Computational Graphs

CS115 - Math for Computer Science

TS. Lương Ngọc Hoàng
TS. Dương Việt Hằng

September 9, 2023

Roadmap

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

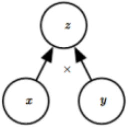
Variables are Nodes in Graph

- So far neural networks described with informal graph language
- To describe back-propagation it is helpful to use more precise computational graph language
- Many possible ways of formalizing computations as graph
- Here we use each node as a variable

The variable may be a

- *Scalar, vector, matrix, tensor, or other type*

Ex: Computational Graph of xy



(a)

(a) Compute $z = xy$

Operations in Graphs

❑ To formalize our graphs we also need the idea of an operation

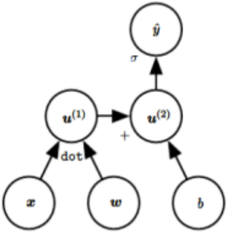
- *An operation is a simple function of one or more variables*
- *Our graph language is accompanied by a set of allowable operations*
- *Functions more complex than operations are obtained by composing operations*
- *If variable y is computed by applying operation to variable x then draw directed edge from x to y*

Edges denote input-output

- To formalize our graphs we also need the idea of an operation

- *If variable y is computed from variable x we*
- *draw an edge from x to y*
- *• We may annotate the output node with the*
- *name of the operation*

Ex: Graph of Logistic Regression



(b)

(b) Logistic Regression Prediction $\hat{y} = \sigma(\boldsymbol{x}^T\boldsymbol{w} + b)$

– Variables in graph $\boldsymbol{u}^{(1)}$ and $\boldsymbol{u}^{(2)}$ are not in original expression, but are needed in graph

Ex: Graph of ReLU

(c) Compute expression $\boldsymbol{H}=\max\{0,\boldsymbol{XW}+\boldsymbol{b}\}$

– Computes a design matrix of Rectified linear unit activations $\boldsymbol{H}$ given design matrix consisting of a minibatch of inputs $\boldsymbol{X}$

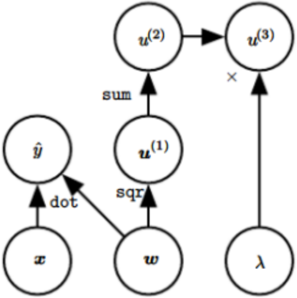
Ex: Two operations on input

(d) Perform more than one operation to a variable

Weights $\boldsymbol{w}$ are used in two operations:

1. To make prediction $\hat{y}$ and
2. The weight decay penalty

$$\lambda \sum_i w_i^2$$



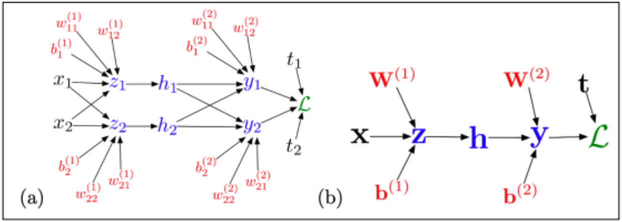
(d)

Ex: Graph of Linear Regression

$$p(C_1|\phi) = y(\phi) = \sigma\,(w^T\phi + b) + ½||w||^2$$

$$\begin{aligned} z &= wx + b \\ y &= \sigma(z) \\ \mathcal{L} &= \frac{1}{2}(y - t)^2 \\ \mathcal{R} &= \frac{1}{2}w^2 \\ \mathcal{L}_{\text{reg}} &= \mathcal{L} + \lambda\mathcal{R}. \end{aligned}$$

Ex: Computational Graph of MLP



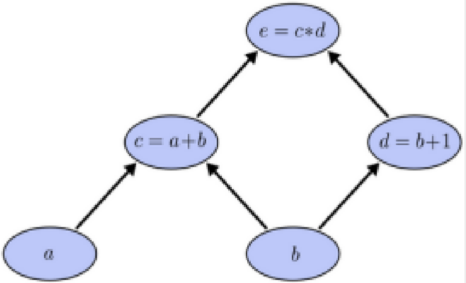
(a) Full computation graph for the loss computation in a multi-layer neural net
(b) Vectorized form of the computation graph

$$
\begin{aligned}
z_i &= \sum_j w_{ij}^{(1)} x_j + b_i^{(1)} \\
h_i &= \sigma(z_i) \\
y_k &= \sum_i w_{ki}^{(2)} h_i + b_k^{(2)} \\
\mathcal{L} &= \frac{1}{2} \sum_k (y_k - t_k)^2
\end{aligned}
$$

Graph of a math expression

- Consider the expression

  e=(a+b)(b+1)*



- *It has two adds, one multiply*
- *Introduce a variable for result of each operation*

  *c=a+b, d=b+1 and e=c *d*

- To make a computational
  - *Operations and inputs are nodes*
  - *Values used in operations are directed edges*

Computational Graph Language

- To describe backpropagation more precisely computational graph language is helpful
- Each node is either
  - *a variable (Scalar, vector, matrix, tensor, or other type*
  - *an Operation (Simple function of one or more variables. Or functions more complex than operations are obtained by composing operations)*
  - *If variable **y** is computed by applying operation to variable **x** then draw directed edge from **x** to **y***

Composite Function

- Consider a composite function *f(g(h(x)))*

  We have an outer function f, an inner function g and a final inner function h(x)

$$f(x) = e^{x}$$

$$f(x) = e^{\sin(x**2)}$$

$g(x) = \sin x$ and

$h(x) = x^2$ or

$$f(g(h(x))) = e^{g(h(x))}$$

x^2 → v=h(x) → u=g(v) → f(u)

Every connection is an input, every node is a function or operation

Chain Rule for Composites

- Chain rule is the process we can use to analytically compute derivatives of composite functions

For example, f(g(h(x))) is a composite function

$$f(x) = e^{\sin(x**2)}$$

$f(x) = e^x$

$g(x) = \sin x$ and

$h(x) = x^2$ or

$f(g(h(x))) = e^{g(h(x))}$

Derivatives of Composite function

- To get derivatives of $f(g(h(x)))= e^{g(h(x))}$ wrt x

Use the chain rule

$$\boxed{\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx}}$$

$$f(x) = e^{\sin(x**2)}$$

$f(x)=e^x$

$g(x)=\sin x$ and

$h(x)=x^2$ or

$f(g(h(x)))=e^{g(h(x))}$

$$\frac{df}{dg} = e^{g(h(x))}$$

$$\frac{dg}{dh} = \cos(h(x))$$

$$\frac{dh}{dx} = 2x$$

$$\frac{df}{dx} = e^{g(h(x))} \cdot \cos\ h(x) \cdot 2x = e^{\sin x**2} \cdot \cos x^2 \cdot 2x$$

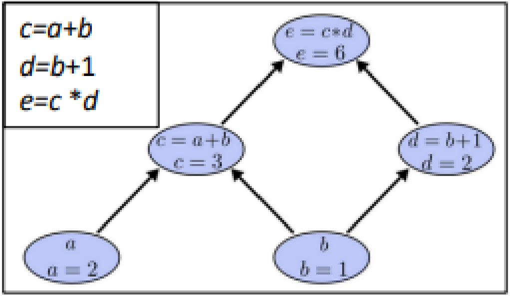
Derivatives of Composite function

- To get derivatives of $f(g(h(x)))= e^{g(h(x))}$ wrt x

Another way???

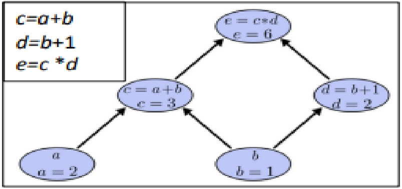
Derivatives for *e=(a+b)* (b+1)*

- Computational graph

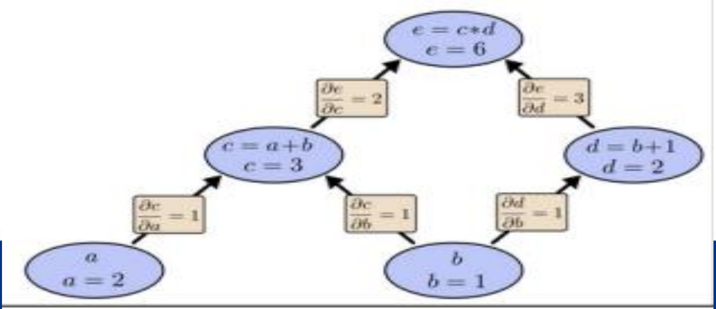


- Need derivatives on the edges (*If a <u>directly</u> affects c=a+b, then we want to know how it affects c. This is called partial derivative of c wrt a*)

Derivatives for *e=(a+b)* (b+1)*



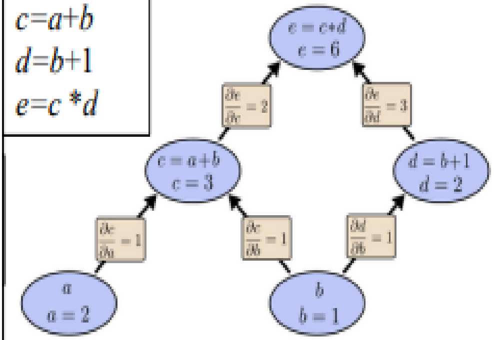
- For partial derivatives of e we need sum & product rules of calculus



Derivative wrt variables indirectly connected

- How is *e* affected by *a*?

$c = a + b$
$d = b + 1$
$e = c * d$



- The general rule (with multiple paths) is:

Sum over all possible paths from one node to the other while multiplying derivatives on each path

$$\frac{\partial e}{\partial b} = 1 * 2 + 1 * 3 = 5$$

Ex. of Backprop Computation

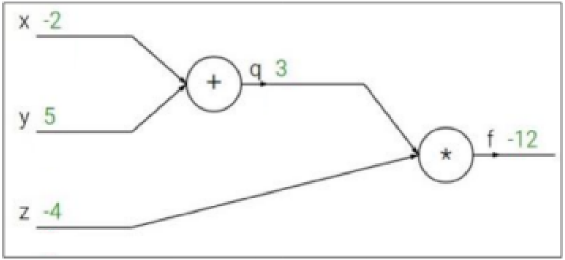
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y$$

$$f = qz$$

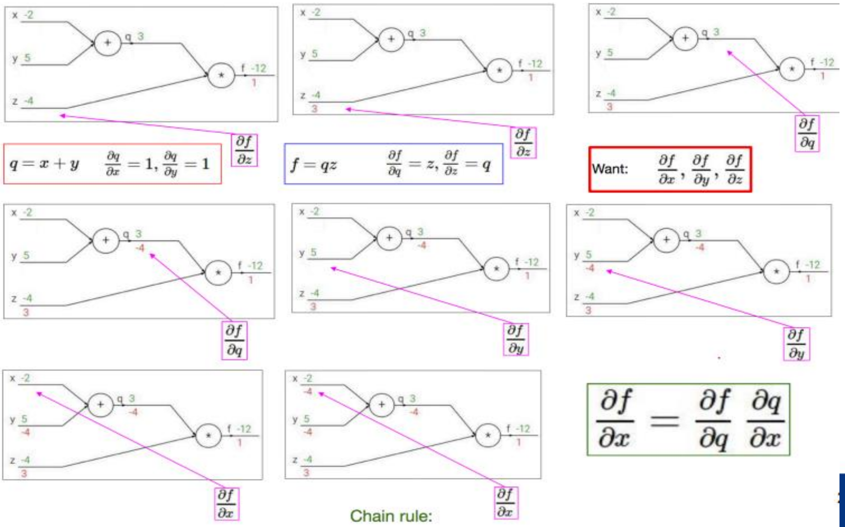
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



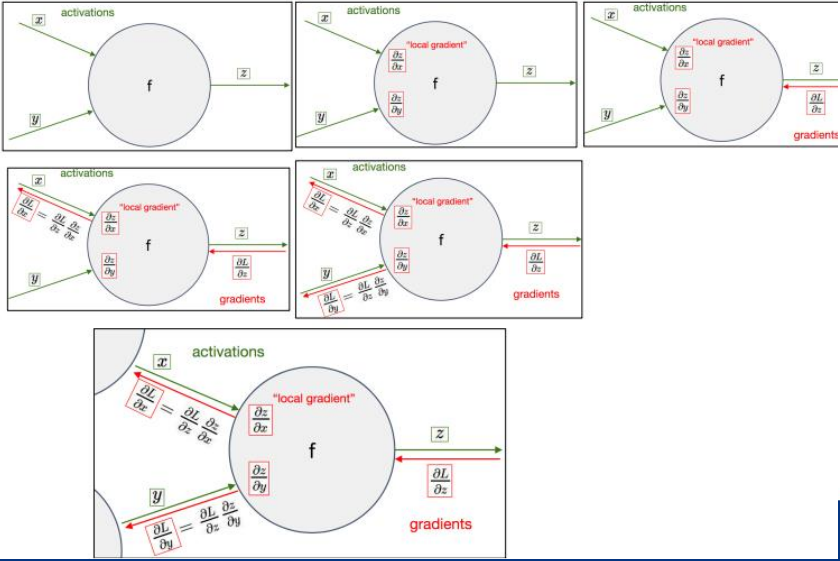
Steps in Backprop

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

e.g. x = -2, y = 5, z = -4

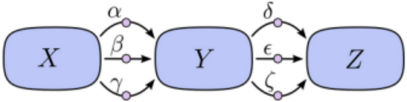


Backprop for a neuron



Factoring Paths

- Summing over paths leads to combinatorial explosion



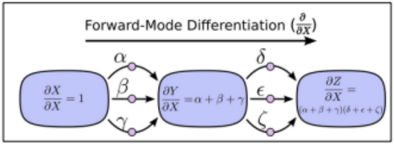
- If we want to get derivative $\boxed{\frac{\partial Z}{\partial X}}$ we need to sum over $3*3=9$ paths: $\boxed{\frac{\partial Z}{\partial X} = \alpha\delta + \alpha\varepsilon + \alpha\zeta + \beta\delta + \beta\varepsilon + \beta\zeta + \gamma\delta + \gamma\varepsilon + \gamma\zeta}$
  - It will grow exponentially
  - Instead we could factor the paths as:

$$\boxed{\frac{\partial Z}{\partial X} = (\alpha + \beta + \gamma)(\delta + \varepsilon + \zeta)}$$

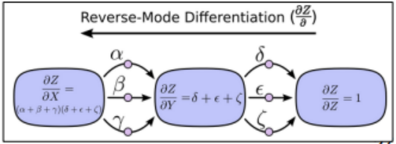
- This is where *forward-mode* and *reverse-mode* differentiation come in

Forward- and Reverse-Mode Differentiation

- Forward mode differentiation tracks how one input affects every node
  - Applies $\boxed{\frac{\partial}{\partial X}}$ to every node



- Reverse mode differentiation tracks how every node affects one output
  - Applies $\boxed{\frac{\partial Z}{\partial}}$ to every node



Reverse Mode Differentiation

Reverse-mode differentiation from e down

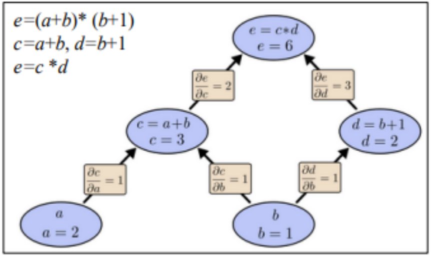
- Apply $\boxed{\frac{\partial e}{\partial}}$ to every node

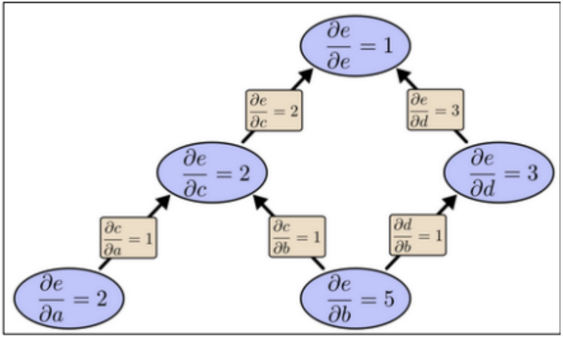
$$\boxed{\frac{\partial e}{\partial c} = \frac{\partial (c * d)}{\partial c} = d = b + 1 = 1 + 1 = 2}$$

$$\boxed{\frac{\partial e}{\partial a} = \frac{\partial (c * d)}{\partial a} = \frac{\partial ((a+b) * (b+1))}{\partial a} = b + 1 = 2}$$

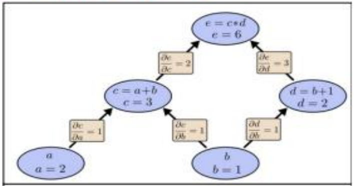
- Gives derivative of e wrt every node
- We get both $\boxed{\frac{\partial e}{\partial a}}$ and $\boxed{\frac{\partial e}{\partial b}}$

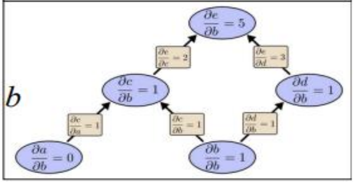
a=2 and b=1





Combining the two modes





Why reverse mode?

Consider Original example

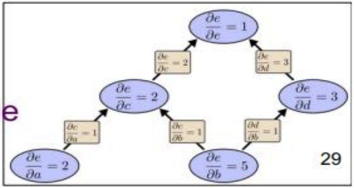
$e=(a+b)*(b+1)$

$c=a+b,\ d=b+1$

$e=c*d$

Forward differentiation from b up

- Gives derivative of every node wrt b
- *i.e.*, wrt a single input
- We get $\boxed{\frac{\partial e}{\partial b}}$



Reverse-mode diff from e down

- Gives derivative of e wrt every node
- We get both $\boxed{\frac{\partial e}{\partial a}}$ and $\boxed{\frac{\partial e}{\partial b}}$

$$\begin{array}{c} y \\ \uparrow \\ u \\ \uparrow \\ x \end{array} \qquad y=y(u);\ u=u(x)$$

$$z=z(u, v);\ u=u(x);\ v=v(x)$$

$$\begin{tikzcd} & z & \\ u \arrow[ur] & & v \arrow[ul] \\ x \arrow[u] & & x \arrow[u] \end{tikzcd}$$

$$\begin{tikzcd} & z & \\ & u \arrow[u] & \\ x \arrow[ur] & & y \arrow[ul] \end{tikzcd}$$

$z=z(u);\ u=u(x,y);$

$z=z(u,v);\ u=u(x,y);\ v=v(x, y)$