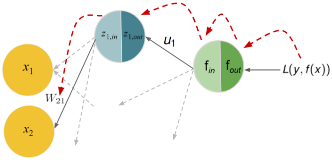
Deep Learning

Chain Rule and Computational Graphs



Learning goals

- Chain rule of calculus
- Computational graphs

CHAIN RULE OF CALCULUS

- The chain rule can be used to compute derivatives of the composition of two or more functions.
- Let $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$,
  $g : \mathbb{R}^m \to \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$.
- If $\mathbf{y} = g(\mathbf{x})$ and $z = f(\mathbf{y})$, the chain rule yields:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i}$$

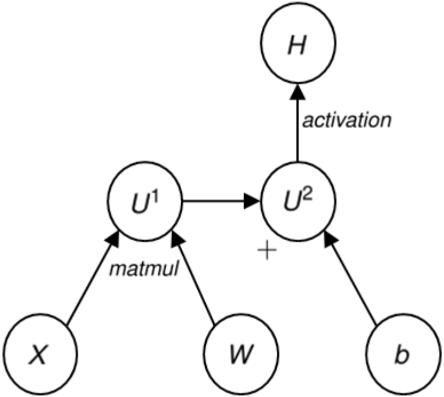
or, in vector notation:

$$\nabla_{\mathbf{x}} z = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^\top \nabla_{\mathbf{y}} z,$$

where $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ is the $(n \times m)$ Jacobian matrix of g.

COMPUTATIONAL GRAPHS

- CGs are nested expresssions, visualized as graphs.
- Each node is a variable, either an input or derived.
- Derived variables are functions applied to other variables.



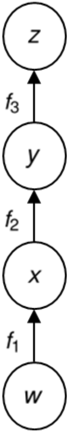
source : Goodfellow et al. (2016)

Figure: The computational graph for the expression $H = \sigma(XW + B)$ with activation function $\sigma(\cdot)$.

CHAIN RULE OF CALCULUS: EXAMPLE 1

- Suppose we have the following computational graph.
- To compute the derivative of $\frac{\partial z}{\partial w}$ we need to recursively apply the chain rule. That is:

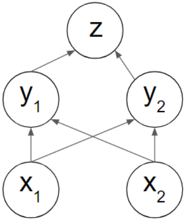
$$
\begin{aligned}
\frac{\partial z}{\partial w} &= \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x} \cdot \frac{\partial x}{\partial w} \\
&= f_3'(y) \cdot f_2'(x) \cdot f_1'(w) \\
&= f_3'(f_2(f_1(w))) \cdot f_2'(f_1(w)) \cdot f_1'(w)
\end{aligned}
$$



source : Goodfellow et al. (2016)

Figure: A computational graph, such that $x = f_1(w)$, $y = f_2(x)$ and $z = f_3(y)$.

CHAIN RULE OF CALCULUS: EXAMPLE 2



To compute $\nabla_{\mathbf{x}} z$, we apply the chain rule

- $\frac{\partial z}{\partial x_1} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_1} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x_1}$
- $\frac{\partial z}{\partial x_2} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_2} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x_2}$

Therefore, the gradient of z w.r.t $\mathbf{x}$ is

- $\nabla_{\mathbf{x}} z = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \end{bmatrix}}_{\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^\top} \underbrace{\begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \frac{\partial z}{\partial y_2} \end{bmatrix}}_{\nabla_{\mathbf{y}} z} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^\top \nabla_{\mathbf{y}} z$

COMPUTATIONAL GRAPH: NEURAL NET

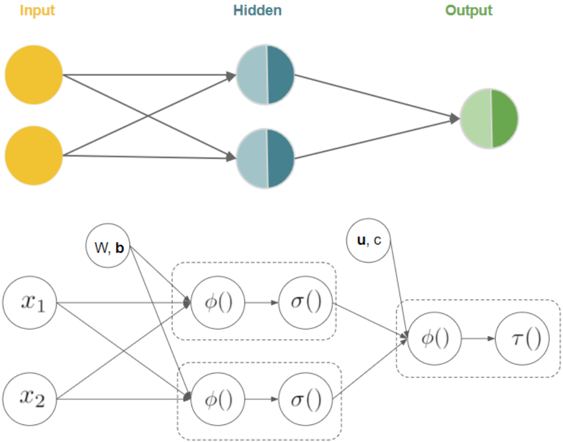


Figure: A neural network can be seen as a computational graph. ϕ is the weighted sum and σ and τ are the activations.
Note: In contrast to the top figure, the arrows in the computational graph below merely indicate **dependence**, not weights.