Machine Learning,
An introduction 3:

Backpropagation

Part 1

This lesson is based on the Standford Course:
CS231n: Convolutional Neural Networks for Visual Recognition

Lecture 4

http://cs231n.stanford.edu/syllabus.html

it also uses slides from that lecture

Glossary

- Score: prediction results of the model (NN, or some classifier) on the data

  [-2,2]==[apple, banana]

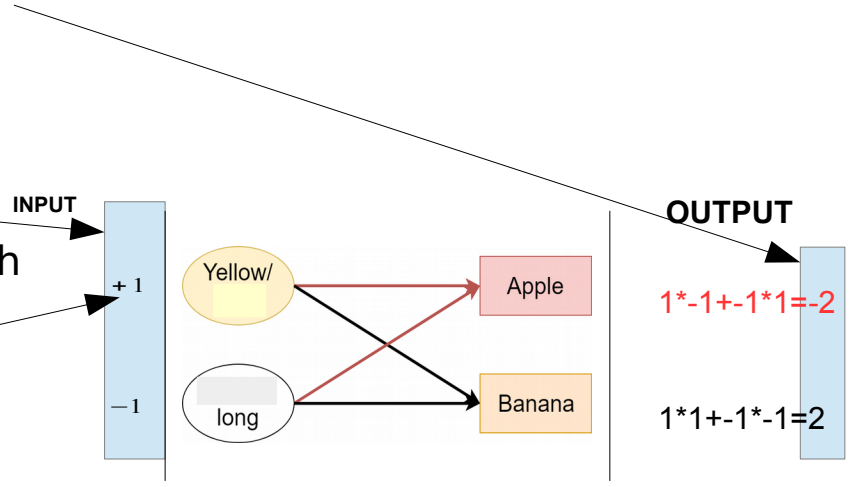
- Example: a collection of features

  e.g. Example: A picture, features: each pixel

  (Yellow & Long)

- Feature: a parameter

  (Yellow/Red)



Last Time:
Loss functions and Gradient Descent

We now use our Linear Classifiers to predict 3 classes: Apple, Banana and Pear

	Correct Label	Apple	Banana	Pear
Class				
Apple		**1.87**	-0.4	0.1
Banana		**1.3**	1.1	-0.3
Pear		0.56	1.35	**1.1**

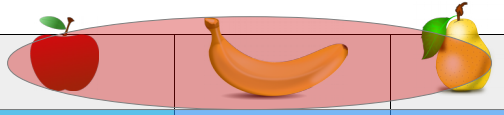
$$f_{pear}(x, W) = W_{pear} * x_{pear} + b_{pear}$$

$$f_{banana}(x, W) = W_{banana} * x_{banana} + b_{banana}$$

$$f_{apple}(x, W) = W_{apple} * x_{apple} + b_{apple}$$

We have used our Linear Classifiers to predict 3 classes: Apple, Banana and Pear

Correct class



Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	**1.3**	1.1	-0.3
Pear	0.56	1.35	**1.1**

Classes (labels)

The Scores for the Classifier. i.e. the predictions for each class

* images are in the public domain

We have used our Linear Classifiers to predict 3 classes: Apple, Banana and Pear

Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	**1.3**	1.1	-0.3
Pear	0.56	1.35	**1.1**

The Scores for the Classifier. i.e. the predictions for each class

We need a way to quantify the inaccuracy of the results.

Then we need a way to find the parameters that minimize the inaccuracy of the results

Loss Function

Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	1.3	1.1	-0.3
Pear	0.56	**1.35**	**1.1**

The loss should reflect the difference between the predicted value and the actual value

$$L = \frac{1}{N} \sum_i L_i\left(f\left(x_i, W\right), y_i\right)$$

f: our classifier
y_i: label for example i
L_i: Loss function for example i

L is a function of the predicted and actual value for the ith example

Scores | inaccuracy | minimize the inaccuracy

Loss Function for multiclass SVM

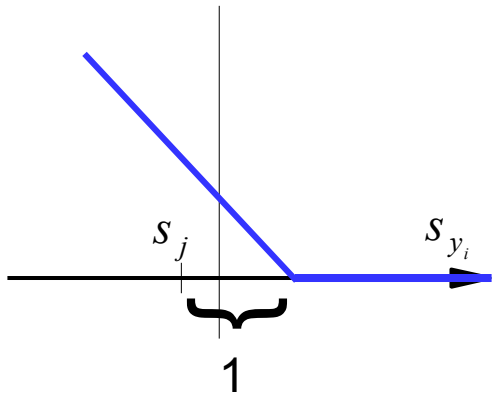
Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	1.3	1.1	-0.3
Pear	0.56	**1.35**	**1.1**

$$L_i = \sum_{i \neq y_i} \begin{cases} 0 & if\ s_{y_i} \geqslant s_i + 1 \\ s_j - s_{y_i} + 1 & otherwise \end{cases}$$

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

$$L_i = L_i(s, y_i)$$

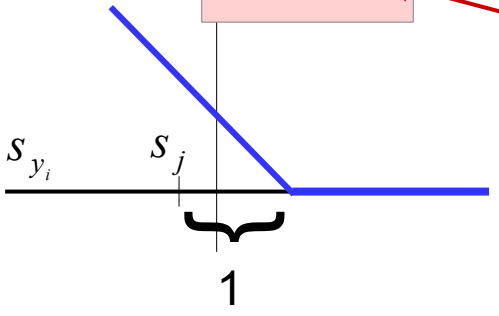
$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$



Scores | inaccuracy | minimize the inaccuracy

Loss Function for multiclass SVM: 'Hinge' Loss

Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	**1.3**	1.1	-0.3
Pear	0.56	**1.35**	**1.1**
Loss	0.33		



For the first example (Apple):

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

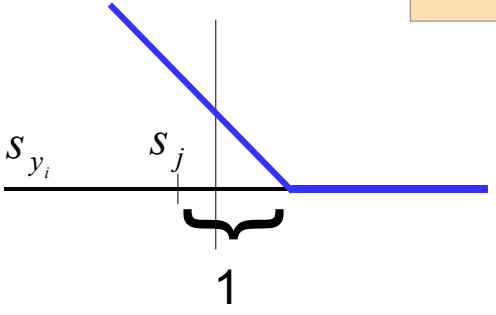
= max (0,1.3-1.87+1)
+max (0,0.56-1.87+1)
=max(0,0.33)+
max(0,-0.31)
=0.33+0
=0.33

The +1 means the loss function penalizes a difference of less than 1 between the correct score and the predicted score

Scores | inaccuracy | minimize the inaccuracy

Loss Function for multiclass SVM: 'Hinge' Loss

Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	**1.3**	1.1	-0.3
Pear	0.56	**1.35**	**1.1**
Loss	0.33	1.25	



For the second example (Banana):

$$L_{ban} = \sum_{j \neq y_{ban}} max\left(0, s_j - s_{y_{ban}} + 1\right)$$

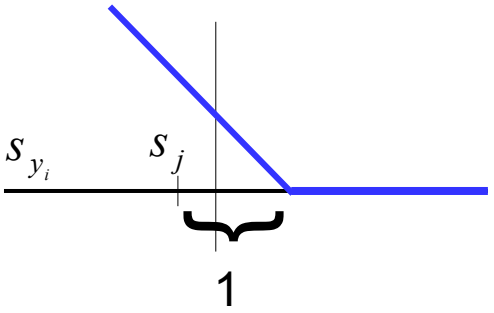
= max (0,-0.4-1.1+1)
+max (0,1.35-1.1.+1)
=max(0,-0,4)
+max(0,1.25)
=0+1.25
=1.25

The +1 means the loss function penalizes a difference of less than 1 between the correct score and the predicted score

Scores | inaccuracy | minimize the inaccuracy

Loss Function for multiclass SVM: 'Hinge' Loss

Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	**1.3**	1.1	-0.3
Pear	0.56	**1.35**	**1.1**
Loss	0.33	1.25	0



For the third example (Pear):

$$L_{pear} = \sum_{j \neq y_{pear}} max(0, s_j - s_{y_{pear}} + 1)$$

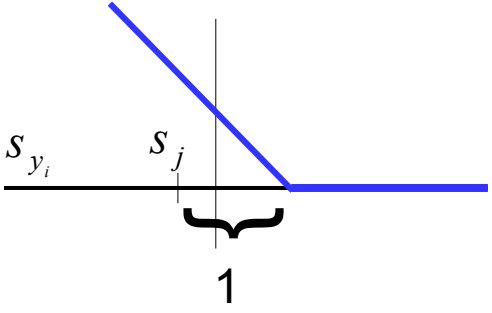
= max (0,0.1-1.1+1)
+max (0,-0.3-1.1.+1)
=max(0,0)
+max(0,-0.4)
=0+0
=0

The +1 means the loss function penalizes a difference of less than 1 between the correct score and the predicted score

Scores | inaccuracy | minimize the inaccuracy

Loss Function for multiclass SVM: 'Hinge' Loss

Class\Pred.			
Apple	**1.87**	-0.4	0.2
Banana	**1.3**	1.1	-0.6
Pear	0.56	**1.35**	**2.2**
Loss	0.33	1.25	0



What if the Weights and therefore scores were doubled?

$$L_{pear} = \sum_{j \neq y_{pear}} max(0, s_j - s_{y_{pear}} + 1)$$

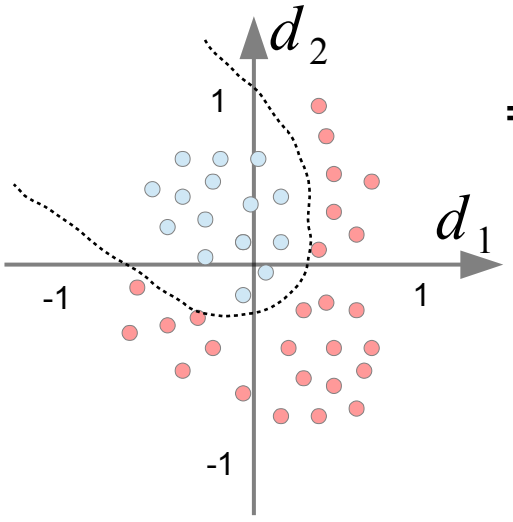
= max (0,0.2-2.2+1)
+max (0,-0.6-2.2.+1)
=max(0,-2)
+max(0,-1.8)
=0+0
=0

It's the same score!
So any multiples of weights that result in 0 will still result in zero.

Scores | inaccuracy | minimize the inaccuracy

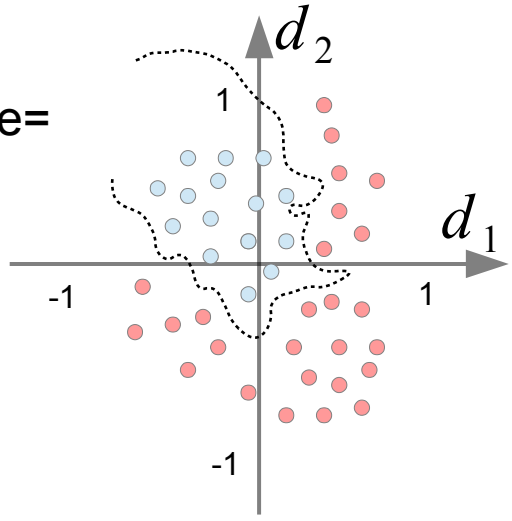
Loss Function for multiclass SVM: 'Hinge' Loss

Remember:
Non-linear boundary



=Same score=

Large Weights:
Overfitting



Scores | inaccuracy | minimize the inaccuracy

Loss Function Regularization

Solution: Regularization!

$$L(W) = \frac{1}{N}\sum_i L_i(f(x_i, W), y_i) + \lambda R(W)$$

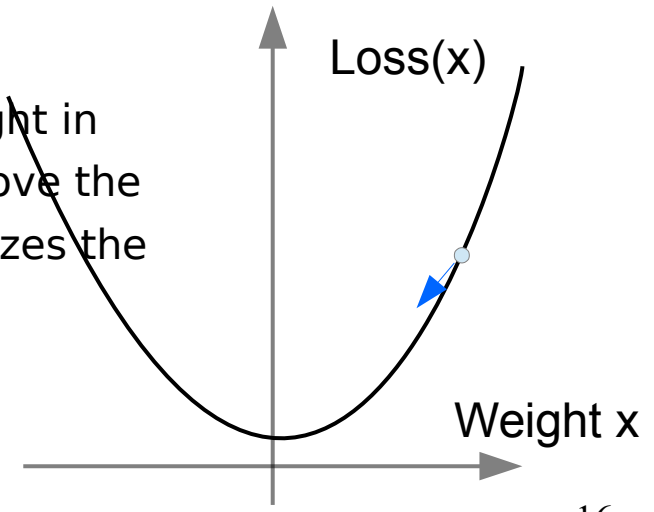
λ Is a scaling parameter (how strong do we want The regularization to be)

$R(W)$ Is a function of the weights

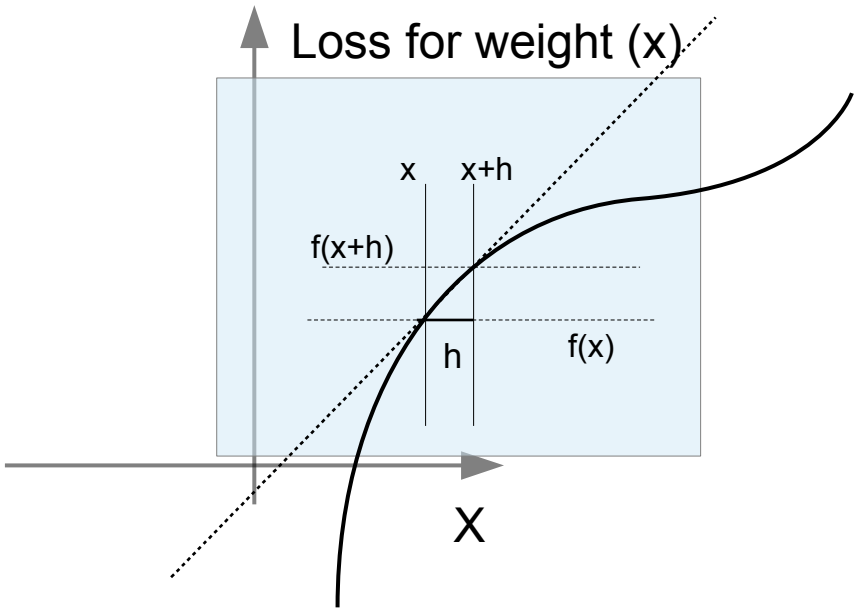
Scores | inaccuracy | minimize the inaccuracy

Optimization!

- Now that we can quantify how good our model is, we try to find a method which allows us to change the weights so that the loss function is minimized:
  - Gradient Descent!
    - We calculate the slope for each weight in regards to the loss function, then move the weight in the direction which minimizes the loss



Optimization via Gradient Descent



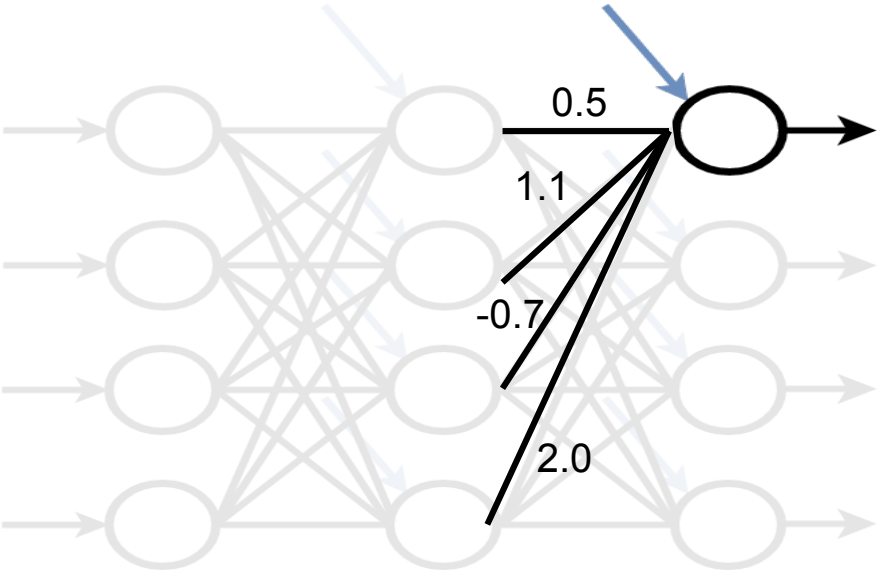
$$Slope = \frac{vertical\ change}{horizontal\ change}$$

$$Slope = \frac{f(x+h) - f(x)}{h}$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Optimization via Gradient Descent

Current W
0.5
1.1
-0.7
2.0



Optimization via Gradient Descent

Current W	W+h	Gradient dW
0.5	0.5+0.0001	-2.5
1.1	1.1	
-0.7	-0.7	
2.0	2.0	
Loss: 1.25347	Loss: 1.25322	

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{1.25322 - 1.25347}{0.0001}$$

$$= -2.5$$

Optimization via Gradient Descent

Current W	W+h	Gradient dW
0.5	0.5	-2.5
1.1	1.1+0.0001	-1.3
-0.7	-0.7	
2.0	2.0	
Loss: 1.25347	Loss: 1.25334	

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

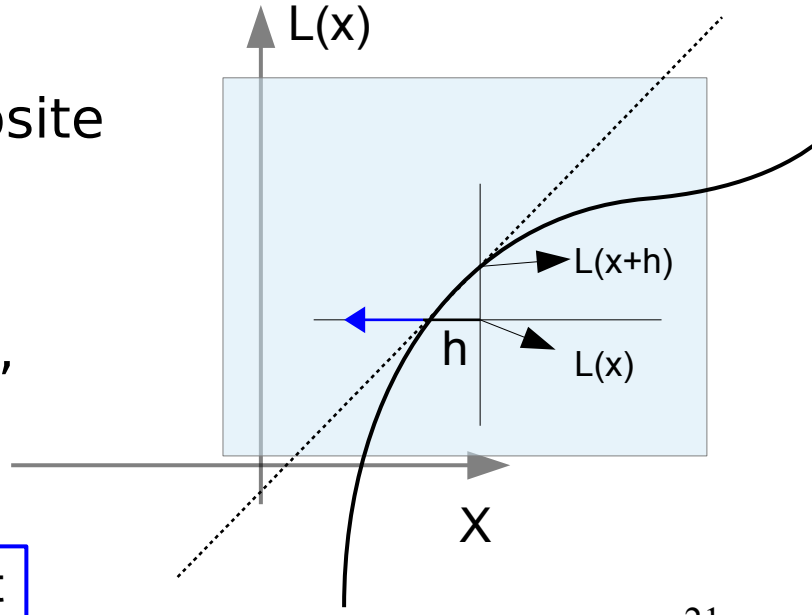
$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{1.25334 - 1.25347}{0.0001}$$

$$= -1.3$$

Optimization via Gradient Descent

1) Calculate Loss
2) Calculate Gradient
3) Move Parameter x in the opposite direction of the gradient (gradient points “uphill”) by a distance called the “step size”
4) repeat

Weight += - step size * gradient



Summary

- The Model predicts the scores,

Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	**1.3**	1.1	-0.3
Pear	0.56	1.35	**1.1**

Summary

- The Model predicts the scores,
- The scores plug into the Loss function

$$L = \frac{1}{N} \sum_i L_i\left(f\left(x_i, W\right), y_i\right)$$

Summary

- The Model predicts the scores,
- The scores plug into the Loss function which
- Gives us the Loss for each class

Class\Pred.			
Apple	**1.87**	-0.4	0.1
Banana	**1.3**	1.1	-0.3
Pear	0.56	**1.35**	**1.1**
Loss	0.33	1.25	0

Summary

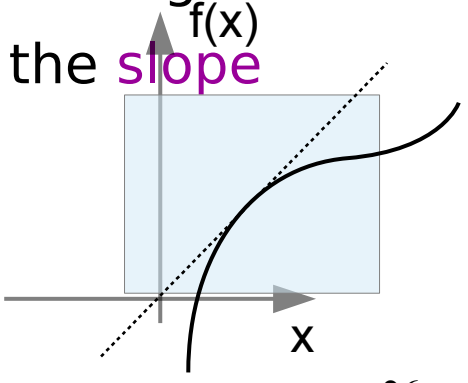
- The Model predicts the scores,
- The scores plug into the Loss function which
- Gives us the Loss for each class
- The Loss can be Regularized to avoid overfitting

$$L(W) = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i) + \lambda R(W)$$

Summary

- The Model predicts the scores,
- The scores plug into the Loss function which
- Gives us the Loss for each class
- The Loss can be Regularized to avoid overfitting
- Using the Loss function we can calculate the slope (Gradient)

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



Summary

- The Model predicts the scores,
- The scores plug into the Loss function which
- Gives us the Loss for each class
- The Loss can be Regularized to avoid overfitting
- Using the Loss function we can calculate the slope (Gradient)
- Adjust weights in the opposite direction of the gradient

$$weights += -stepSize * weightsGrad$$

Summary

- The Model predicts the scores,
- The scores plug into the Loss function which
- Gives us the Loss for each class
- The Loss can be Regularized to avoid overfitting
- Using the Loss function we can calculate the slope (Gradient)
- Adjust weights in the opposite direction of the gradient
- Profit!

Today:

Backpropagation!

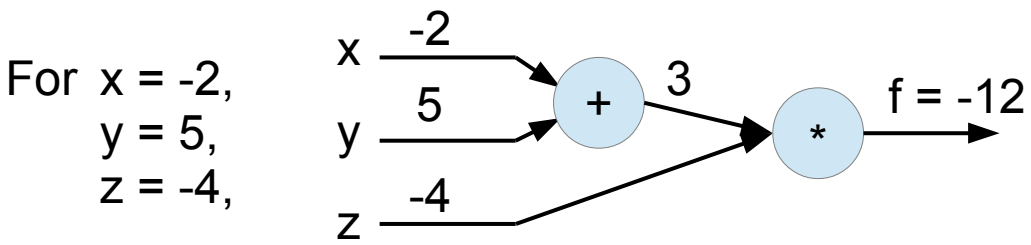
A neural network is a series of functions applied to the input:

$$f(x, W) = W * x + b$$

Let's take a simple function as an example:

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

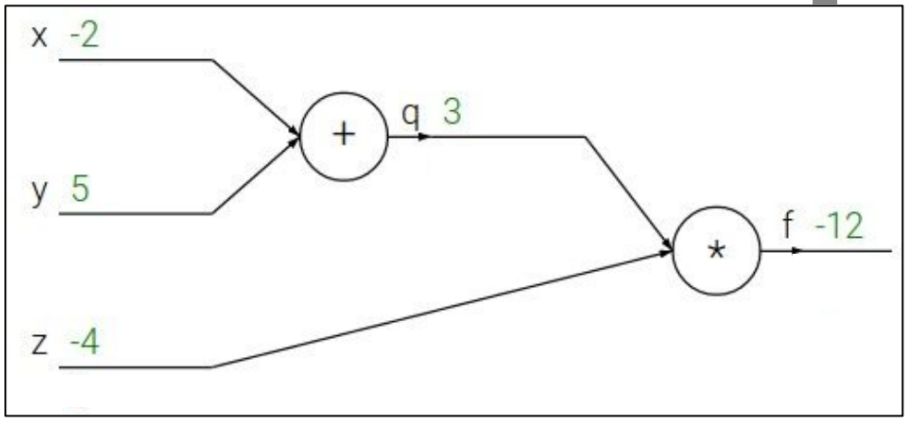
And create a computational graph:



Backpropagation: a simple example

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

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

Backpropagation: a simple example

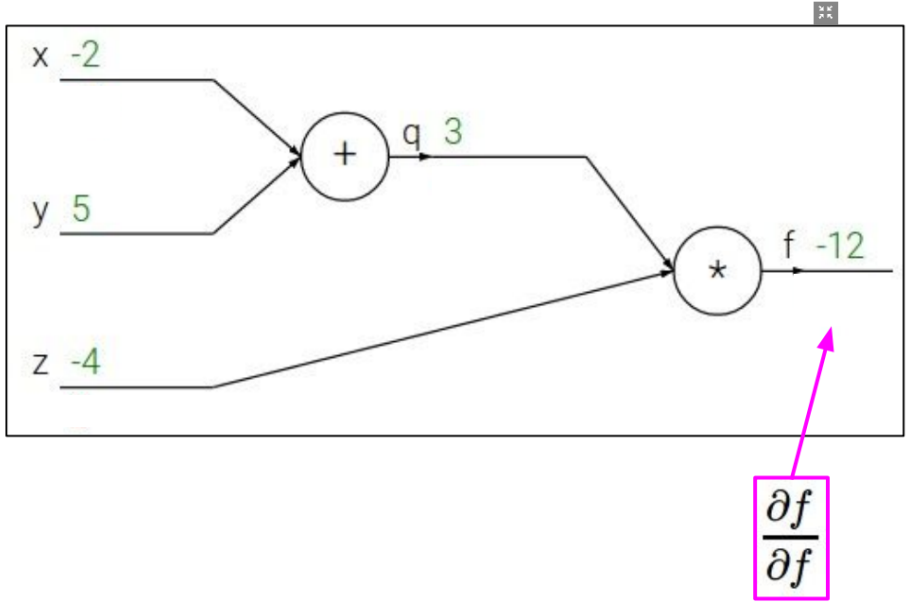
$$f(x, y, z) = (x + y)z$$

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



Backpropagation: a simple example

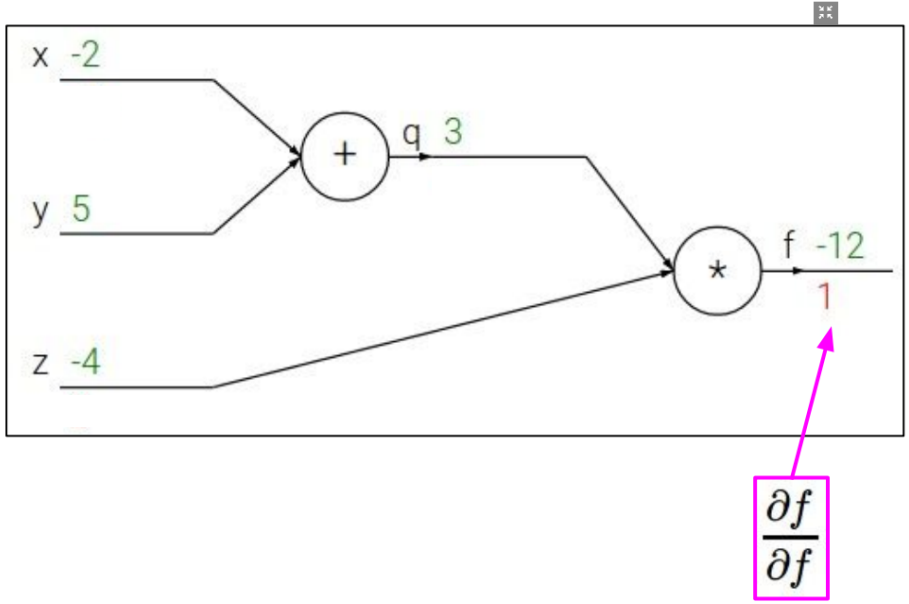
$$f(x, y, z) = (x + y)z$$

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



Backpropagation: a simple example

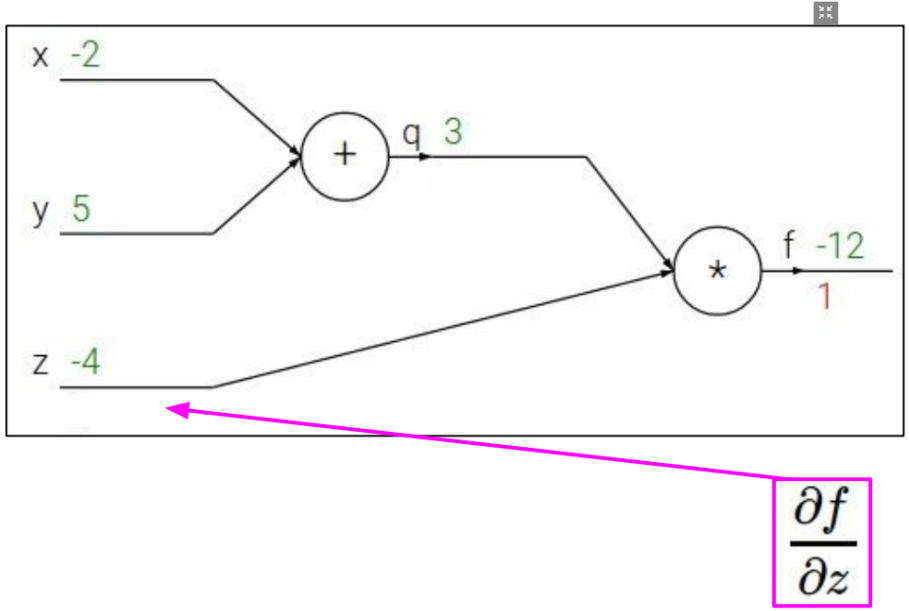
$$f(x, y, z) = (x + y)z$$

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



Backpropagation: a simple example

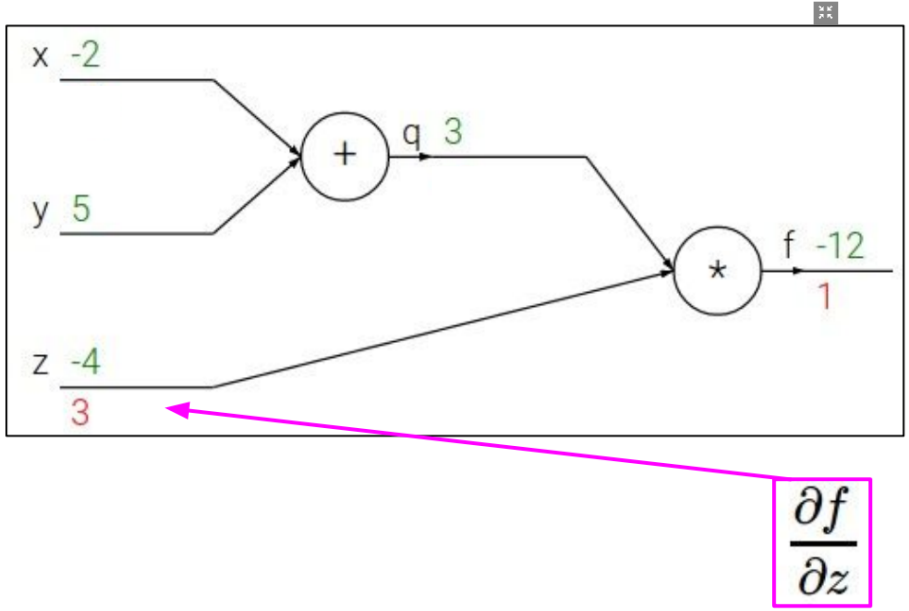
$$f(x, y, z) = (x + y)z$$

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



Backpropagation: a simple example

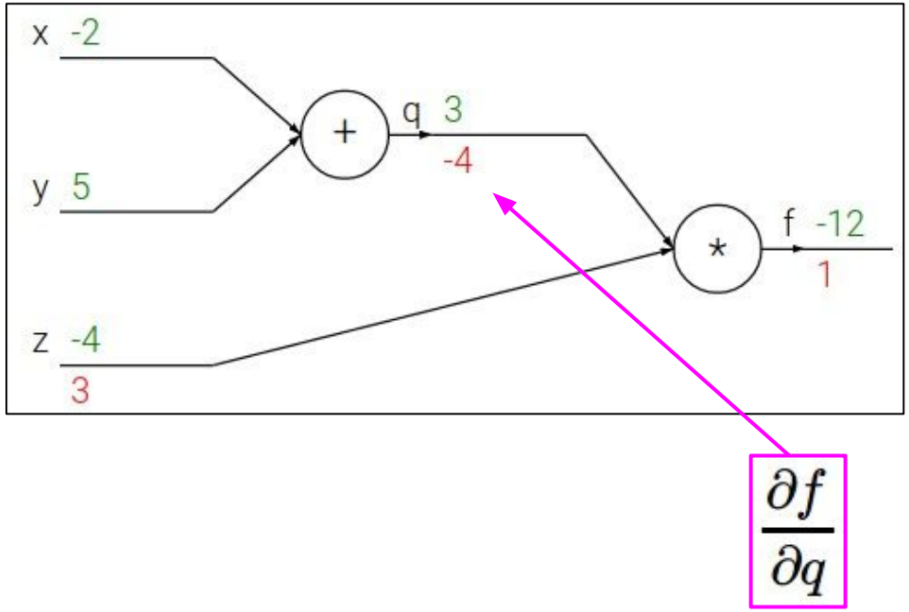
$$f(x, y, z) = (x + y)z$$

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



Backpropagation: a simple example

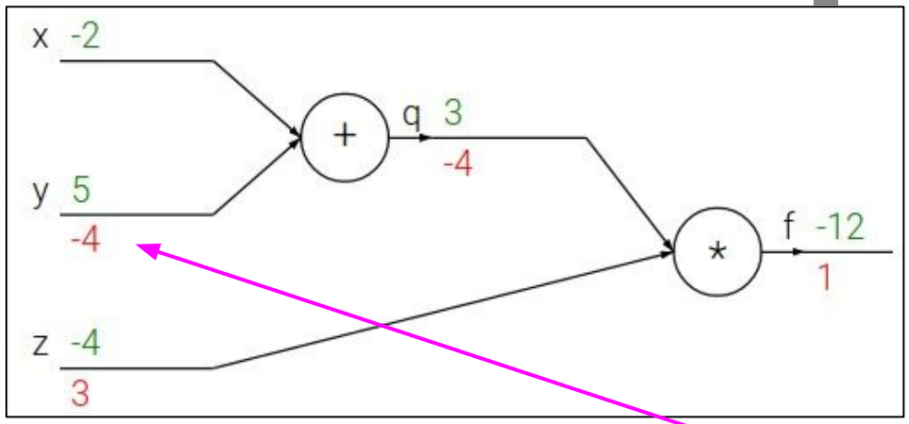
$$f(x, y, z) = (x + y)z$$

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



Chain rule:

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

Backpropagation: a simple example

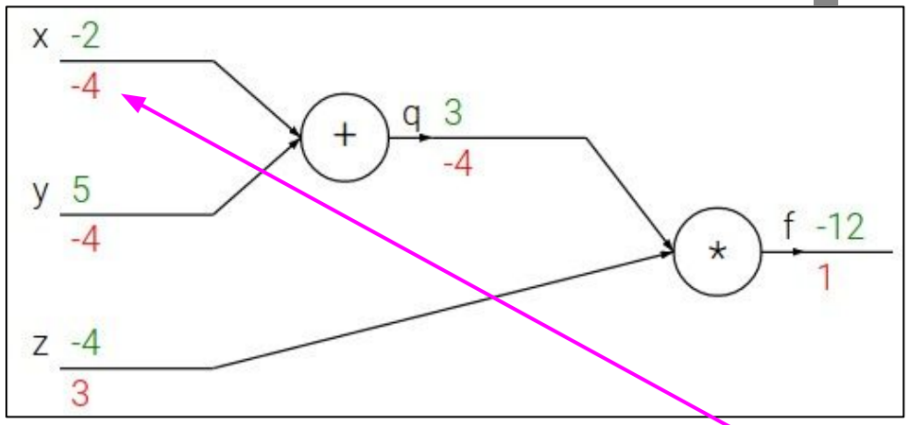
$$f(x, y, z) = (x + y)z$$

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

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

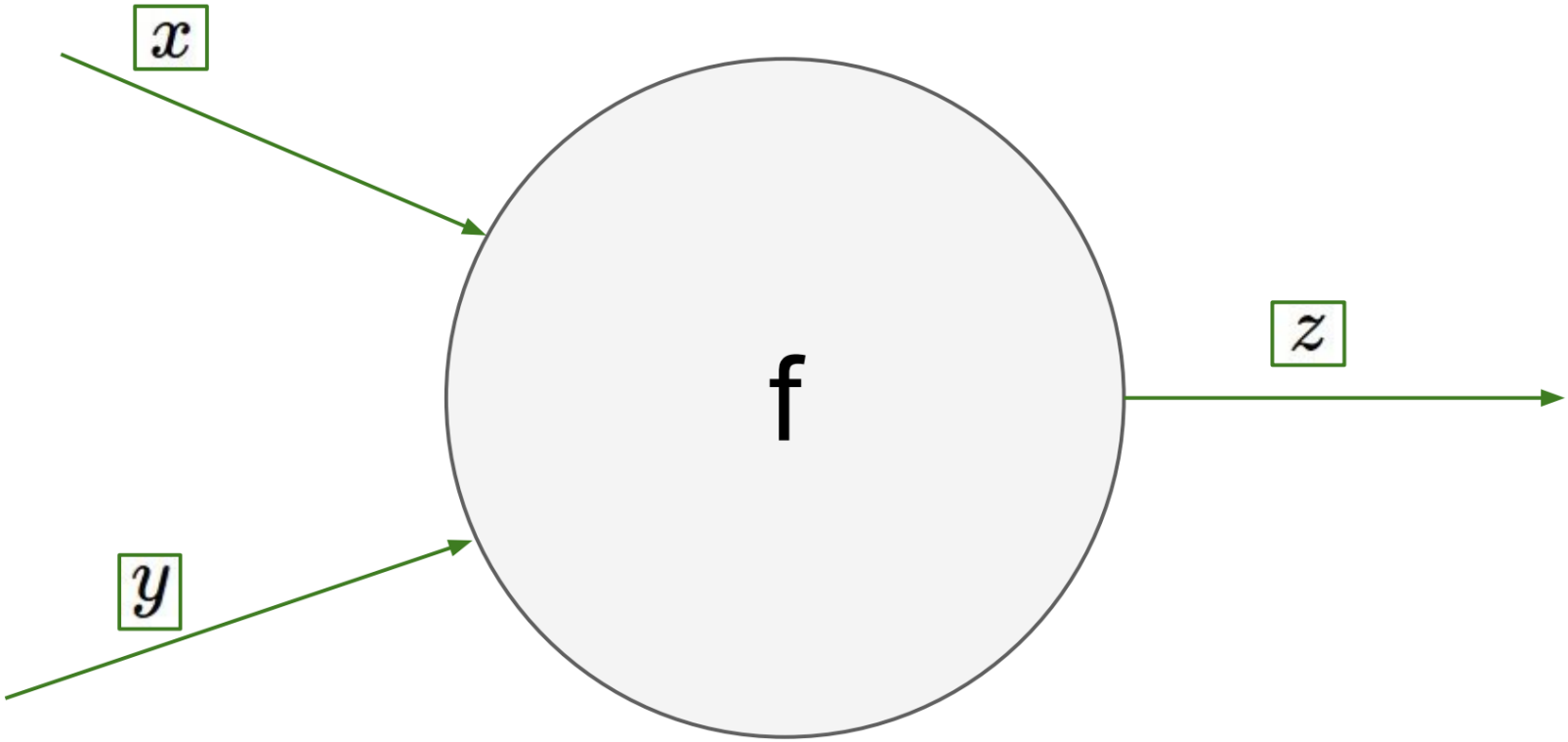
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

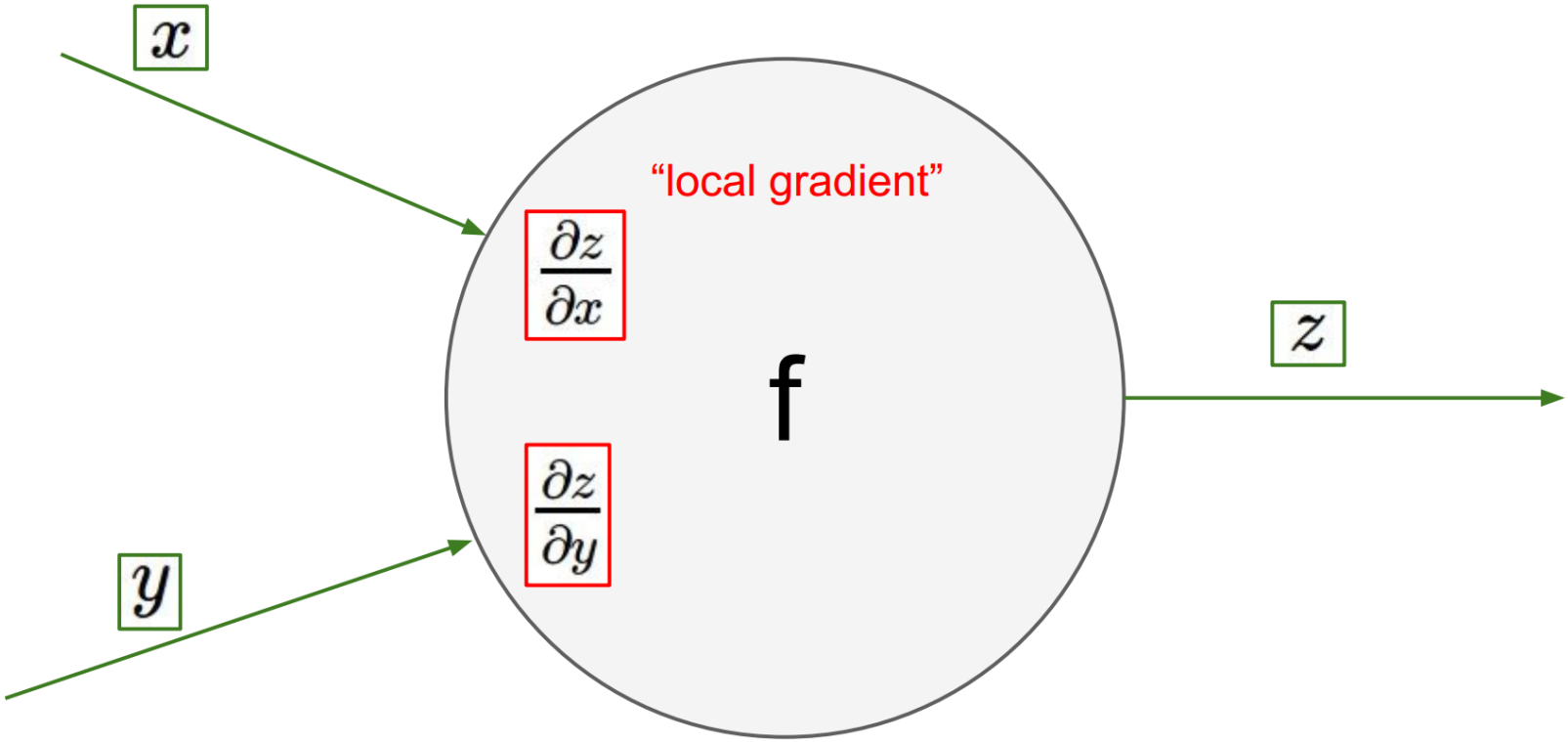
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

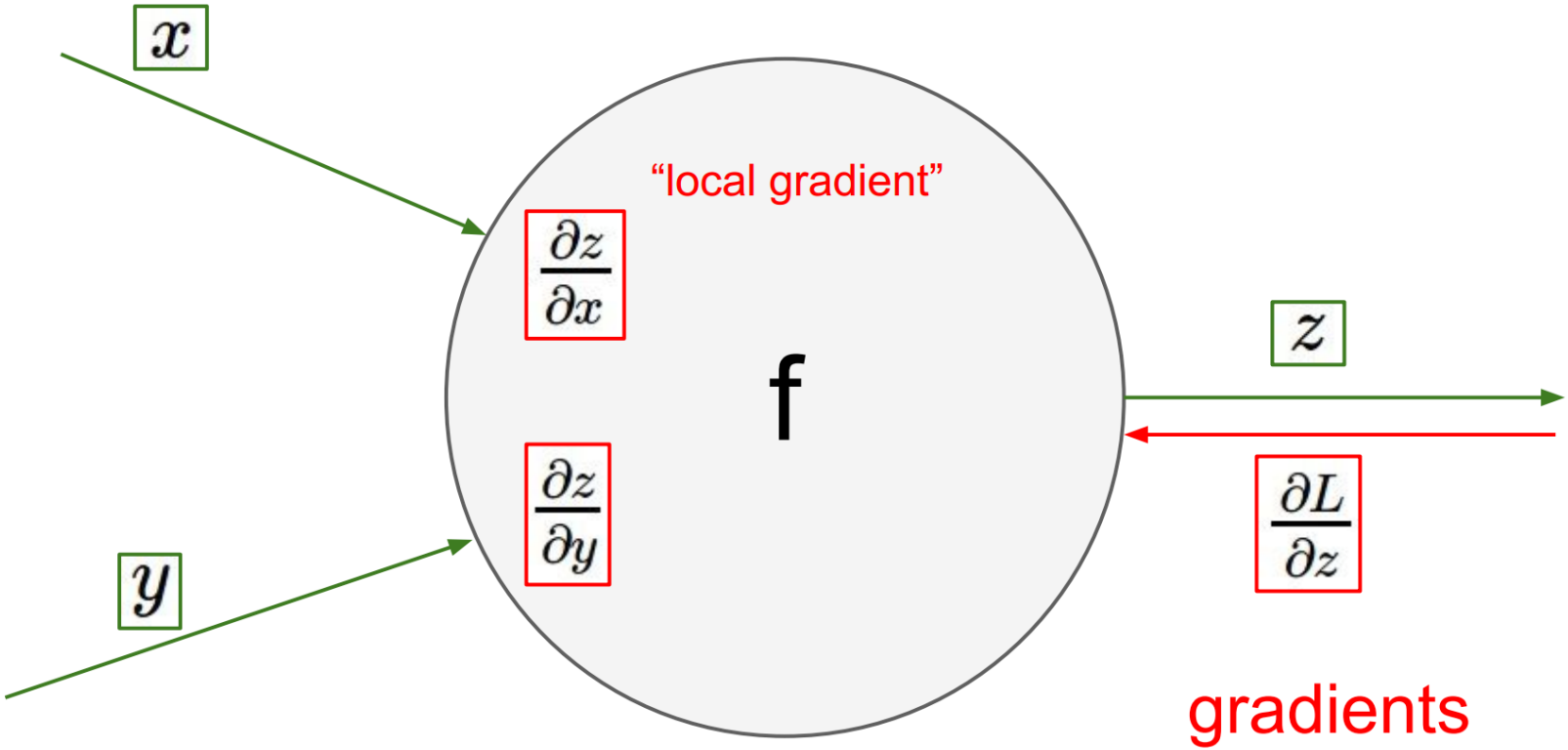


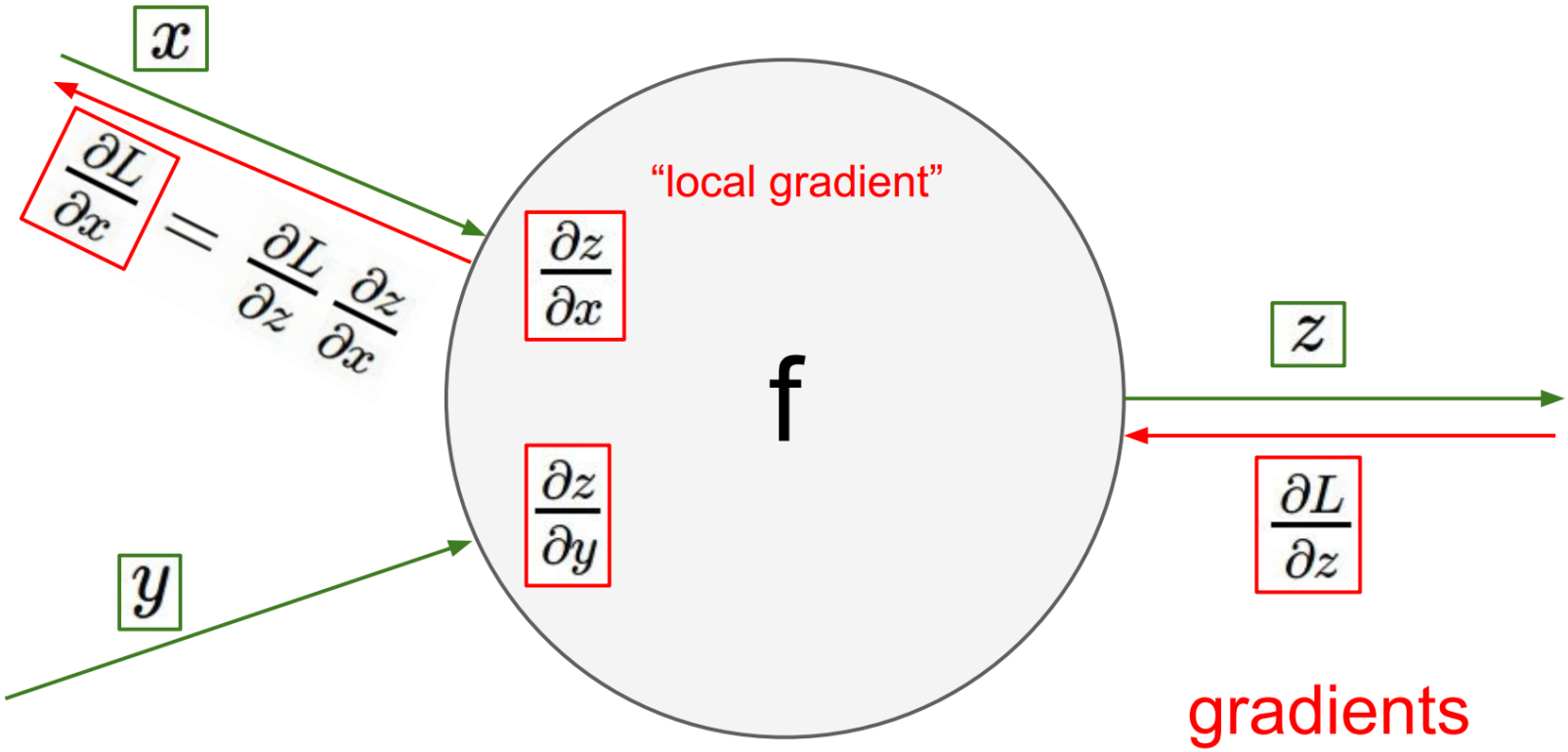
Chain rule:

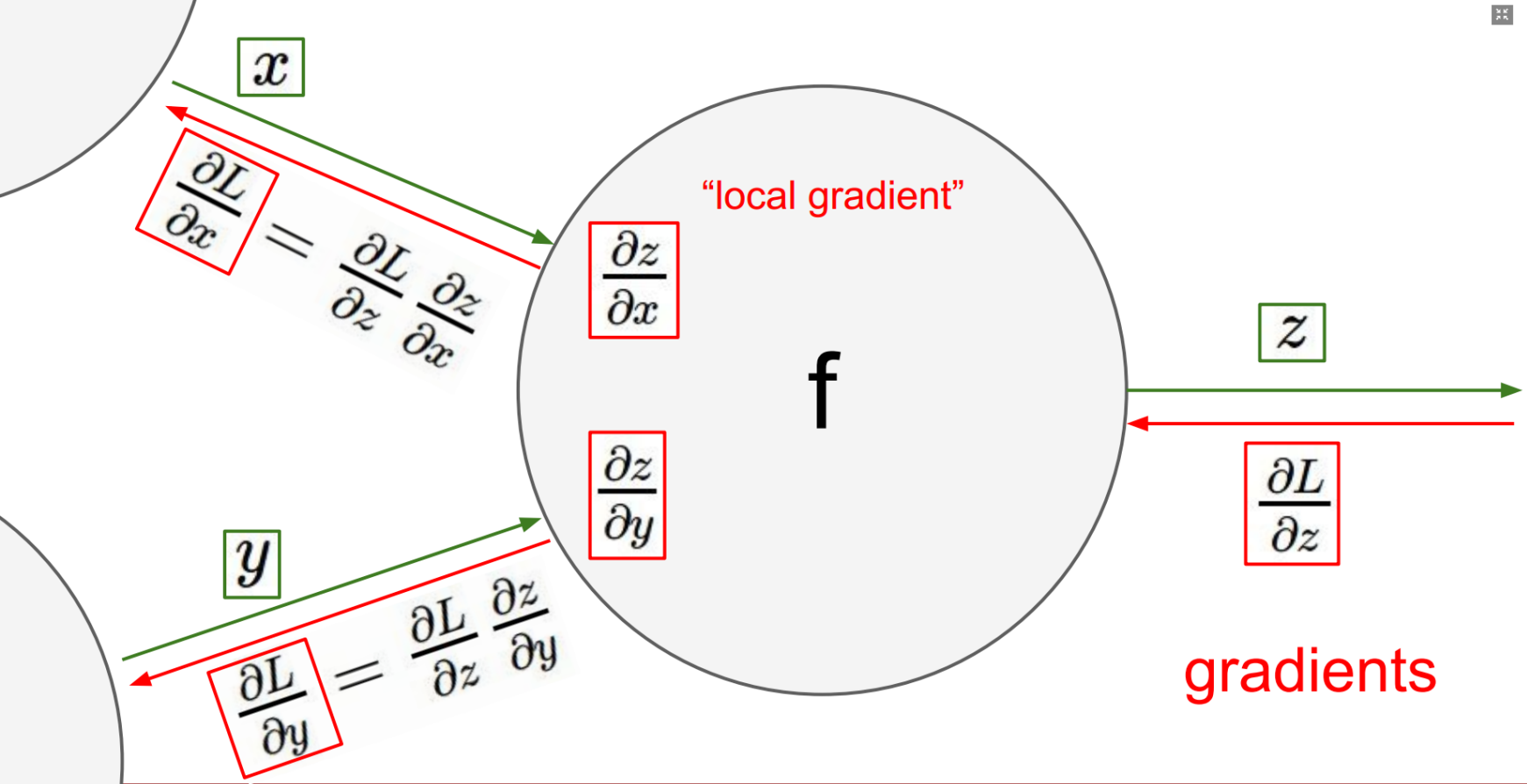
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$


x
y
f
z

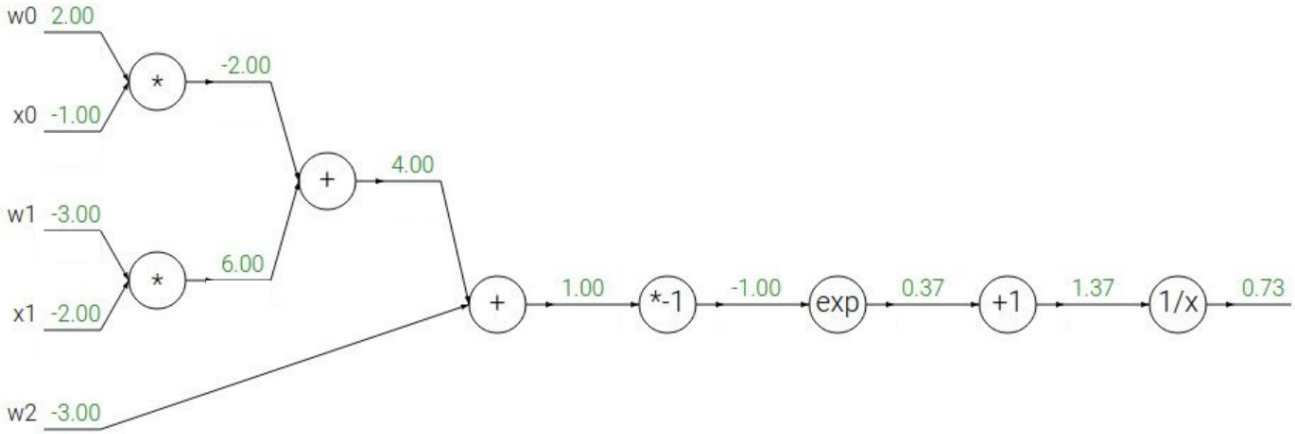

x
y
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z


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

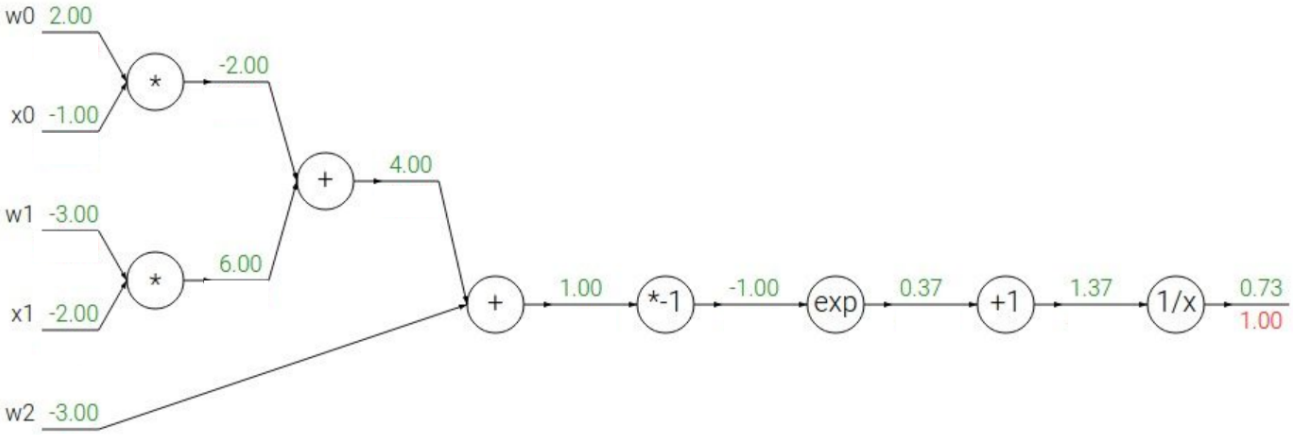

x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
gradients


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

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



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



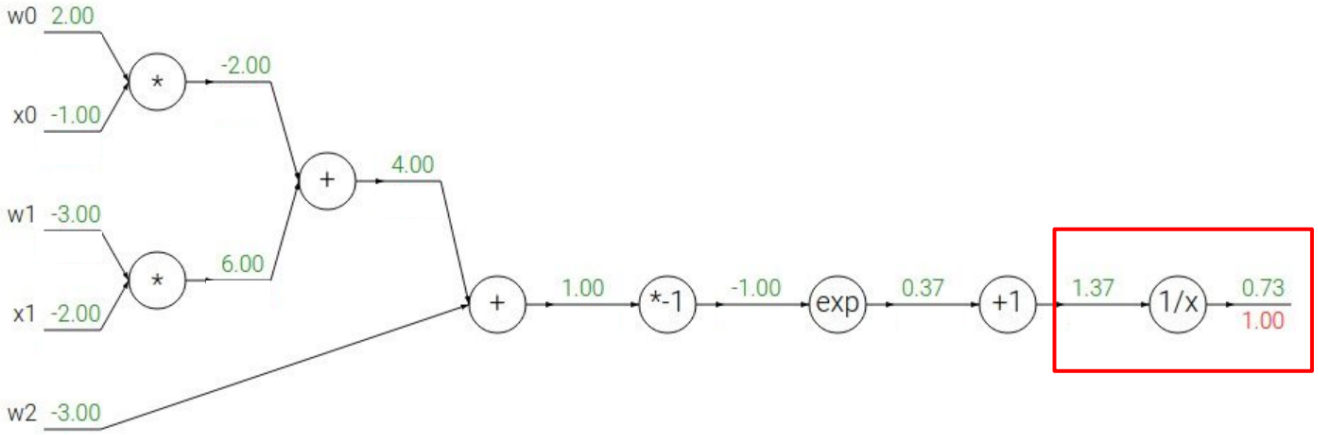
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



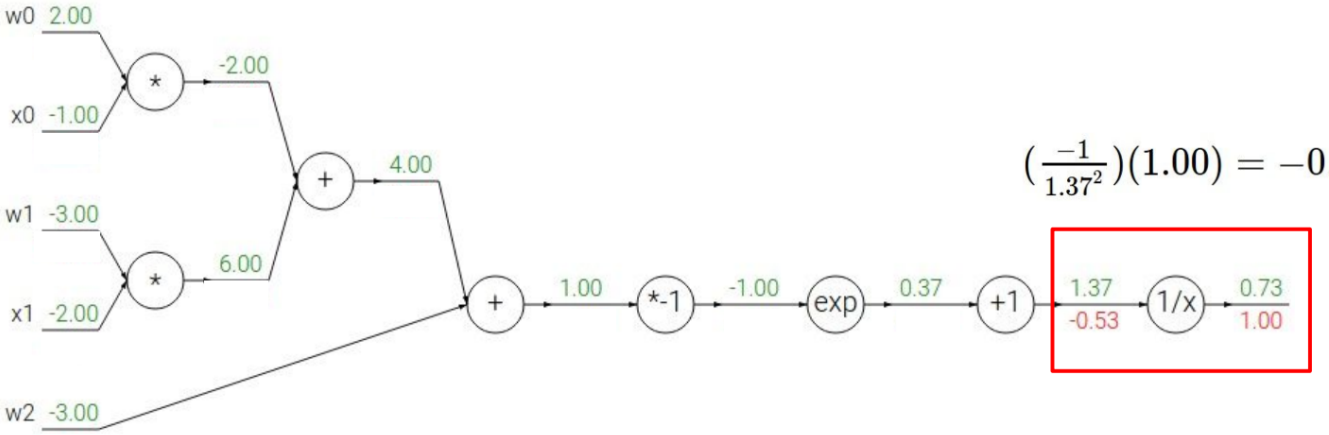
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



$$(\frac{-1}{1.37^2})(1.00) = -0.53$$

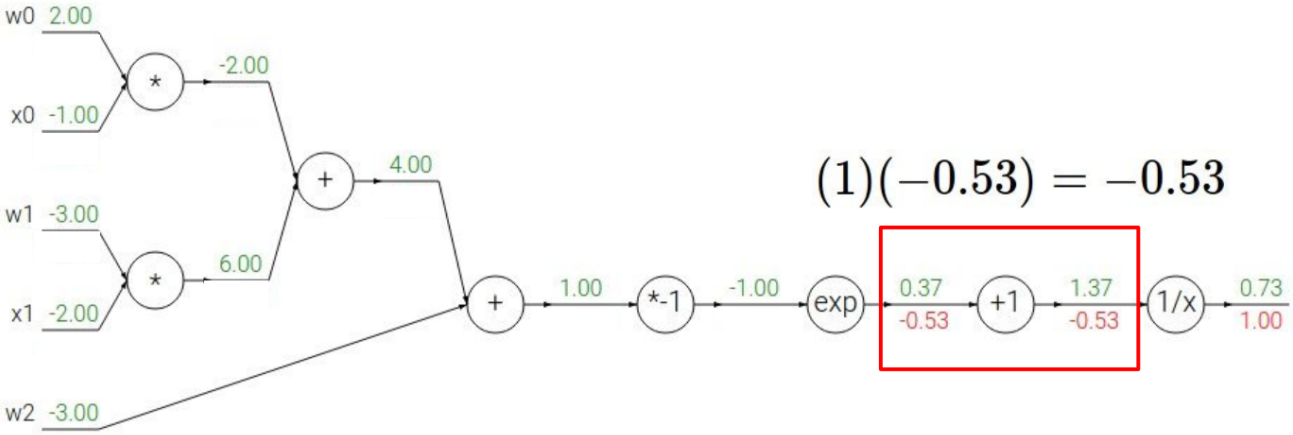
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



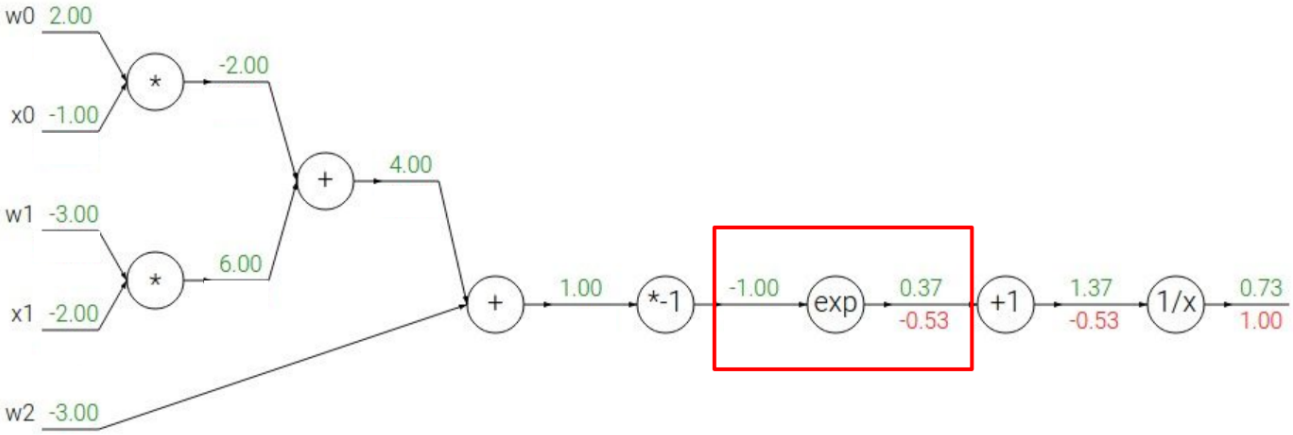
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



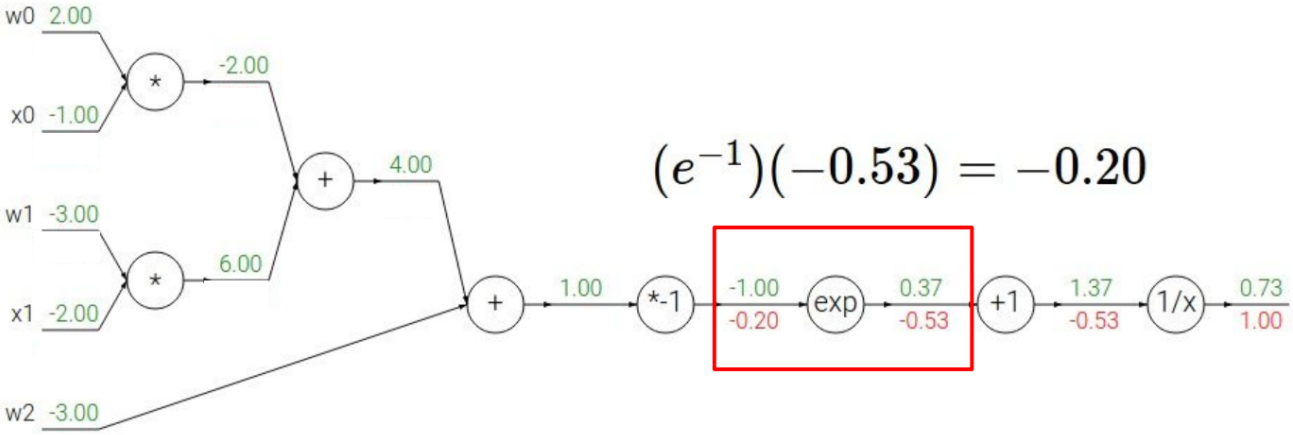
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



$$(e^{-1})(-0.53) = -0.20$$

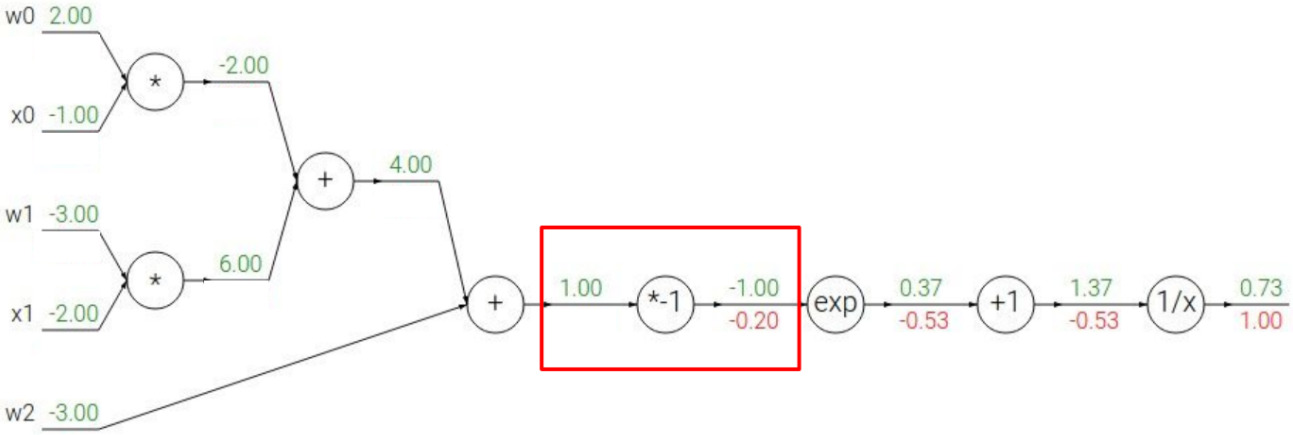
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



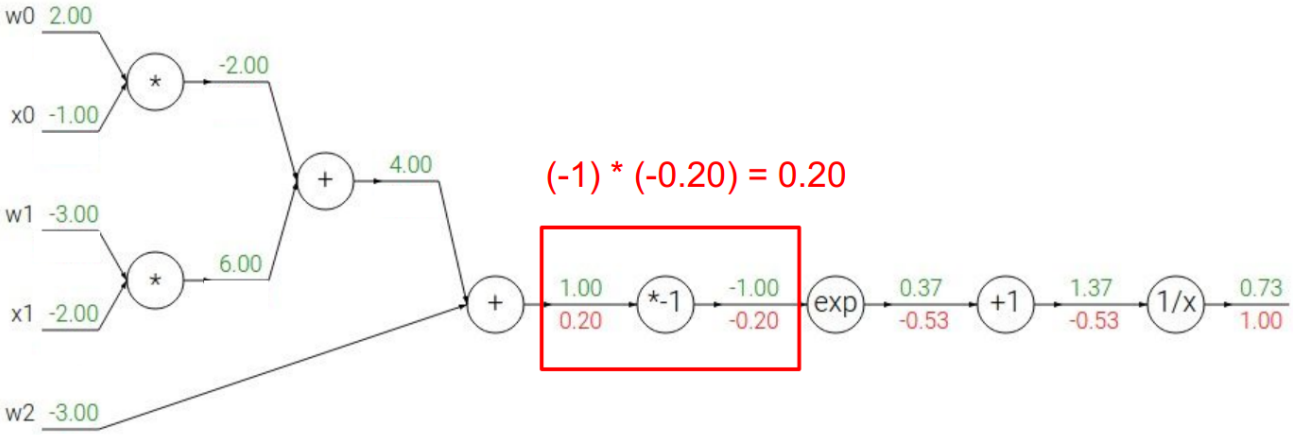
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



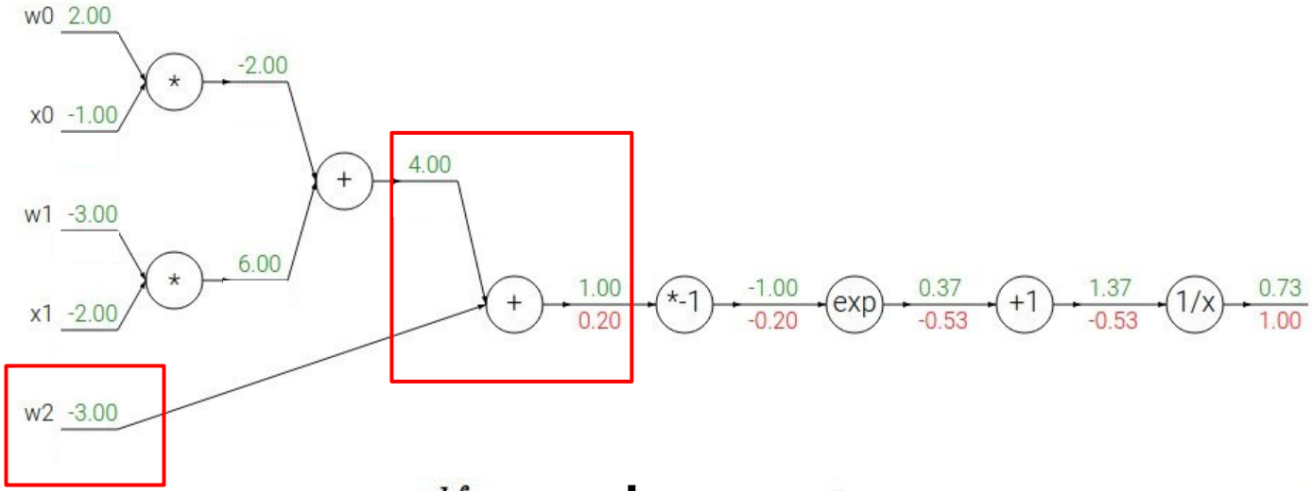
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



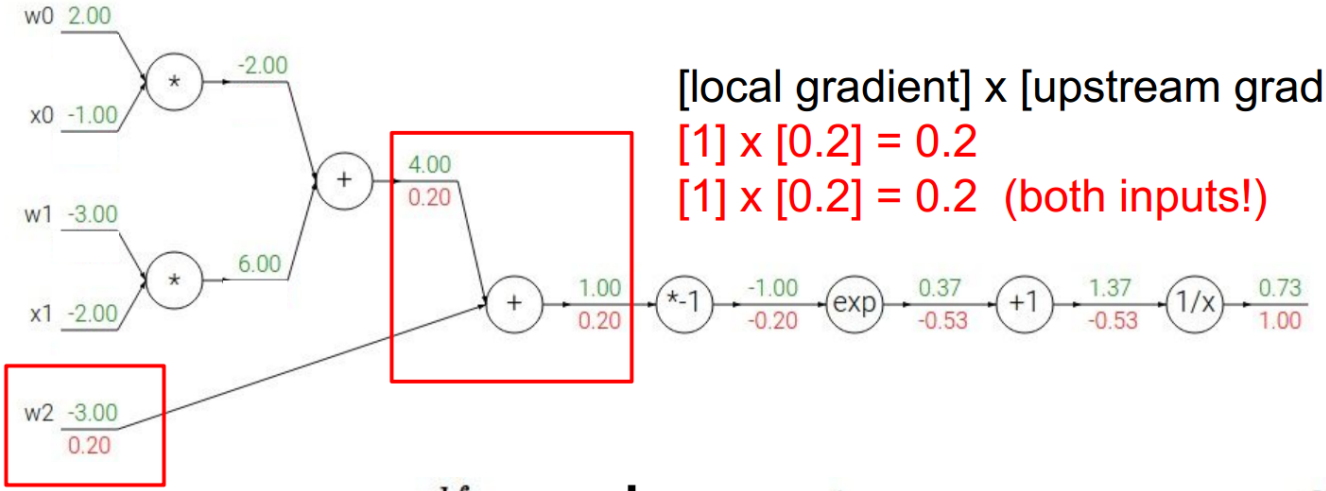
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



[local gradient] x [upstream gradient]

[1] x [0.2] = 0.2

[1] x [0.2] = 0.2 (both inputs!)

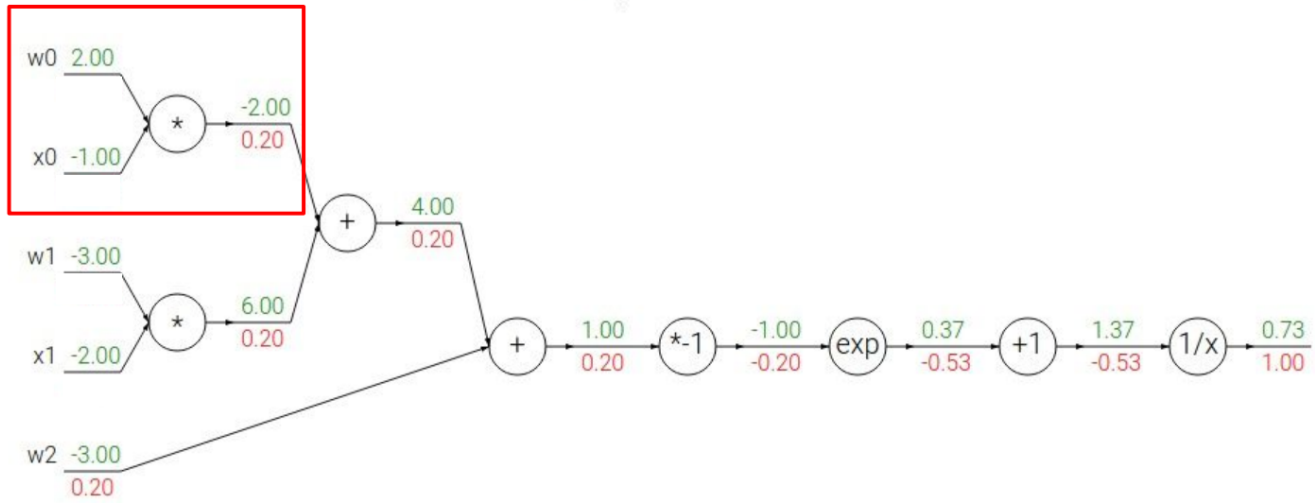
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

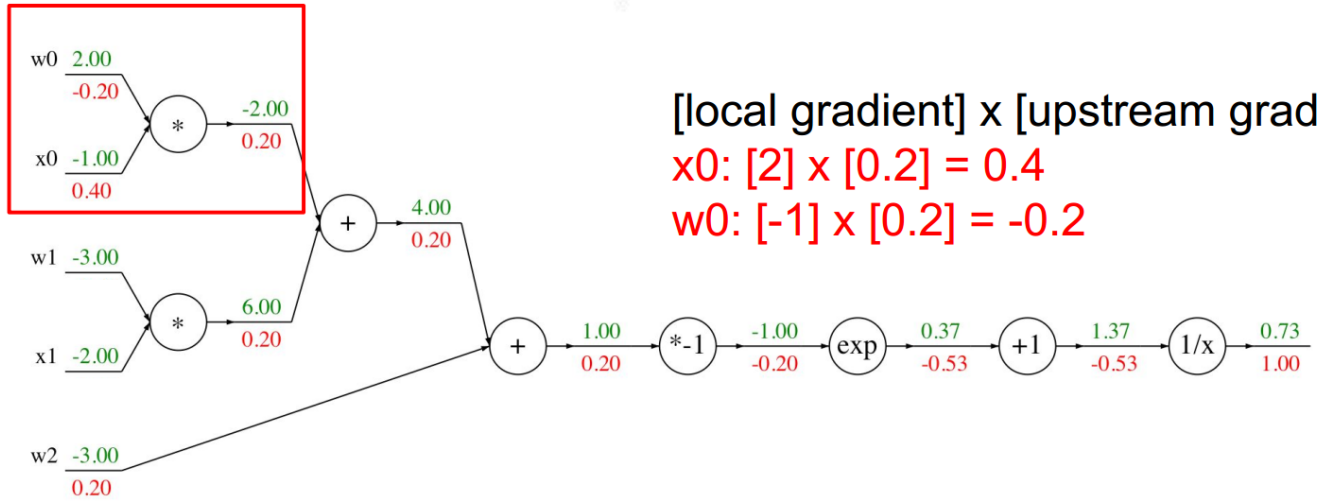
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Another example:

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



[local gradient] x [upstream gradient]

x0: [2] x [0.2] = 0.4

w0: [-1] x [0.2] = -0.2

$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

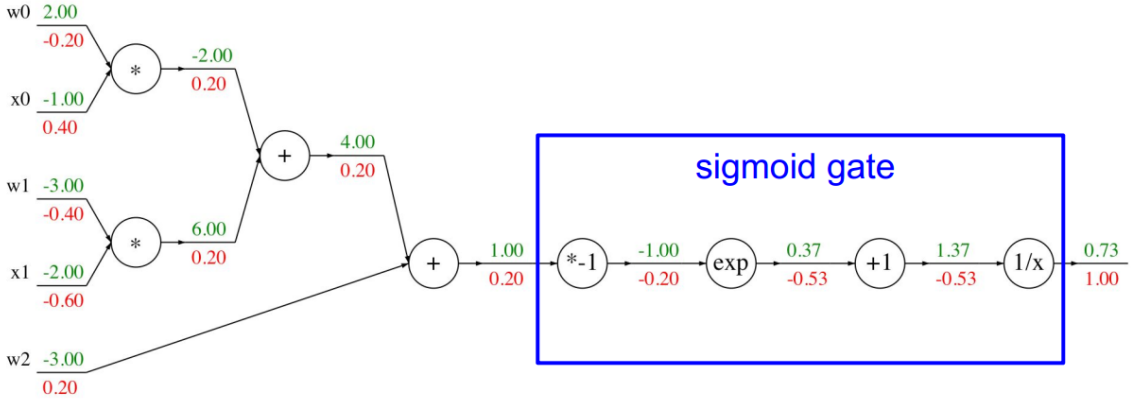
$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

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

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

sigmoid function

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

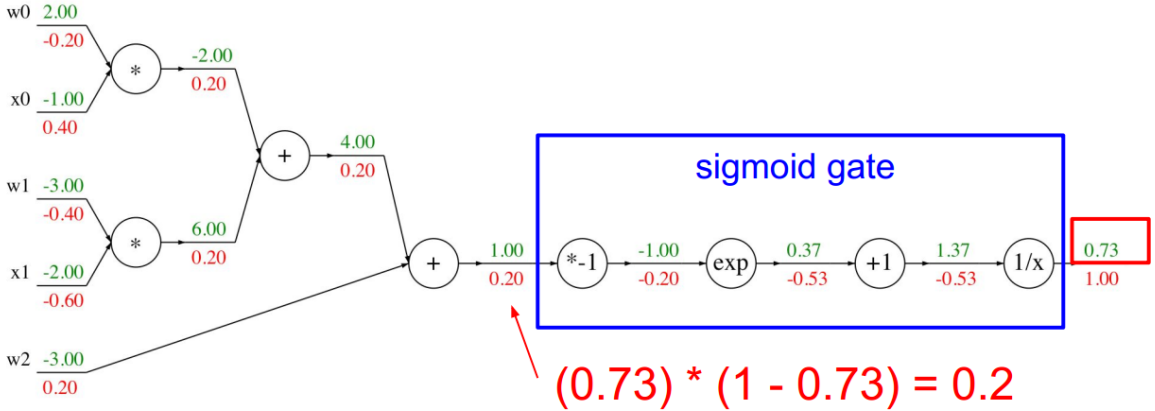


$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

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

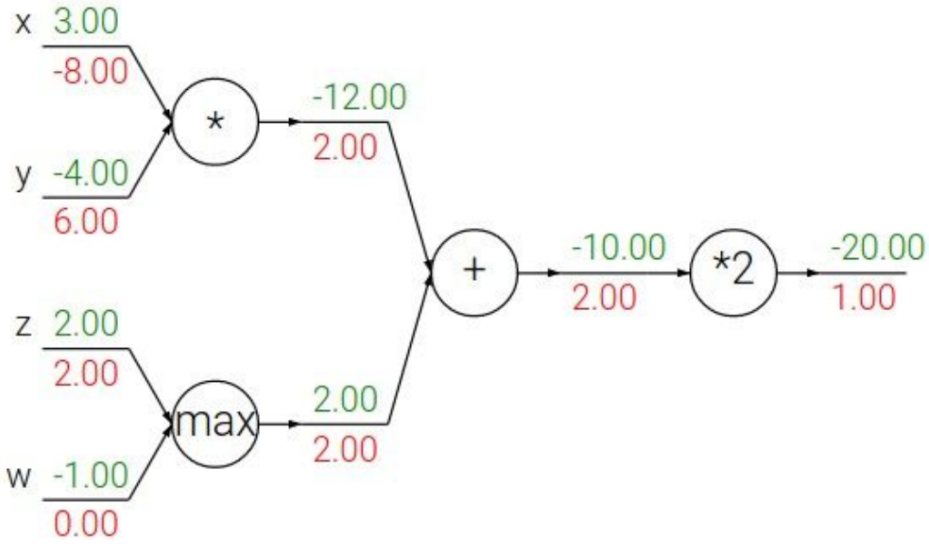
sigmoid function

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



Patterns in backward flow

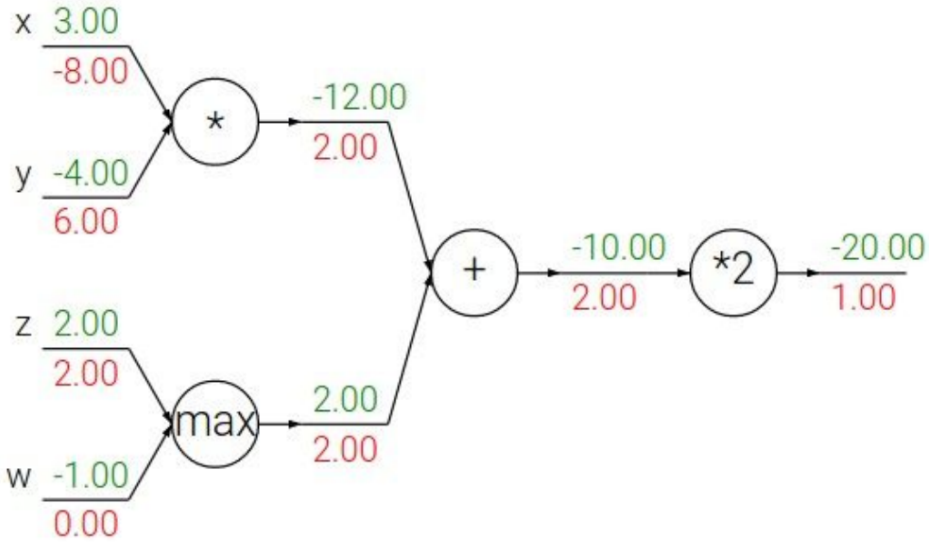
add gate: gradient distributor



Patterns in backward flow

add gate: gradient distributor

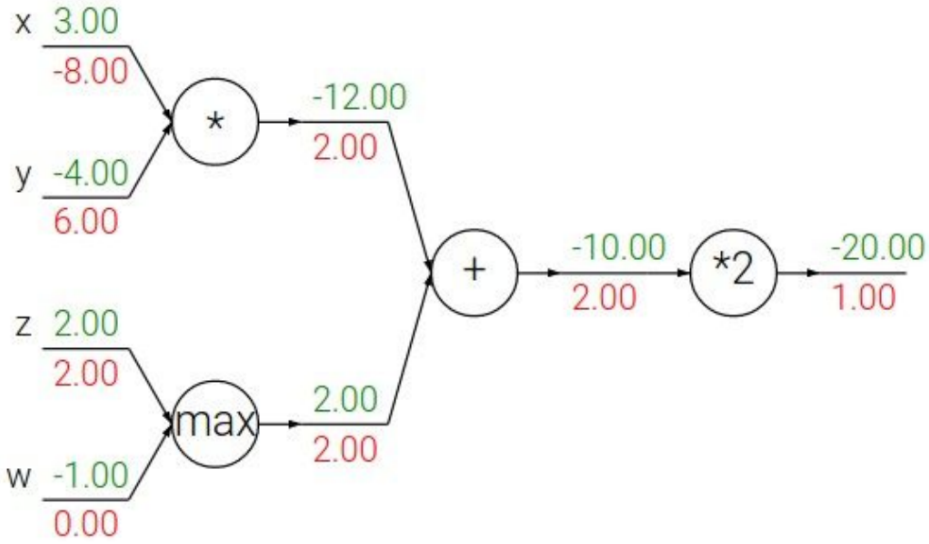
Q: What is a **max** gate?



Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

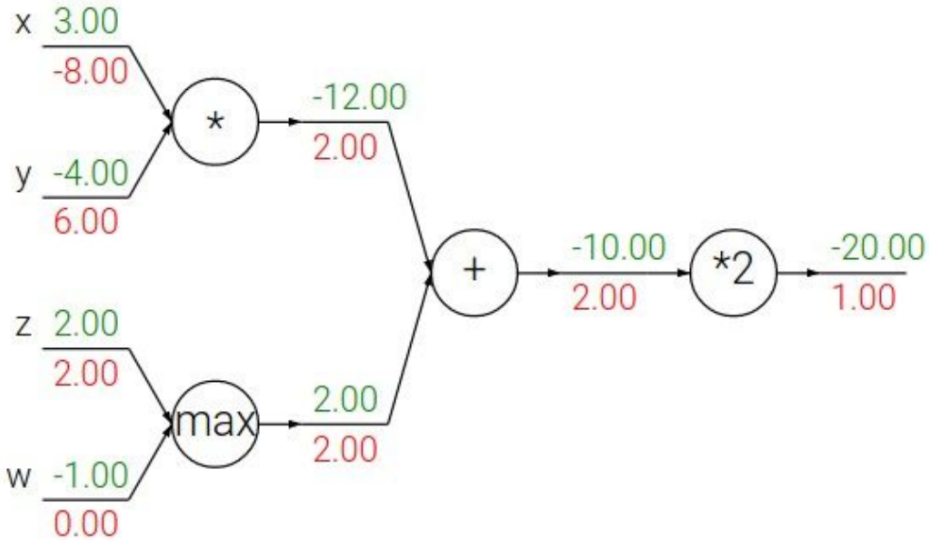


Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

Q: What is a **mul** gate?

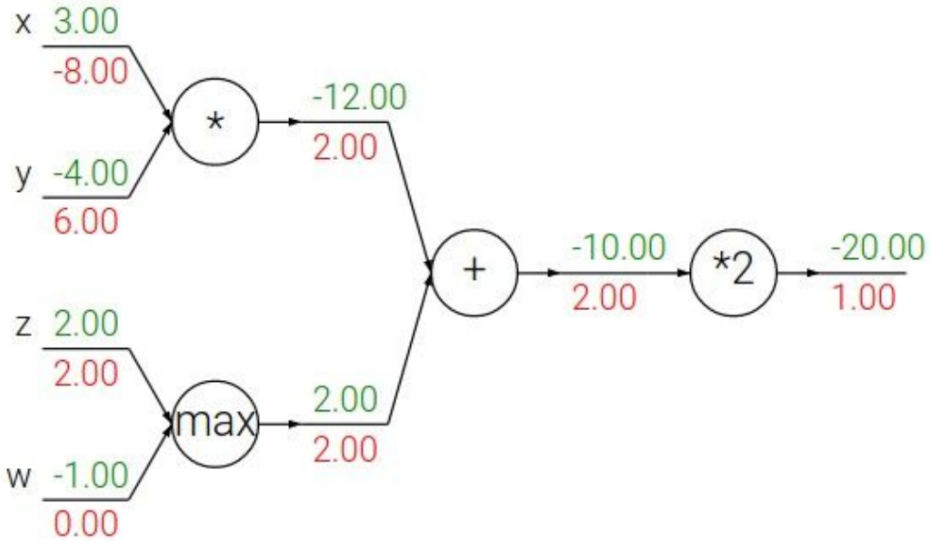


Patterns in backward flow

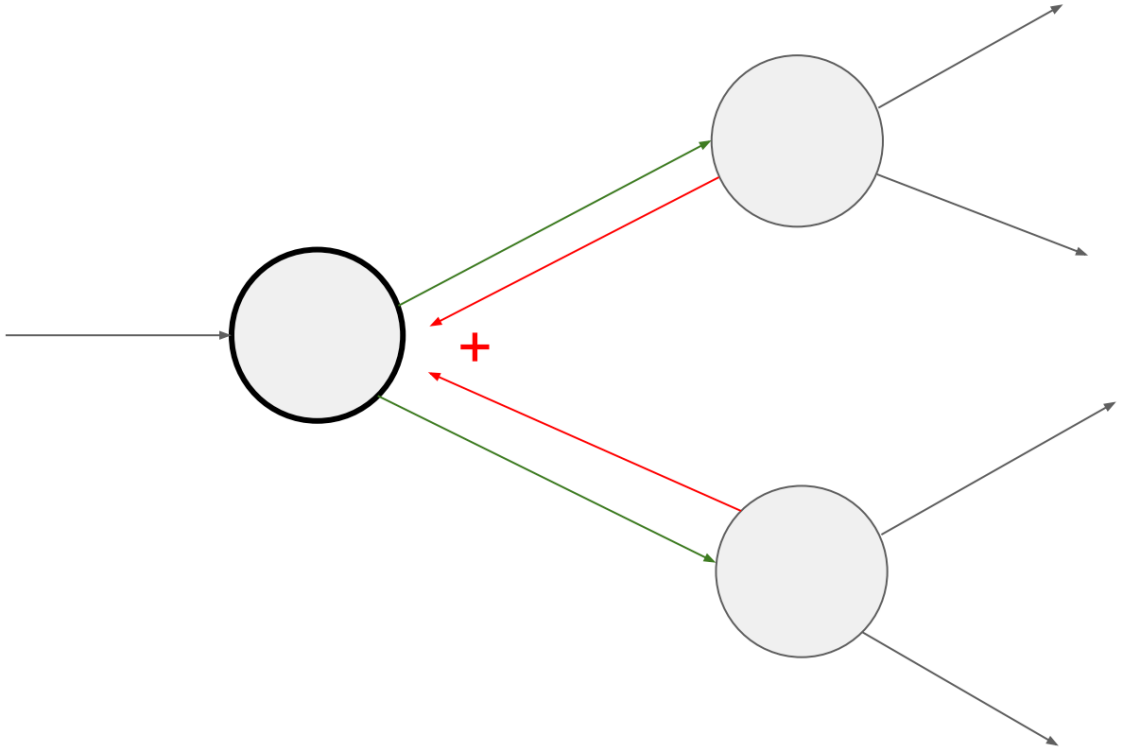
add gate: gradient distributor

max gate: gradient router

mul gate: gradient switcher



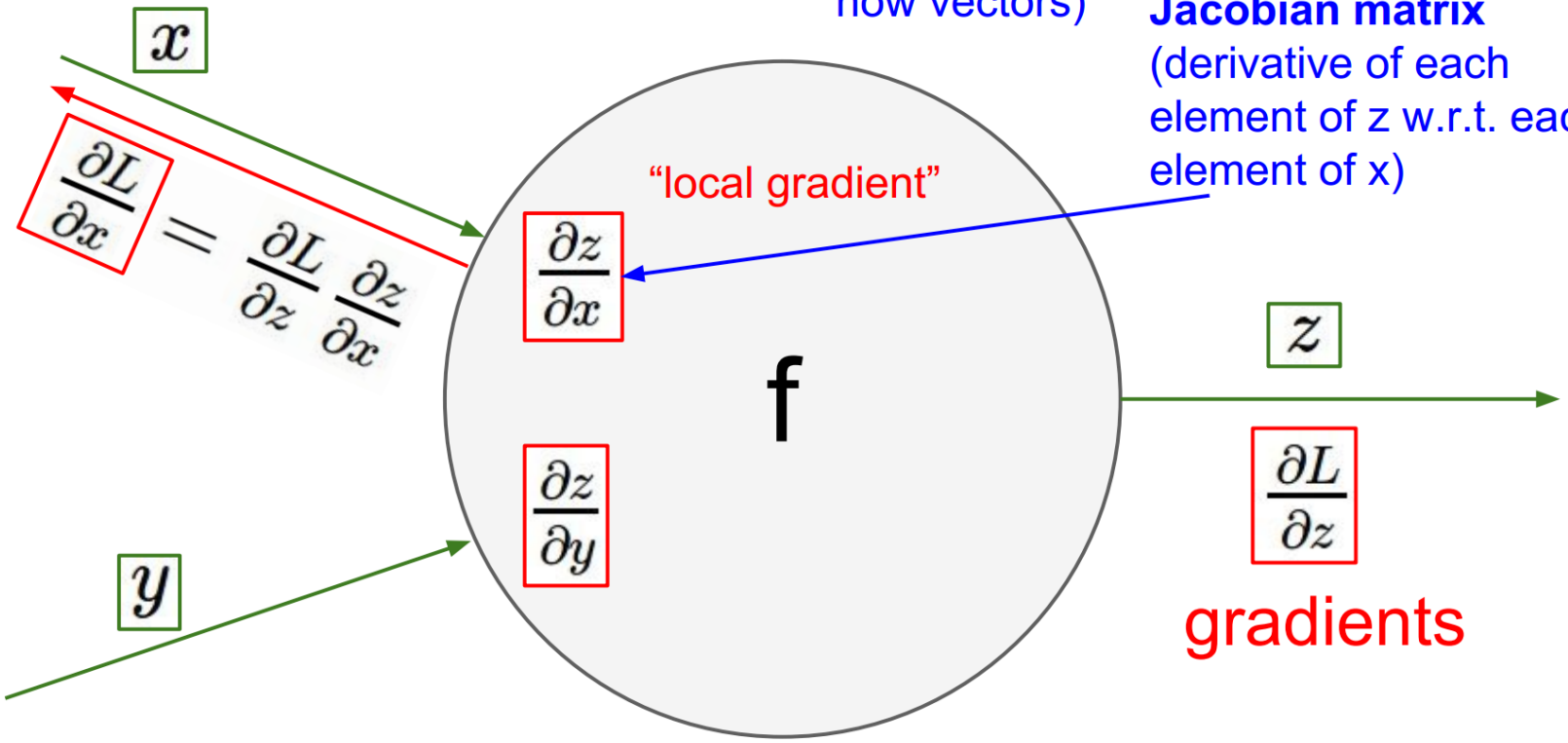
Gradients add at branches



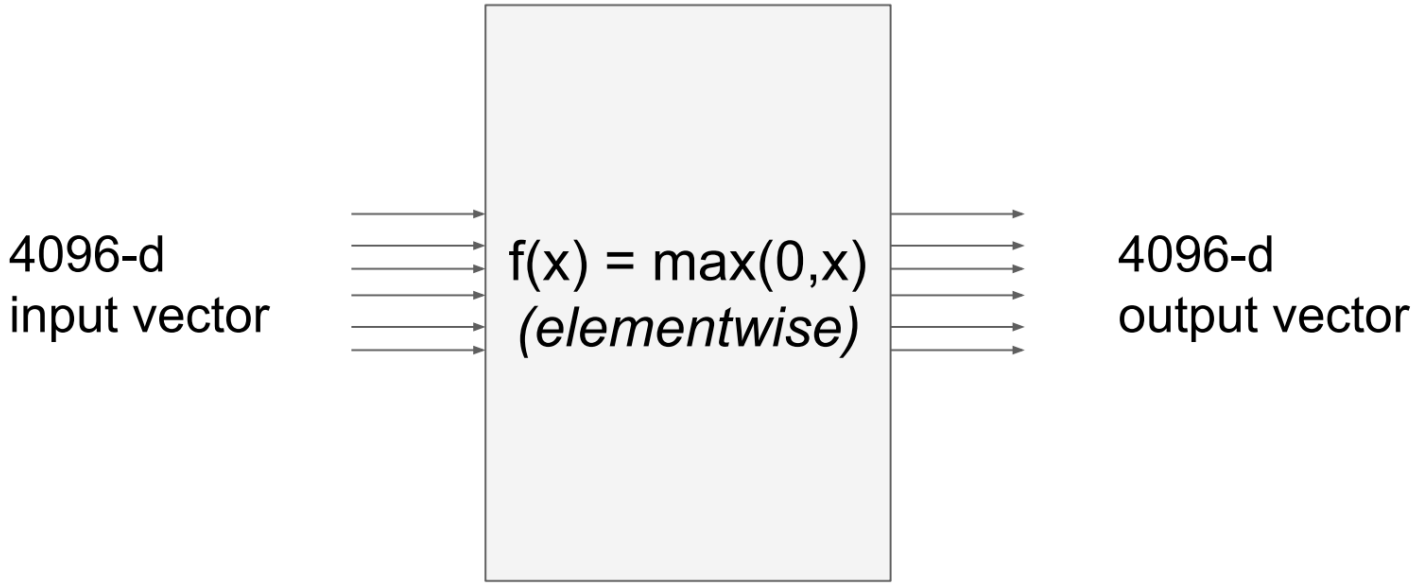
Gradients for vectorized code

(x,y,z are now vectors)

This is now the **Jacobian matrix** (derivative of each element of z w.r.t. each element of x)



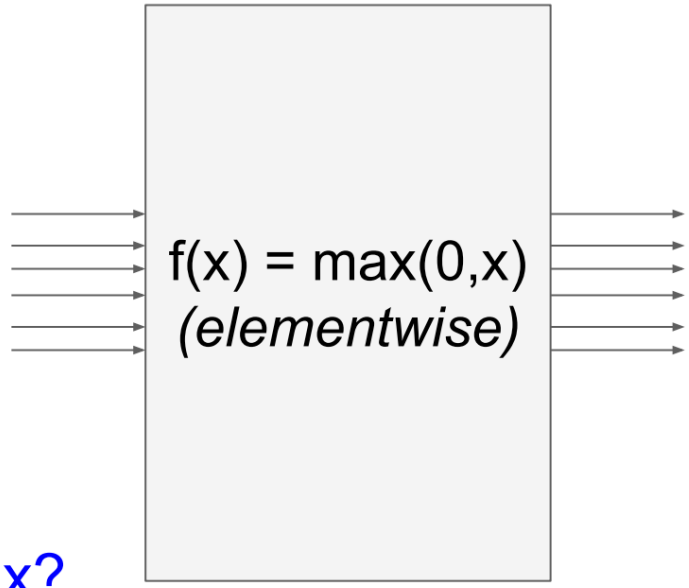
Vectorized operations



Vectorized operations

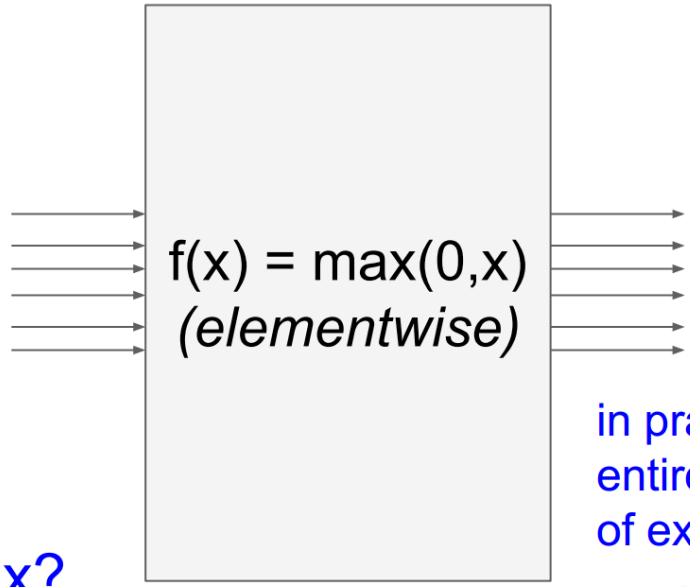
$$\frac{\partial L}{\partial x} = \boxed{\frac{\partial f}{\partial x}} \frac{\partial L}{\partial f}$$

Jacobian matrix



Q: what is the size of the Jacobian matrix?

Vectorized operations



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 :\

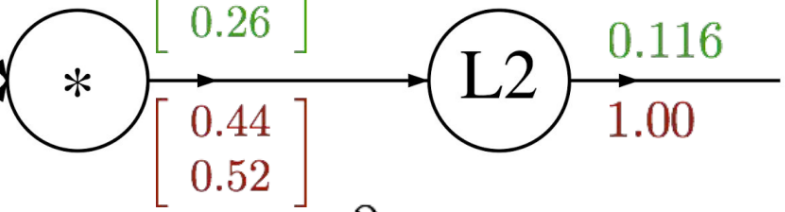
A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix}$$ W

$$\begin{bmatrix} 0.088 & 0.176 \\ 0.104 & 0.208 \end{bmatrix}$$

$$\begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$ x

$$\begin{bmatrix} -0.112 \\ 0.636 \end{bmatrix}$$



$$\begin{bmatrix} 0.22 \\ 0.26 \end{bmatrix}$$

$$\begin{bmatrix} 0.44 \\ 0.52 \end{bmatrix}$$

0.116

1.00

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

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

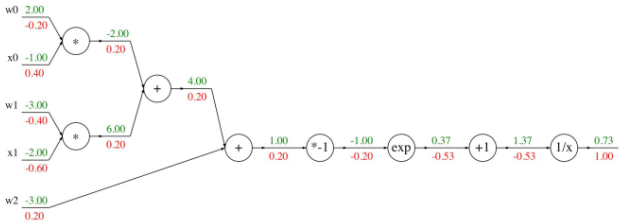
$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

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

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

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

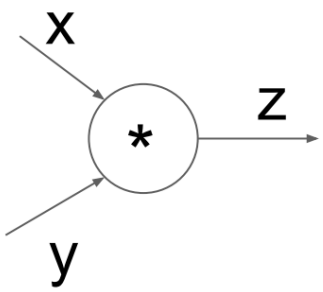
Modularized implementation: forward / backward API



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
```

Modularized implementation: forward / backward API



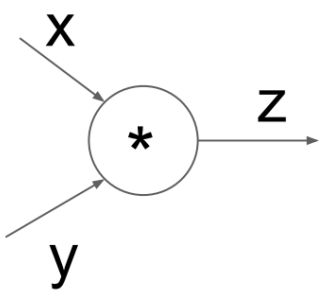
(x,y,z are scalars)

```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        return z
    def backward(dz):
        # dx = ... #todo
        # dy = ... #todo
        return [dx, dy]
```

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

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

Modularized implementation: forward / backward API



(x,y,z are scalars)

```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        self.x = x # must keep these around!
        self.y = y
        return z
    def backward(dz):
        dx = self.y * dz # [dz/dx * dL/dz]
        dy = self.x * dz # [dz/dy * dL/dz]
        return [dx, dy]
```

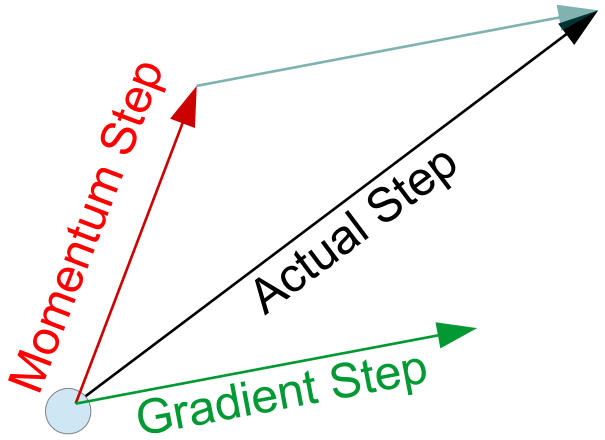
Bonus Round!

Optimization methods:

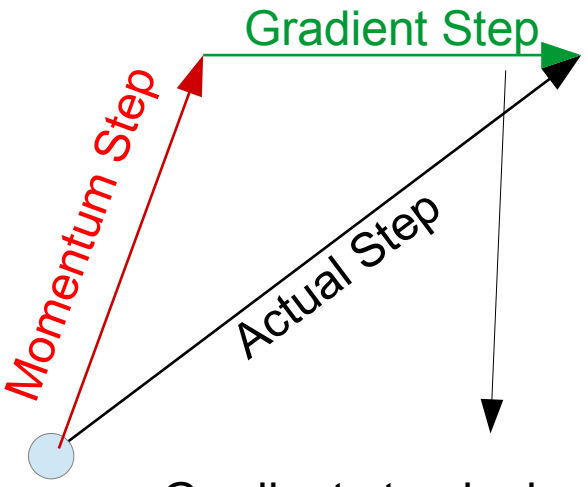
- (Stochastic) Gradient Descent
  - Parameter = parameter – learning rate * gradient
- Momentum:
  - Parameter = parameter + momentum(gradient)

    momentum accumulates an exponentially decaying average of past gradients
- Nesterov momentum:
  - Parameter = parameter + Nesterov_momentum(gradient)

Momentum Update



Nesterov momentum Update



Gradient step is done after the momentum step as a 'correction'

- AdaGrad
  - Parameter = parameter – learning rate x gradient / sqrt(accumulated gradient norms)
  - Basically slower learning rate at high gradients and fast learning rate for small gradients
- And others, see:

  https://towardsdatascience.com/neural-network-optimization