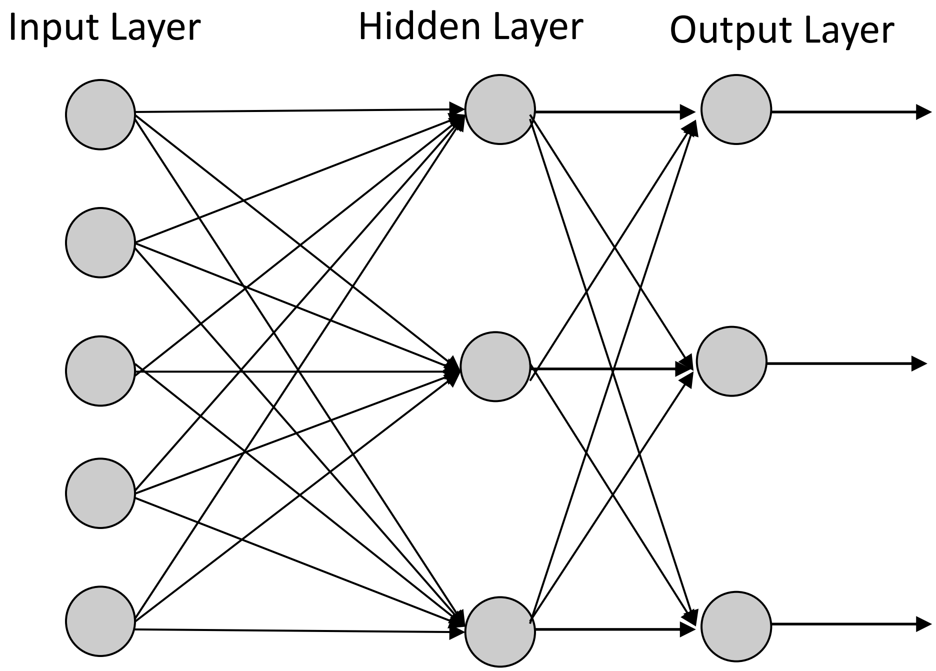
Today: Outline

- **Neural networks cont'd:** learning via gradient descent; chain rule review; gradient computation using the backpropropagation algorithm; neural network architectures

- Reminder: Pre-lecture Material for Tue Sept 29

Neural Networks II

Artificial Neural Network



Neural Network (fully connected)

Deep Network: many hidden layers

Artificial Neural Network:

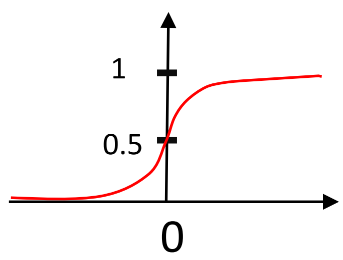
general notation

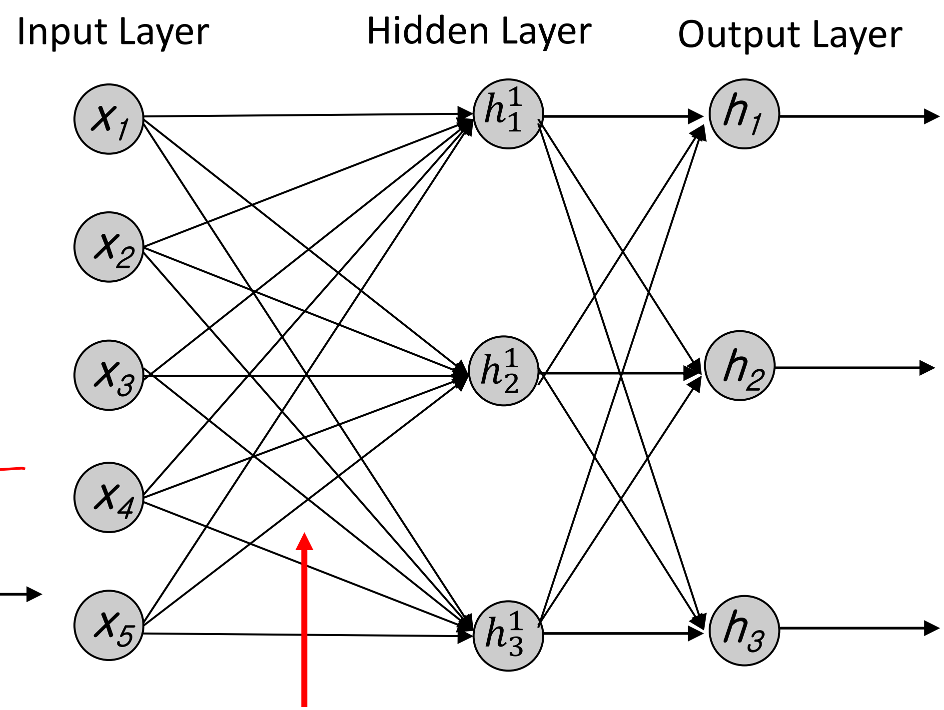
input $x = \begin{bmatrix} x_1 \\ \cdots \\ x_5 \end{bmatrix}$

hidden layer activations

$$h^i = g(\Theta^{(i)} x)$$

$$g(z) = \frac{1}{1 + \exp(-z)}$$





weights $\Theta^{(1)} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{15} \\ \vdots & \ddots & \vdots \\ \theta_{31} & \cdots & \theta_{35} \end{pmatrix}$

Artificial Neural Network:

general notation

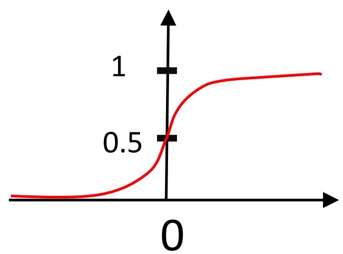
input

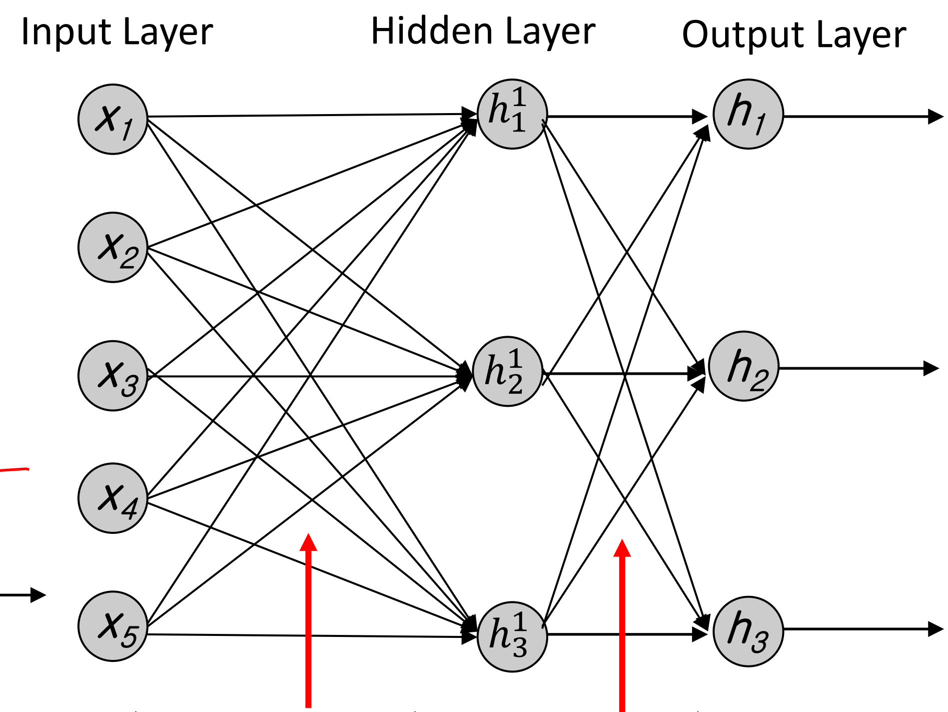
$$x = \begin{bmatrix} x_1 \\ \cdots \\ x_5 \end{bmatrix}$$

hidden layer activations

$$h^i = g(\Theta^{(i)}x)$$

$$g(z) = \frac{1}{1+\exp(-z)}$$





output

$$h_\Theta(\mathrm{x}) = g(\Theta^{(2)}a)$$

weights

$$\Theta^{(1)} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{15} \\ \vdots & \ddots & \vdots \\ \theta_{31} & \cdots & \theta_{35} \end{pmatrix} \qquad \Theta^{(2)} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{13} \\ \vdots & \ddots & \vdots \\ \theta_{31} & \cdots & \theta_{33} \end{pmatrix}$$

Cost function

Neural network: $h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$

training error

$$J(\Theta) = \boxed{-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}\log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right]}$$
$$\boxed{+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2}$$

regularization

Gradient computation

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta(x^{(i)})_k + (1-y_k^{(i)})\log(1-h_\theta(x^{(i)})_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_j^{(l)})^2$$

$$\min_{\Theta} J(\Theta)$$

Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta)$

Gradient computation

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta(x^{(i)})_k + (1-y_k^{(i)})\log(1-h_\theta(x^{(i)})_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_j^{(l)})^2$$

$$\min_{\Theta} J(\Theta)$$

Need code to compute:

- $J(\Theta)$
- $\boxed{\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta)}$ ← ***Backpropagation***

Neural Networks II

Backpropagation

Partial Derivatives

- Re-cap:

$$f(x, y) = x^2 + 2xy^2 + y^3$$

$$\frac{\partial f}{\partial x} = 2x + 2y^2$$

Chain Rule

- Need to compute gradient of

$$\log(h_\Theta(\mathrm{x})) = \log(g(\Theta^{(2)}g(\Theta^{(1)}x))) \quad \text{w.r.t } \Theta$$

- How can we compute the gradient of several chained functions?

$$f(\theta) = f_1(f_2(\theta)) \qquad f'(\theta) = f_1'(f_2(\theta)) * f_2{}'(\theta)$$

$$f'(\theta) = \frac{\partial f}{\partial \theta} = \frac{\partial f_1}{\partial f_2}\frac{\partial f_2}{\partial \theta}$$

- What about functions of multiple variables?

$$f(\theta_1, \theta_2) = f_1(f_2(\theta_1, \theta_2)) \qquad \frac{\partial f}{\partial \theta_1} = \qquad \frac{\partial f}{\partial \theta_2} =$$

https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/multivariable-chain-rule/v/multivariable-chain-rule

Backpropagation: Efficient Chain Rule

- Partial gradient computation via chain rule:

$$\frac{\partial f}{\partial \theta_1} = \frac{\partial f_1}{\partial f_2}(f_2(f_3(\theta))) * \frac{\partial f_2}{\partial f_3}(f_3(\theta)) * \frac{\partial f_3}{\partial \theta_1}(\theta)$$

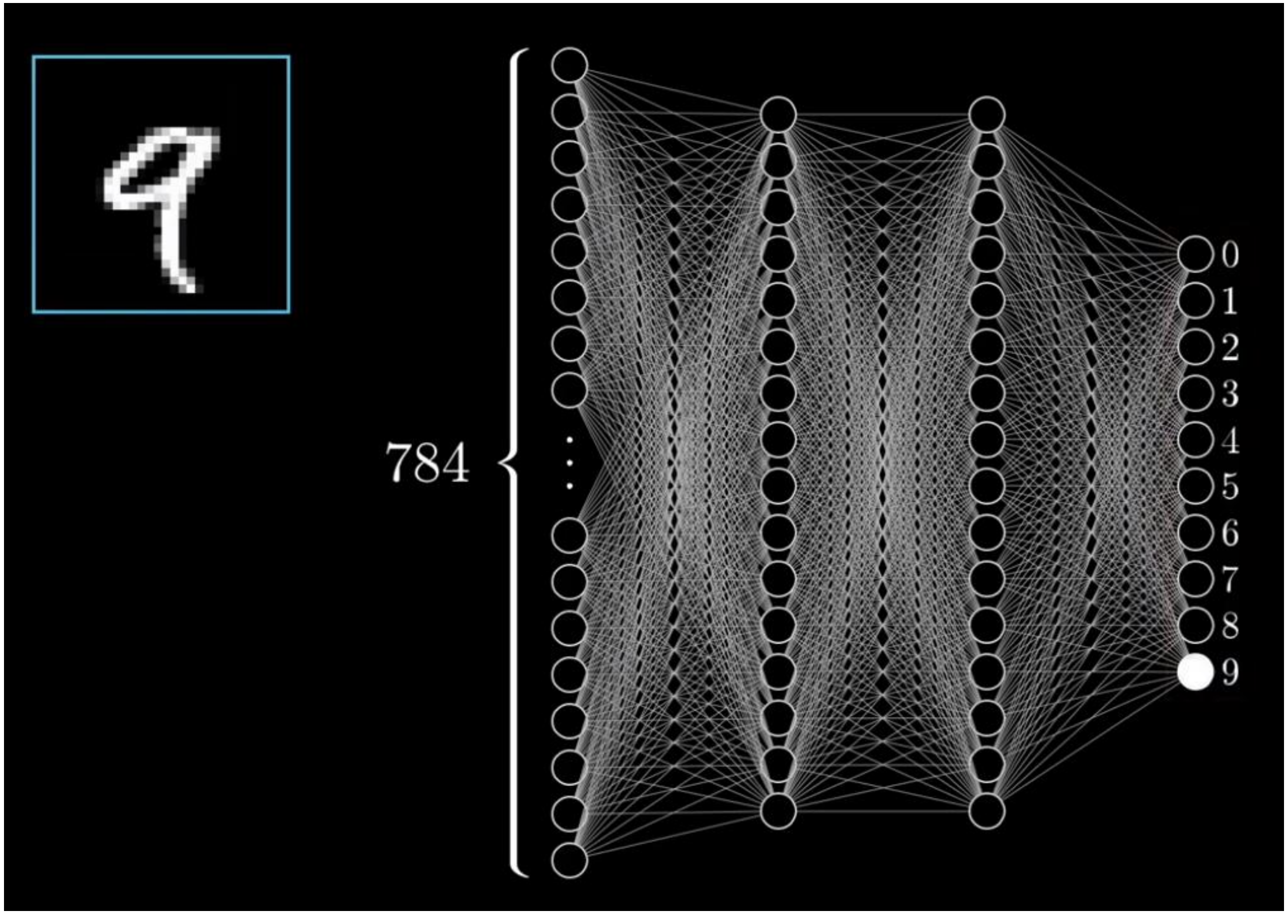
$$\frac{\partial f}{\partial \theta_2} = \frac{\partial f_1}{\partial f_2}(f_2(f_3(\theta))) * \frac{\partial f_2}{\partial f_3}(f_3(\theta)) * \frac{\partial f_3}{\partial \theta_2}(\theta)$$

$$\frac{\partial f}{\partial \theta_3} = \frac{\partial f_1}{\partial f_2}(f_2(f_3(\theta))) * \frac{\partial f_2}{\partial f_3}(f_3(\theta)) * \frac{\partial f_3}{\partial \theta_3}(\theta)$$

- need to re-evaluate functions many times
- Very inefficient! E.g. 100,000-dim parameters

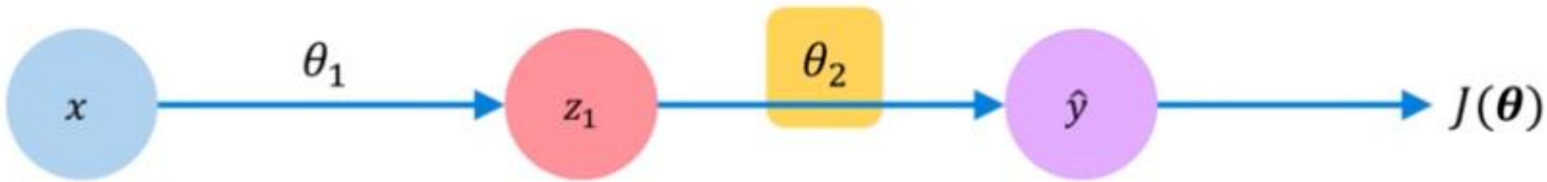
Example: Classification

- A deep network is a massive composite function!



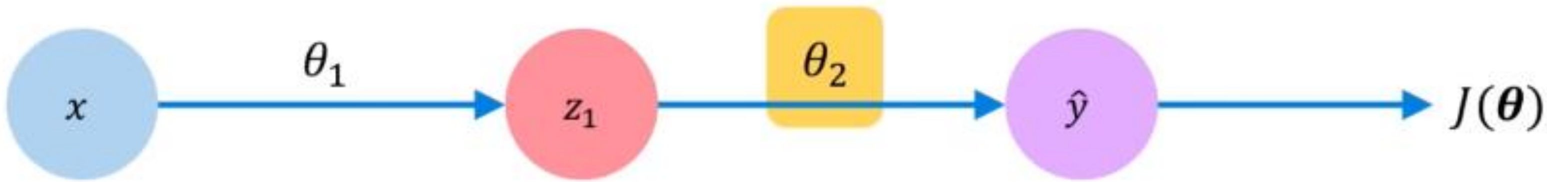
BLUE1BROWN SERIES

Computing Gradients: Backpropagation



How does a small change in one weight (ex. $\boldsymbol{\theta_2}$) affect the final loss $\boldsymbol{J(\theta)}$?

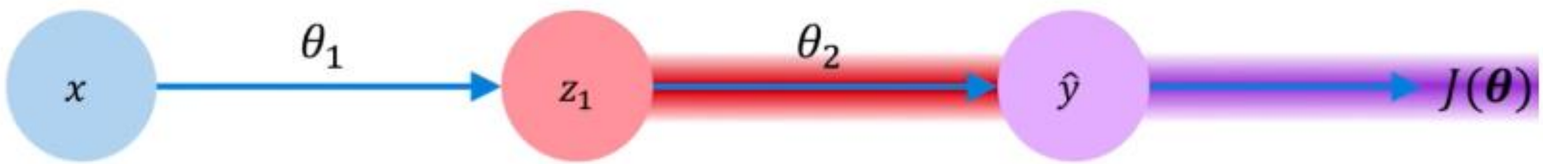
Computing Gradients: Backpropagation



$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} =$$

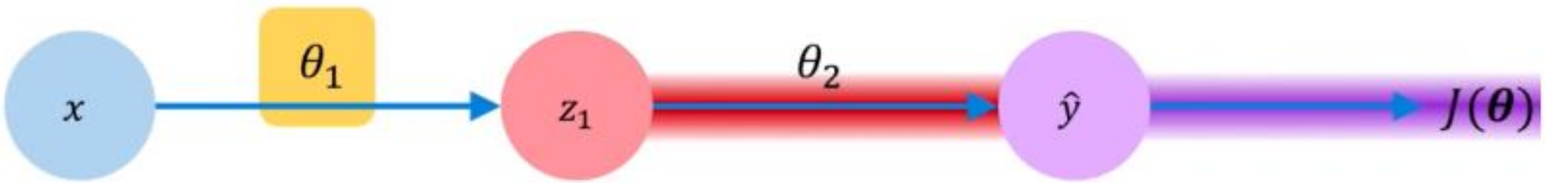
Let's use the chain rule!

Computing Gradients: Backpropagation



$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial \theta_2}$$

Computing Gradients: Backpropagation

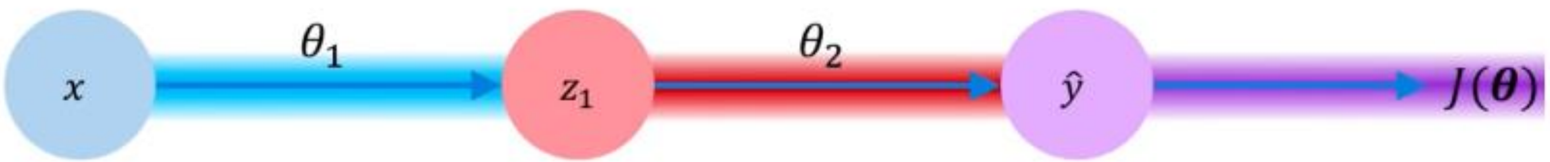


$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial \theta_1}$$

Apply chain rule!

Apply chain rule!

Computing Gradients: Backpropagation



$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial \theta_1}$$

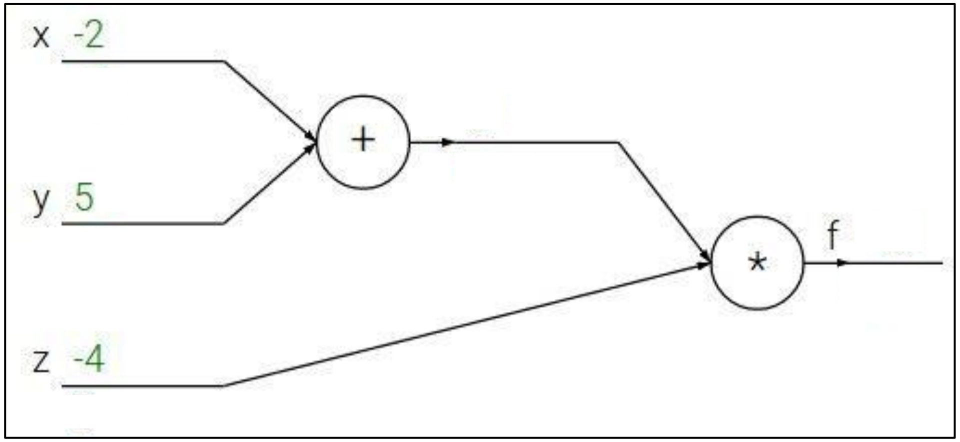
Neural Networks II

Analytical Gradients with Computational Graphs

Chain Rule with a Computational Graph

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

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



Chain Rule with a Computational Graph

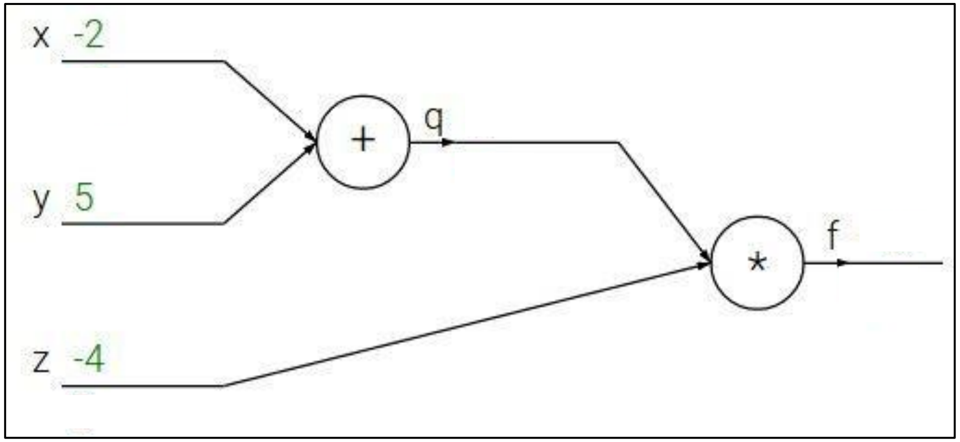
$$f(x,y,z) = (x+y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



Computation Graph: Forward

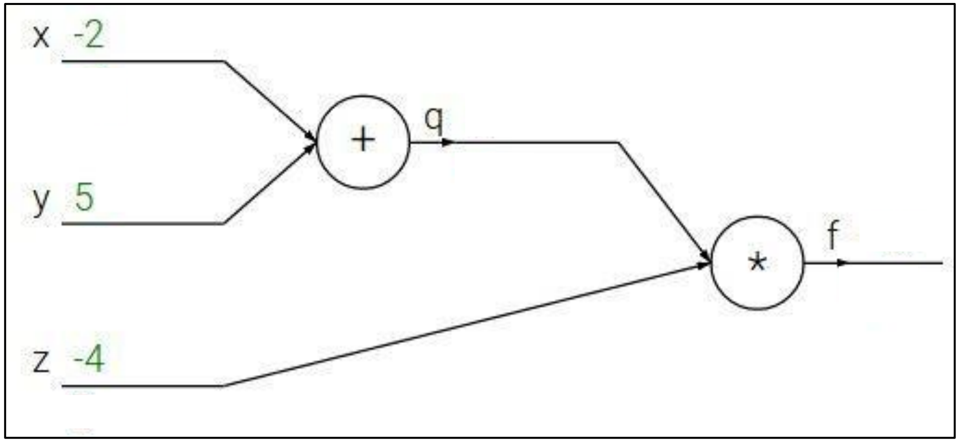
$$f(x,y,z) = (x+y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



compute values

Computation Graph: Backward

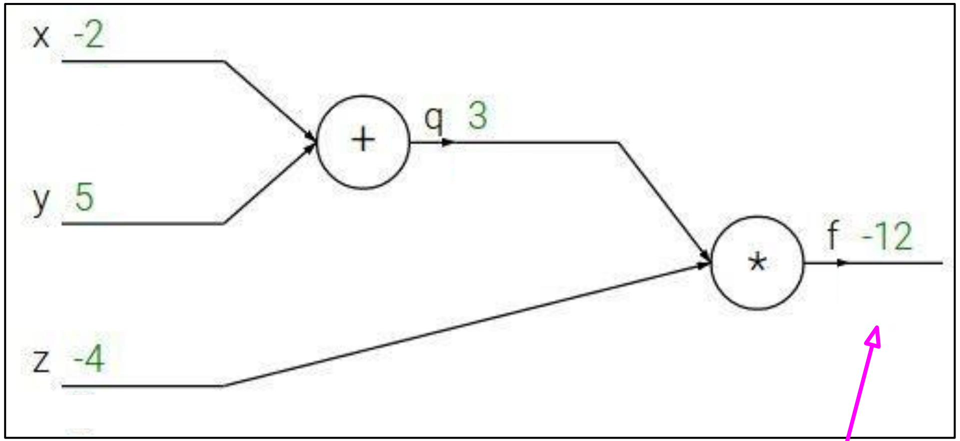
$$f(x,y,z) = (x+y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



$\frac{\partial f}{\partial f}$

compute gradients

Computation Graph: Backward

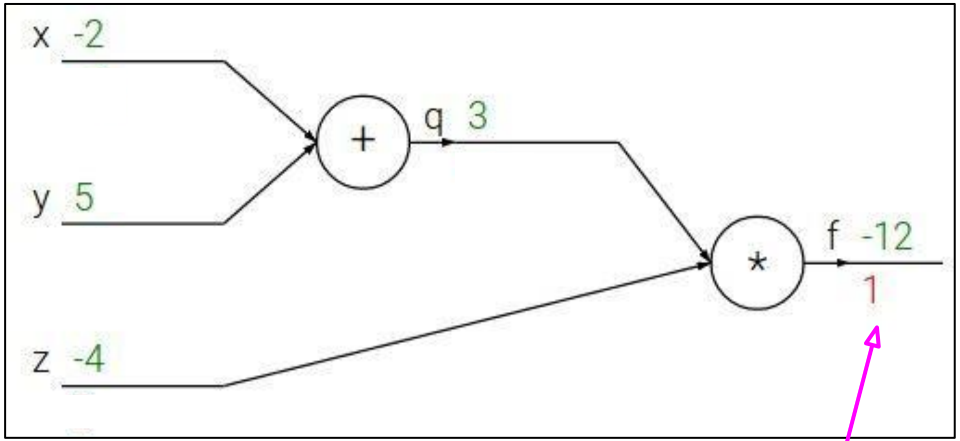
$$f(x, y, z) = (x + y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



$$\frac{\partial f}{\partial f}$$

Computation Graph: Backward

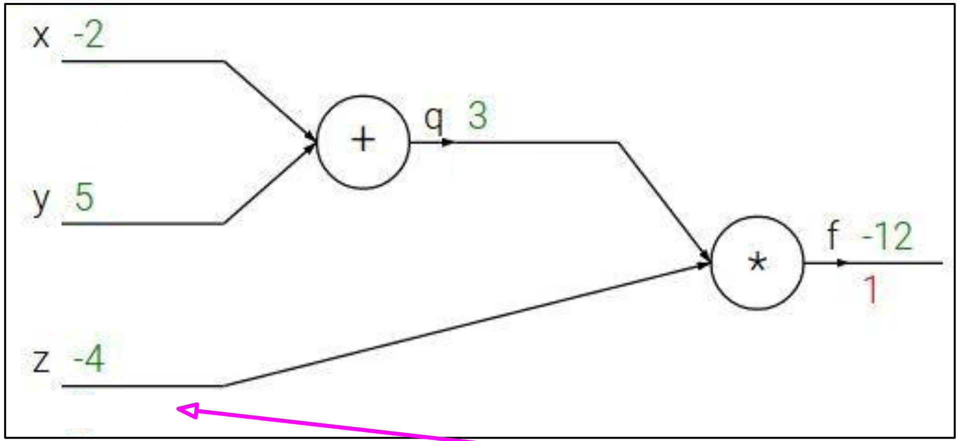
$$f(x, y, z) = (x + y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

Computation Graph: Backward

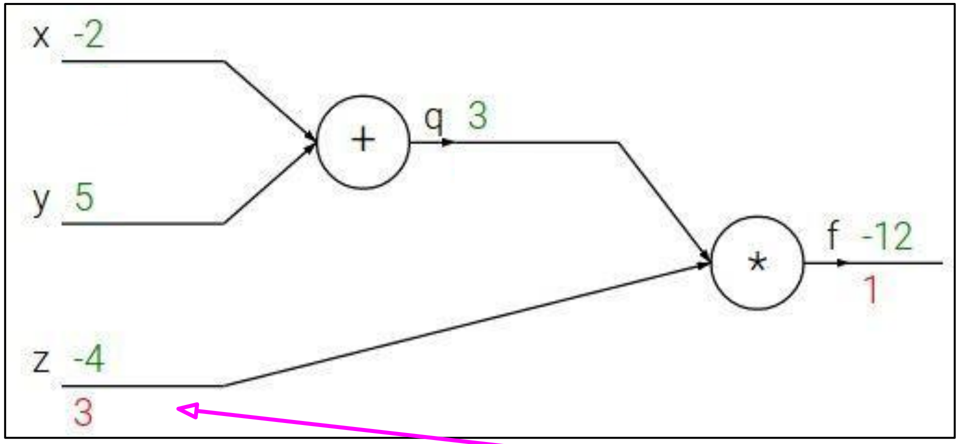
$$f(x,y,z) = (x+y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



$$\boxed{\frac{\partial f}{\partial z}}$$

Computation Graph: Backward

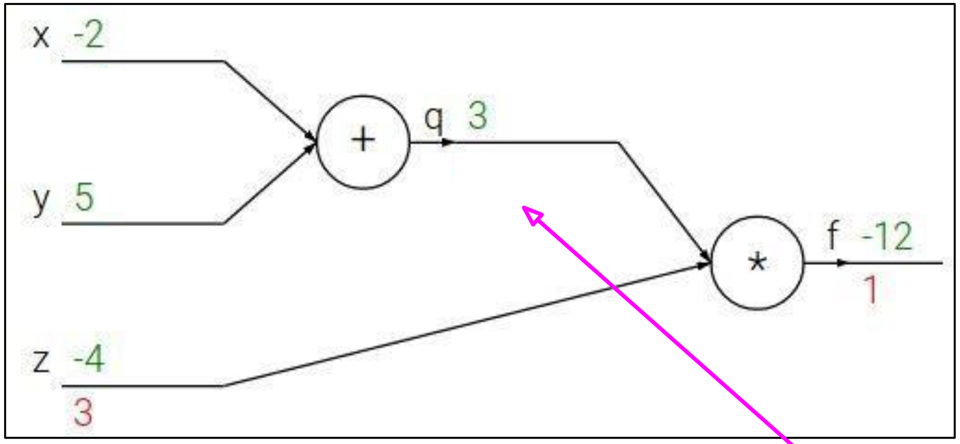
$$f(x,y,z) = (x+y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



$$\boxed{\frac{\partial f}{\partial q}}$$

Computation Graph: Backward

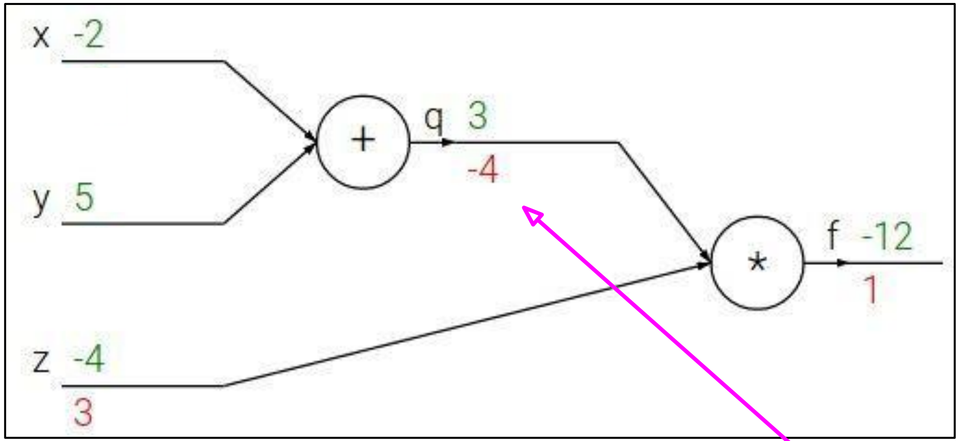
$$f(x,y,z) = (x+y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



$$\frac{\partial f}{\partial q}$$

Computation Graph: Backward

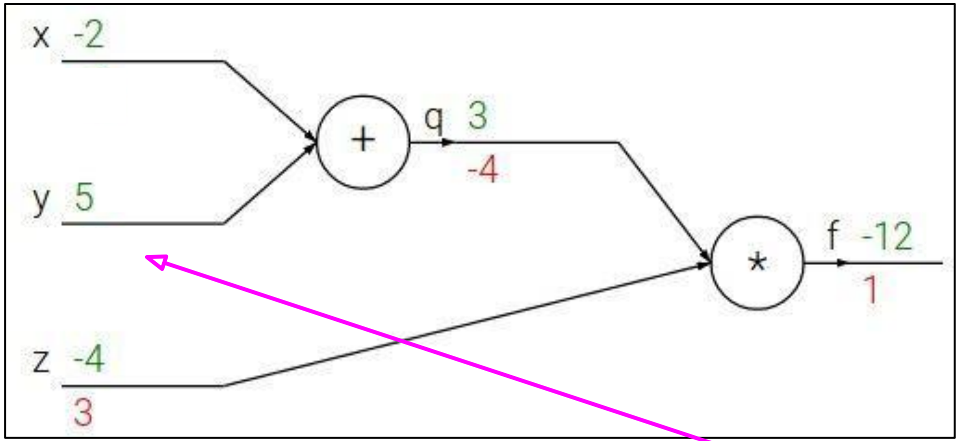
$$f(x, y, z) = (x + y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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

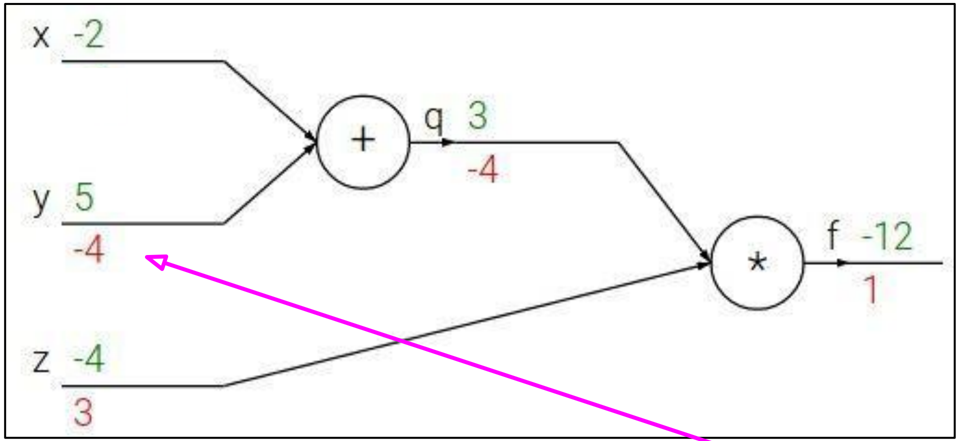


$$\frac{\partial f}{\partial y}$$

Computation Graph: Backward

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

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



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\boxed{\frac{\partial f}{\partial y}}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}}$$

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

Computation Graph: Backward

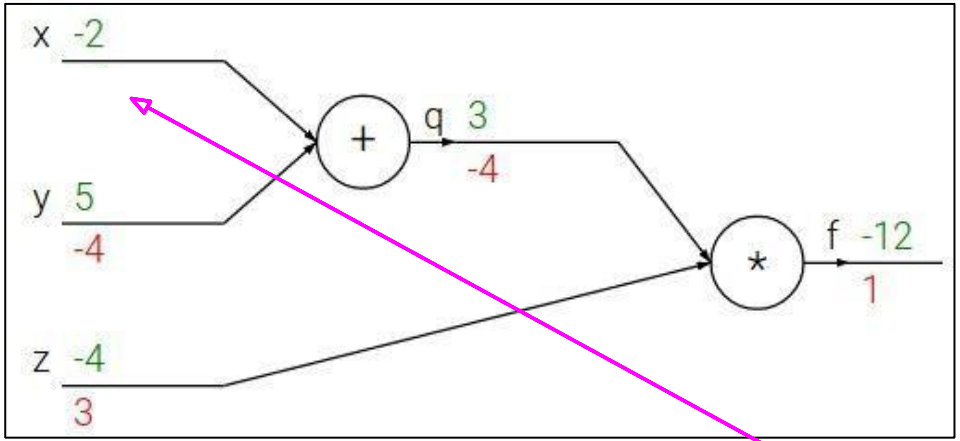
$$f(x, y, z) = (x + y)z$$

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

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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

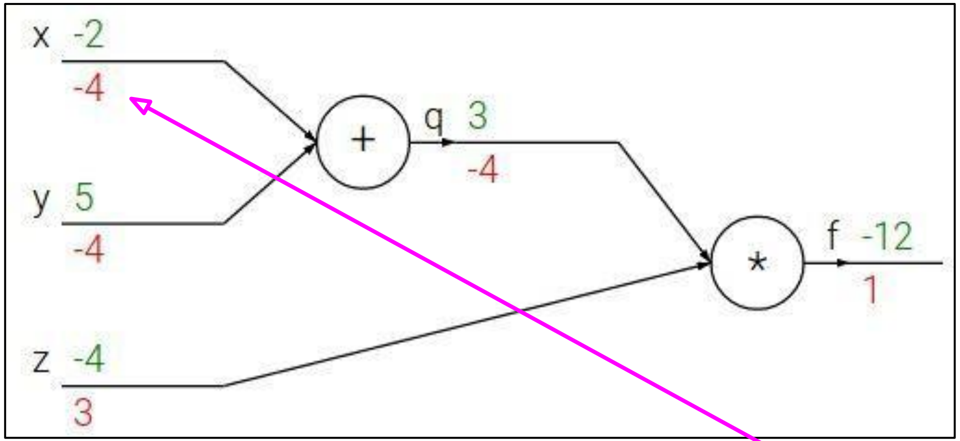


$$\frac{\partial f}{\partial x}$$

Computation Graph: Backward

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

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



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

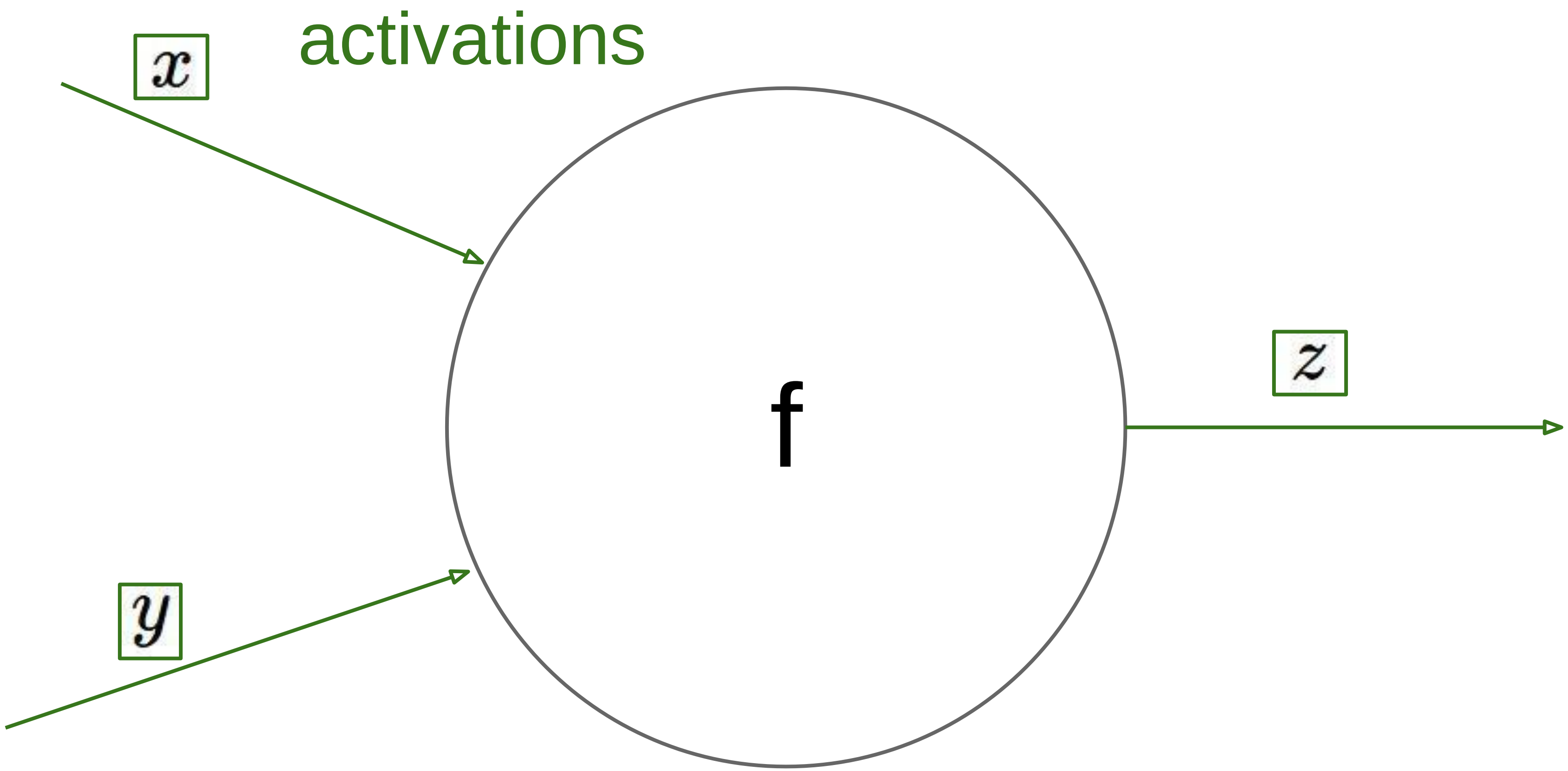
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

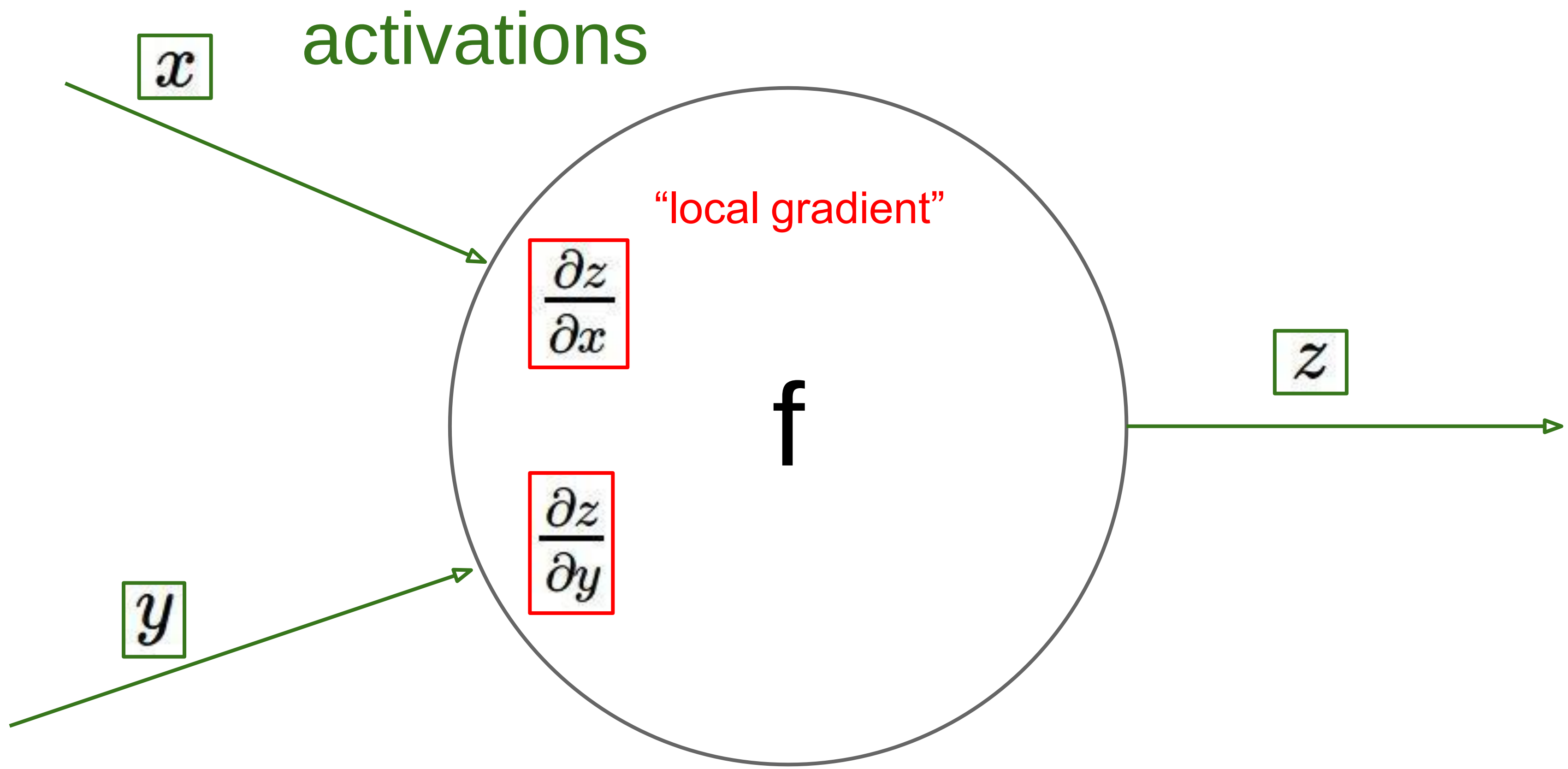
$$\boxed{\frac{\partial f}{\partial x}}$$

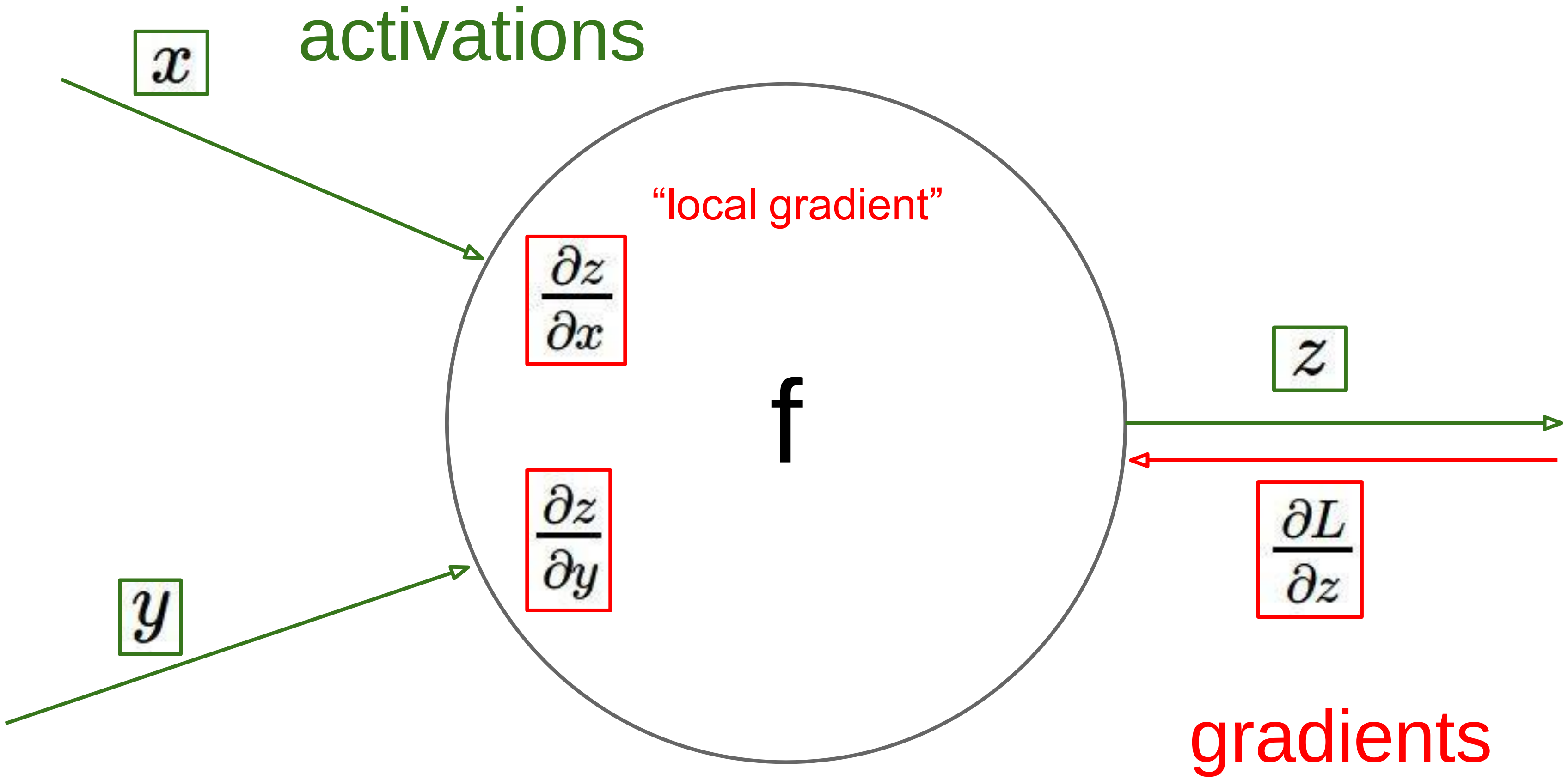
Chain rule:

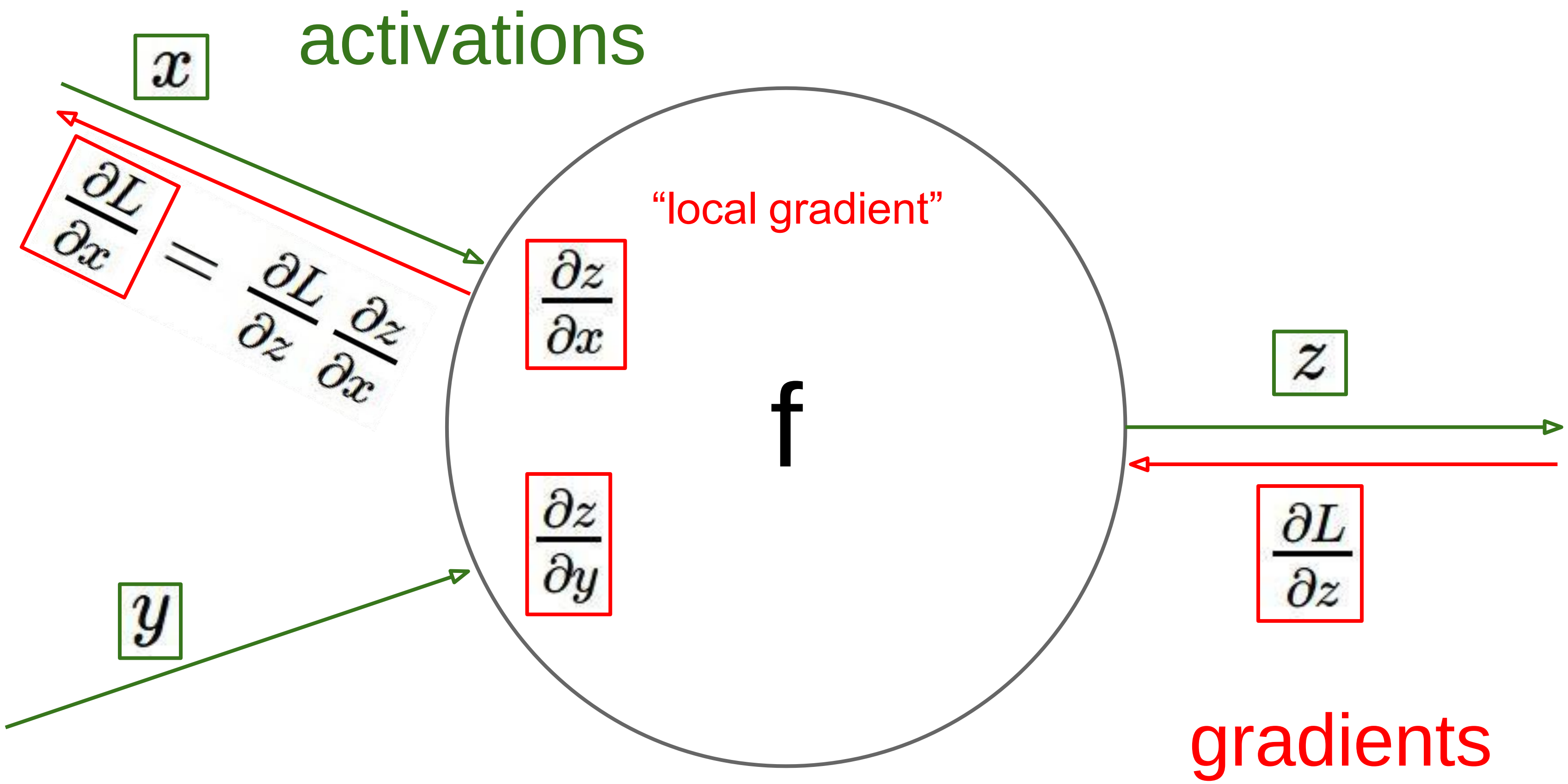
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

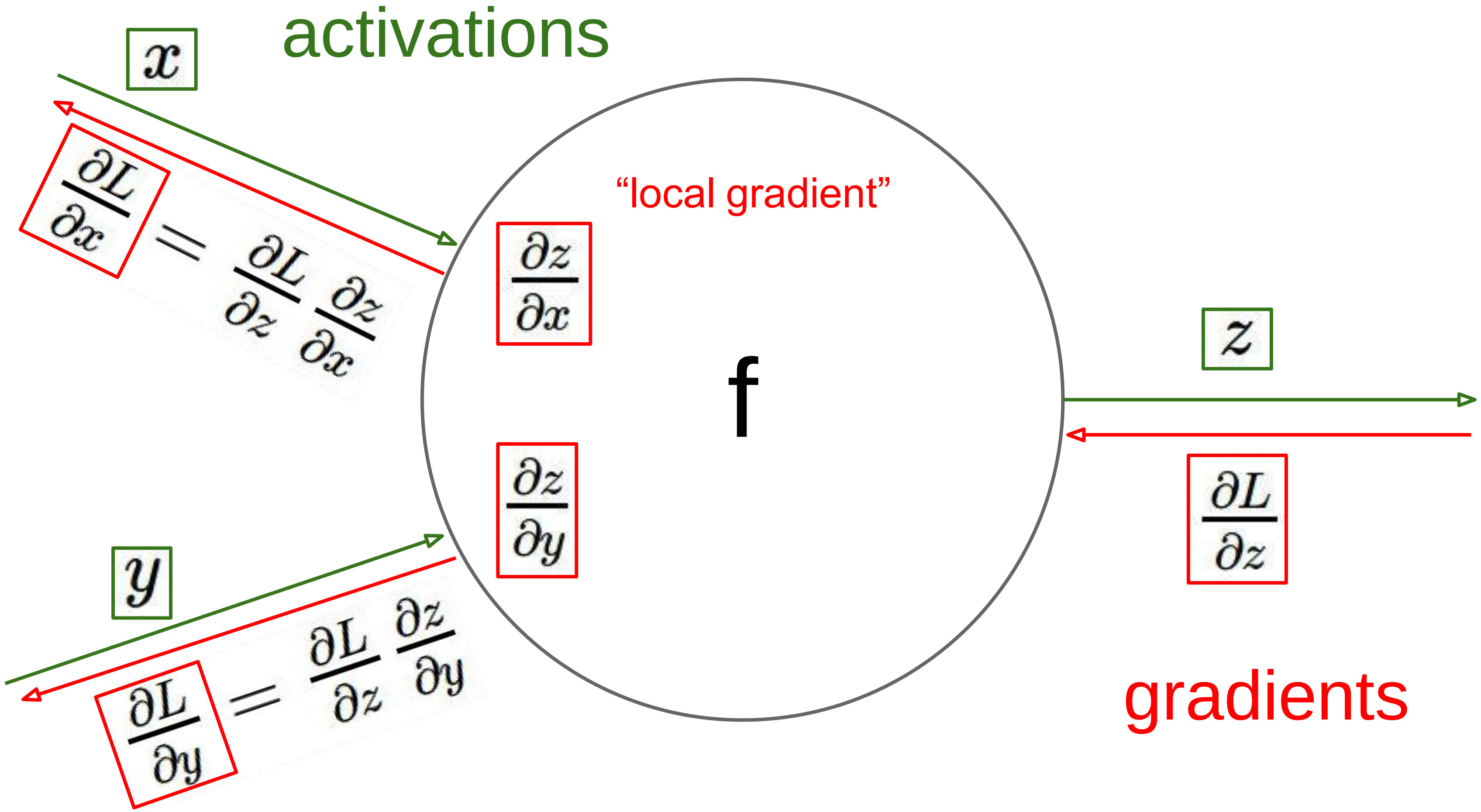
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

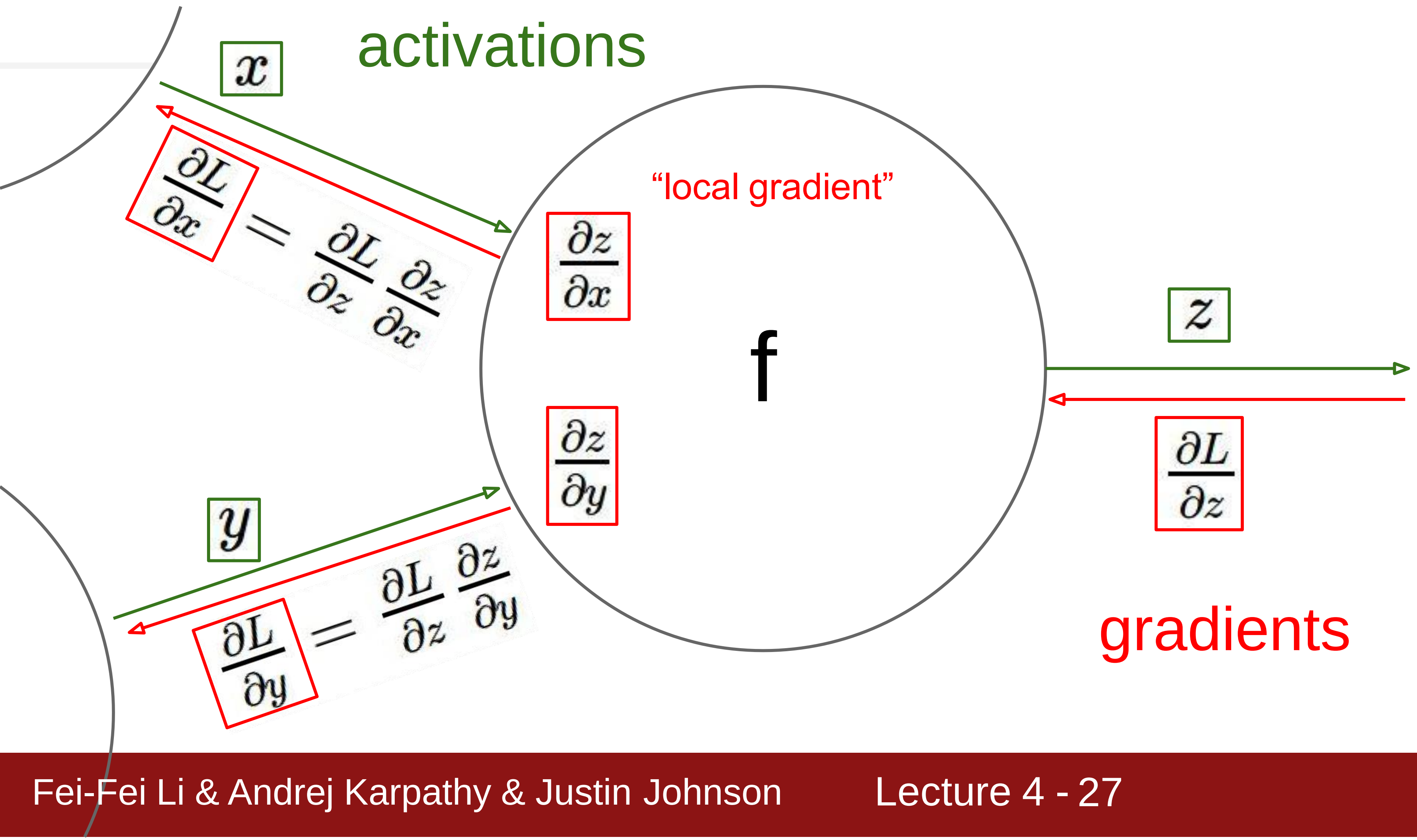

activations
x
y
f
z


activations
x
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
y


activations
x
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
y
$\frac{\partial L}{\partial z}$
gradients


activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
gradients


activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients


activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial y}$
gradients

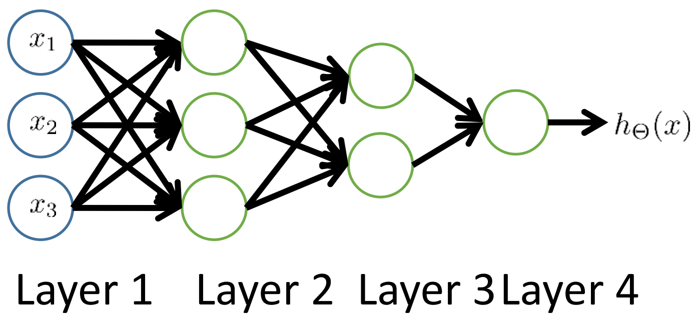
Neural Networks II

Architectures and Learning

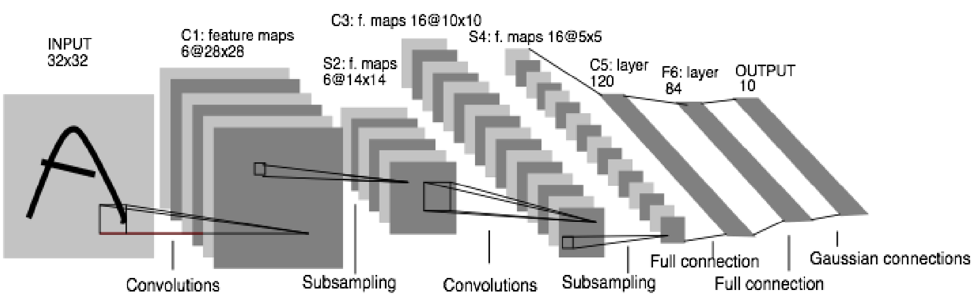
Network architectures

Feed-forward

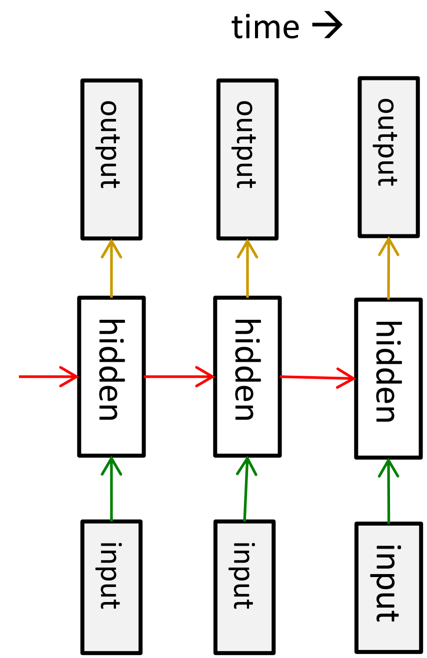
Fully connected



Convolutional



Recurrent



Summary so far

- Neural network chains together many layers of “neurons” such as logistic units
- Hidden neurons learn more and more abstract non-linear features
- Backpropagation is the algorithm used to compute the partial derivatives used to update parameters in the learning process.