Neural Networks (Part III)

Automatic differentiation

Setup: Evaluate the derivative of a function $F: \mathbb{R}^d \to \mathbb{R}$, $\rightarrow y = F(x)$, $x \in \mathbb{R}^d$, y

e.g. evaluate the gradients of the loss of a neural nentwork.

↳ $x \in \mathbb{R}^d$ is a vector of containing all NN params.
$y \in \mathbb{R}$ is the value of loss, for a given x. $(\mathcal{L}(\theta))$

$$F : \underbrace{[\,\cdot \;\; \cdots \;\; \cdot\cdot\,]}_{x \in \mathbb{R}^d} \longrightarrow \underbrace{\square}_{y \in \mathbb{R}}$$

Let's focus on cases where F has a compositional form:

$$F(x) = D(C(B(A(x)))), \quad F = D \circ C \circ B \circ A$$

$$y = D(c), \quad c = C(b), \quad b = B(a), \quad a = A(x)$$

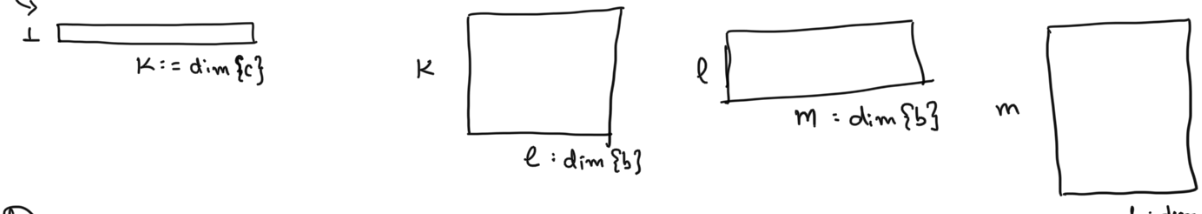
A B C D

We want to evaluate the Jacobian of F, i.e.

$$\underset{1\times d}{\nabla_x F} = F'(x) = \frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \ldots, \frac{\partial y}{\partial x_d} \right]$$

$$\xRightarrow[\text{rule}]{\text{chain}} \nabla_x F = \frac{\partial y}{\partial c} \cdot \frac{\partial c}{\partial b} \cdot \frac{\partial b}{\partial a} \cdot \frac{\partial a}{\partial x}$$

$$\frac{\partial y}{\partial c} = D'(c), \quad \frac{\partial c}{\partial b} = C'(b) \quad \frac{\partial b}{\partial a} = B'(a) \quad \frac{\partial a}{\partial x} = A'(x)$$



- Does it matter in which order we multiply these matrices?

i) Forward accumulation:

$$\nabla_x F = \frac{\partial y}{\partial c}\Bigg(\overbrace{\frac{\partial c}{\partial b}\left(\frac{\partial b}{\partial a} \cdot \frac{\partial a}{\partial x} \right)}^{K \times d} \Bigg)$$

Jacobian vector product

$$\nabla_x F \cdot \vec{v} = \frac{\partial y}{\partial c}\left(\frac{\partial c}{\partial b}\left(\frac{\partial b}{\partial a}\left(\frac{\partial}{\partial}\right.\right.\right.$$

$I = \begin{bmatrix} 1 & 0 \end{bmatrix}$ → row of I

$1 \times d$ $\quad 1 \times k$ $\quad m \times d$

$V = [0\ 1\ ]$, $d \times d$

$$\underset{m \times d}{\frac{\partial b}{\partial x}} = \begin{bmatrix} \frac{\partial b_1}{\partial x_1} & \cdots & \frac{\partial b_1}{\partial x_d} \\ \vdots & & \\ \frac{\partial b_m}{\partial x_1} & \cdots & \frac{\partial b_m}{\partial x_d} \end{bmatrix}$$

Constructs the Jacobian m one column at a tim

ii) Reverse accumulation:

$$\underset{1 \times d}{\nabla_x F} = \left(\left(\underbrace{\frac{\partial y}{\partial c} \cdot \frac{\partial c}{\partial b}} \right) \frac{\partial b}{\partial a} \right) \frac{\partial a}{\partial x}$$

$$\underset{1 \times m}{\frac{\partial y}{\partial b}} = \left[\frac{\partial y}{\partial b_1}, \frac{\partial y}{\partial b_2}, \cdots, \frac{\partial y}{\partial b_m} \right]$$

Vector jacobian product

$$\vec{v}^T \nabla_x F = \left(\left(\left(v^T \frac{\partial y}{\partial c} \right) \frac{\partial c}{\partial b} \right) \frac{\partial b}{\partial a}$$

Constructs the Jacobian matrix one row at a time

(*) Use fwd mode auto-diff:
when $\dim\{y\} \gg \dim\{x\}$

Use reverse mode auto-diff:
when $\dim\{x\} \gg \dim\{y\}$ (this is the most commo case in ML

Network initialization

Glorot initialization :

Suppose we have an input X and a linear neuron with randou weights that generates outputs Y. Also assume that $\{X, Y\}$ are normalized to have zero mean and unit variance.

$$Y = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

(*) Also assume that has a zero mean.

What's $\mathrm{Var}[Y]$?

$$\mathrm{Var}[w_i x_i] = \cancel{\mathbb{E}[x_i]^2} \, \mathrm{Var}[w_i] + \cancel{\mathbb{E}[w_i]^2} \, \mathrm{Var}[x_i] + \mathrm{Var}[w_i]\mathrm{Var}[$$

$= \mathrm{Var}[w_i]\mathrm{Var}[x_i]$

$= \quad \mathrm{Var}[W_i]\,\mathrm{Var}[X_i]$

$$\Rightarrow \mathrm{Var}[Y] = \underline{d} \cdot \underbrace{\mathrm{Var}[W_i]}_{?}\,\mathrm{Var}[X_i]$$

Choose $\mathrm{Var}[W_i]$ such : $\mathrm{Var}[W_i] = \frac{1}{d_{in}}$, $d_{in} : \dim\{X\}$

If we repeat the same analysis for the back-propagated gradien

we get a similar result : $\mathrm{Var}[W_i] = \frac{1}{d_{out}}$, $d_{out} := \dim\{Y\}$

These constraints can only be simultaneously satisfied if $d_{in} = d_{out}$.

In general we use an empirical rule to chose $\mathrm{Var}[W_i]$ as:

$$\mathrm{Var}[W_i] = \frac{2}{d_{in} + d_{out}}$$

This leads to the so-called Glorot initialization scheme:

- Uniform : $W \sim U\left[-\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}, \frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}} \right]$

- Normal : $W \sim N\left(0, \frac{2}{d_{in}+d_{out}}\right)$