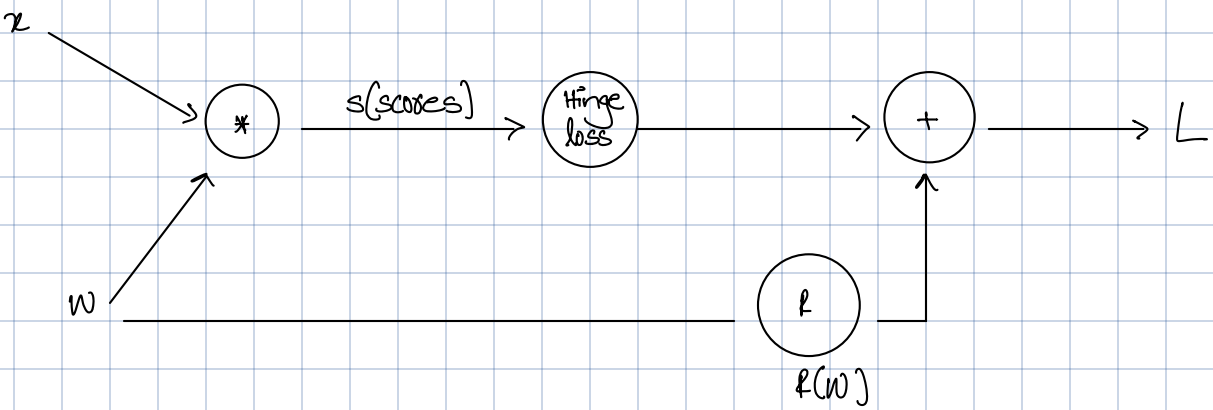
Introduction to Nueral networks and Backpropogation:

Computational graphs:



If we express a function as a computational graph. Then, we can use back-propogation to recursively, using chain rule, compute the gradient.

Example,

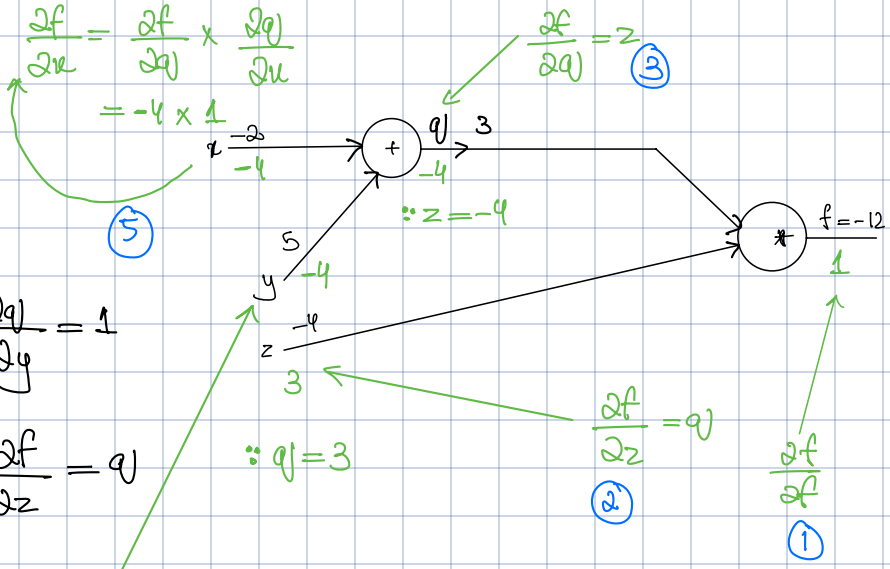
$$f(x,y,z) = (x+y)z$$

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

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

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

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

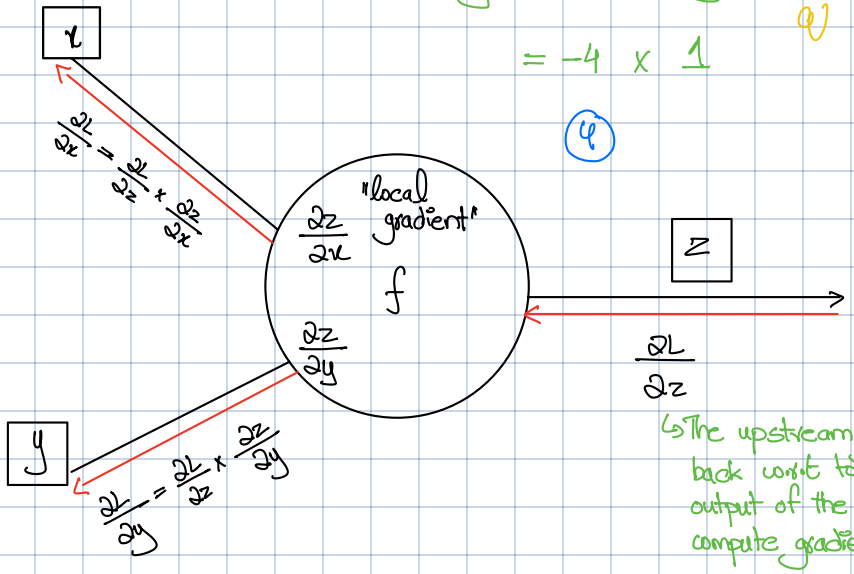


$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \times \frac{\partial q}{\partial y}$ } In to get to effect y on f, we need to know effect of q * q on f.

$= -4 \times 1$

4

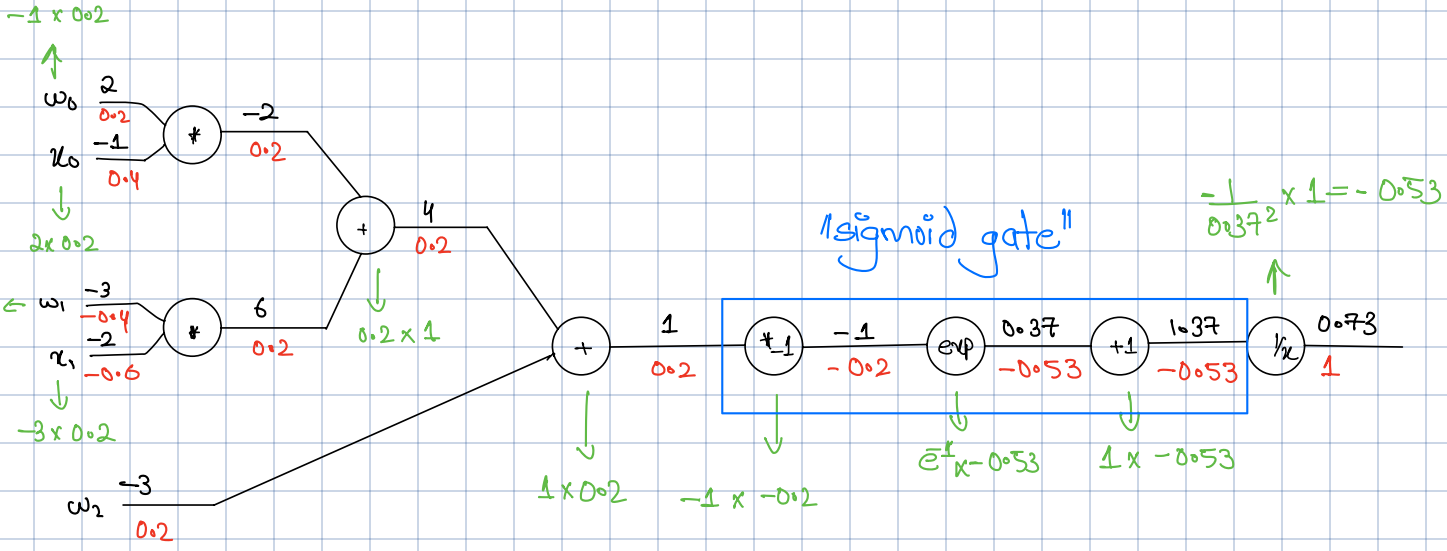
$\frac{\partial f}{\partial z}$



↳ The upstream gradient coming back wrt to the immediate output of the node. Now, we compute gradient to inputs

Another example,

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



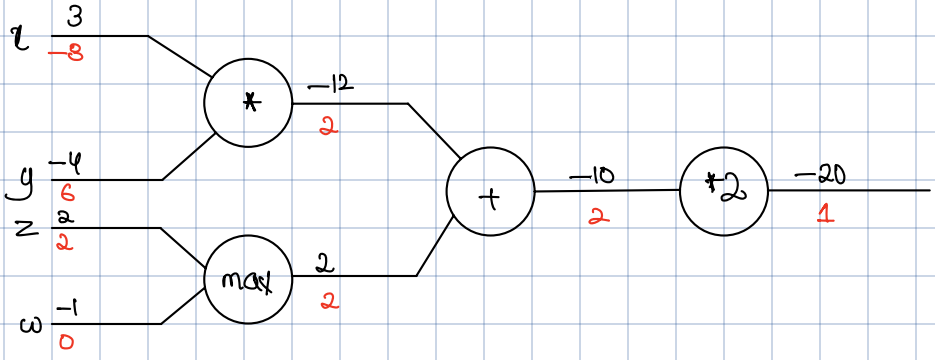
- As long as we know local gradient of a function, we can use it as node
- In above example, we can replace sigmoid gate with

$$\frac{d\,\sigma(x)}{dx} = (1-\sigma(x))\,\sigma(x)$$

$(1-0.73)\,0.73 = 0.2$

- Find local gradient
- Multiply it with "upstream gradient"
- For + node, gradient of each input is 1
- For *, mult Uh with other input

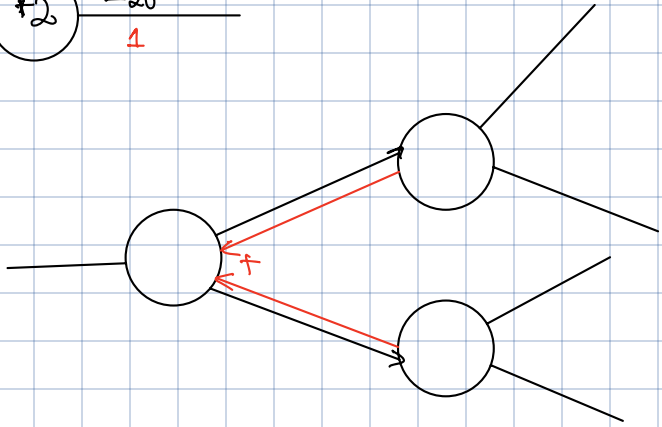
Patterns in backward flow:



add gate: gradient distributor → equally distributes the gradient

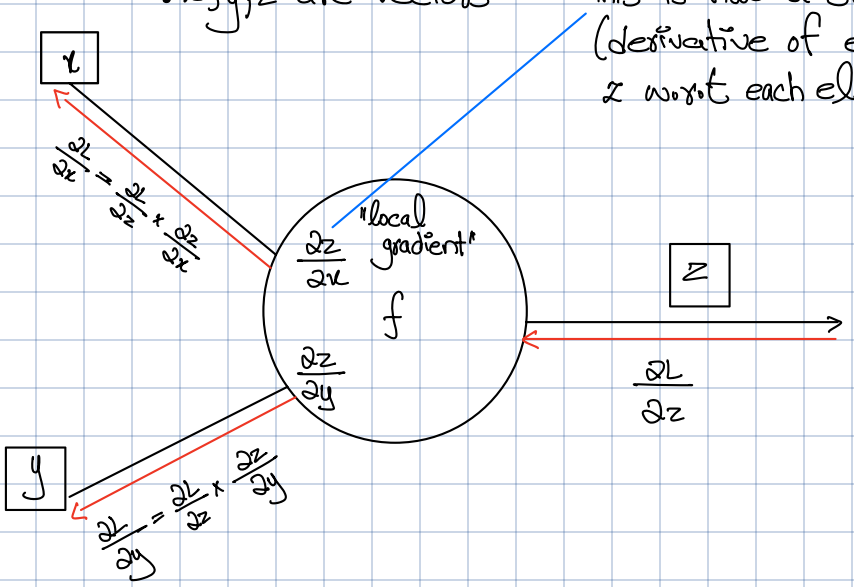
Max gate: gradient router → Route the gradient to one of the branches

mult gate: gradient switcher → Scales the gradient by the value of the other branch



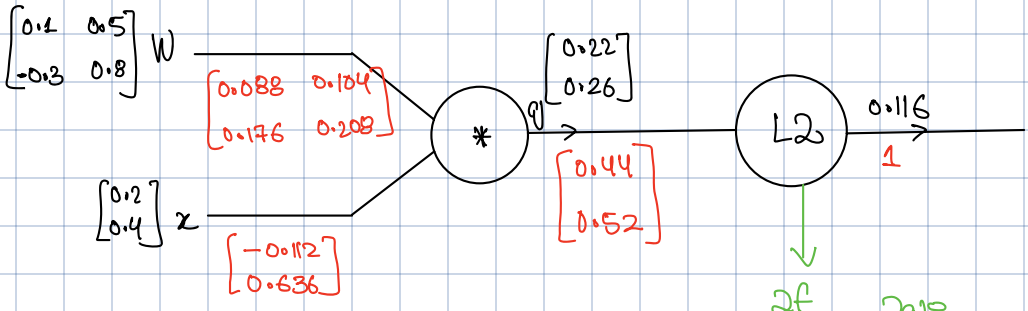
Gradients for vectorized code:

- x, y, z are vectors



Example,

$$f(x, w) = \|W \cdot x\|^2 = \sum_{i=1}^{2} (W \cdot x)^2$$



$$q = W \cdot x = \begin{bmatrix} w_{11} x_1 + \cdots + w_{1n} x_n \\ w_{n1} x_1 + \cdots + w_{nn} x_n \end{bmatrix}$$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

$$\nabla_q f = 2q$$

$2q \times 1$

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

> Gradient w.r.t to a variable should have same shape as that variable

For gradient of W:

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

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

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

$$= 2q_i x_j$$

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

For gradient of x:

$$\frac{\partial q_k}{\partial x_i} = W_{ki}$$

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

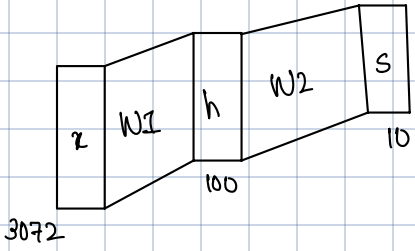
$$= \sum_k 2q_k W_{ki}$$

$$\nabla_x f = 2W^T \cdot q$$

Nueral Networks:

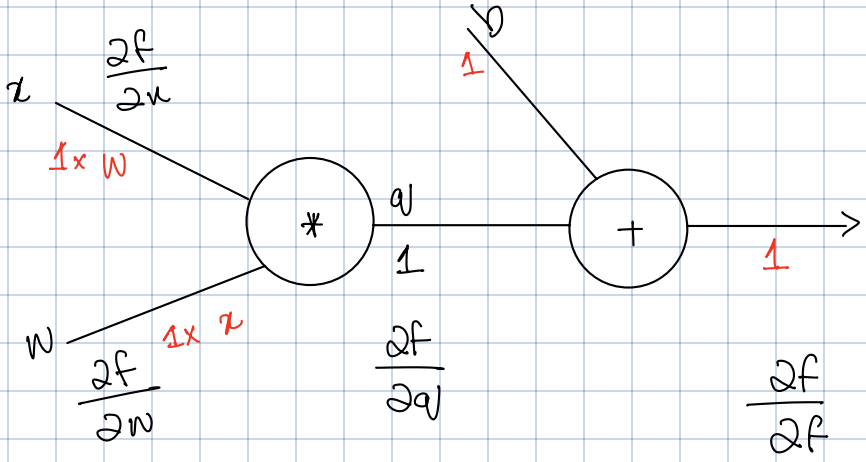
(Before) Linear Score function: $f = W_1 x$

(Now) 2-layer NN: $f = W_2 \max(0, W_1 x)$



- Simpler function stacked on top of each other in hierarchical way with a non-linear in between

Homework helper:



$q = x \cdot W$

$\frac{\partial q}{\partial x} = W$

$\frac{\partial q}{\partial W} = x$

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

$= 1 \times W$

$\frac{\partial f}{\partial W} = \frac{\partial f}{\partial q} \times \frac{\partial q}{\partial W}$

$= 1 \times x$

Softmax loss:

$$J(w,b) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{N} 1\{y^{(i)}==j\}\log\frac{e^{z_j^{(i)}}}{\sum_{k=1}^{N} e^{z_k^{(i)}}}\right]$$

$$\frac{\partial J}{\partial z_j^{(i)}} = z_j^{(i)} - \log\left(\sum_{k=1}^{N} e^{z_k^i}\right)$$

$$= 1 - \frac{e^{z_j^i}}{\sum_{k=1}^{N} e^{z_k^i}}$$

$$= \frac{(1-a)}{-N}$$

$$= \frac{a-1}{N}$$

$$\frac{\partial J}{\partial z_j^i} = \left(\frac{a-1}{N}\right)$$

SVM loss:

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} \max(0, s_j - s_i + 1)$$

if margin >0;

$$\frac{\partial L}{\partial s_j} = \sum_{i=1}^{N} 1$$

$j == y_i$

$$\frac{\partial L}{\partial s_{y_i}} = \sum_{i=1}^{N} -1$$