Last time

- want $\theta^* = \underset{\theta \in \Theta}{\text{argmin}}\ L(\theta, D)$ instead $\hat{\theta} = \underset{\theta \in \Theta}{\text{opt}}\ L(\theta, D)$

- gradient-based methods (a subset of opt)
  - GD, SGD, Momentum, RMSprop, Adam
  - all rely on obtaining $[\nabla_\theta L(\theta, D)]\big|_{\theta = \theta^{(t)}}$
  - however this quantity can be very difficult to calculate (e.g. complex L, high-dim θ)

    complexity driven by f_θ

Today

- How to systematically obtain $[\nabla_\theta L(\theta, D)]\big|_{\theta = \theta^{(t)}}$ for arbitrarily complex classes of functions (e.g. neural networks)

Backpropagation Algorithm

- A modularized procedure for evaluating derivatives of arbitrarily complex functions, in particular neural networks, at a given input value

Idea:

Step 1: Break a complex function into simple components connected through a directed computational graph.

Def computational graph: A graph comprised of nodes (representing "sub-functions") and edges (representing function arguments and outputs) that together describe the computations required to evaluate a given function.

Step 2: Obtain the derivatives of each node/"sub-function"'s outputs w.r.t. its inputs

Step 3: Evaluate the function at a given input value ("forward pass")

Step 4: Obtain the derivate of the function at this input value by applying the chain rule to the node derivatives along the computational graph. ("backward pass")

Recall:

→ univariate chain rule:

$$\frac{d}{dt} f(g(t)) = \frac{df}{dg} \cdot \frac{dg}{dt} = f'(g(t)) \cdot g'(t)$$

e.g. $h(t) = \max(0, t^2)$ $\qquad t \to g(t) \xrightarrow{v} \max(0,v) \to h(t)$

$f(v) = \max(0, v)$

$v = g(t) = t^2$

$f'(v) = \mathbb{1}\{v \geq 0\}$ ⚠ not differentiable at $v = 0$

$g'(t) = 2t$

$$h(t) = \max(0, g(t)) = f(g(t))$$

$$\frac{d}{dt} h(t) = f'(g(t)) \cdot g'(t) = \mathbb{1}\{g(t) \geq 0\} \cdot 2t$$

$$= \mathbb{1}\{t^2 \geq 0\} \cdot 2t$$

→ multivariate chain rule

$$\frac{d}{dt} f(g_1(t), \ldots g_p(t)) = \sum_{i=1}^{p} \frac{\partial f}{\partial g_i} \cdot \frac{dg_i}{dt}$$

e.g. $h(t) = \frac{\sin(t)}{t}$

$f(v_1, v_2) = v_1 \cdot v_2$ $\qquad \frac{\partial f}{\partial v_1} = v_2 \qquad \frac{\partial f}{\partial v_2} = v_1$

$v_1 = g_1(t) = \sin(t) \qquad \frac{dg_1(t)}{dt} = \cos(t)$

$v_2 = g_2(t) = \frac{1}{t} \qquad \frac{dg_2(t)}{dt} = -\frac{1}{t^2}$

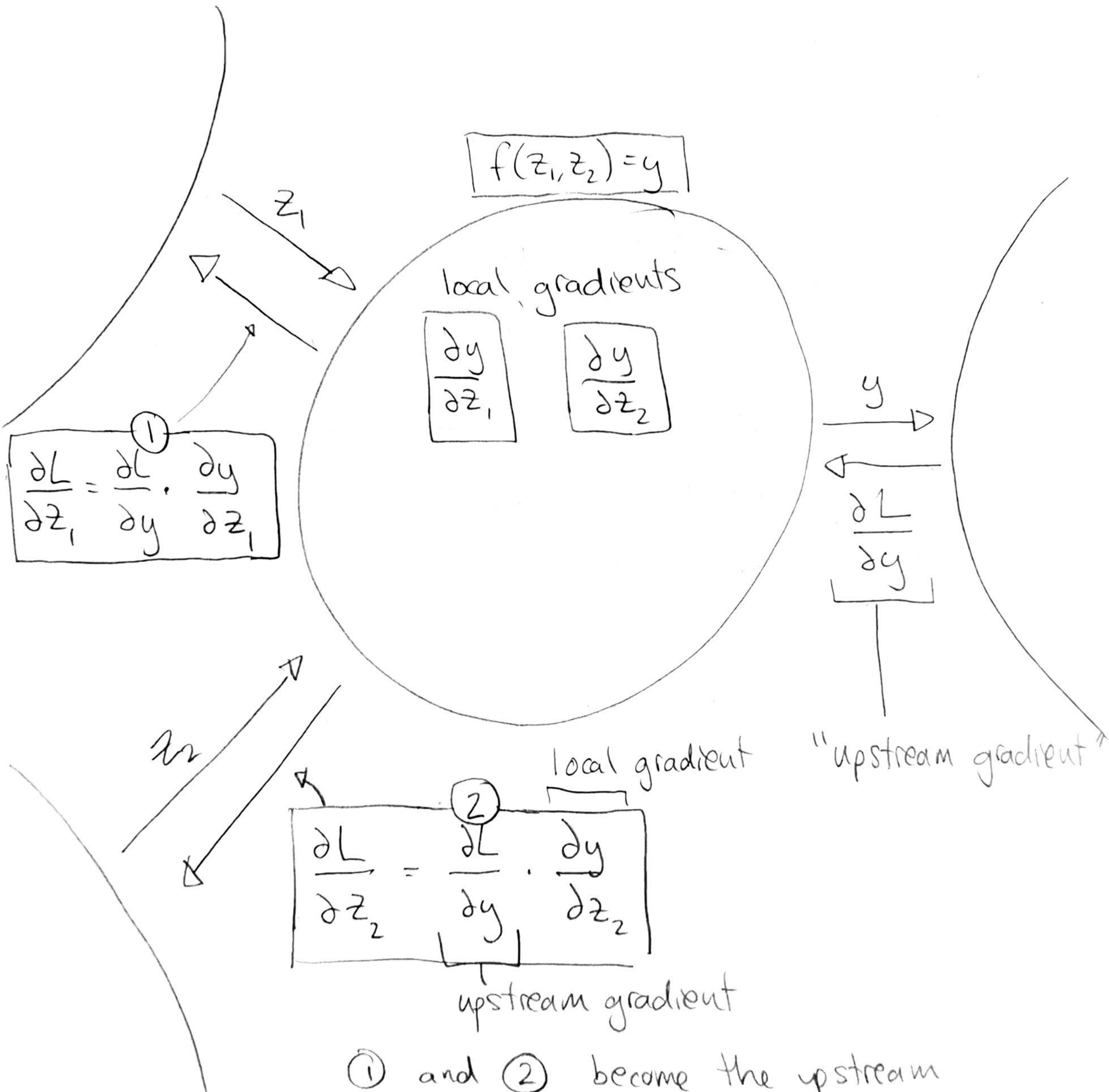
$$\frac{d\,h(t)}{dt} = \frac{d}{dt} f(g_1(t), g_2(t))$$

$$= \frac{\partial f}{\partial g_1} \cdot \frac{dg_1}{dt} + \frac{\partial f}{\partial g_2} \cdot \frac{dg_2}{dt}$$

$$= \left(\frac{1}{t}\right) \cdot \cos(t) + \sin(t) \cdot \left(\frac{-1}{t^2}\right)$$

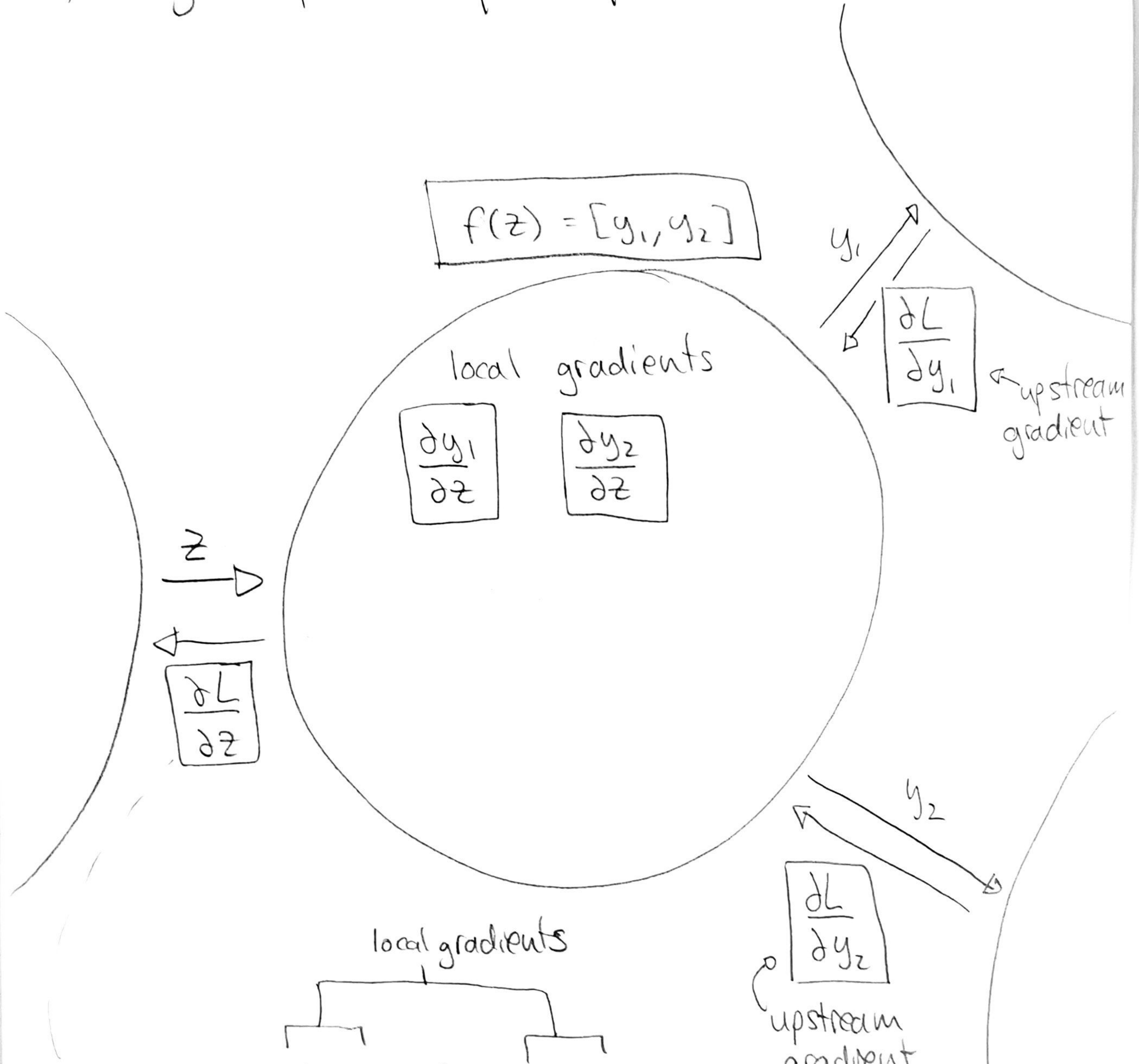
2 Scenarios within a computational graph

(1) multiple inputs, single output : $f(z_1, z_2) = y$



(1) and (2) become the upstream gradients in the z_1 and z_2 nodes

(2) single input multiple outputs



local gradients

$$\frac{\partial L}{\partial z} = \underbrace{\frac{\partial L}{\partial y_1}} \cdot \overbrace{\frac{\partial y_1}{\partial z}} + \underbrace{\frac{\partial L}{\partial y_2}} \cdot \overbrace{\frac{\partial y_2}{\partial z}}$$

upstream gradients