\partial x} = ye^{xy}$ $\frac{\partial z}{\partial y} = xe^{xy}$
• $z = e^{x/y}$	10. $\frac{\partial z}{\partial x} = \frac{1}{y}e^{x/y}$ $\frac{\partial z}{\partial y} = -\frac{x}{y^2}e^{x/y}$
• $z = x^2e^{2y}$	11. $\frac{\partial z}{\partial x} = 2xe^{2y}$ $\frac{\partial z}{\partial y} = 2x^2e^{2y}$
• $z = ye^{y/x}$	12. $\frac{\partial z}{\partial x} = -\frac{y^2}{x^2}e^{y/x}$ $\frac{\partial z}{\partial y} = e^{y/x} + \frac{y}{x}e^{y/x}$