- Recall

$s = f(x;W) = Wx$ — score function

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$ — SVM loss

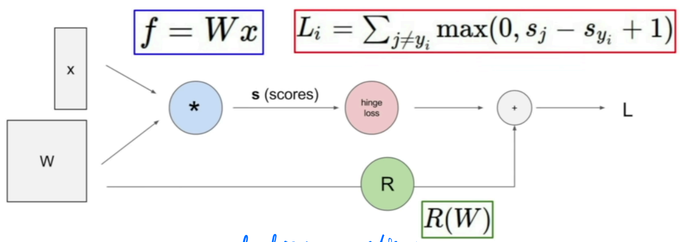
$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \sum_k W_k^2$ — data loss + regularization

want $\nabla_W L$

→ solve it with optimization — analytic gradient / numerical gradient

- analytic gradient (thru computational graphs)

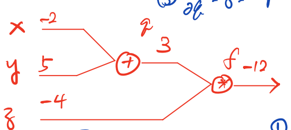
Computational graphs



- important to calculate gradient for neural network (complex function)
- e.g. find gradient

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

@ $x=-2$, $y=5$, $z=-4$



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

$f = q \times z$ $\quad \frac{\partial f}{\partial q} = z$ $\quad \frac{\partial f}{\partial z} = q$

find $\frac{df}{dx}$ $\frac{df}{dy}$ $\frac{df}{dz}$

① $\frac{\partial f}{\partial f} = 1$

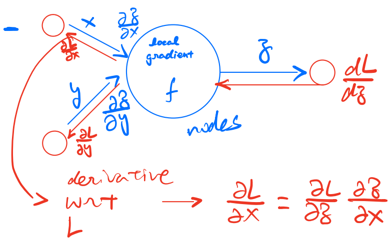
② $\frac{\partial f}{\partial z} = q = 3$

③ $\frac{\partial f}{\partial q} = z = -4$

④ $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y} = -4 \cdot 1 = -4$

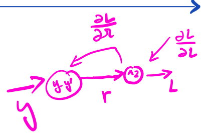
⑤ $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} = -4 \cdot 1 = -4$

back-propagation $\quad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$ chain rule



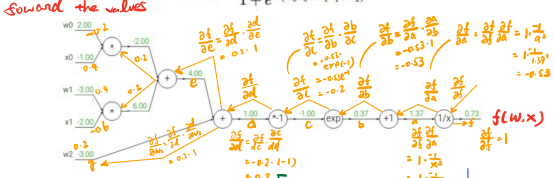
derivative wrt L $\quad \frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$

e.g.



Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

forward the values



backprop the gradient

backpropagation!

sigmoid gate

$\sigma(x) = \frac{1}{1+e^{-x}}$

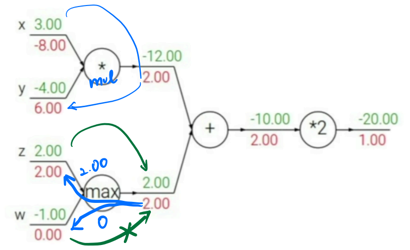
$\frac{\partial \sigma(x)}{\partial x} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)$

$= (1-\sigma(x))\sigma(x)$

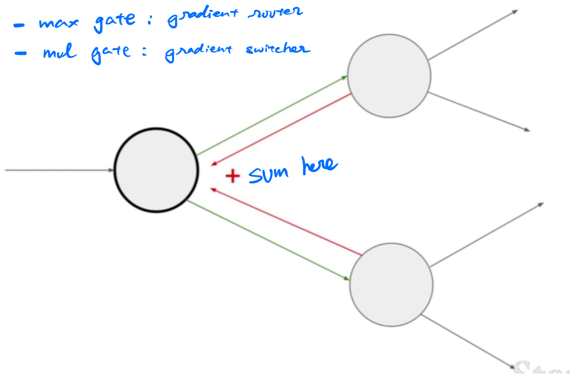
$(1-0.73)0.73 = 0.2$

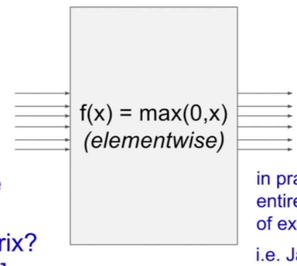
→ to combine into a more complex node

- other nodes (more complex ones)



- max gate : gradient router
- mul gate : gradient switcher





Q: what is the size of the Jacobian matrix? [4096 x 4096!]

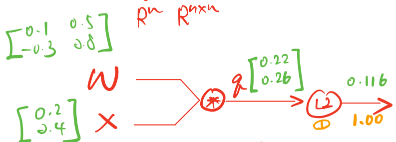
in practice we process an entire minibatch (e.g. 100) of examples at one time:

i.e. Jacobian would technically be a [409,600 x 409,600] matrix :\

yet it will be diagonal (very sparse)

e.g. $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$\mathbb{R}^n$ $\quad \mathbb{R}^{n \times n}$



- $q = W \cdot (x) = \begin{bmatrix} W_{11}x_1 + \cdots + W_{1n}x_n \\ W_{21}x_1 + \cdots + W_{2n}x_n \\ \vdots \\ W_{n1}x_1 + \cdots + W_{nn}x_n \end{bmatrix}$

- $f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$

- ① $\nabla_q f = 2q = \begin{bmatrix} 0.44 \\ 0.52 \end{bmatrix} \in \mathbb{R}^2$

② $\frac{\partial q_k}{\partial W_{ij}} = 1_{k=i} x_j$

$\therefore \frac{\partial f}{\partial W_{ij}} = \sum_k \frac{\partial f}{\partial q_k}\frac{\partial q_k}{\partial W_{ij}}$

$= \sum_k (2q_k)(1_{k=i} x_j)$

$= 2q_i x_j$

$\nabla_W f = 2q \cdot x^T$

$\frac{\partial q_k}{\partial x_i} = W_{k,i}$

$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k}\frac{\partial q_k}{\partial x_i}$

$= \sum_k 2q_k W_{k,i}$

$\nabla_x f = 2W^T q$

Note that: the gradient with respect to a variable should hv the same shape as the variable

Modularized implementation: forward / backward API

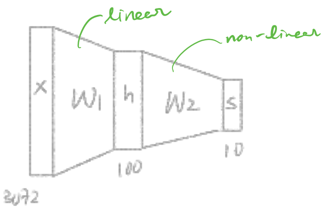
Graph (or Net) object *(rough psuedo code)*

```
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

- Neural Network

  - Before | Linear score function : $f = Wx$

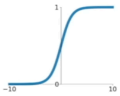
    after | 2-Layer Neural Network : $f = W_2 \max(0, W_1 x)$



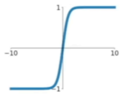
  or even 3-layer ...

Activation functions

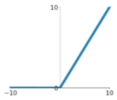
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



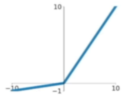
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

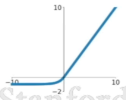


Leaky ReLU
$\max(0.1x, x)$

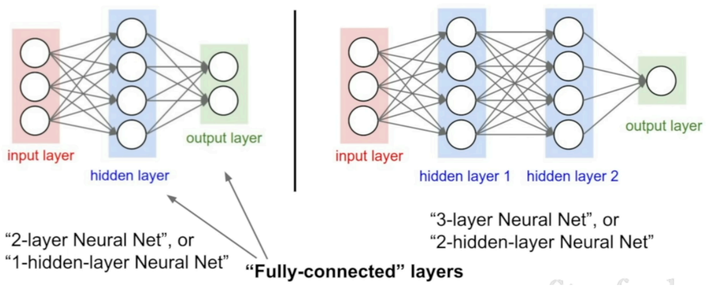


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Neural networks: Architectures



"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"Fully-connected" layers

"3-layer Neural Net", or
"2-hidden-layer Neural Net"