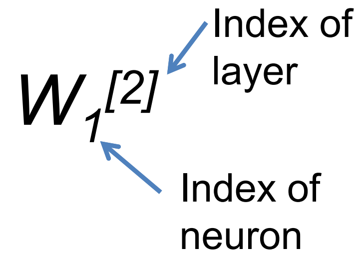
Deep Learning Crash Course

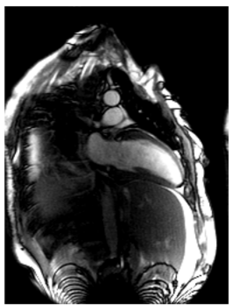
www.deeplearningcrashcourse.org

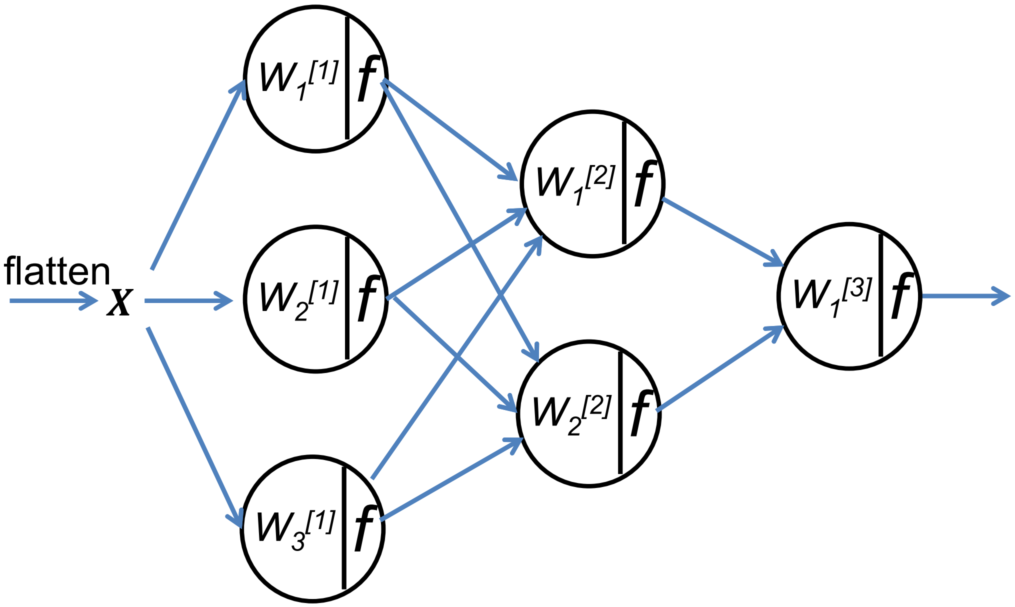
Hui Xue

Fall 2021

Recap : MLP







$\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]}$

$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$
$$\boldsymbol{a}^{[1]} = \boldsymbol{f}(\boldsymbol{Z}^{[1]})$$
$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$
$$\boldsymbol{a}^{[2]} = \boldsymbol{f}(\boldsymbol{Z}^{[2]})$$
$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$
$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]} = \boldsymbol{f}(\boldsymbol{Z}^{[3]})$$

The i-th sample $(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)})$

Recap : Cross-Entropy Loss

Given two probability distribution p and q, cross-entropy:

$$\boldsymbol{H(p,q) = -\sum_{for\ all\ possible\ x} p(x) log[q(x)]}$$

Multi-class classification:

$$\boldsymbol{L_{CE\_loss}(y,\hat{y}) = -\sum_{k=0}^{4} y[k]\, log(\hat{y}[k])}$$

$$\boldsymbol{L_{CE\_loss}(y,\hat{y}) = -log(\hat{y}[y_k])}$$

Binary classification:

$$\boldsymbol{L_{BCE\_loss}(y,\hat{y}) = -[y \cdot log(\hat{y}) + (1-y) \cdot log(1-\hat{y})]}$$

Recap : Mini-batch Gradient Descent

Initialize weights and bias

Random shuffle dataset

BatchSize = 32

for epoch in range(E):

  select #BatchSize **samples**

  Evaluate loss function (forward pass) at this **batch**

  $L = \frac{1}{B}\sum_{i=0}^{B-1} L(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}(\boldsymbol{W}, \boldsymbol{b}))$

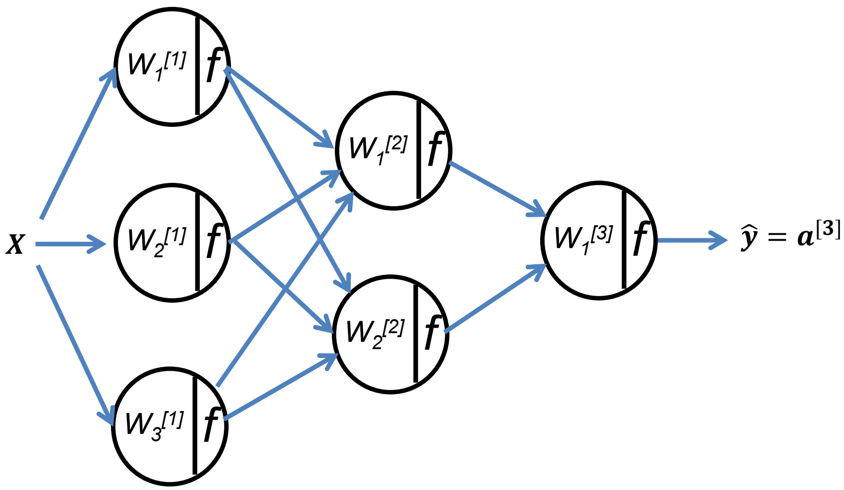
  Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[l]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[l]}}$

  Update parameter $\boldsymbol{W}^{[l]} = \boldsymbol{W}^{[l]} - \alpha \frac{\partial L}{\partial \boldsymbol{W}^{[l]}}, \ \boldsymbol{b}^{[l]} = \boldsymbol{b}^{[l]} - \alpha \frac{\partial L}{\partial \boldsymbol{b}^{[l]}}$

Lecture 3

- **Backprop**

To find the good model parameter



Evaluate loss function (forward pass) at this **batch**

$$L = \frac{1}{B}\sum_{i=0}^{B-1} L(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}(\boldsymbol{W}, \boldsymbol{b}))$$

Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[l]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[l]}}$

Update parameter $\boldsymbol{W}^{[l]} = \boldsymbol{W}^{[l]} - \alpha \frac{\partial L}{\partial \boldsymbol{W}^{[l]}}, \ \boldsymbol{b}^{[l]} = \boldsymbol{b}^{[l]} - \alpha \frac{\partial L}{\partial \boldsymbol{b}^{[l]}}$

Evaluate loss function : A forward pass

Evaluate loss function (forward pass) at this **batch**

$$L = \frac{1}{B}\sum_{i=0}^{B-1} L(\boldsymbol{y}^{(\boldsymbol{i})}, \hat{\boldsymbol{y}}^{(\boldsymbol{i})}(\boldsymbol{W}, \boldsymbol{b}))$$

Let's look at one sample and write $\boldsymbol{y}^{(\boldsymbol{i})}$ as $\boldsymbol{y}$:

$$L\big(y, \hat{y}(\boldsymbol{W}, \boldsymbol{b})\big) = -[y\,\boldsymbol{log}(\hat{y}) + (1-y)\,\boldsymbol{log}(1-\hat{y})]$$

For every sample, we input $(\boldsymbol{X}, y)$ and model outputs $\hat{y}$

$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{f}(\boldsymbol{Z}^{[1]})$$

$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{f}(\boldsymbol{Z}^{[2]})$$

$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$

$$\hat{y} = a^{[3]} = \boldsymbol{f}(\boldsymbol{Z}^{[3]})$$

Compute gradient : Is it easy to do?

Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[3]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[3]}}$

$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{f}(\boldsymbol{Z}^{[1]})$$

$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{f}(\boldsymbol{Z}^{[2]})$$

$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]} = \boldsymbol{f}(\boldsymbol{Z}^{[3]})$$

First idea, compute analytical derivative

$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]} = \boldsymbol{f}\left(\boldsymbol{Z}^{[3]}\right)$$

$$= \boldsymbol{f}\left(\boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}\right)$$

$$= \boldsymbol{f}\left(\boldsymbol{W}^{[3]}\boldsymbol{f}(\boldsymbol{Z}^{[2]}) + \boldsymbol{b}^{[3]}\right)$$

$$= \boldsymbol{f}\left(\boldsymbol{W}^{[3]}\boldsymbol{f}(\boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}) + \boldsymbol{b}^{[3]}\right)$$

$$= \boldsymbol{f}\left(\boldsymbol{W}^{[3]}\boldsymbol{f}(\boldsymbol{W}^{[2]}\boldsymbol{f}(\boldsymbol{Z}^{[1]}) + \boldsymbol{b}^{[2]}) + \boldsymbol{b}^{[3]}\right)$$

$$= \boldsymbol{f}\left(\boldsymbol{W}^{[3]}\boldsymbol{f}(\boldsymbol{W}^{[2]}\boldsymbol{f}(\boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}) + \boldsymbol{b}^{[2]}) + \boldsymbol{b}^{[3]}\right)$$

Compute analytical gradient : hard to do

Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[3]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[3]}}$

$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{f}(\boldsymbol{Z}^{[1]})$$

$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{f}(\boldsymbol{Z}^{[2]})$$

$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]} = \boldsymbol{f}(\boldsymbol{Z}^{[3]})$$

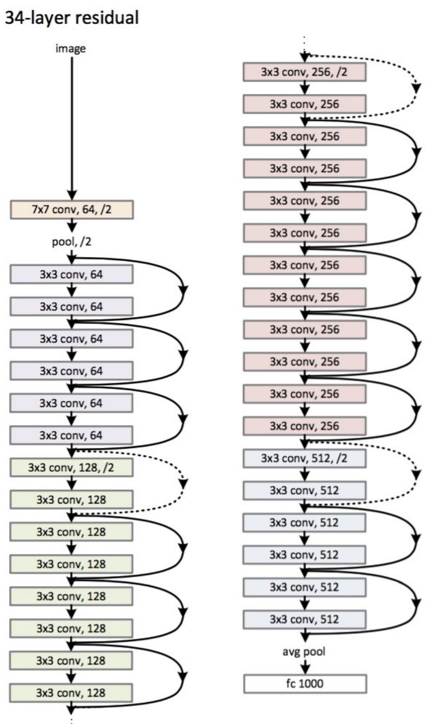
So, let's compute:

$$\frac{\partial L}{\partial \boldsymbol{W}^{[1]}} = -\frac{\partial\{[\boldsymbol{ylog}(\hat{\boldsymbol{y}}) + (\boldsymbol{1} - \boldsymbol{y})\log(\boldsymbol{1} - \hat{\boldsymbol{y}})]\}}{\partial \boldsymbol{W}^{[1]}}$$

$$= -\frac{\partial\left\{\begin{bmatrix} \boldsymbol{ylog}\left(\boldsymbol{f}\left(\boldsymbol{W}^{[3]}\boldsymbol{f}(\boldsymbol{W}^{[2]}\boldsymbol{f}(\boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}) + \boldsymbol{b}^{[2]}) + \boldsymbol{b}^{[3]}\right)\right) \\ +(\boldsymbol{1} - \boldsymbol{y})\log\left(\boldsymbol{1} - \boldsymbol{f}\left(\boldsymbol{W}^{[3]}\boldsymbol{f}(\boldsymbol{W}^{[2]}\boldsymbol{f}(\boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}) + \boldsymbol{b}^{[2]}) + \boldsymbol{b}^{[3]}\right)\right) \end{bmatrix}\right\}}{\partial \boldsymbol{W}^{[1]}}$$

Compute derivative gradient is hard … for this small MLP

Compute derivative gradient : cannot be done



Deep Residual Learning for Image Recognition. https://arxiv.org/pdf/1512.03385.pdf

Going Deeper with Convolutions. CVPR 2015

Compute analytical gradient becomes impossible for deep neural networks !

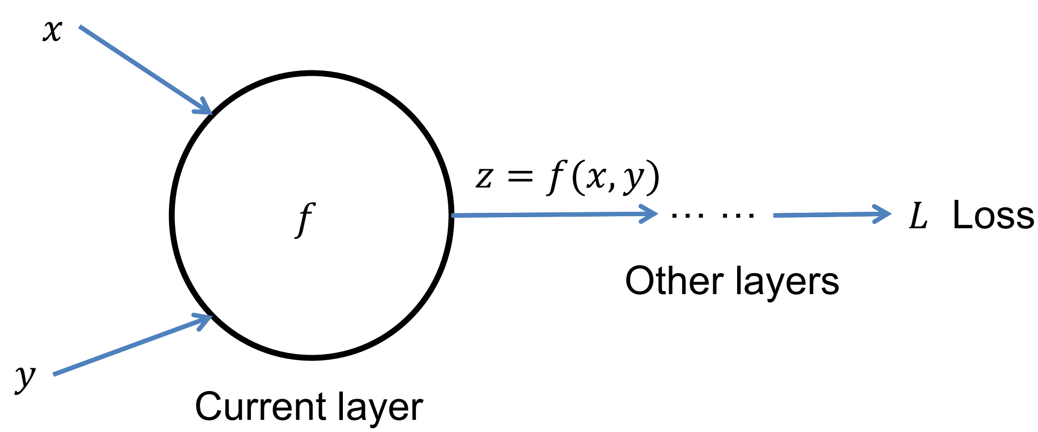
Don’t try it!

Compute gradient : Back propagation

Compute derivative gradient becomes impossible for deep neural networks. We need back prop.

Three ideas behind BackProp:

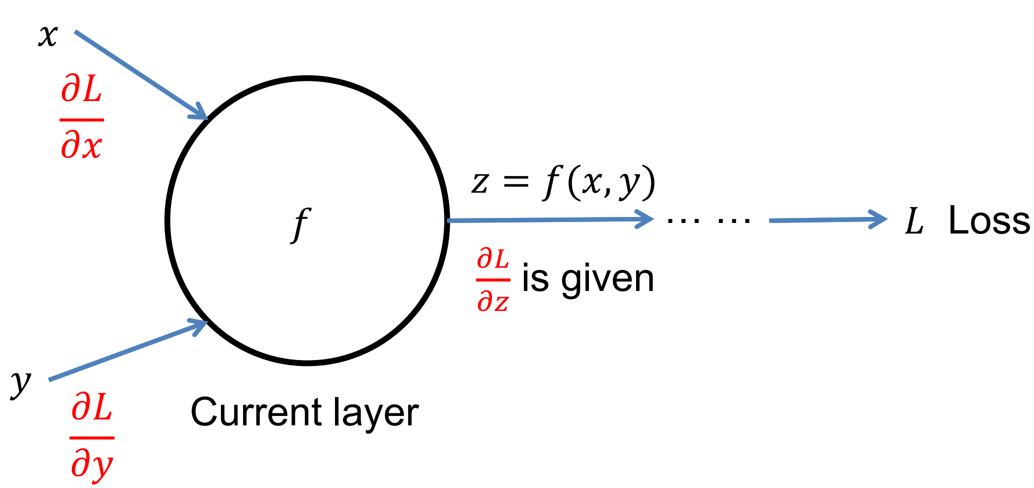
- Divide and conquer
- Chain-rule
- Let computer do the job - AutoDiff



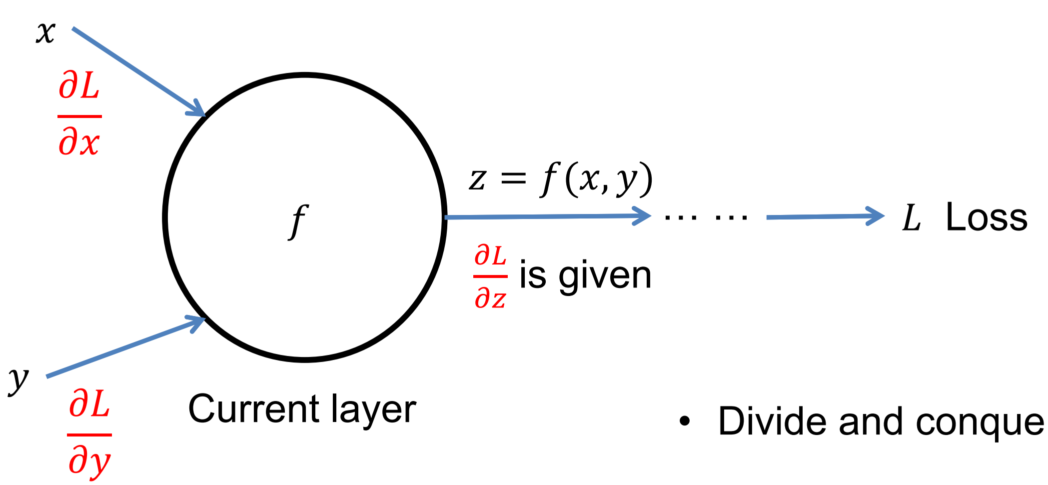
Compute gradient : Back propagation

For this current layer, what we want to do:

- Given the gradient from the loss to output parameter z
- Compute gradient from the loss to input x and y



Compute gradient : Back propagation



- Chain rule

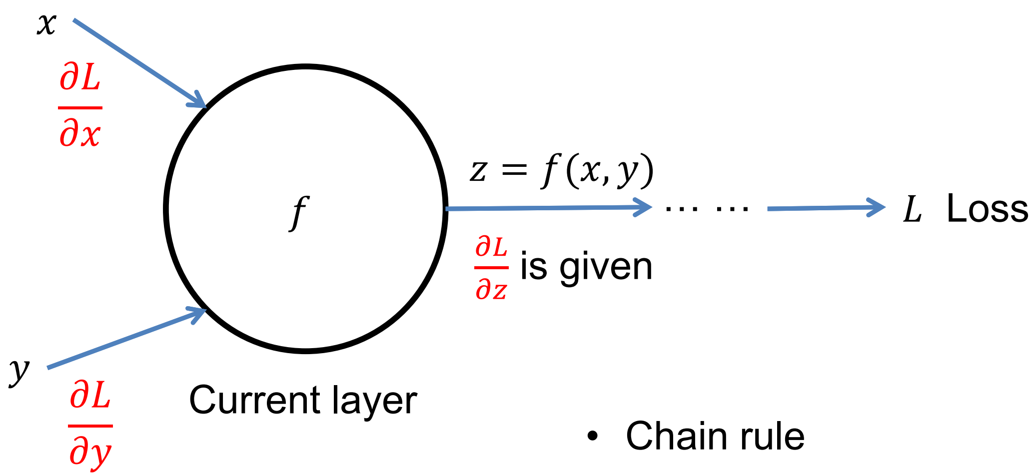
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$$

- Divide and conquer

We only need to know:
1) the current layer to produce z from x and y
2) the upstream gradient

Compute gradient : Back propagation



- Divide and conquer

  We know the current **function** to compute z from x and y

  We know the gradient of loss to output z

  Enough information to compute derivative of loss to input x and y

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

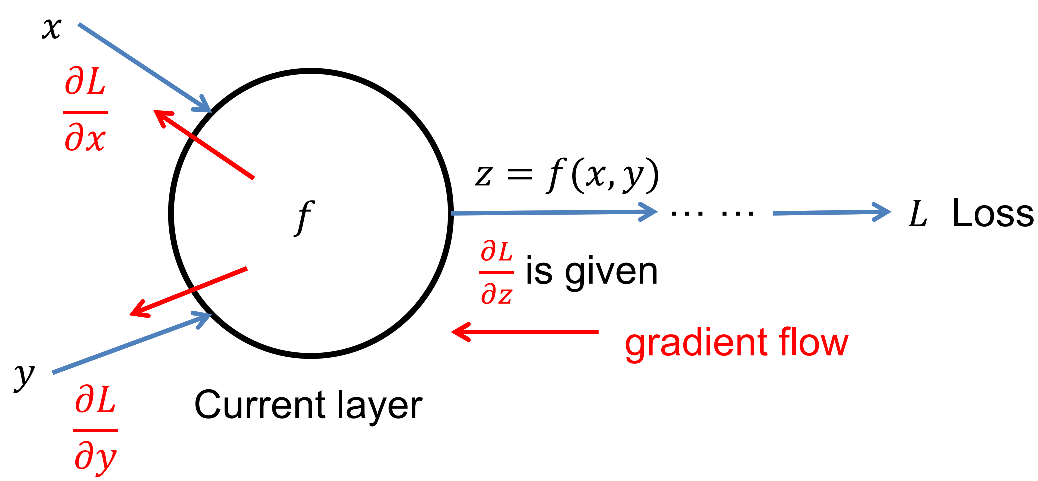
- Chain rule

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

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

Compute gradient : Back propagation

To compute current gradient, only need **upstream gradient** and **local function**

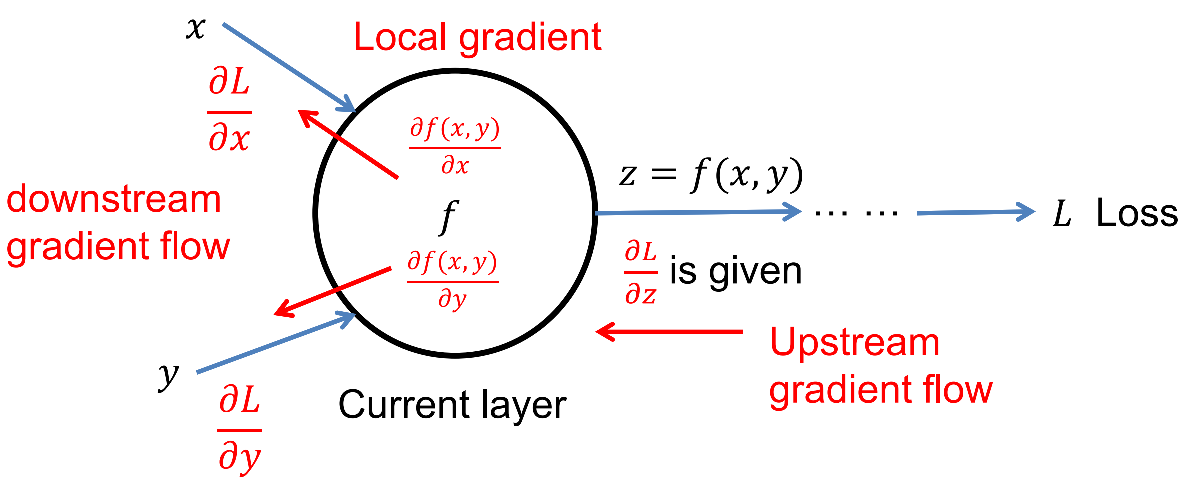


$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y} = \boxed{\frac{\partial L}{\partial z}}\boxed{\frac{\partial f(x,y)}{\partial y}}$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x} = \boxed{\frac{\partial L}{\partial z}}\boxed{\frac{\partial f(x,y)}{\partial x}}$$

Compute gradient : Back propagation

To compute current gradient, only need **upstream gradient** and **local gradient**



1) Inputs flow through the model to compute loss

2) Gradient of loss flows back to compute gradient to inputs (and all others …)

3) Upstream gradient is combined with local gradient to compute downstream gradient

Computational Graph

$$L = x^2 + 2xy \qquad x = 1, y = 4$$

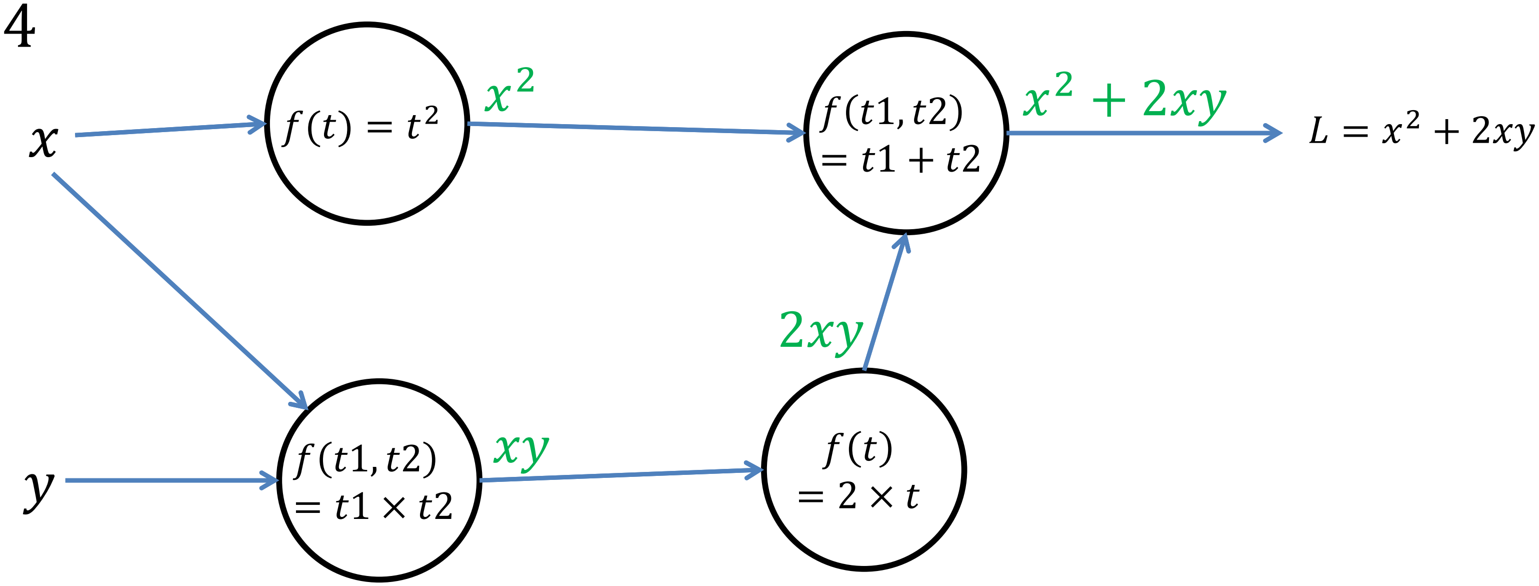
$$\frac{\partial L}{\partial x} = 2x + 2y = 2 + 8 = 10$$

$$\frac{\partial L}{\partial y} = 2x = 2$$

Computational Graph

$L = x^2 + 2xy$

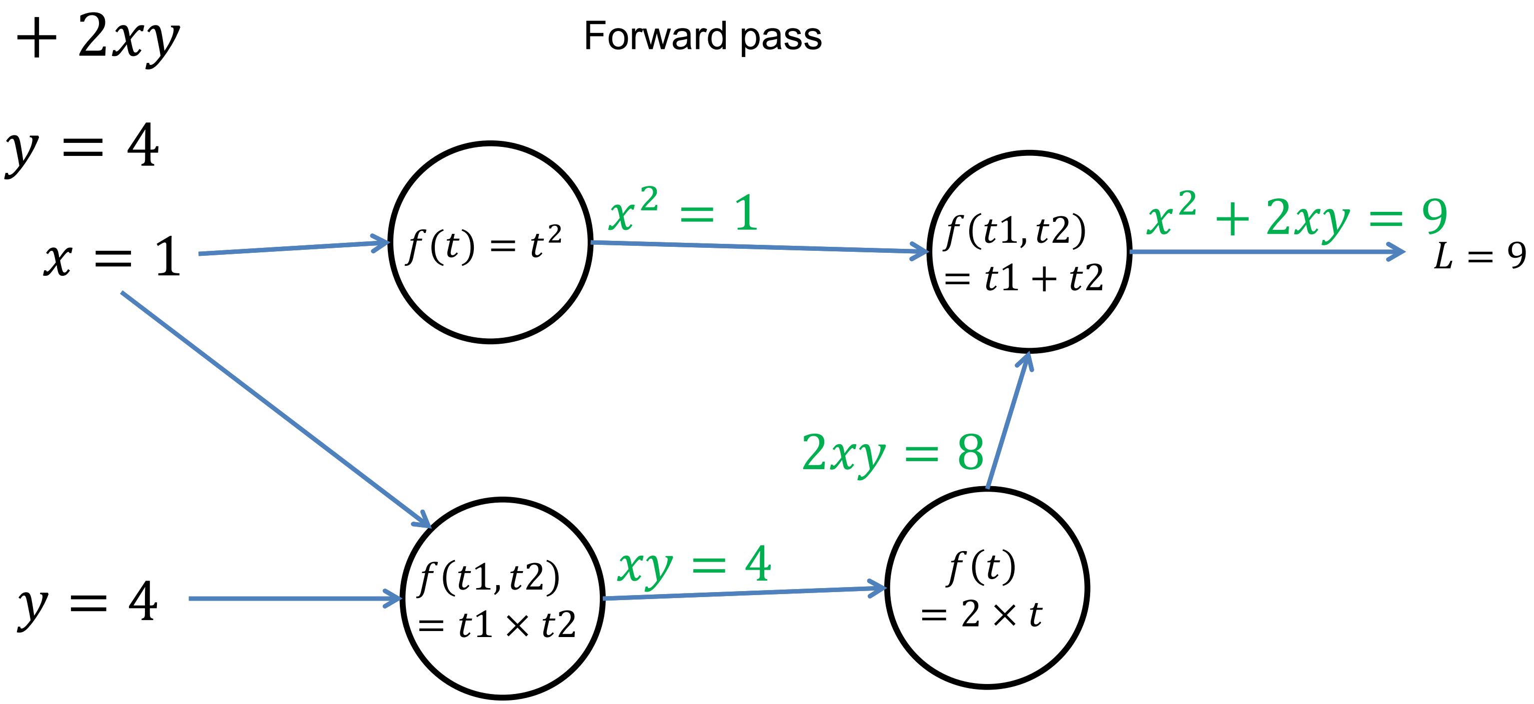
$x = 1, y = 4$



Computational Graph

$L = x^2 + 2xy$

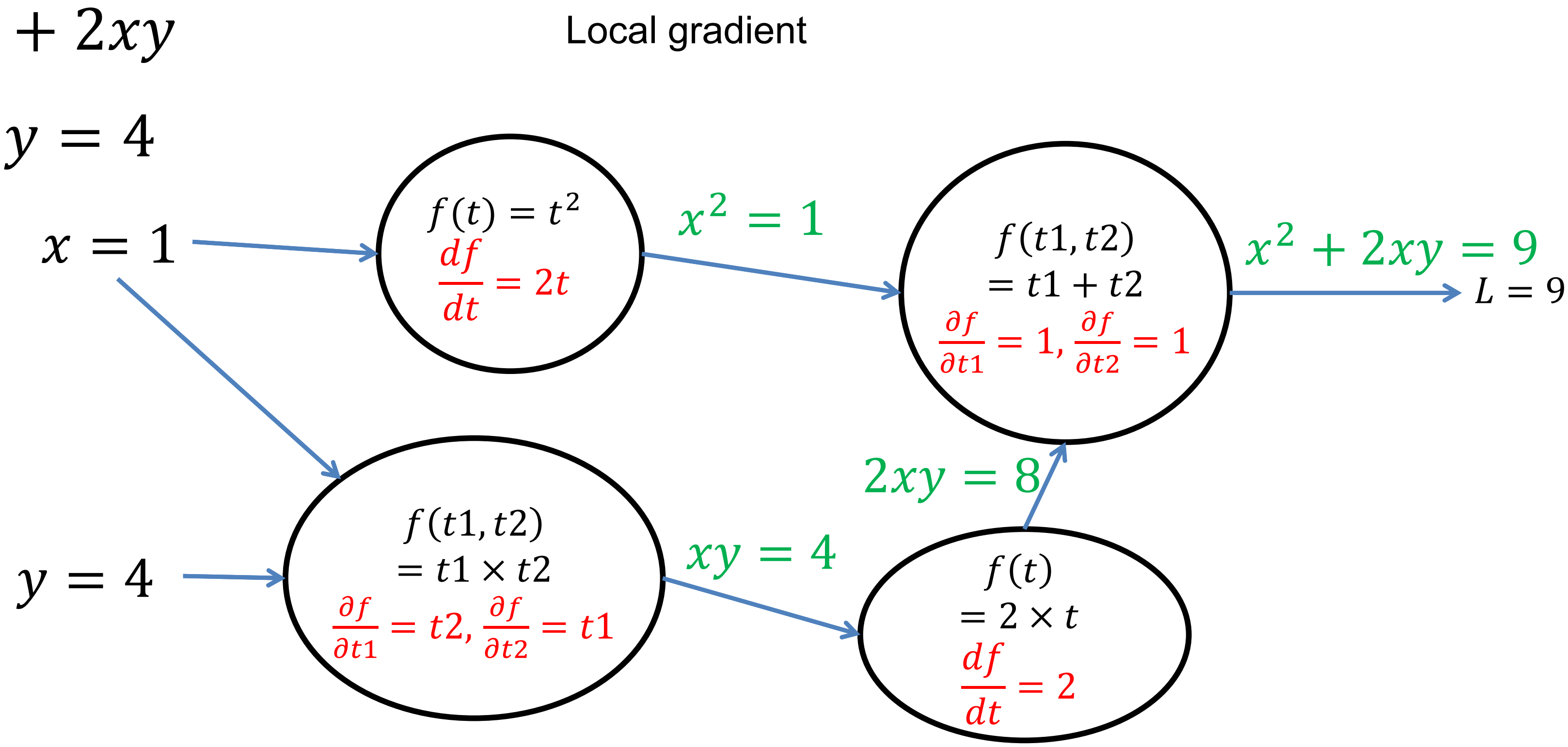
$x = 1, y = 4$



Computational Graph

$L = x^2 + 2xy$

$x = 1, y = 4$

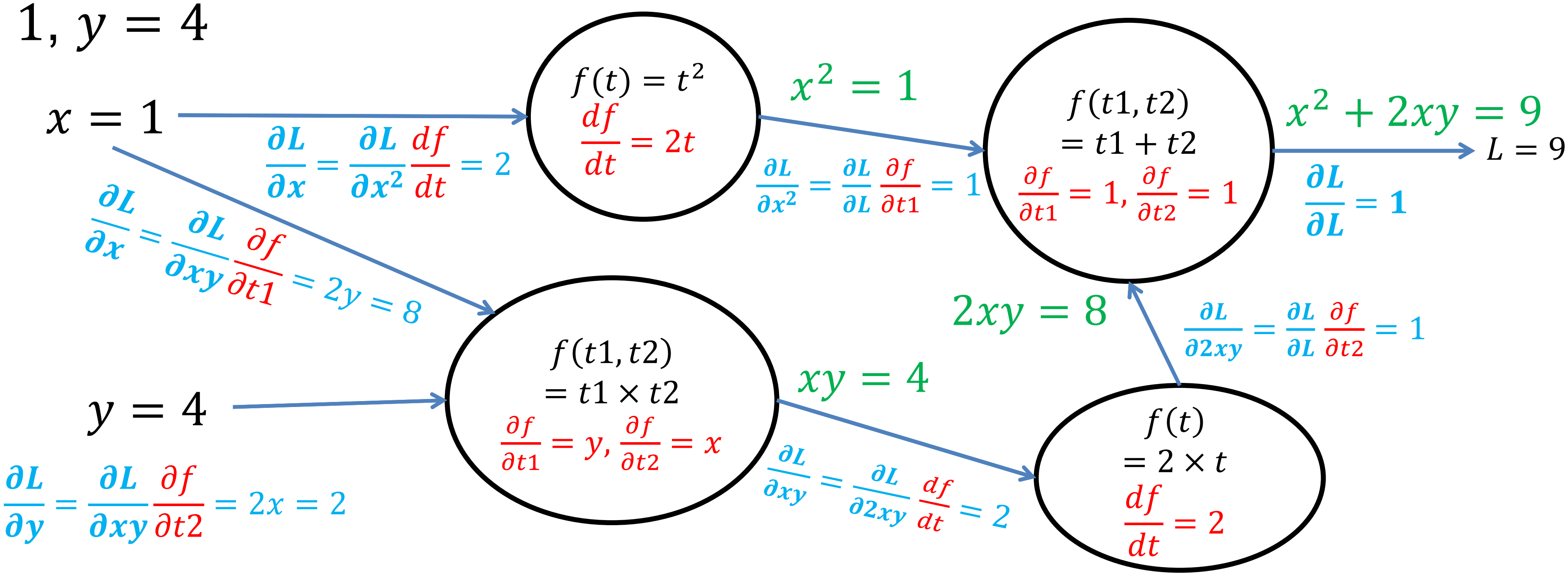


Computational Graph

$$L = x^2 + 2xy$$

Back prop

$$x = 1,\ y = 4$$



Computational Graph

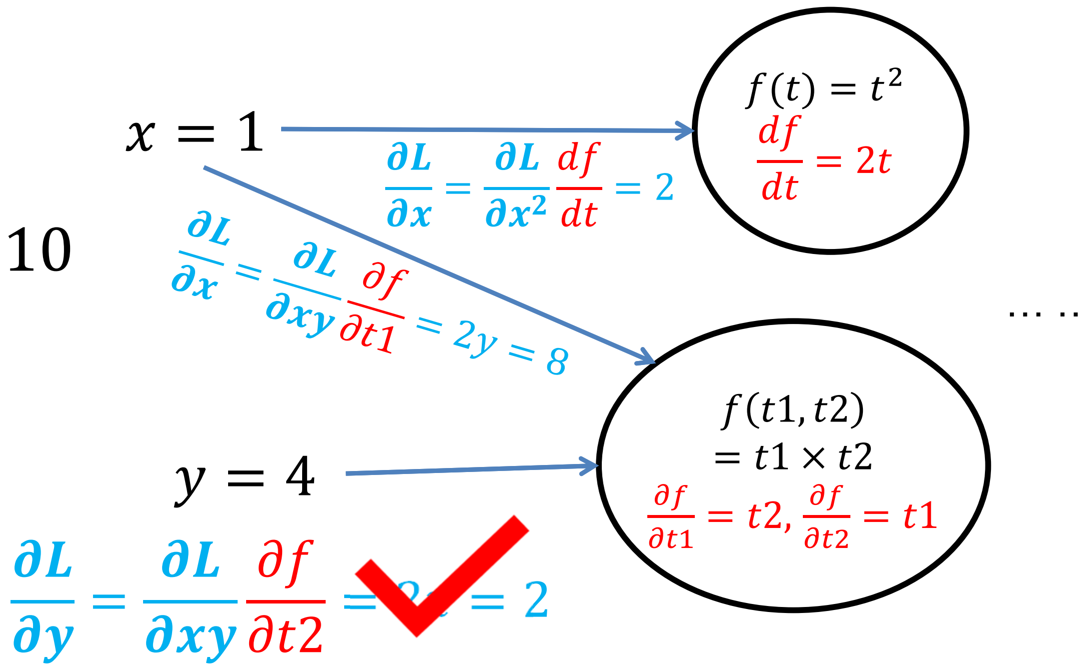
$$L = x^2 + 2xy$$

$$x = 1, y = 4$$

$$\frac{\partial L}{\partial x} = 2x + 2y = 2 + 8 = 10$$

$$\frac{\partial L}{\partial y} = 2x = 2$$

Back prop



Computational Graph

$$L = x^2 + 2xy$$

$$x = 1,\ y = 4$$

$$\frac{\partial L}{\partial x} = 2x + 2y = 2 + 8 = 10$$

Back prop

$x = 1$

$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial x^2}\frac{df}{dt} = 2$

$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial xy}\frac{\partial f}{\partial t1} = 2y = 8$

$f(t) = t^2$

$\frac{df}{dt} = 2t$

$f(t1, t2)$
$= t1 \times t2$

$\frac{\partial f}{\partial t1} = t2, \frac{\partial f}{\partial t2} = t1$

… …

Two paths contribute to one variable :

Sum contributions from all paths

Derivative of Scalar to Vector and Matrix

Loss is a scalar $L \in \mathbb{R}$

If x is a scalar, then

$\frac{\partial L}{\partial x}$ is a scalar

Derivative of a scalar function to a scalar variable is a scalar.

If $\boldsymbol{x}$ is a vector, $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{N \times 1}}$

$\frac{\partial L}{\partial \boldsymbol{x}}$ is a vector in the same shape as $\boldsymbol{x}$

$$\boldsymbol{x} = [x_0 \quad x_1 \quad \ldots . \; x_{N-2} \quad x_{N-1}]^T$$

$$\frac{\partial L}{\partial \boldsymbol{x}} = [\frac{\partial L}{\partial x_0} \quad \frac{\partial L}{\partial x_1} \quad \ldots . \frac{\partial L}{\partial x_{N-2}} \quad \frac{\partial L}{\partial x_{N-1}}]^T$$

If $\boldsymbol{x}$ is a matrix, $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{N \times M}}$

$\frac{\partial L}{\partial \boldsymbol{x}}$ still has in the same shape as $\boldsymbol{x}$

$$\boldsymbol{x} = \left[\begin{array}{cccc} x_{00} & x_{01} & \ldots & x_{0,M-1} \\ x_{10} & x_{11} & \ldots & x_{1,M-1} \\ \ldots & \ldots & \ldots & \ldots \\ x_{N-1,0} & x_{N-1,M-1} & \ldots & x_{N-1,M-1} \end{array} \right]$$

$$\frac{\partial L}{\partial \boldsymbol{x}} = [\frac{\partial L}{\partial x_{ij}}]_{i=0:N-1, j=0:M-1}$$

Vector to Vector : Jacobian matrix

$f \in \mathbb{R}^N \rightarrow \mathbb{R}^M$ $\quad \boldsymbol{y} = f(\boldsymbol{x})$ $\quad \boldsymbol{x} \in \mathbb{R}^N$ $\quad \boldsymbol{y} \in \mathbb{R}^M$

$\boldsymbol{x}$ is a vector of length N $\quad \boldsymbol{x} = [x_0 \quad x_1 \quad \ldots . \; x_{N-2} \quad x_{N-1}]^T$

$\boldsymbol{y}$ is a vector of length M $\quad \boldsymbol{y} = [y_0 \quad y_1 \quad \ldots . \; y_{M-2} \quad x_{M-1}]^T$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = [\frac{\partial y_i}{\partial x_j}]_{i=0:M-1, j=0:N-1}$$

$$\begin{bmatrix} \frac{\partial y_0}{\partial x_0} & \cdots & \frac{\partial y_0}{\partial x_{N-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_{M-1}}{\partial x_0} & \cdots & \frac{\partial y_{M-1}}{\partial x_{N-1}} \end{bmatrix}$$

Jacobian matrix:

MxN

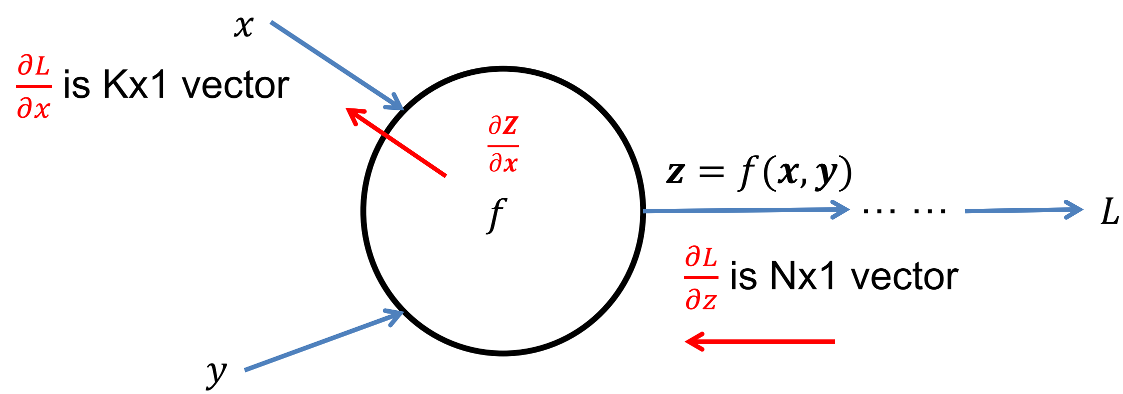
output length x input length

Derivative of every element of y to every element of x

Backprop for Vector

Apply Jacobian matrix as local gradient

$\boldsymbol{x} \in \mathbb{R}^K \qquad \boldsymbol{z} \in \mathbb{R}^N \qquad \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{x}}$ is the Jacobian matrix NxK



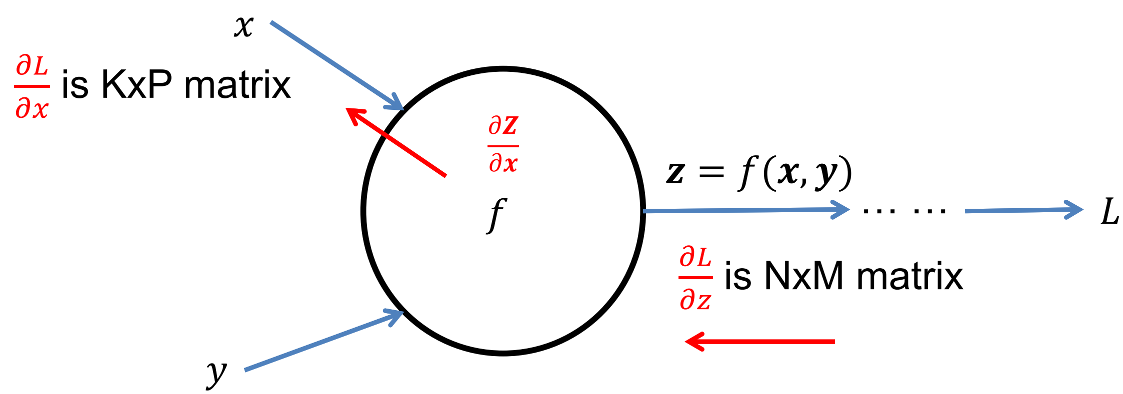
- Unlike the scalar case, backprop for vectors requires matrix multiplication
- Only one way to put together equation

$$\frac{\partial L}{\partial x} = \left[\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{x}}\right]^T \cdot \frac{\partial L}{\partial \boldsymbol{z}} \qquad \frac{\partial L}{\partial y} = \left[\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{y}}\right]^T \cdot \frac{\partial L}{\partial \boldsymbol{z}}$$

Backprop for Matrix and Tensors

$\boldsymbol{x} \in \mathbb{R}^{K \times P}$ $\boldsymbol{z} \in \mathbb{R}^{N \times M}$

$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}$ is the Jacobian Tensor
(NxM) x (KxP)

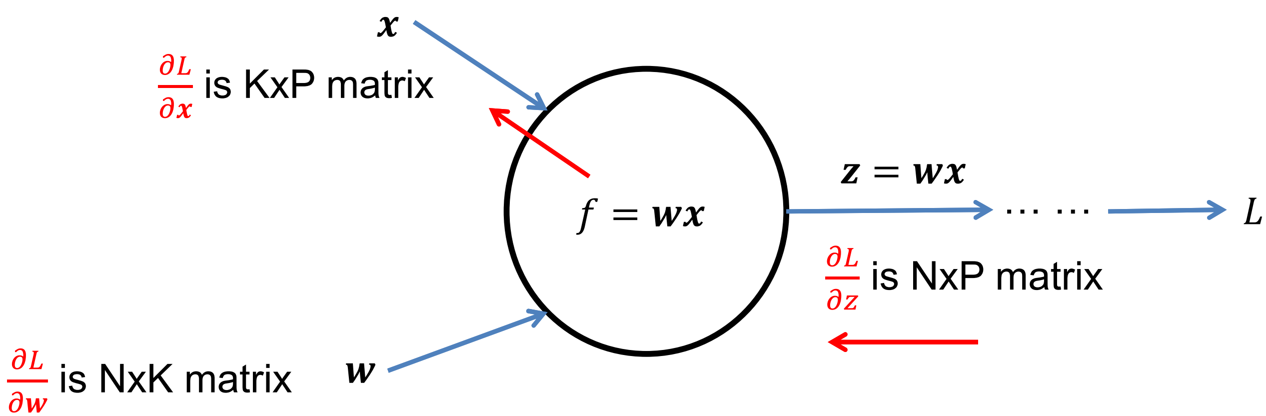


- Often use backprop to computer Jacobian Tensor
- Still only one way to put together equation

$$\frac{\partial L}{\partial \boldsymbol{x}} = \left[\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right]^T \cdot \frac{\partial L}{\partial \boldsymbol{z}} \qquad \frac{\partial L}{\partial \boldsymbol{y}} = \left[\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}}\right]^T \cdot \frac{\partial L}{\partial \boldsymbol{z}}$$

Backprop for Matrix multiplication

$$\boldsymbol{x} \in \mathbb{R}^{K\times P} \quad \boldsymbol{w} \in \mathbb{R}^{N\times K} \quad \boldsymbol{z} \in \mathbb{R}^{N\times P}$$

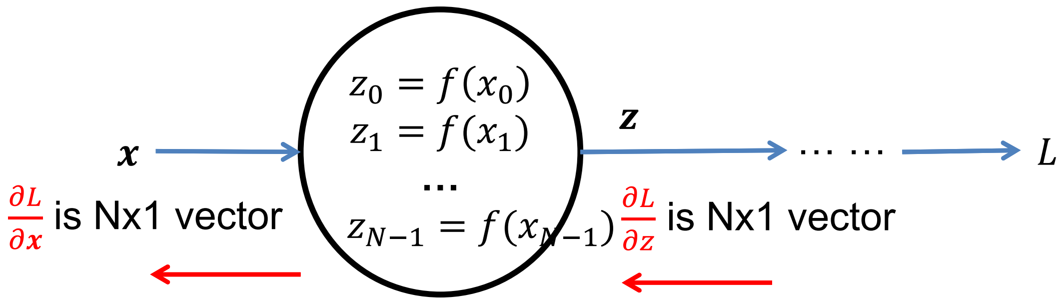


- Matrix multiplication formula is useful
- Only one way to put together them

$$\frac{\partial L}{\partial \boldsymbol{x}} = w^T \cdot \frac{\partial L}{\partial \boldsymbol{z}} \qquad \frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \boldsymbol{z}} \cdot \boldsymbol{x}^T$$

Backprop for Element-wise operation

$\boldsymbol{x} \in \mathbb{R}^{N\times 1}$ $\quad$ $\boldsymbol{z} \in \mathbb{R}^{N\times 1}$

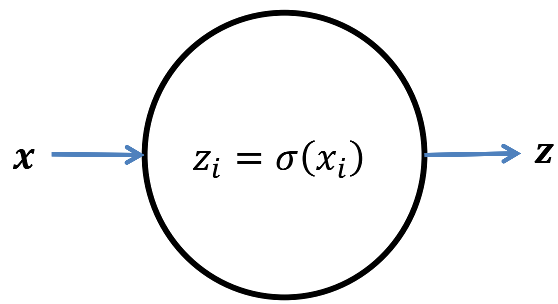


- Simply compute backprop element by element

$$\frac{\partial L}{\partial x_i} = \frac{df(x_i)}{dx_i}\frac{\partial L}{\partial z_i} \qquad i = 0{:}\, N-1$$

Backprop for Sigmoid

$\boldsymbol{x} \in \mathbb{R}^{N\times 1}$ $\quad$ $\boldsymbol{z} \in \mathbb{R}^{N\times 1}$ $\quad$ $\frac{\partial L}{\partial x_i} = \frac{df(x_i)}{dx_i}\frac{\partial L}{\partial z_i}$ $\quad$ $i = 0{:}\,N-1$



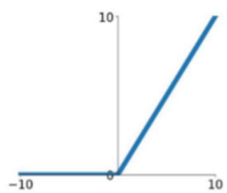
- Simply compute backprop element by element

$$\frac{d\sigma(t)}{dt} = \sigma(t)[1-\sigma(t)] \qquad \frac{\partial L}{\partial x_i} = \sigma(x_i)[1-\sigma(x_i)]\frac{\partial L}{\partial z_i}$$

Backprop for ReLU activation function

$\boldsymbol{x} \in \mathbb{R}^{N\times 1}$ $\quad$ $\boldsymbol{z} \in \mathbb{R}^{N\times 1}$ $\quad$ $\frac{\partial L}{\partial x_i} = \frac{df(x_i)}{dx_i}\frac{\partial L}{\partial z_i}$ $\quad$ $i = 0{:}\,N-1$

ReLU
$\max(0, x)$



$$\boldsymbol{x} \rightarrow z_i = max(0, x_i) \rightarrow \boldsymbol{z}$$

- compute backprop element by element

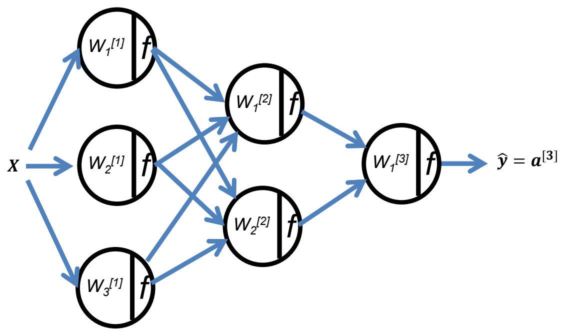
$$\frac{dmax(0,t)}{dt} = \begin{cases} 1, t > 0 \\ 0.5, t = 0 \\ 0, otherwise \end{cases}$$

$$\frac{\partial L}{\partial x_i} = \begin{cases} \frac{\partial L}{\partial z_i}, x_i > 0 \\ \frac{\partial L}{2\partial z_i}, x_i = 0 \\ 0, x_i < 0 \end{cases}$$

Computational Graph for MLP

Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[3]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[3]}}$

$$\frac{\partial L}{\partial \boldsymbol{x}} = \boldsymbol{w}^T \cdot \frac{\partial L}{\partial \boldsymbol{z}} \qquad \frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \boldsymbol{z}} \cdot \boldsymbol{x}^T \qquad \boldsymbol{z} = \boldsymbol{w}\boldsymbol{x}$$



$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$

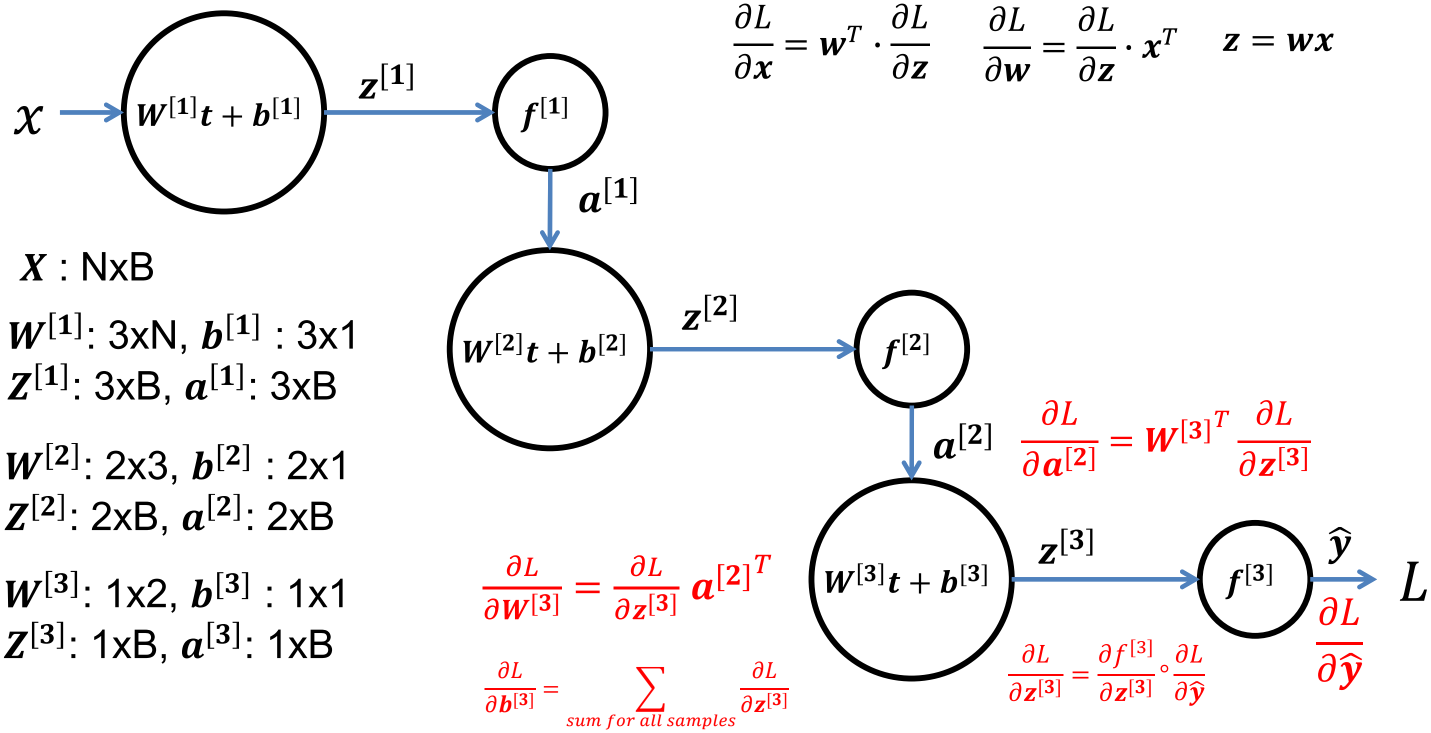
$$\boldsymbol{a}^{[1]} = \boldsymbol{f}(\boldsymbol{Z}^{[1]})$$

$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{f}(\boldsymbol{Z}^{[2]})$$

$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]} = \boldsymbol{f}(\boldsymbol{Z}^{[3]})$$



$\boldsymbol{X}$: NxB

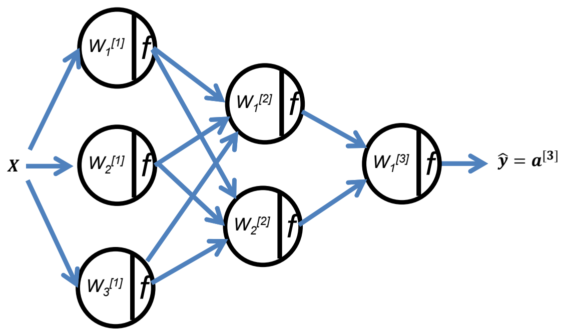
$\boldsymbol{W}^{[1]}$: 3xN, $\boldsymbol{b}^{[1]}$: 3x1
$\boldsymbol{Z}^{[1]}$: 3xB, $\boldsymbol{a}^{[1]}$: 3xB

$\boldsymbol{W}^{[2]}$: 2x3, $\boldsymbol{b}^{[2]}$: 2x1
$\boldsymbol{Z}^{[2]}$: 2xB, $\boldsymbol{a}^{[2]}$: 2xB

$\boldsymbol{W}^{[3]}$: 1x2, $\boldsymbol{b}^{[3]}$: 1x1
$\boldsymbol{Z}^{[3]}$: 1xB, $\boldsymbol{a}^{[3]}$: 1xB

Computational Graph for MLP

Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[3]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[3]}}$



$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{f}(\boldsymbol{Z}^{[1]})$$

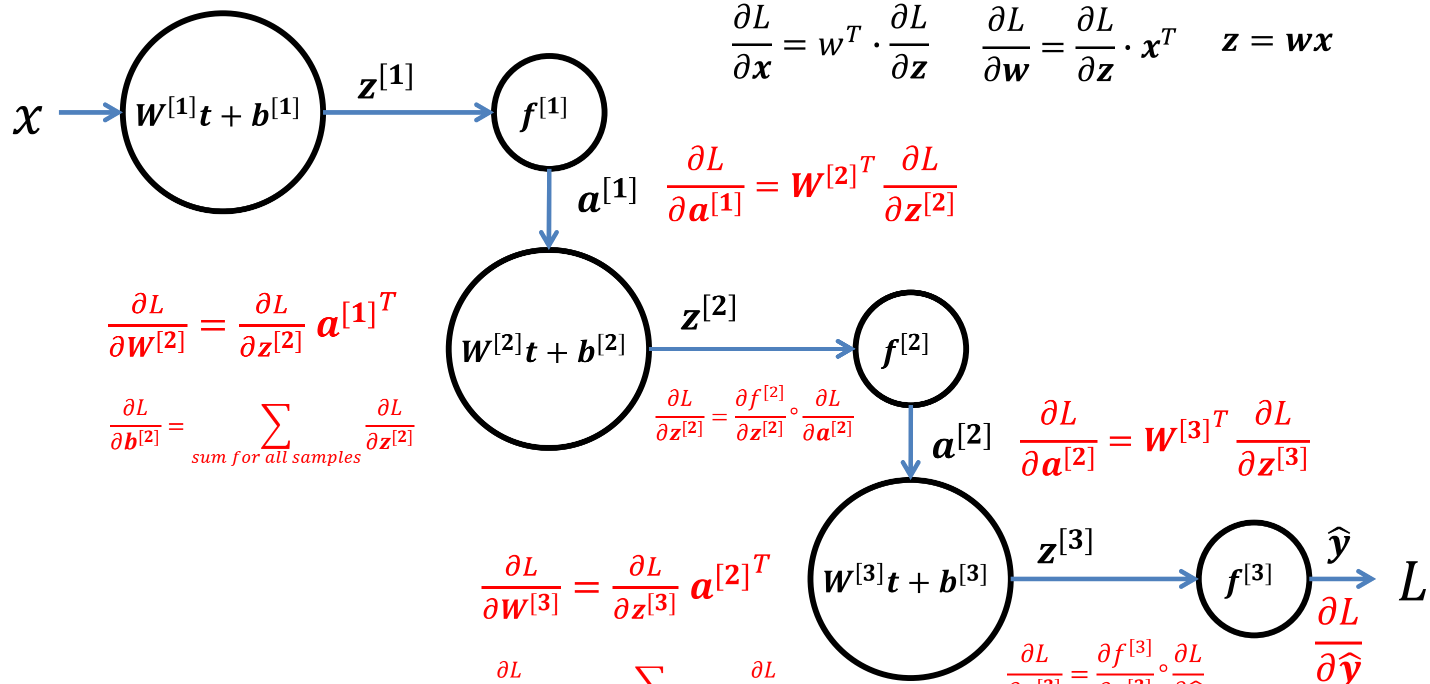
$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{f}(\boldsymbol{Z}^{[2]})$$

$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]} = \boldsymbol{f}(\boldsymbol{Z}^{[3]})$$

$$\frac{\partial L}{\partial \boldsymbol{x}} = w^T \cdot \frac{\partial L}{\partial \boldsymbol{z}} \qquad \frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \boldsymbol{z}} \cdot \boldsymbol{x}^T \qquad \boldsymbol{z} = \boldsymbol{w}\boldsymbol{x}$$



$$\frac{\partial L}{\partial \boldsymbol{a}^{[1]}} = \boldsymbol{W}^{[2]^T} \frac{\partial L}{\partial \boldsymbol{z}^{[2]}}$$

$$\frac{\partial L}{\partial \boldsymbol{W}^{[2]}} = \frac{\partial L}{\partial \boldsymbol{z}^{[2]}} \boldsymbol{a}^{[1]^T}$$

$$\frac{\partial L}{\partial \boldsymbol{b}^{[2]}} = \sum_{sum\ for\ all\ samples} \frac{\partial L}{\partial \boldsymbol{z}^{[2]}}$$

$$\frac{\partial L}{\partial \boldsymbol{z}^{[2]}} = \frac{\partial f^{[2]}}{\partial \boldsymbol{z}^{[2]}} \circ \frac{\partial L}{\partial \boldsymbol{a}^{[2]}}$$

$$\frac{\partial L}{\partial \boldsymbol{a}^{[2]}} = \boldsymbol{W}^{[3]^T} \frac{\partial L}{\partial \boldsymbol{z}^{[3]}}$$

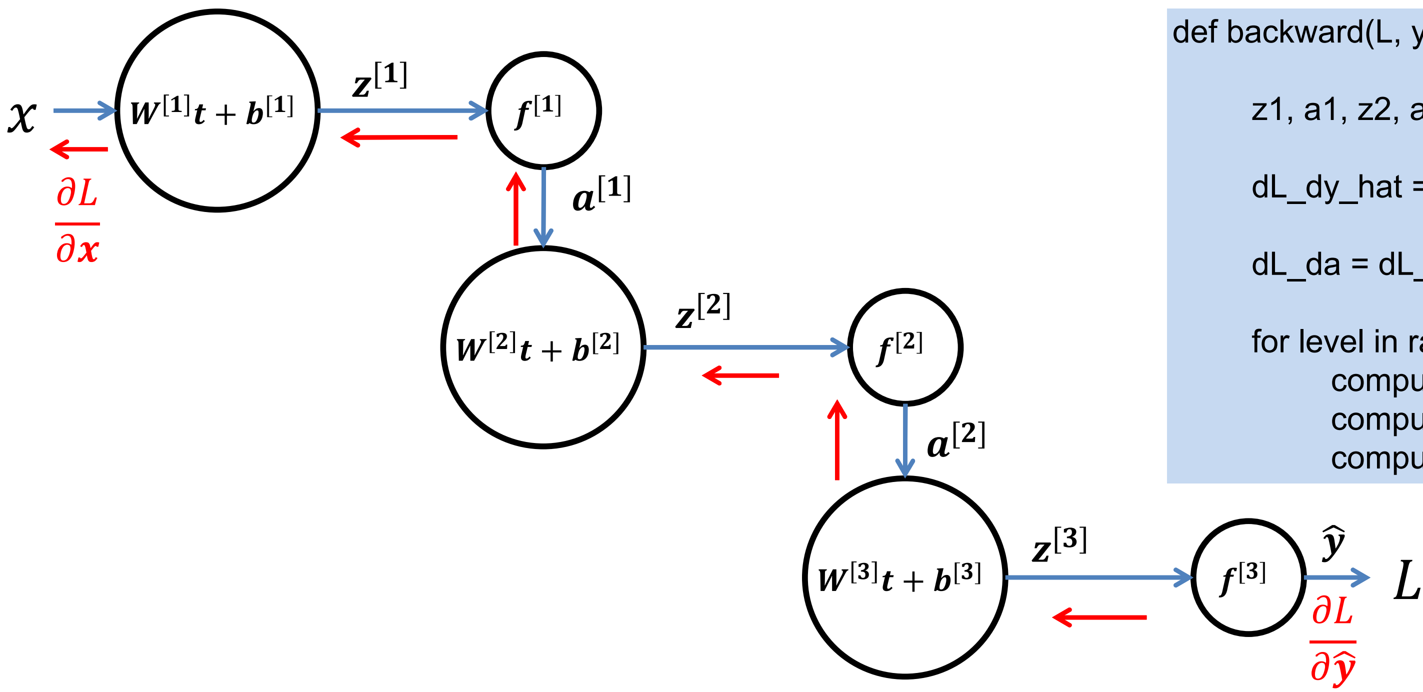
$$\frac{\partial L}{\partial \boldsymbol{W}^{[3]}} = \frac{\partial L}{\partial \boldsymbol{z}^{[3]}} \boldsymbol{a}^{[2]^T}$$

$$\frac{\partial L}{\partial \boldsymbol{b}^{[3]}} = \sum_{sum\ for\ all\ samples} \frac{\partial L}{\partial \boldsymbol{z}^{[3]}}$$

$$\frac{\partial L}{\partial \boldsymbol{z}^{[3]}} = \frac{\partial f^{[3]}}{\partial \boldsymbol{z}^{[3]}} \circ \frac{\partial L}{\partial \hat{\boldsymbol{y}}}$$

Backprop for MLP

Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[1]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[2]}}, \frac{\partial L}{\partial \boldsymbol{W}^{[3]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[3]}}$




```
def backward(L, y, y_hat, buffer):

    z1, a1, z2, a2, z3 = buffer

    dL_dy_hat = compute_loss_gradient(L, y, y_hat)

    dL_da = dL_dy_hat

    for level in range(3,0, -1):
        compute dL_dz
        compute dL_dW, dL_db
        compute new dL_da
```

Start the backprop

Given a batch $(\boldsymbol{x}, \boldsymbol{y})$, the mini-batch loss is:

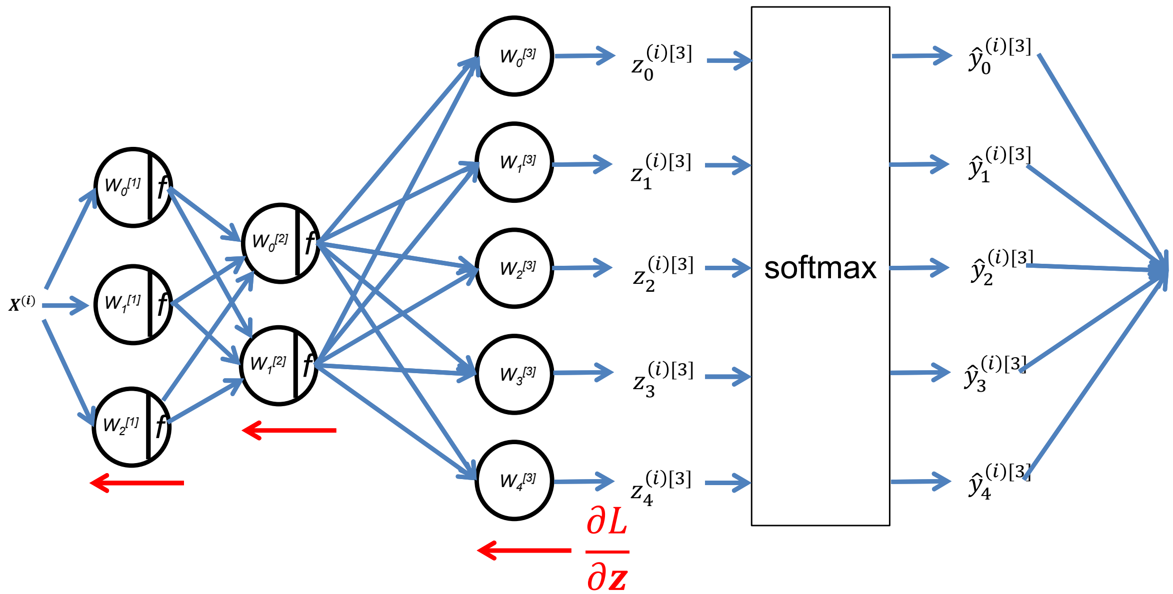
$$L = \frac{1}{B}\sum_{i=0}^{B-1} L^{(i)}\,(y^{(i)}, \hat{y}^{(i)})$$

$$L^{(i)}\,(y^{(i)}, \hat{y}^{(i)}) = -[y^{(i)}\boldsymbol{log}\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right)\boldsymbol{log}(1 - \hat{y}^{(i)})]$$

$$\frac{\partial L}{\partial \hat{\boldsymbol{y}}} = \left[\frac{\partial L}{\partial \hat{y}^{(i)}}\right]_{i=0:B-1} \qquad \frac{\partial L}{\partial \hat{y}^{(i)}} = \frac{1}{B}\frac{\partial L^{(i)}\,(y^{(i)},\hat{y}^{(i)})}{\partial \hat{y}^{(i)}} = \frac{1}{B}\left[\frac{y^{(i)}}{\hat{y}^{(i)}} - \frac{1-y^{(i)}}{1-\hat{y}^{(i)}}\right]$$

Start the backprop

K=5, 5 class classification



$$L_{CE}\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = -\sum_{j=0}^{K-1} \boldsymbol{y}_j^{(i)} \log(\hat{\boldsymbol{y}}_j^{(i)})$$

$$\frac{\partial L}{\partial \boldsymbol{z}} = [\hat{y}_0 - y_0, \hat{y}_1 - y_1, \dots, \hat{y}_{k-1} - y_{k1}]^T$$

We have all ingredients for MB-SGD

Initialize weights and bias

Random shuffle dataset

BatchSize = 32

for epoch in range(E):

select #BatchSize **samples**

1. Evaluate loss function (forward pass) at this **batch**
$L = \frac{1}{B}\sum_{i=0}^{B-1} L(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}(\boldsymbol{W}, \boldsymbol{b}))$

2. Compute gradient $\frac{\partial L}{\partial \boldsymbol{W}^{[l]}}, \frac{\partial L}{\partial \boldsymbol{b}^{[l]}}$

3. Update parameter $\boldsymbol{W}^{[l]} = \boldsymbol{W}^{[l]} - \alpha \frac{\partial L}{\partial \boldsymbol{W}^{[l]}}, \ \boldsymbol{b}^{[l]} = \boldsymbol{b}^{[l]} - \alpha \frac{\partial L}{\partial \boldsymbol{b}^{[l]}}$

Forward pass
Run data through model to compute output

Backward pass or backprop
Starting from loss, compute gradient to all parameters

Update
Use gradient information to update parameters