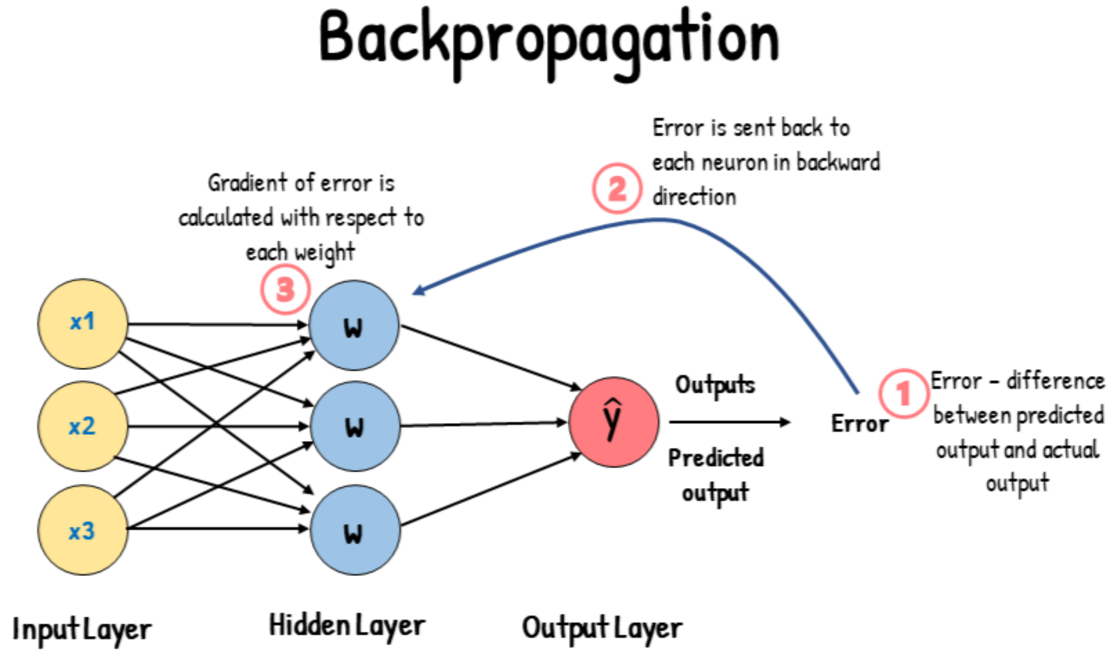


Introduction to Optimisation for ML

Derivatives and Gradients

Andreas Makris, 2nd year PhD student

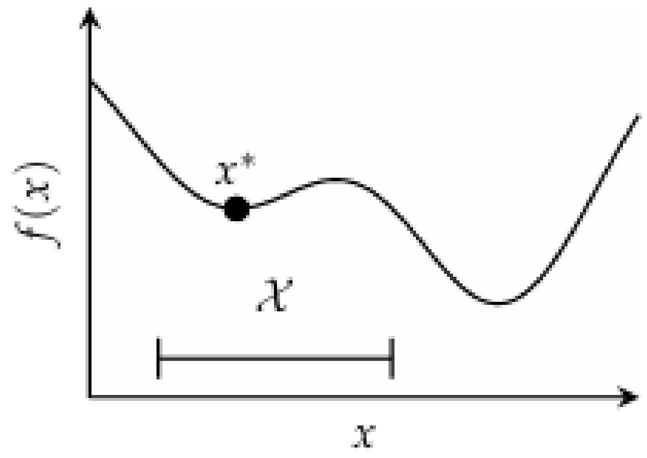
Lancaster University, ProbAI Hub

Based on chapter 2 of the book **Algorithms for Optimization** by Mykel J. Kochenderfer Tim A. Wheeler

What is optimization?

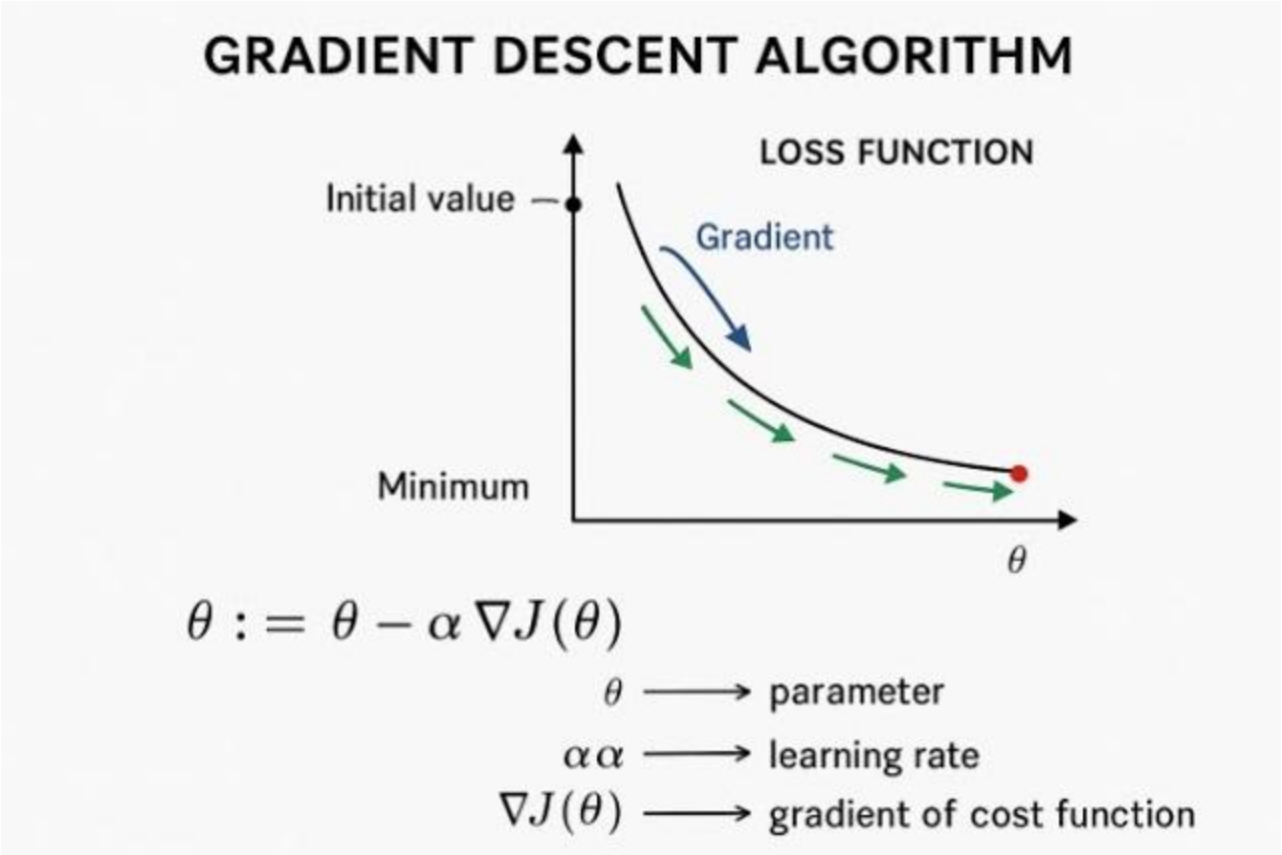
- We have a function f that depends on some input x. We want to find x that minimizes f subject to some constrain. Mathematically:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned}$$



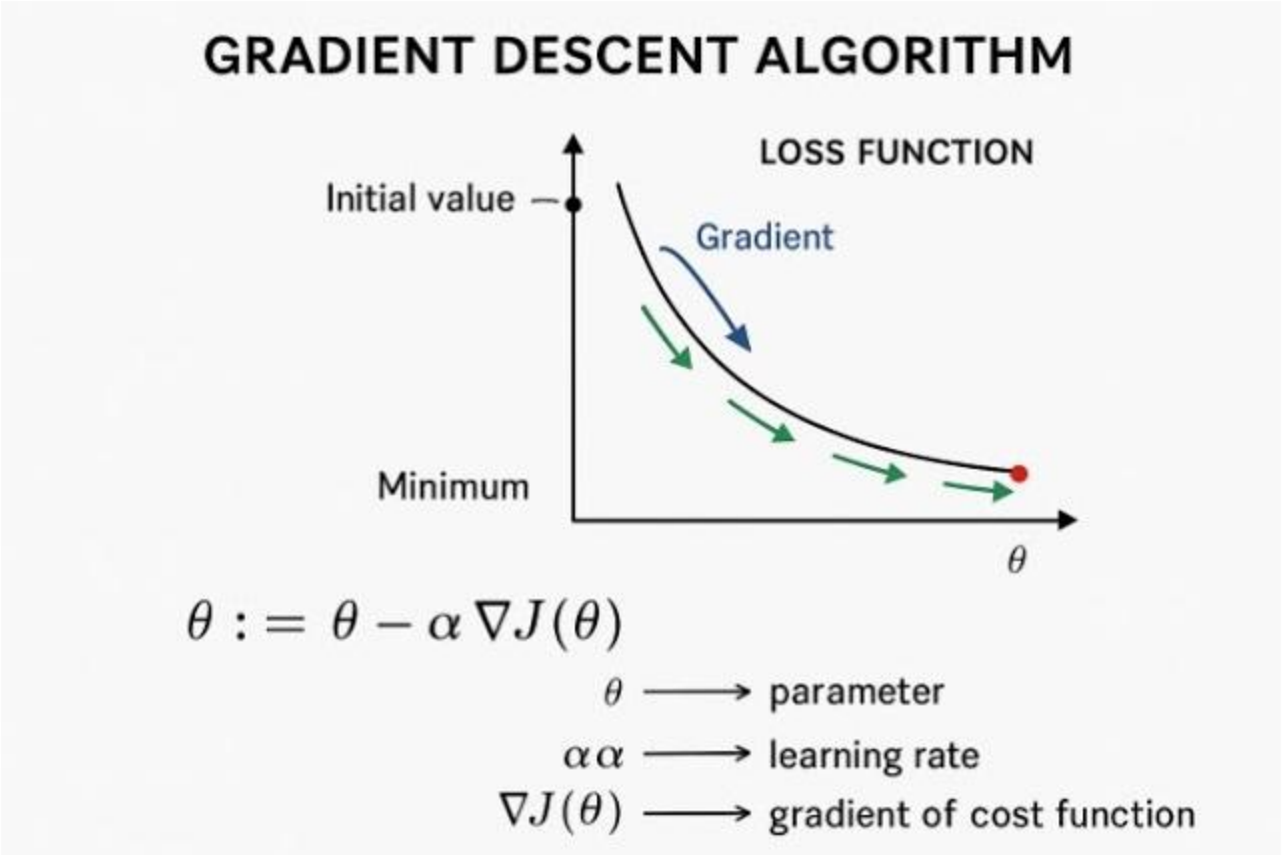
Why do we need optimization in ML?

- The function is the loss function J. The input are the parameters of the model θ.



Why do we need optimization in ML?

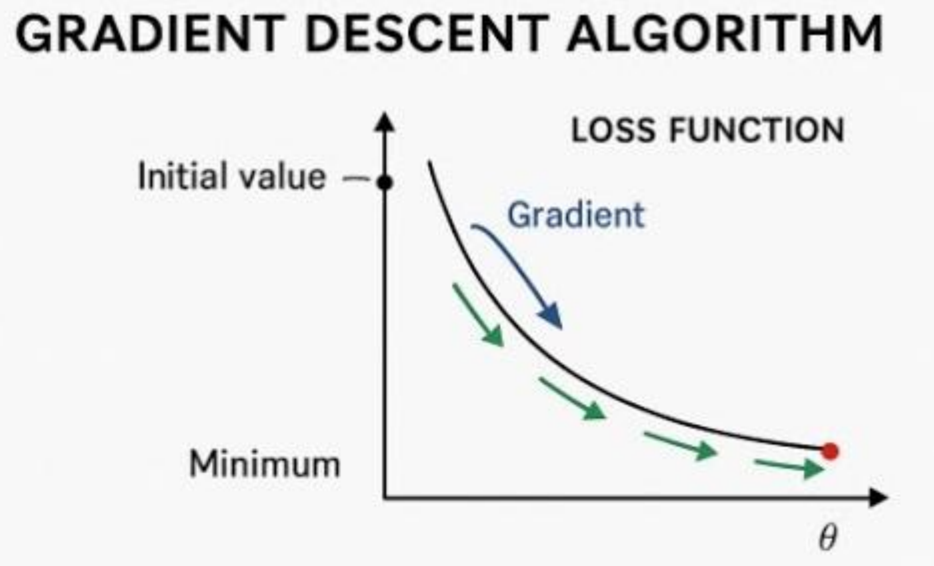
- The function is the loss function J. The input are the parameters of the model θ.
- We want to find the parameters of the model that minimize the loss function.



Why do we need optimization in ML?

- The function is the loss function J. The input are the parameters of the model θ.
- We want to find the parameters of the model that minimize the loss function.
- There are a lot of optimization algorithms that use the gradient of the function with respect to the input (e.g. gradient descent, ADAM).
- Today we will focus on how to calculate the gradient of the loss with respect to the parameters of the model.

$$\frac{\partial J}{\partial \theta}$$



$$\theta := \theta - \alpha \nabla J(\theta)$$

$\theta \longrightarrow$ parameter

$\alpha\alpha \longrightarrow$ learning rate

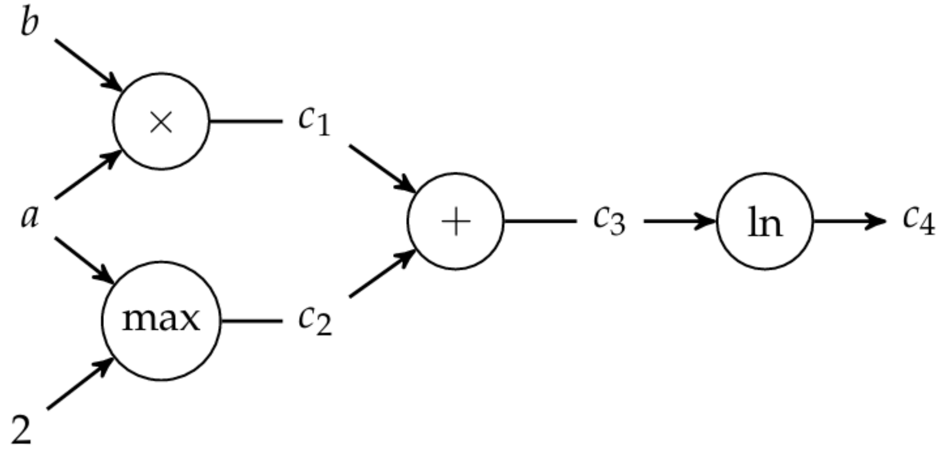
$\nabla J(\theta) \longrightarrow$ gradient of cost function

https://www.youtube.com/watch?v=30IMyrQo53g

Automatic Differentiation

- Two types (modes) of autodiff; forward mode and reverse mode (backpropagation).

$$f(a,b) = \ln(ab + \max(a,2))$$

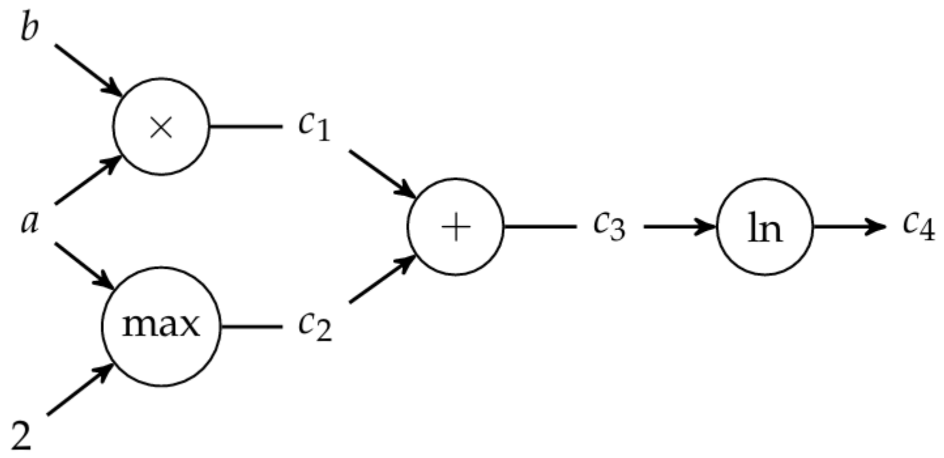


$c_4 = \ln(c_3) = \ln(ab + \max(a,2))$ $c_3 = c_1 + c_2 = ab + \mathrm{ma}$

Automatic Differentiation

- Two types (modes) of autodiff; forward mode and reverse mode (backpropagation).
- Can be used when a function can be expressed as a computation graph with all elementary functions being differentiable.

$$f(a,b) = \ln(ab + \max(a,2))$$

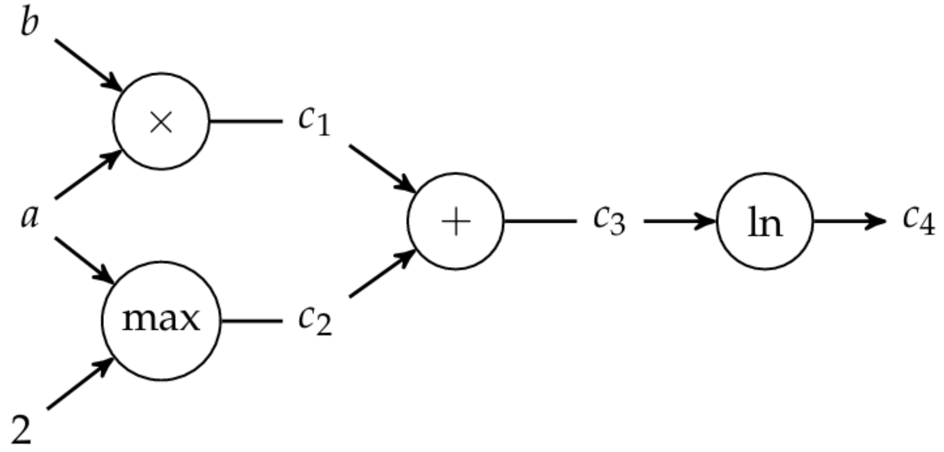


$c_4 = \ln(c_3) = \ln(ab + \max(a,2))$ $c_3 = c_1 + c_2 = ab + \mathrm{ma}$

Automatic Differentiation

- Two types (modes) of autodiff; forward mode and reverse mode (backpropagation).
- Can be used when a function can be expressed as a computation graph with all elementary functions being differentiable.
- Start by building the computation graph; inputs on the left, operations are nodes, introduce intermediate variables, outputs on the right.

$$f(a,b) = \ln(ab + \max(a,2))$$

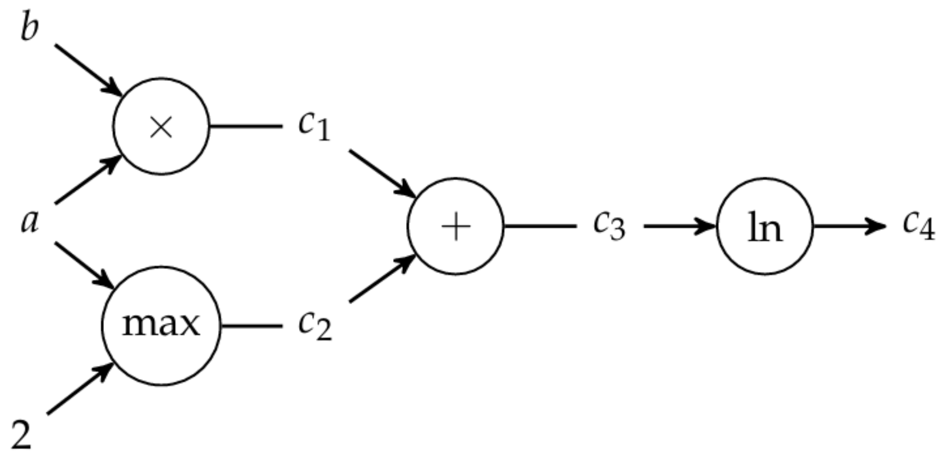


$c_4 = \ln(c_3) = \ln(ab + \max(a,2))$ $c_3 = c_1 + c_2 = ab + \mathrm{ma}$

Automatic Differentiation

- Two types (modes) of autodiff; forward mode and reverse mode (backpropagation).
- Can be used when a function can be expressed as a computation graph with all elementary functions being differentiable.
- Start by building the computation graph; inputs on the left, operations are nodes, introduce intermediate variables, outputs on the right.
- Both modes are based on the **chain rule**.
- Our goal is to calculate $\frac{\partial f}{\partial \alpha}$ (and $\frac{\partial f}{\partial b}$).

$$f(a, b) = \ln(ab + \max(a, 2))$$

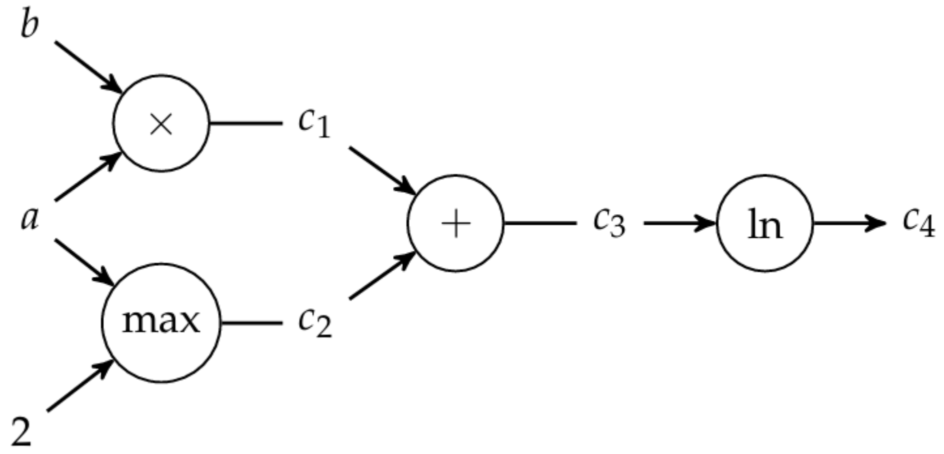


$c_4 = \ln(c_3) = \ln(ab + \max(a, 2))$ $c_3 = c_1 + c_2 = ab + \mathrm{ma}$

Automatic Differentiation

- Forward mode: Calculate in order $\frac{\partial c_1}{\partial \alpha}$, $\frac{\partial c_2}{\partial \alpha}, \frac{\partial c_3}{\partial \alpha}, \frac{\partial f}{\partial \alpha}$.

$$f(a, b) = \ln(ab + \max(a, 2))$$

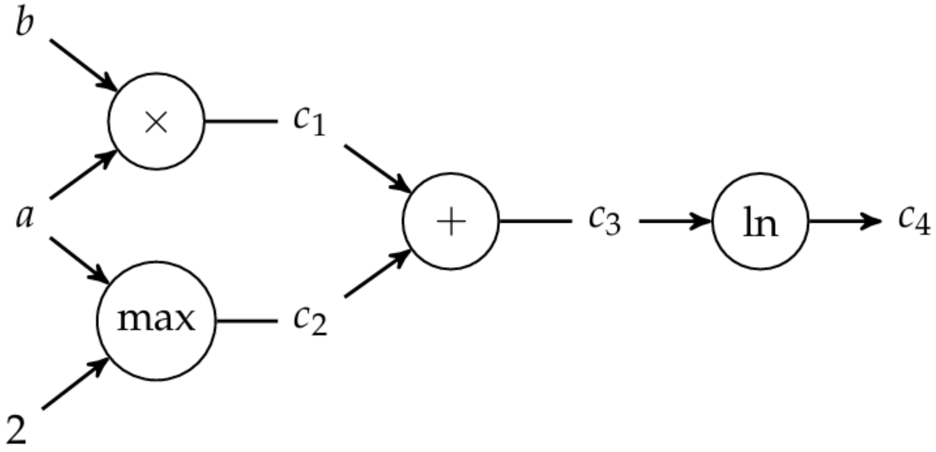


$c_4 = \ln(c_3) = \ln(ab + \max(a, 2))$ $c_3 = c_1 + c_2 = ab + \mathrm{ma}$

Automatic Differentiation

- Forward mode: Calculate in order $\frac{\partial c_1}{\partial \alpha}$, $\frac{\partial c_2}{\partial \alpha}, \frac{\partial c_3}{\partial \alpha}, \frac{\partial f}{\partial \alpha}$.
- Reverse mode: Calculate in order $\frac{\partial f}{\partial c_3}$, $\frac{\partial f}{\partial c_2}, \frac{\partial f}{\partial c_1}, \frac{\partial f}{\partial \alpha}$.

$$f(a,b) = \ln(ab + \max(a,2))$$



$$c_4 = \ln(c_3) = \ln(c_1 + c_2) = \ln(ab + \max(a,2))\ c_3 = c_1 + c_2$$

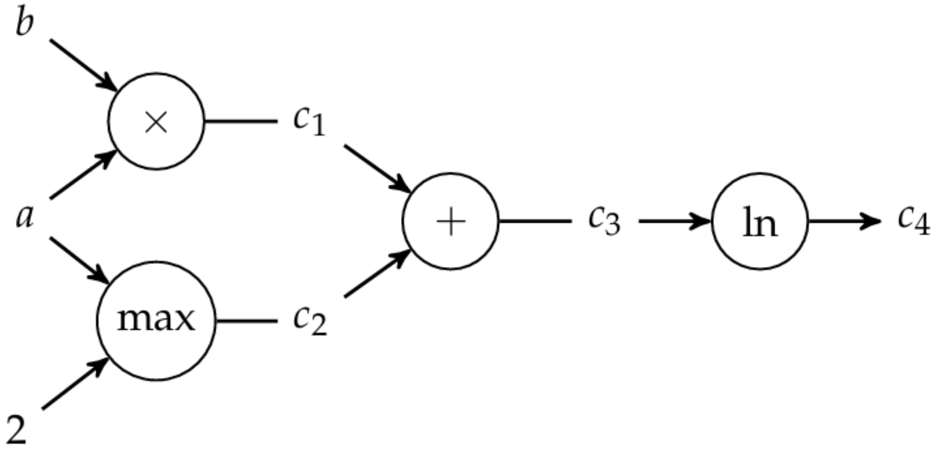
Automatic Differentiation

- Forward mode: Calculate in order $\frac{\partial c_1}{\partial \alpha}$, $\frac{\partial c_2}{\partial \alpha}$, $\frac{\partial c_3}{\partial \alpha}$, $\frac{\partial f}{\partial \alpha}$.
- Reverse mode: Calculate in order $\frac{\partial f}{\partial c_3}$, $\frac{\partial f}{\partial c_2}$, $\frac{\partial f}{\partial c_1}$, $\frac{\partial f}{\partial \alpha}$.

- When the input dimensionality is higher than the output dimensionality reverse mode is cheaper.

- When the input dimensionality is lower than the output dimensionality forward mode is cheaper.

$$f(a,b) = \ln(ab + \max(a,2))$$

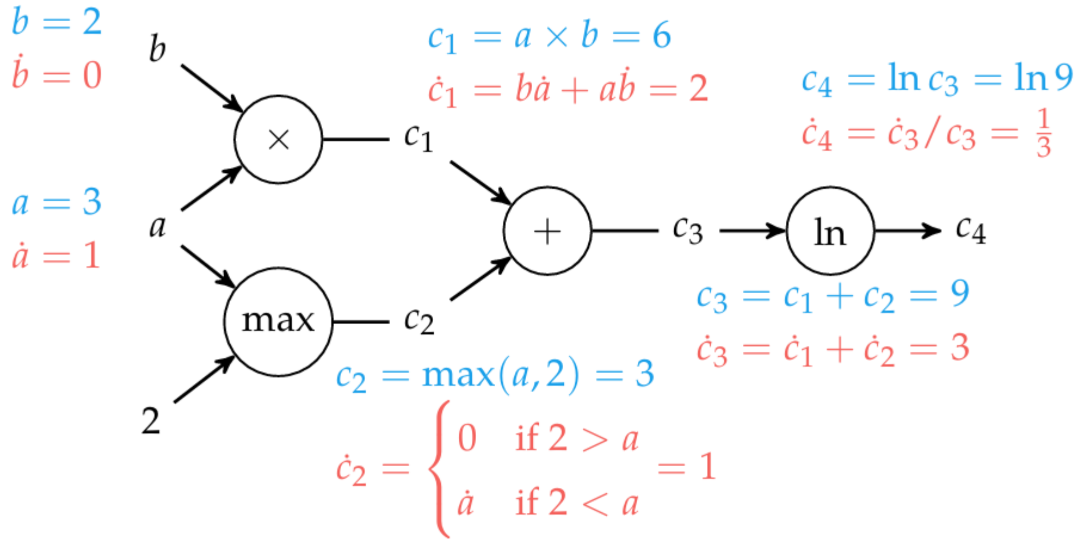


$c_4 = \ln(c_3) = \ln(ab + \max(a,2))$ $c_3 = c_1 + c_2 = ab + \text{ma}$

Forward Mode

- Let a=3 and b=2. Use the forward mode autodiff to find $\frac{\partial f}{\partial \alpha}$.

$$f(a,b) = \ln(ab + \max(a,2))$$



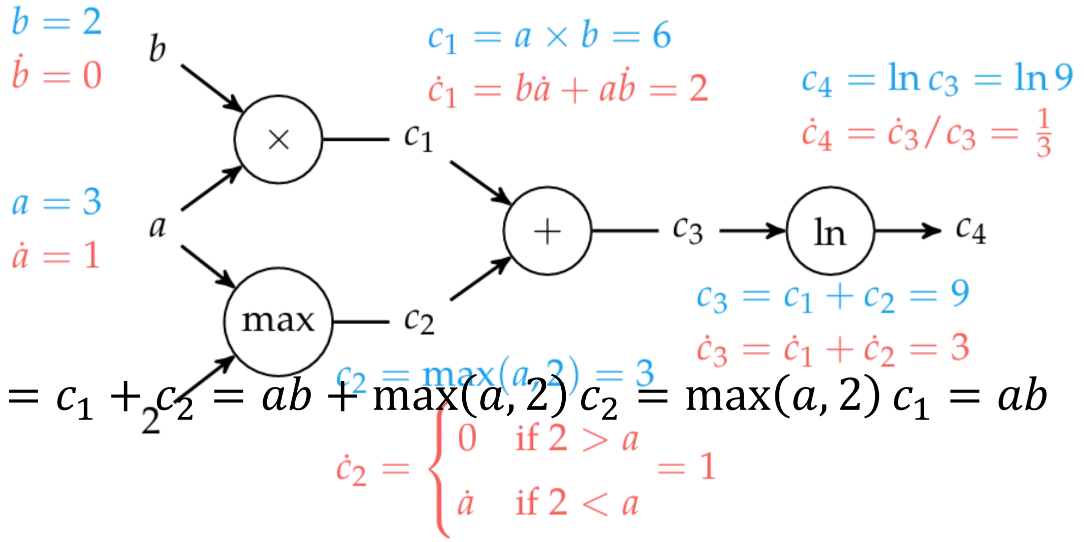
Forward Mode

- Let a=3 and b=2. Use the forward mode autodiff to find $\frac{\partial f}{\partial \alpha}$.
- For each node calculate both the value and partial derivative with respect to a.

- Example use of chain rule;

$$\frac{\partial c_3}{\partial \alpha} = \frac{\partial c_3}{\partial c_1}\frac{\partial c_1}{\partial \alpha} + \frac{\partial c_3}{\partial c_2}\frac{\partial c_2}{\partial \alpha}$$

$$f(a,b) = \ln(ab + \max(a,2))$$



$c_4 = \ln(c_3) = \ln(c_1 + c_2) = \ln(ab + \max(a,2))$ $c_3 = c_1 + c_2 = ab + \max(a,2)$ $c_2 = \max(a,2)$ $c_1 = ab$

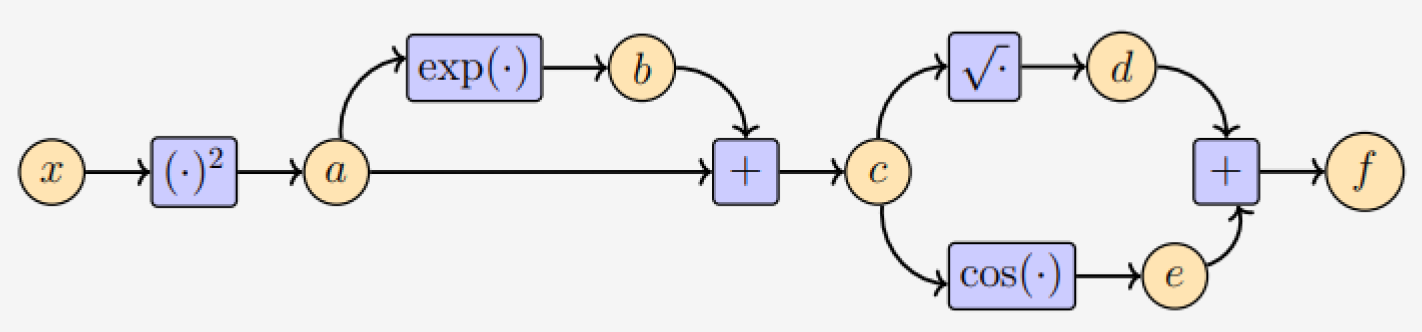
Reverse Mode

- Start by a forward pass to calculate the values (only).
- Do a reverse pass for the gradients.

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial c}$$
$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial b}$$
$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}\,.$$

- This is what neural networks use to calculate the gradients.

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right)$$

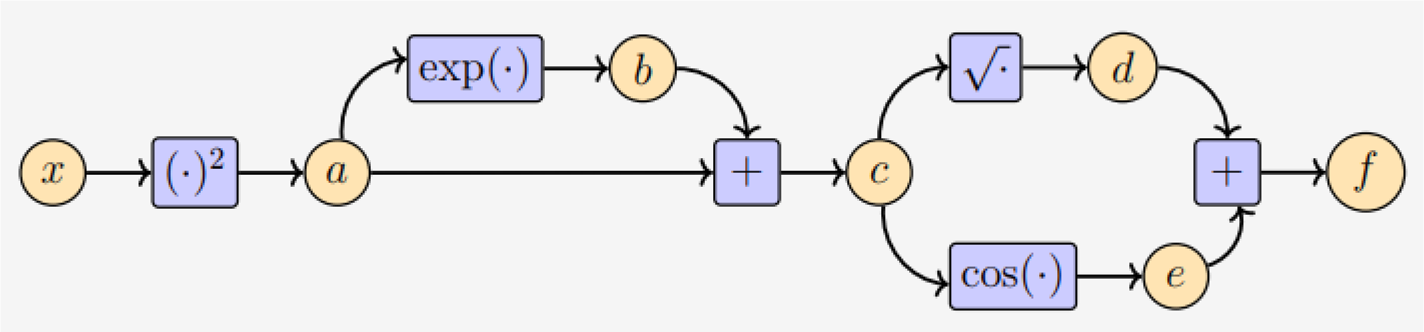


Reverse Mode

- Start by a forward pass to calculate the values.
- Do a reverse pass for the gradients.

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right)$$

$$\begin{aligned}
\frac{\partial f}{\partial c} &= \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial c} \\
\frac{\partial f}{\partial b} &= \frac{\partial f}{\partial c}\frac{\partial c}{\partial b} \\
\frac{\partial f}{\partial a} &= \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a} \\
\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}\,.
\end{aligned}$$



- This is what neural networks use to calculate the gradients.

What if f cannot be expressed as a computational graph with differentiable functions?

Numerical Differentiation

- Estimate derivatives numerically (not exact!!).
- **Finite difference methods**; use the definition of differentiation and plug in a small value of h.
- Forward difference $O(h)$ but central difference $O(h^2)$.
- If h is too small, we might face numerical subtractive cancellation issues.

$$f'(x) \approx \underbrace{\frac{f(x+h)-f(x)}{h}}_{\text{forward difference}} \approx \underbrace{\frac{f(x+h/2)-f(x-h/2)}{h}}_{\text{central difference}} \approx \underbrace{\frac{f(x)-f(x-h)}{h}}_{\text{backward difference}}$$

Numerical Differentiation

- Estimate derivatives numerically (not exact!!).
- **Finite difference methods**; use the definition of differentiation and plug in a small value of h.
- Forward difference $O(h)$ but central difference $O(h^2)$.
- If h is too small, we might face numerical subtractive cancellation issues.
- **Complex step method;**

$$f'(x) \approx \frac{Im\big(f(x+ih)\big)}{h}$$

- No subtractive cancellation issues. $O(h^2)$.
- All proofs with Taylor series.

$$f'(x) \approx \underbrace{\frac{f(x+h)-f(x)}{h}}_{\text{forward difference}} \approx \underbrace{\frac{f(x+h/2)-f(x-h/2)}{h}}_{\text{central difference}} \approx \underbrace{\frac{f(x)-f(x-h)}{h}}_{\text{backward difference}}$$

Numerical Differentiation

- Estimate derivatives numerically (not exact!!).
- **Finite difference methods**; use the definition of differentiation and plug in a small value of h.
- Forward difference $O(h)$ but central difference $O(h^2)$.
- If h is too small, we might face numerical subtractive cancellation issues.
- **Complex step method;**

$$f'(x) \approx \frac{Im\big(f(x+ih)\big)}{h}$$

- No subtractive cancellation issues. $O(h^2)$.
- All proofs with Taylor series.

$$f'(x) \approx \underbrace{\frac{f(x+h)-f(x)}{h}}_{\text{forward difference}} \approx \underbrace{\frac{f(x+h/2)-f(x-h/2)}{h}}_{\text{central difference}} \approx \underbrace{\frac{f(x)-f(x-h)}{h}}_{\text{backward difference}}$$

These methods do not scale well with the number of parameters.

Regression Gradient

- Used for problems with noisy objective functions because the regression helps smooth out the noise when producing a gradient estimate.
- Need a dataset of perturbations and their function evaluations.
- Use first-order Taylor expansion.
- Find g using linear regression.

$$\mathbf{g} = \Delta\mathbf{X}^{+}\Delta\mathbf{f}$$

$$\left(\Delta\mathbf{x}^{(1)}, f(\mathbf{x}+\Delta\mathbf{x}^{(1)})\right), \left(\Delta\mathbf{x}^{(2)}, f(\mathbf{x}+\Delta\mathbf{x}^{(2)})\right), \ldots, \left(\Delta\mathbf{x}^{(m)}, f(\mathbf{x}+\Delta\mathbf{x}^{(m)})\right)$$

$$\hat{f}(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \mathbf{g}^{\top}\Delta\mathbf{x}$$

$$\Delta\mathbf{X} = \begin{bmatrix} (\Delta\mathbf{x}^{(1)})^{\top} \\ \vdots \\ (\Delta\mathbf{x}^{(m)})^{\top} \end{bmatrix}$$

$$\Delta\mathbf{f} = \left[f(\mathbf{x}+\Delta\mathbf{x}^{(1)}) - f(\mathbf{x}), \ \ldots, \ f(\mathbf{x}+\Delta\mathbf{x}^{(m)}) - f(\mathbf{x})\right]$$

Directional Derivatives

- "The directional derivative $\nabla_s f(x)$ of a multivariate function f is the instantaneous rate of change of $f(x)$ as x is moved with velocity s."
- We can calculate the directional derivative using the following formula.

$$\nabla_{\mathbf{s}} f(\mathbf{x}) = \nabla f(\mathbf{x})^\top \mathbf{s}$$

- The directional derivative is a scalar (when the function output is scalar)!
- The directional derivative is highest in the gradient direction and lowest in the opposite direction of the gradient.

$$\nabla_{\mathbf{s}} f(\mathbf{x}) \equiv \underbrace{\lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{s}) - f(\mathbf{x})}{h}}_{\text{forward difference}} = \underbrace{\lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{s}/2) - f(\mathbf{x} - h\mathbf{s}/2)}{h}}_{\text{central difference}} = \underbrace{\lim_{h \to 0} \frac{f(\mathbf{x}) - f(\mathbf{x} - h\mathbf{s})}{h}}_{\text{backward difference}}$$

Simultaneous Perturbation Stochastic Gradient Approximation (SPSA)

- SPSA can estimate the gradient with as few as two function evaluations, regardless of the number of variables. Can work in deep learning.
- Uses directional derivatives.

$$\nabla_{\mathbf{z}} f(\mathbf{x}) \approx \frac{f(\mathbf{x}+\delta\mathbf{z}) - f(\mathbf{x}-\delta\mathbf{z})}{2\delta}$$

$$\nabla f(\mathbf{x}) \approx (\nabla_{\mathbf{z}} f(\mathbf{x}))\mathbf{z}$$

Simultaneous Perturbation Stochastic Gradient Approximation (SPSA)

- SPSA can estimate the gradient with as few as two function evaluations, regardless of the number of variables. Can work in deep learning.
- Uses directional derivatives.
- $z \sim N(0, I)$.
- Average many samples to improve the estimate.
- "The sample count is typically left quite small or even set to 1".

$$\nabla_{\mathbf{z}} f(\mathbf{x}) \approx \frac{f(\mathbf{x} + \delta \mathbf{z}) - f(\mathbf{x} - \delta \mathbf{z})}{2\delta}$$

$$\nabla f(\mathbf{x}) \approx (\nabla_{\mathbf{z}} f(\mathbf{x}))\mathbf{z}$$

Thank you for listening! Questions?

Stable Diffusion 3 (uses Flow model)