# Gradient Descent

# Gradient Descent

- In previous two examples OR and XOR we assumed some values for model parameters(w, b).
- But those parameters learned by network by using learning algorithm Gradient Descent.

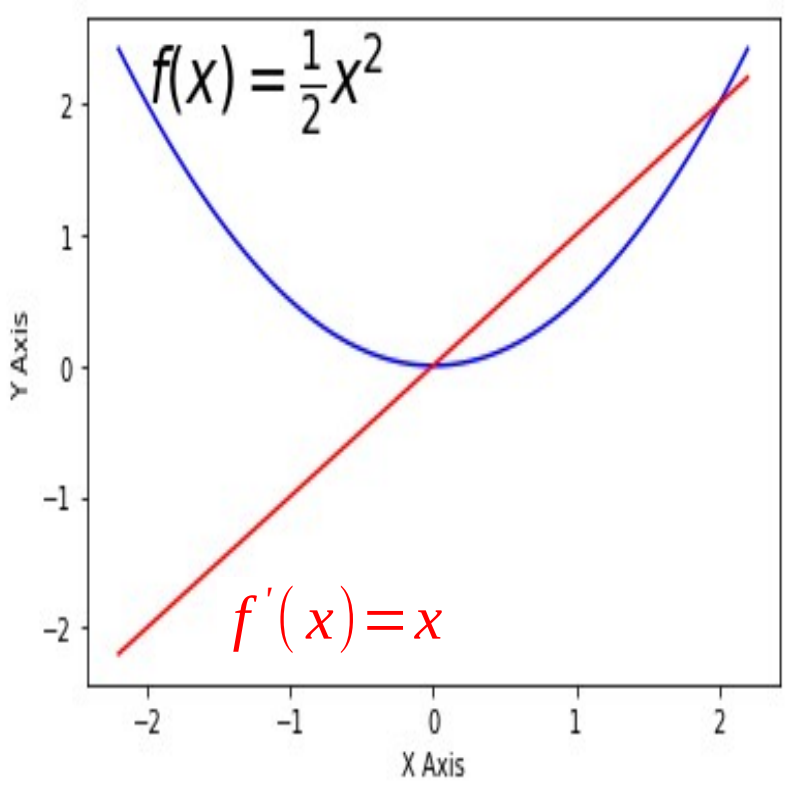
# Gradient-single variable function

- Optimization refers to the task of either minimizing or maximizing some function f(x) by altering x.
- Suppose we have a function y = f(x)
- The derivative of this function is denoted as f'(x) or as dy/dx.
- The derivative f'(x) gives the slope of f(x) at the point x
- In other words, its specifies how to scale a small change in the input inorder to obtain the corresponding change in the output: $f(x+h) \approx f(x) + f'(x)h$

**Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.**

# Gradient-single variable function

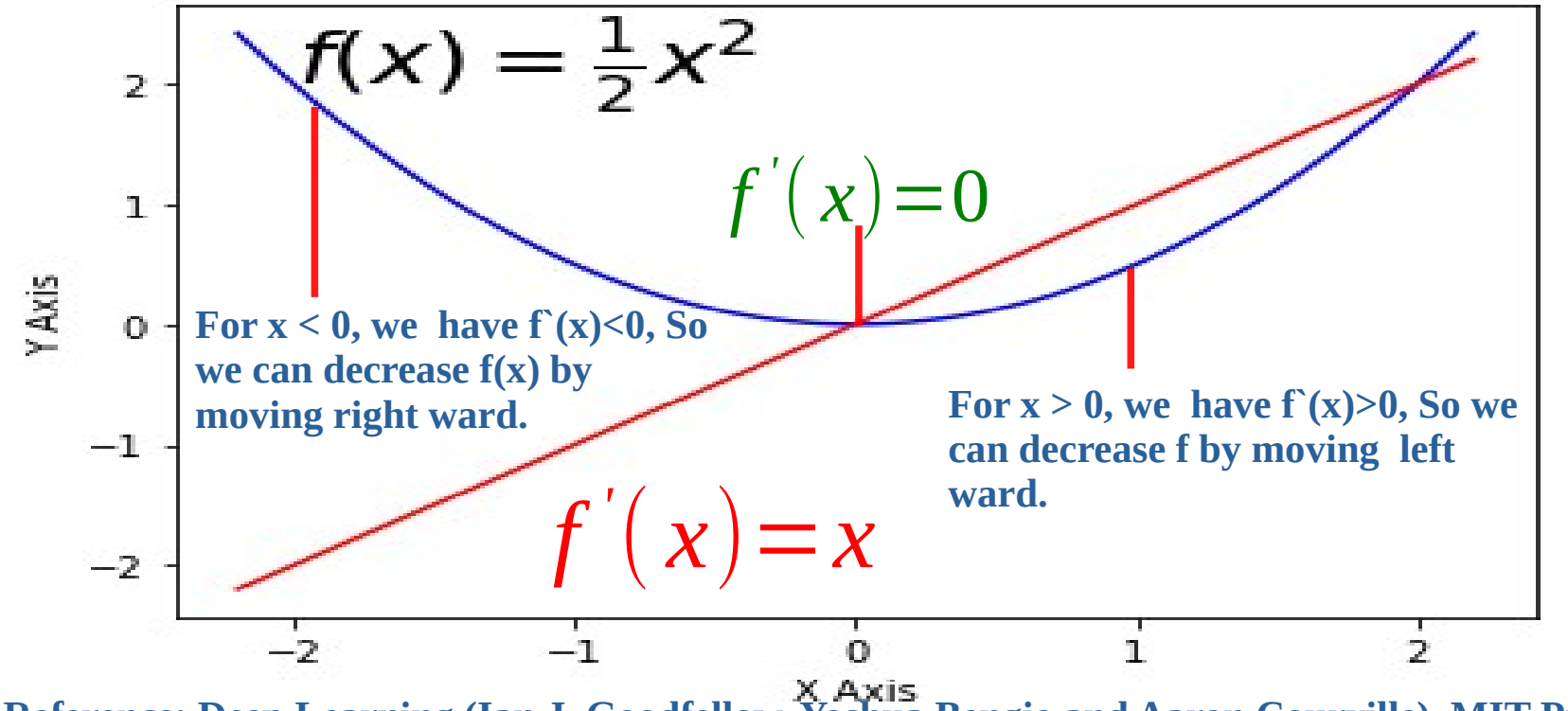
- If f(x) is differentiable at x
- f(x) will increase if its derivative is positive
- f(x) will decrease if its derivative is negative
- So the derivative is useful for minimizing a function
- Because it tells us how to change x inorder to make a small improvement in f(x)




Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

# Gradient-single variable function

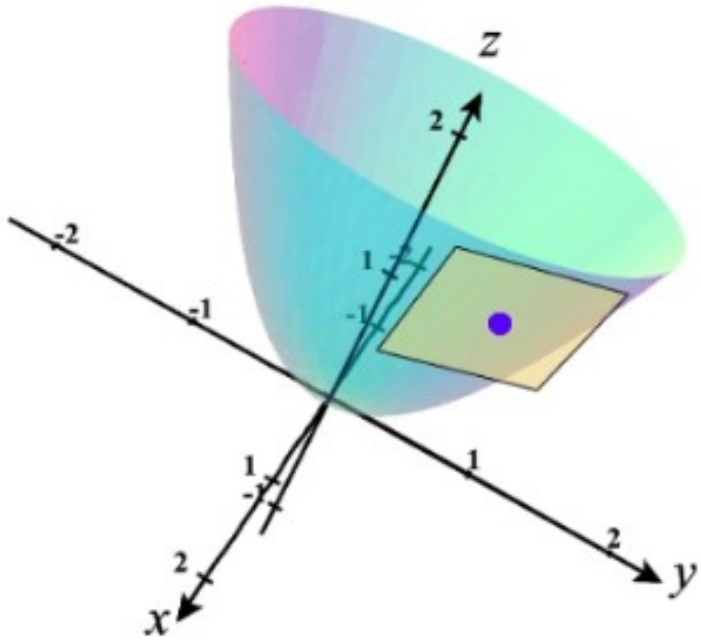
- Here for f(x) global minimum is at x = 0 Since
- Derivative will be zero at local minimum or maximum




Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

# Gradient-Multiariate function

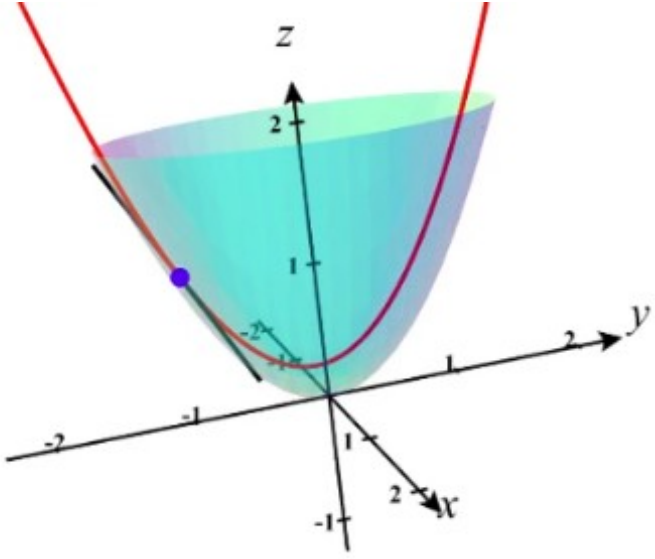
- Gradient is a kind of slop for heigher dimensional function
- The gradient of f(x,y) to be the ‘slope’ of the tangent plane.
- But a plane doesn’t have a single slope;
- it slopes differently in different directions

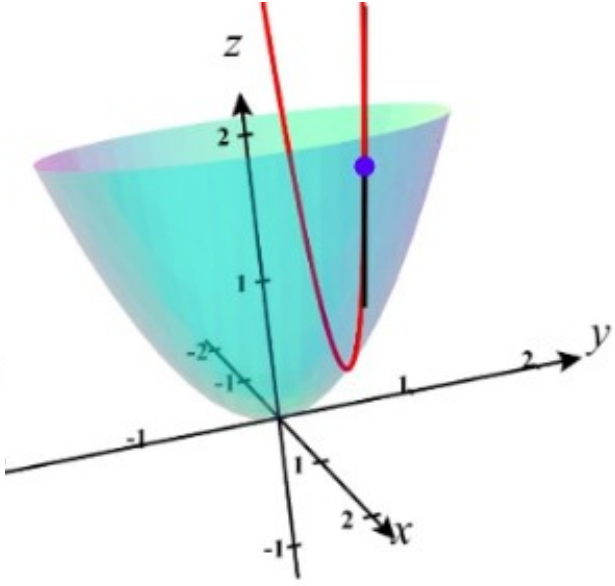


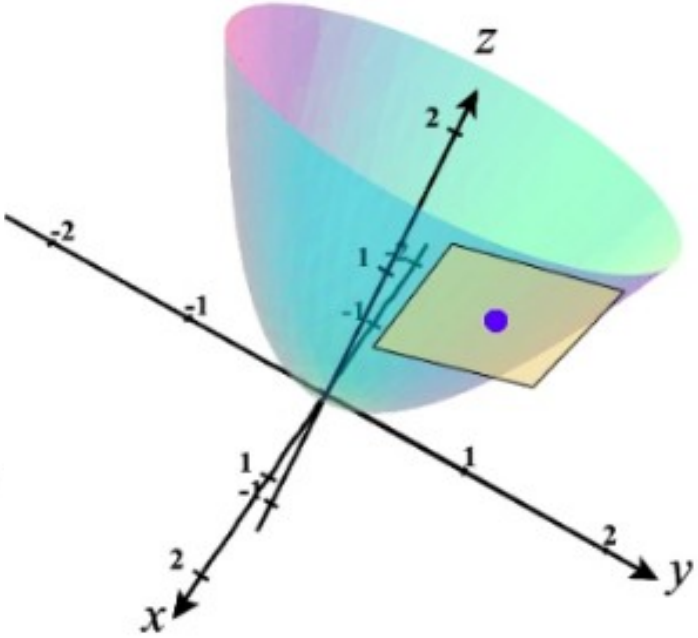

Reference: https://mathcs.clarku.edu/~djoyce/ma131/gradients.pdf

# Gradient-Multiariate function

- It slopes differently in different directions



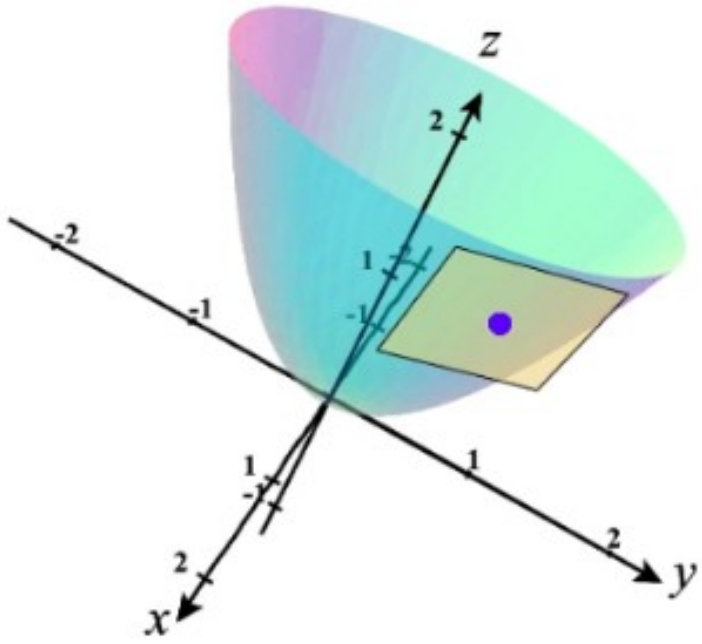




Reference: https://mathcs.clarku.edu/~djoyce/ma131/gradients.pdf

# Gradient-Multiariate function

- What is the ‘slope’ of the plane?
- the partial derivatives in all directions, taken together, form the slop of the plane
- f(x,y) has two variables and two partial derivatives: $\partial z/\partial x, \partial z/\partial y$
- We’ll call that vector of these partial derivatives the gradient of f,denoted ∇f
- The gradient of a multi-variable function has a component for each direction.

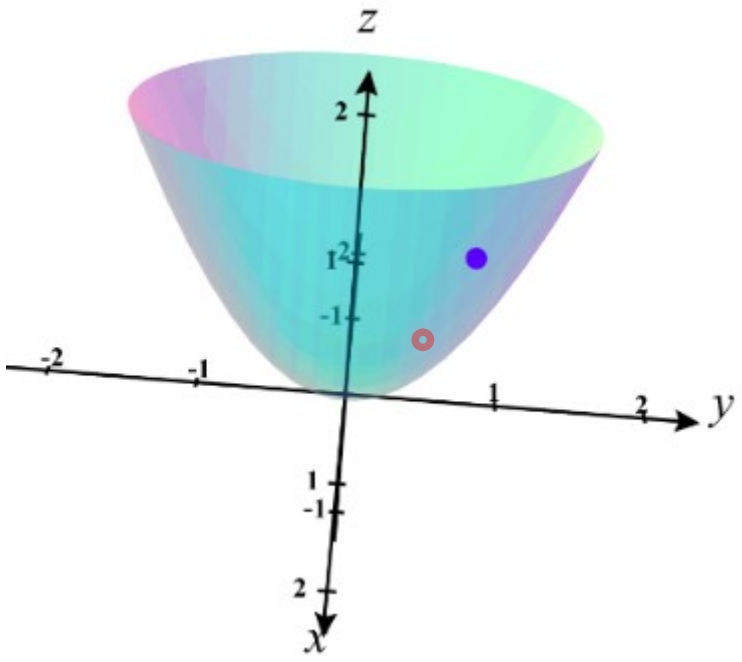



Reference: https://mathcs.clarku.edu/~djoyce/ma131/gradients.pdf

# Gradient-Multiariate function

- The change in height, Δz, is given by
- Δz = (change in x direction)+ (change in y direction)
- Therefore

$$\Delta z = \frac{\partial z}{\partial x}\Delta x + \frac{\partial z}{\partial y}\Delta y$$




Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Based Optimization

- If you imagine standing at a point $(x_0,y_0)$ in the input space of f
- the vector $\nabla f$ tells you which direction you should travel to increase the value of f most rapidly.
- **It is a vector that points in the direction in which the function grows the fastest**.
- Its coordinates are partial derivatives of that function

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Reference: https://mathcs.clarku.edu/~djoyce/ma131/gradients.pdf

# Gradient-Based Optimization

- Consider a simple multiplication function of two numbers $f(x, y) = xy$
- The partial derivative $\frac{\partial f(x, y)}{\partial x} = \frac{\partial f}{\partial x} = y$ measures how f changes as only the variable x increases.
- The partial derivative $\frac{\partial f(x, y)}{\partial y} = \frac{\partial f}{\partial y} = x$ measures how f changes as only the variable y increases.

Refrence: https://cs231n.github.io/optimization-2/

# Gradient-Based Optimization

- **The derivative on each variable tells you the sensitivity of the whole expression on its value.**
- For example, $if\ x=4, y=-3\ \ then\ f(x,y)=-12$
- and the derivative on x $\frac{\partial f}{\partial x}=y=-3$
- This tells us that if we were to increase the value of this variable by a tiny amount h, the effect on the whole expression would be to decrease it (due to the negative sign), and by three times that amount(-3h).

Refrence: https://cs231n.github.io/optimization-2/

# Gradient-Based Optimization

- Similarly, since $\frac{\partial f}{\partial y} = x = 4$, we expect that increasing the value of y by some very small amount h would also increase the output of the function (due to the positive sign), and by 4h.

$$\nabla f = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$$

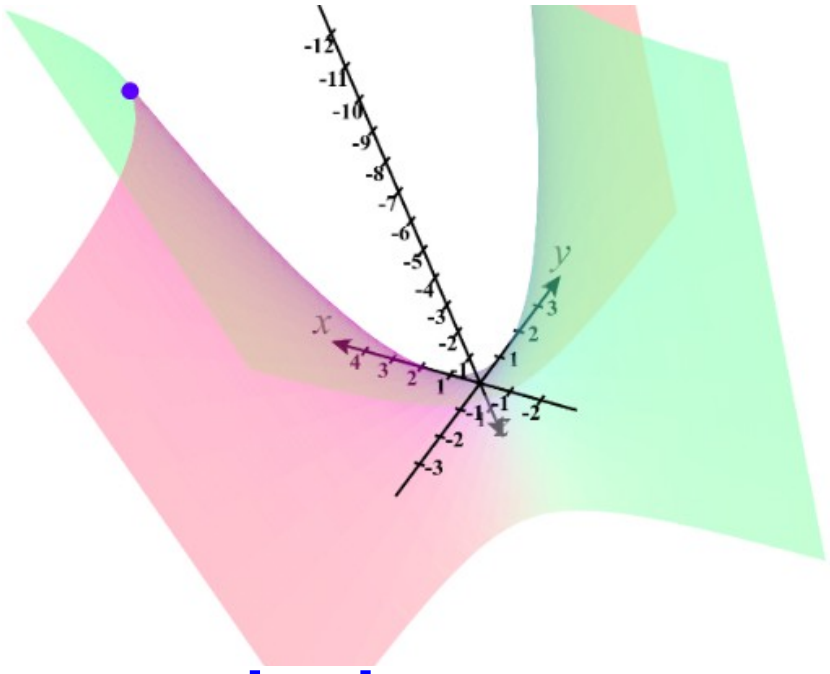
**Refrence: https://cs231n.github.io/optimization-2/**

# Gradient-Based Optimization

- At point (4,-3) f(xy) = -12
- Gradient tell us in which direction we need to move to maximixe function
- Let us take small step in the direction of gradient
- $x_{new} = x + \Delta x = x + h . \partial x$

  $x_{new} = x + \Delta x = 4 + (0.2).(-3)$

  $x_{new} = x + \Delta x = 4 - 0.6 = 3.4$




$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} \partial x \\ \partial y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$$

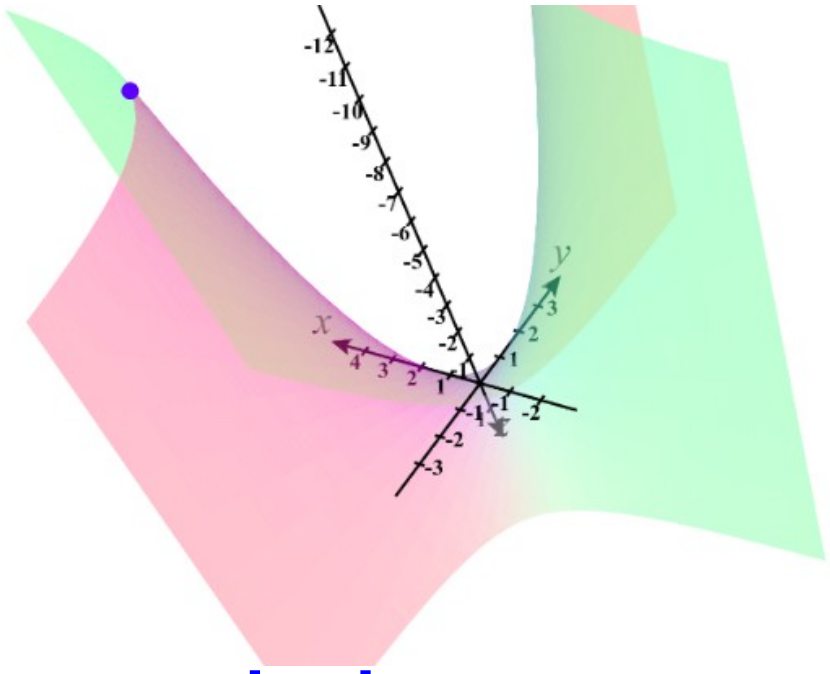
Refrence: https://cs231n.github.io/optimization-2/

# Gradient-Based Optimization

- At point (4,-3) f(xy) = -12
- Gradient tell us in which direction we need to move to maximixe function
- Let us take small step in the direction of gradient
- $y_{new} = y + \Delta y = y + h . \partial y$

  $y_{new} = y + \Delta y = -3 + (0.2).4$

  $y_{new} = y + \Delta y = -3 + 0.8 = -2.2$




$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} \partial x \\ \partial y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$$

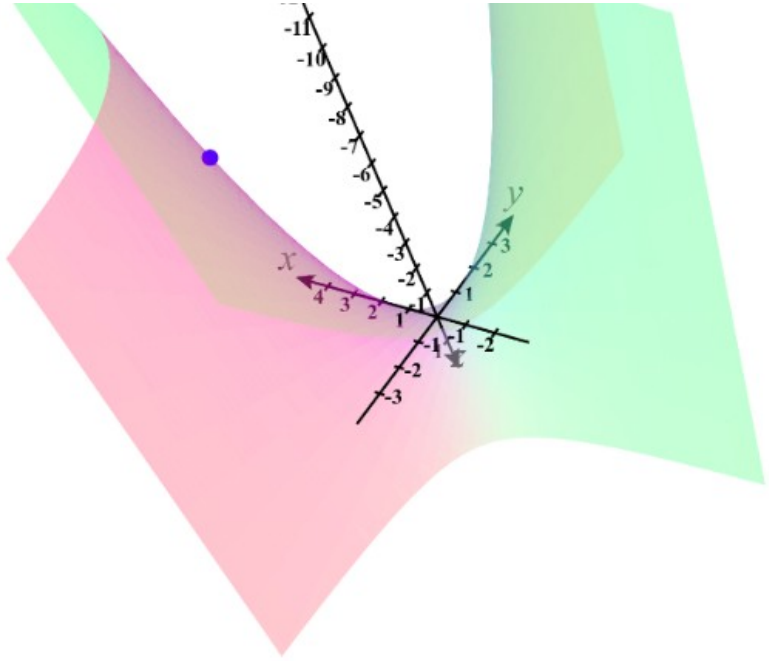
**Refrence: https://cs231n.github.io/optimization-2/**

# Gradient-Based Optimization

- At point (3.4,-2.2) f(xy) = -7.48

- $$x_{new} = x + \Delta x = 3.4 + (0.2).(-2.2)$$
  $$x_{new} = x + \Delta x = 4 - 0.44 = 2.96$$

  $$y_{new} = y + \Delta y = -2.2 + (0.2).(3.4)$$
  $$y_{new} = y + \Delta y = -2.2 + 0.68 = -1.52$$




$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} \partial x \\ \partial y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} -2.2 \\ 3.4 \end{bmatrix}$$

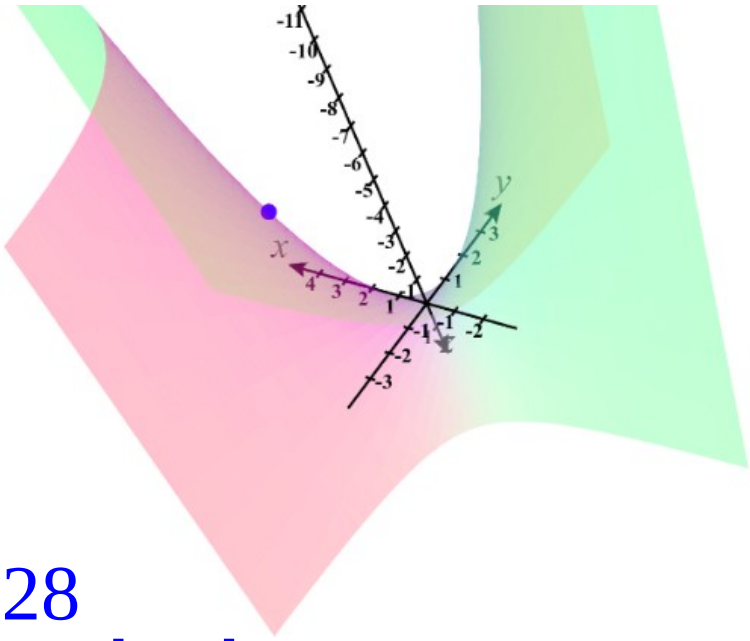
**Refrence: https://cs231n.github.io/optimization-2/**

# Gradient-Based Optimization

- At point (2.96,-1.52) f(xy) = -4.49

- $$x_{new} = x + \Delta x = 2.96 + (0.2).(-1.52)$$
  $$x_{new} = x + \Delta x = 2.96 - 0.304 = 2.656$$

$$y_{new} = y + \Delta y = -1.52 + (0.2).(2.96)$$
$$y_{new} = y + \Delta y = -1.52 + 0.592 = -0.928$$




$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} \partial x \\ \partial y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} -1.52 \\ 2.96 \end{bmatrix}$$

**Refrence: https://cs231n.github.io/optimization-2/**

# Gradient-Descent

- Consider Linear regression Cost function

$$MSE = J(\theta) = J(w, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad y = wx + b$$

- Here we have two parameters w and b

$$\Delta J = \frac{\partial J}{\partial w}\Delta w + \frac{\partial J}{\partial b}\Delta b$$

- The gradient of J is the vector of partial derivatives

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial w} \\ \frac{\partial J}{\partial b} \end{bmatrix}$$

**Reference: Michael A. Nielsen, "Neural Networks and Deep Learning" Determination Press 2015**

# Gradient-Descent

- Consider Linear regression Cost function

$$MSE = J(\theta) = J(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad y = wx + b$$

- Here we have two parameters w and b

$$\Delta J = \frac{\partial J}{\partial w}\Delta w + \frac{\partial J}{\partial b}\Delta b \qquad J(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))^2$$

- The gradient of J is the vector of partial derivatives

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial w} \\ \frac{\partial J}{\partial b} \end{bmatrix} \qquad \nabla J = \begin{bmatrix} \frac{\partial J}{\partial w} \\ \frac{\partial J}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{-2}{n}\sum_{i=1}^{n} x_i(y_i - (wx_i + b)) \\ \frac{-2}{n}\sum_{i=1}^{n}(y_i - (wx_i + b)) \end{bmatrix}$$

Reference: Michael A. Nielsen, "Neural Networks and Deep Learning" Determination Press 2015

# Gradient-Descent

- ΔJ can be rewritten as $\Delta J = \nabla J . \Delta \theta$ *here* $\Delta \theta = \begin{bmatrix} \Delta w \\ \Delta b \end{bmatrix}$

$$\Delta J = \frac{\partial J}{\partial w} \Delta w + \frac{\partial J}{\partial b} \Delta b \qquad \theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

- ∇J relates changes in θ to changes in J
- lets us see how to choose Δθ so as to make ΔJ negative

$$J = J + \Delta J$$

- suppose we choose $\Delta \theta = -\alpha \nabla J$

$$\Delta J = -\alpha \nabla J . \nabla J . = -\alpha \|\nabla J\|^2$$

**Reference: Michael A. Nielsen, "Neural Networks and Deep Learning" Determination Press 2015**

# Gradient descent

- Becaus $\|\nabla J\|^2 \geq 0$ this guarantees that $\Delta J \leq 0$
- Means J will always decrease, never increase, if we change θ according to the $\Delta \theta = -\alpha \nabla J$

$$\Delta w = -\alpha \frac{\partial J}{\partial w} \qquad \Delta b = -\alpha \frac{\partial J}{\partial b}$$

$$w_{new} = w + \Delta w = w - \alpha \frac{\partial J}{\partial w} \qquad b_{new} = b + \Delta b = b - \alpha \frac{\partial J}{\partial b}$$

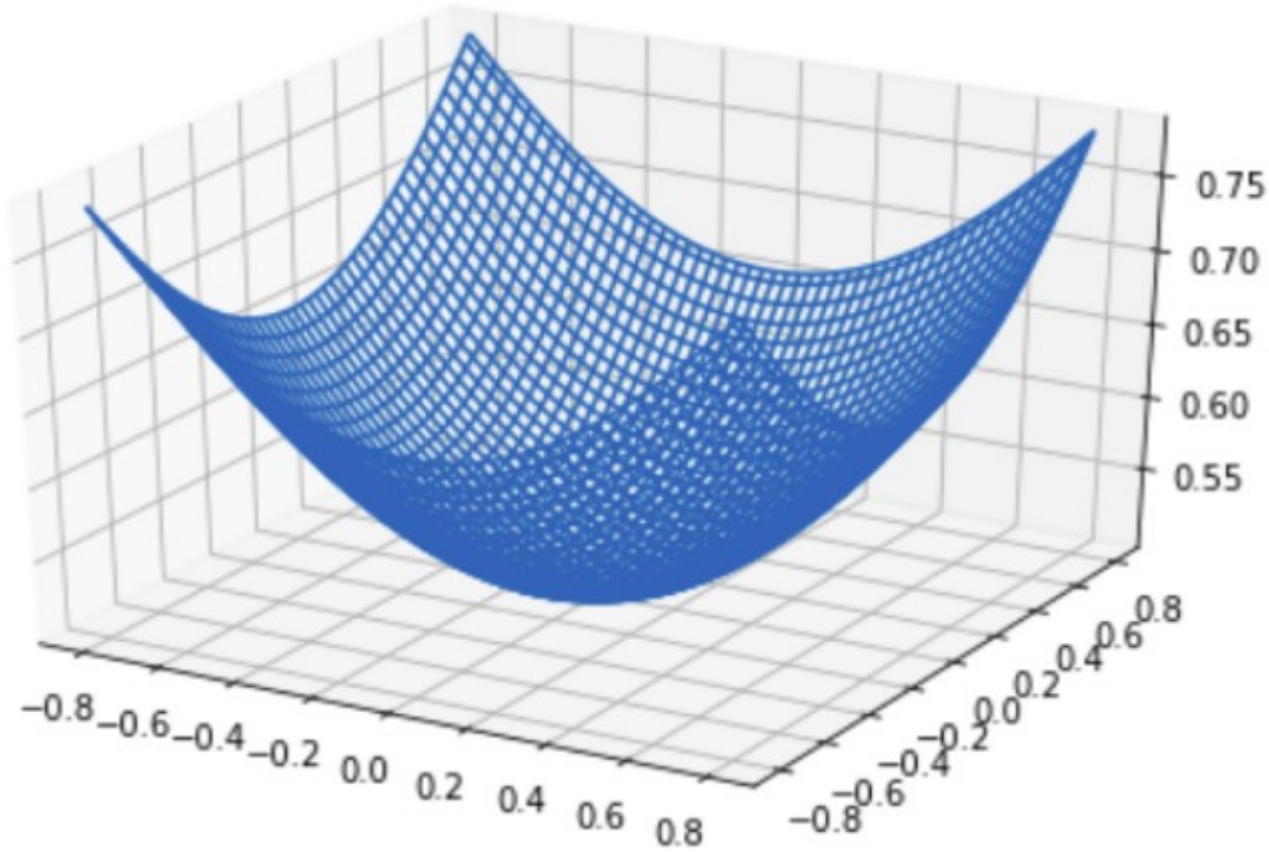
Reference: Michael A. Nielsen, “Neural Networks and Deep Learning” Determination Press 2015

# Gradient descent

$$w_{new} = w - \alpha \frac{\partial J}{\partial w}$$

$$b_{new} = b - \alpha \frac{\partial J}{\partial b}$$

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial w} \\ \frac{\partial J}{\partial b} \end{bmatrix}$$




**Reference: Michael A. Nielsen, "Neural Networks and Deep Learning" Determination Press 2015**

# Back-Propagation

- Computing an analytical expression for the gradient is straight forward, but numerically evaluating such an expression can be computationally expensive
- Theback propagation algorithm does so using a simple and inexpensive procedure
- The term back-propagation is often misunderstood as meaning the whole learning algorithm for multi-layer neural networks
- Actually,back- propagation refers **only** to **the method for computing the gradient**

**Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.**

# Back-Propagation

- Gradient descent, is used to perform learning using this gradient.
- In learning algorithms, the gradient we most often require is the gradient of the cost function with respect to the parameters, $\nabla_\theta J(\theta)$

**Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.**

# Gradient-Based Optimization

- **The derivative on each variable tells you the sensitivity of the whole expression on its value.**
- For example, $if\ x=4, y=-3\ \ then\ f(x,y)=-12$
- and the derivative on x $\frac{\partial f}{\partial x}=y=-3$
- This tells us that if we were to increase the value of this variable by a tiny amount h, the effect on the whole expression would be to decrease it (due to the negative sign), and by three times that amount(-3h).

Refrence: https://cs231n.github.io/optimization-2/

# Gradient-Based Optimization

- Similarly, since $\frac{\partial f}{\partial y} = x = 4$, we expect that increasing the value of y by some very small amount h would also increase the output of the function (due to the positive sign), and by 4h.

$$\nabla f = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$$

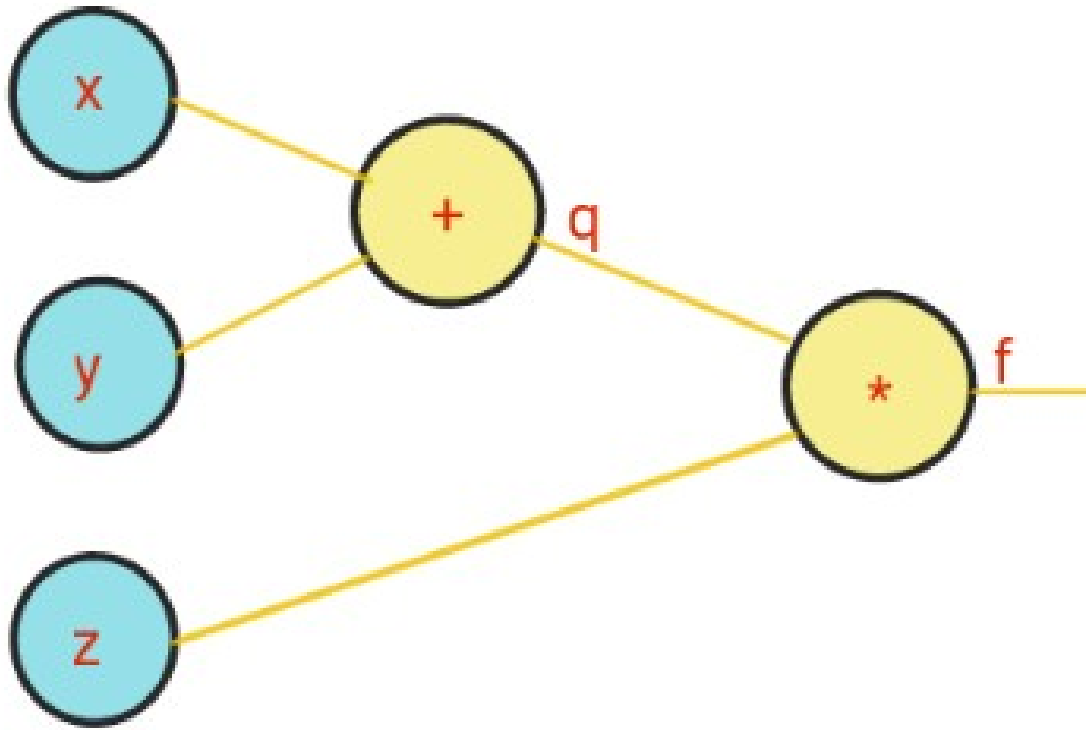
**Refrence: https://cs231n.github.io/optimization-2/**

# Back Propagation with chain rule

- consider more complicated expressions that involve multiple composed functions $f(x, y, z) = (x + y)z$
- this expression can be broken down into two expressions: $q = (x + y)$ and $f = qz$
- f is just multiplication of q and z, so $\frac{\partial f}{\partial q} = z$ , $\frac{\partial f}{\partial z} = q$
- q is addition of x and y so $\frac{\partial q}{\partial x} = 1$ , $\frac{\partial q}{\partial y} = 1$
- From chain rule $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$

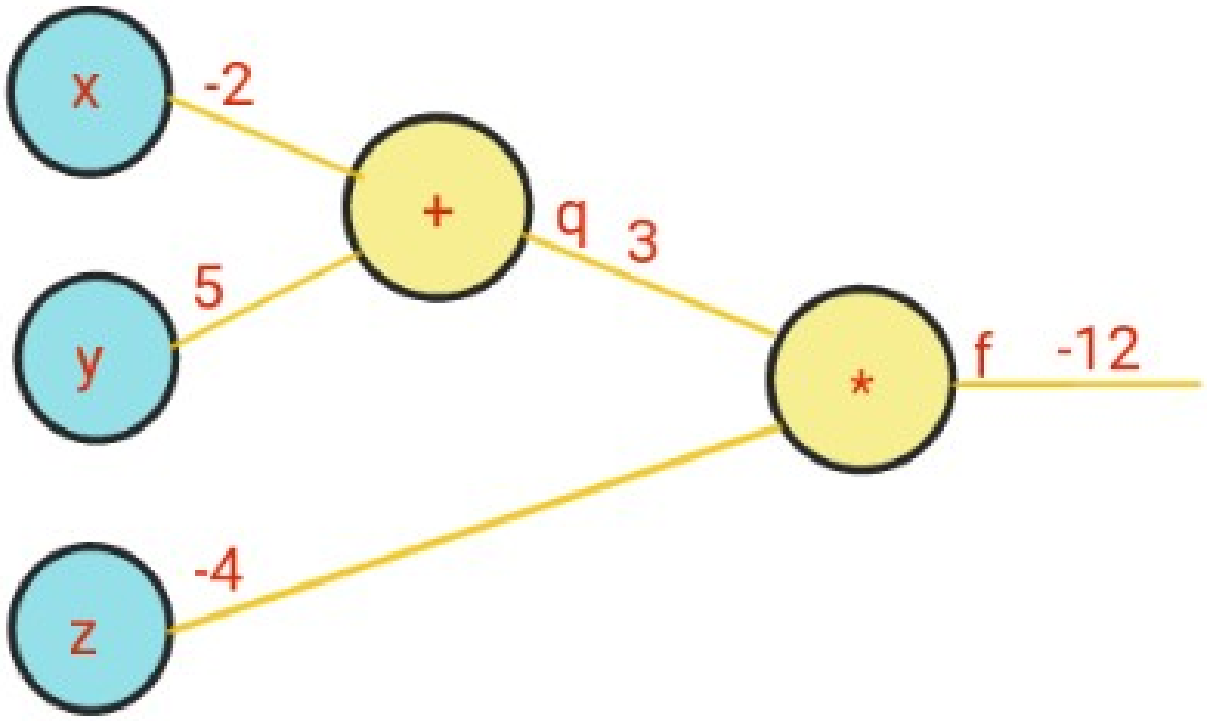
Refrence: https://cs231n.github.io/optimization-2/

# Computational Graph



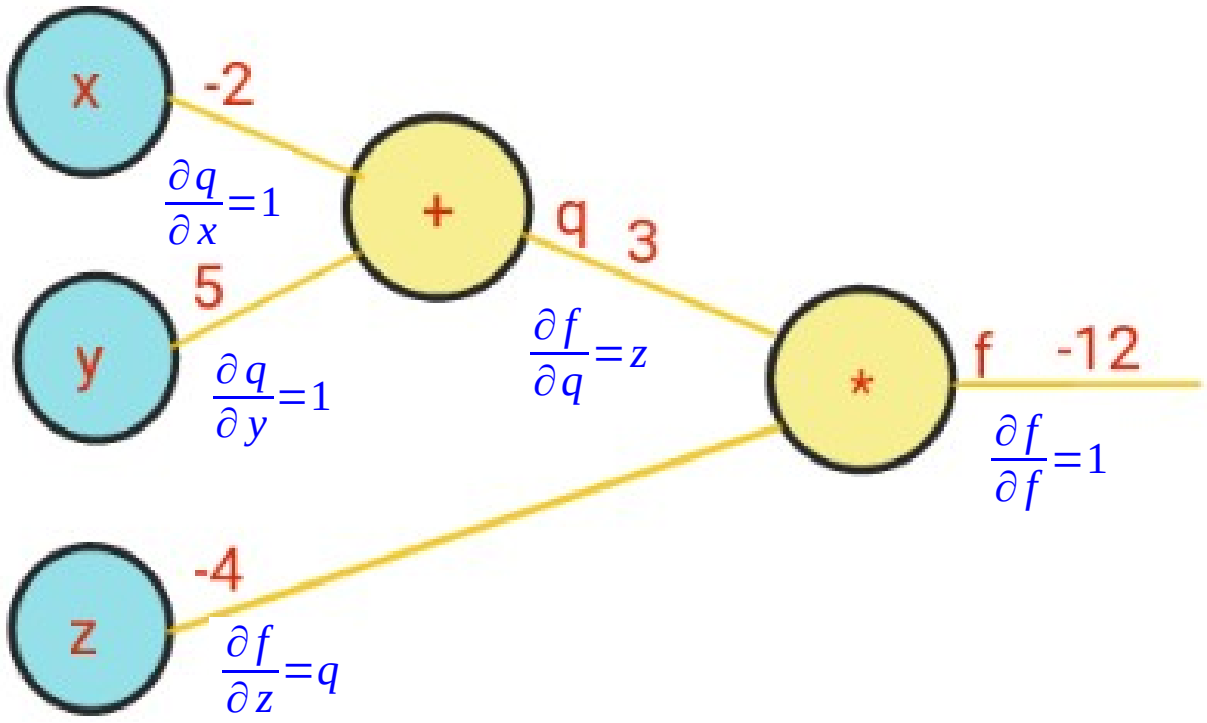

Refrence: https://cs231n.github.io/optimization-2/

# Forward Propagation



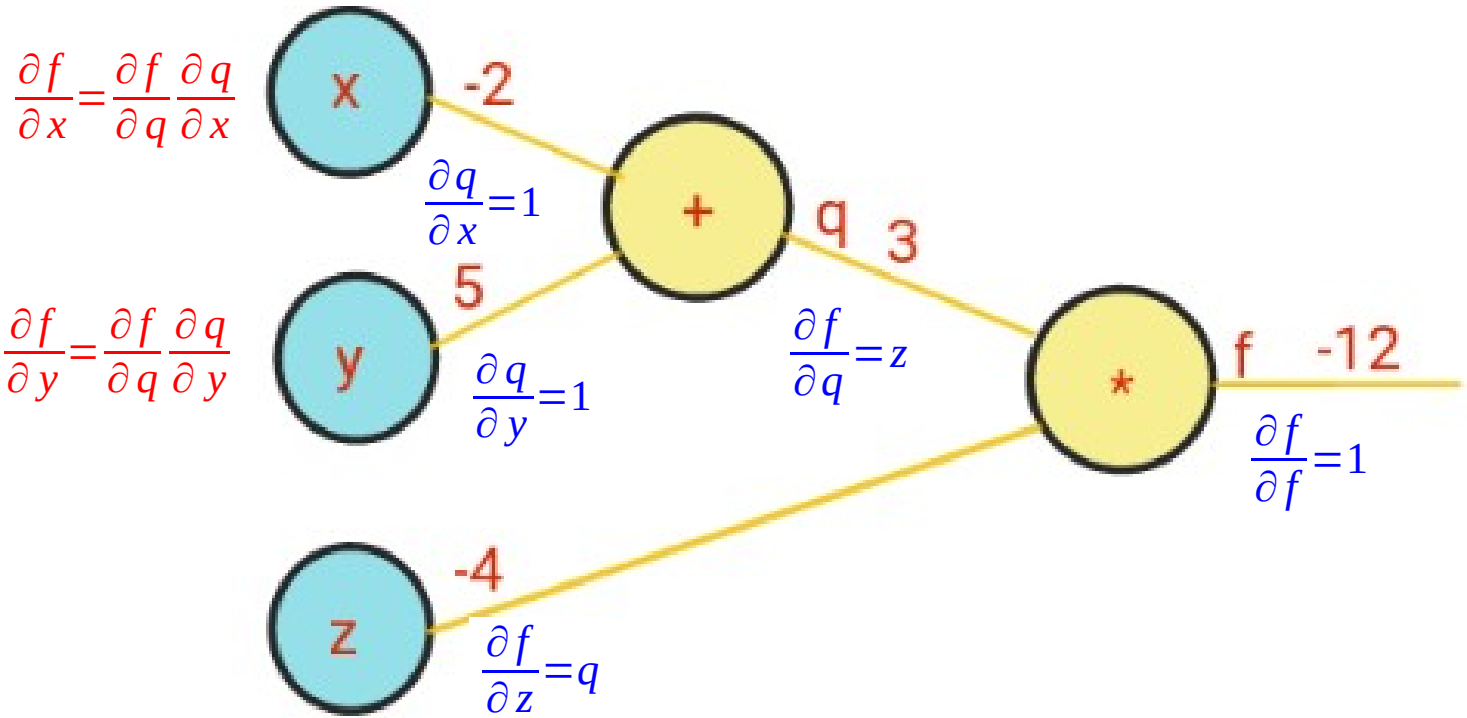

Refrence: https://cs231n.github.io/optimization-2/

# Back Propagation



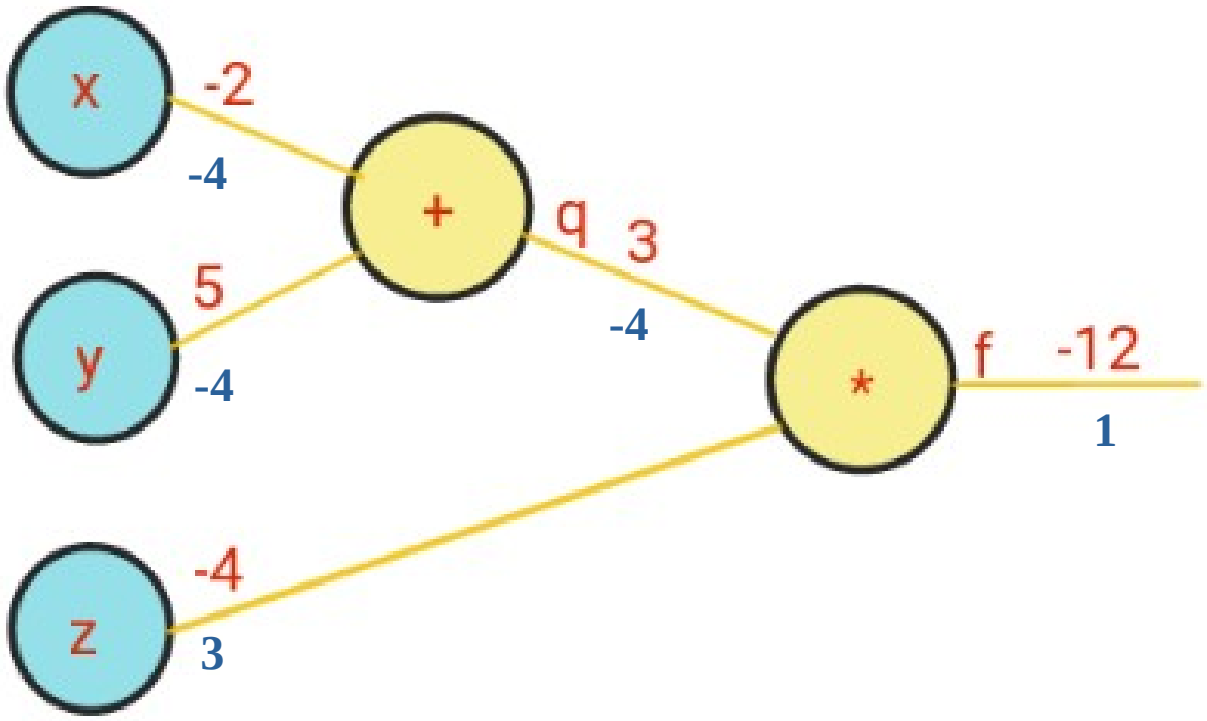

Refrence: https://cs231n.github.io/optimization-2/

# Back Propagation



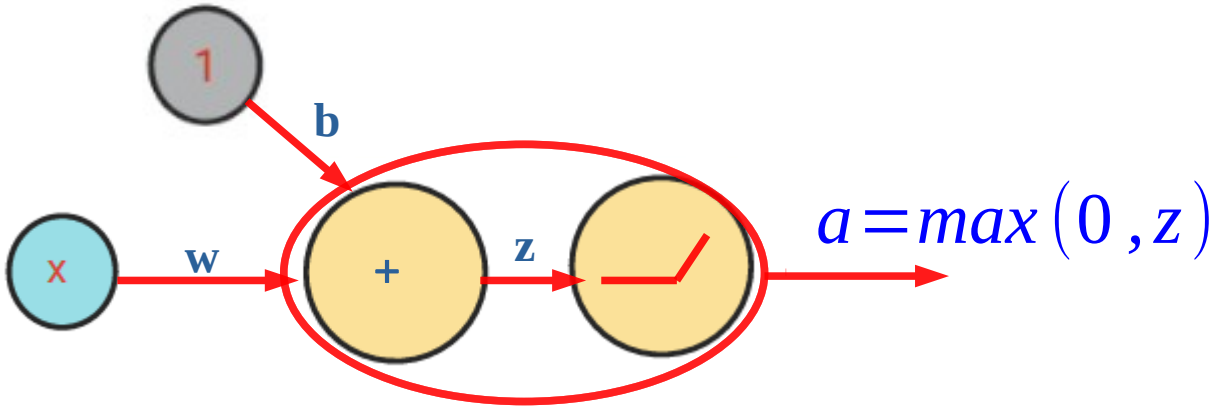

Refrence: https://cs231n.github.io/optimization-2/

# Back Propagation




Refrence: https://cs231n.github.io/optimization-2/

# Back Propagation (Linear Regression)




$\hat{y} = a$

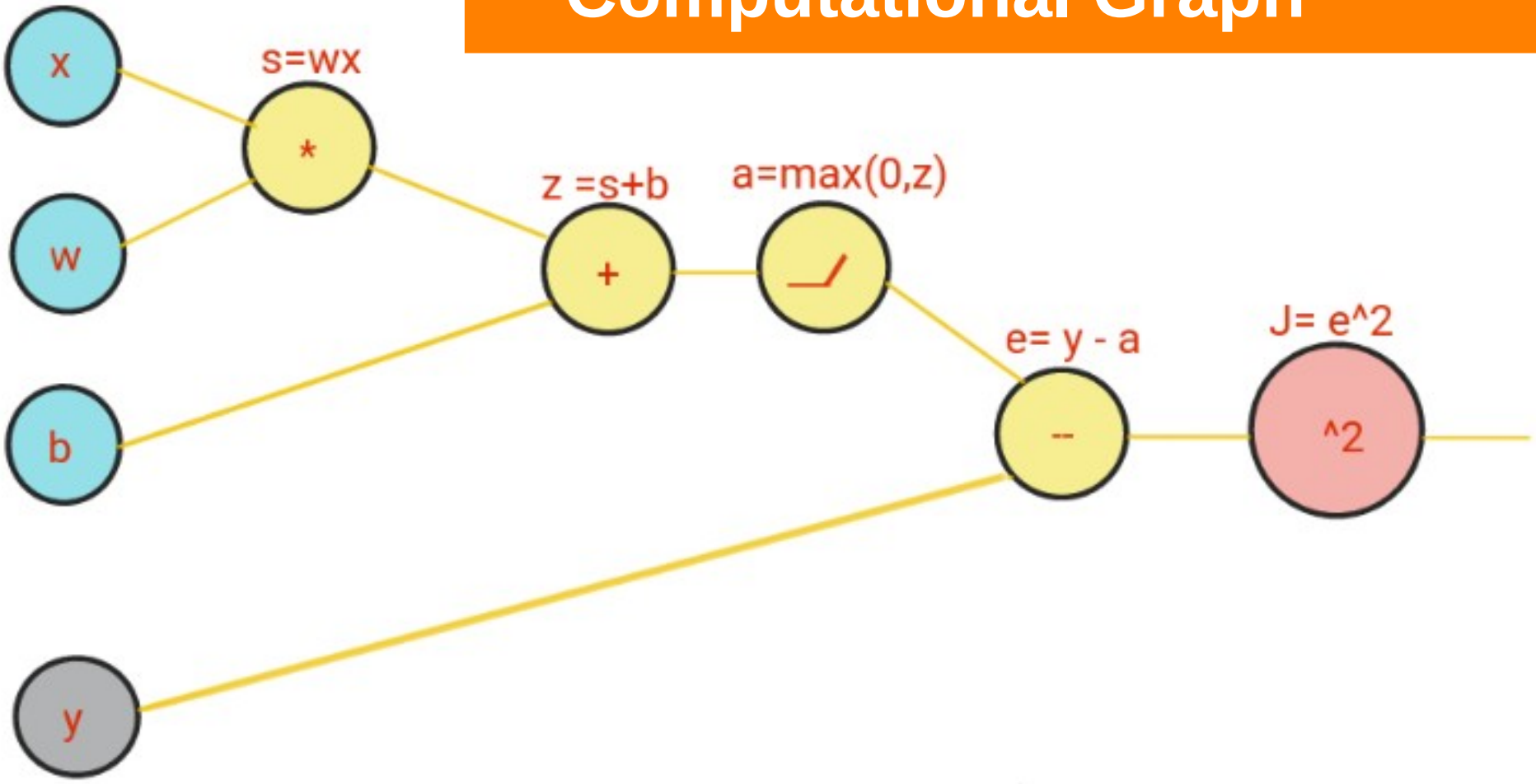
# Back Propagation (Linear Regression)

$$J = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}\right)^2$$

$$\hat{y} = max\left(0, wx + b\right)$$

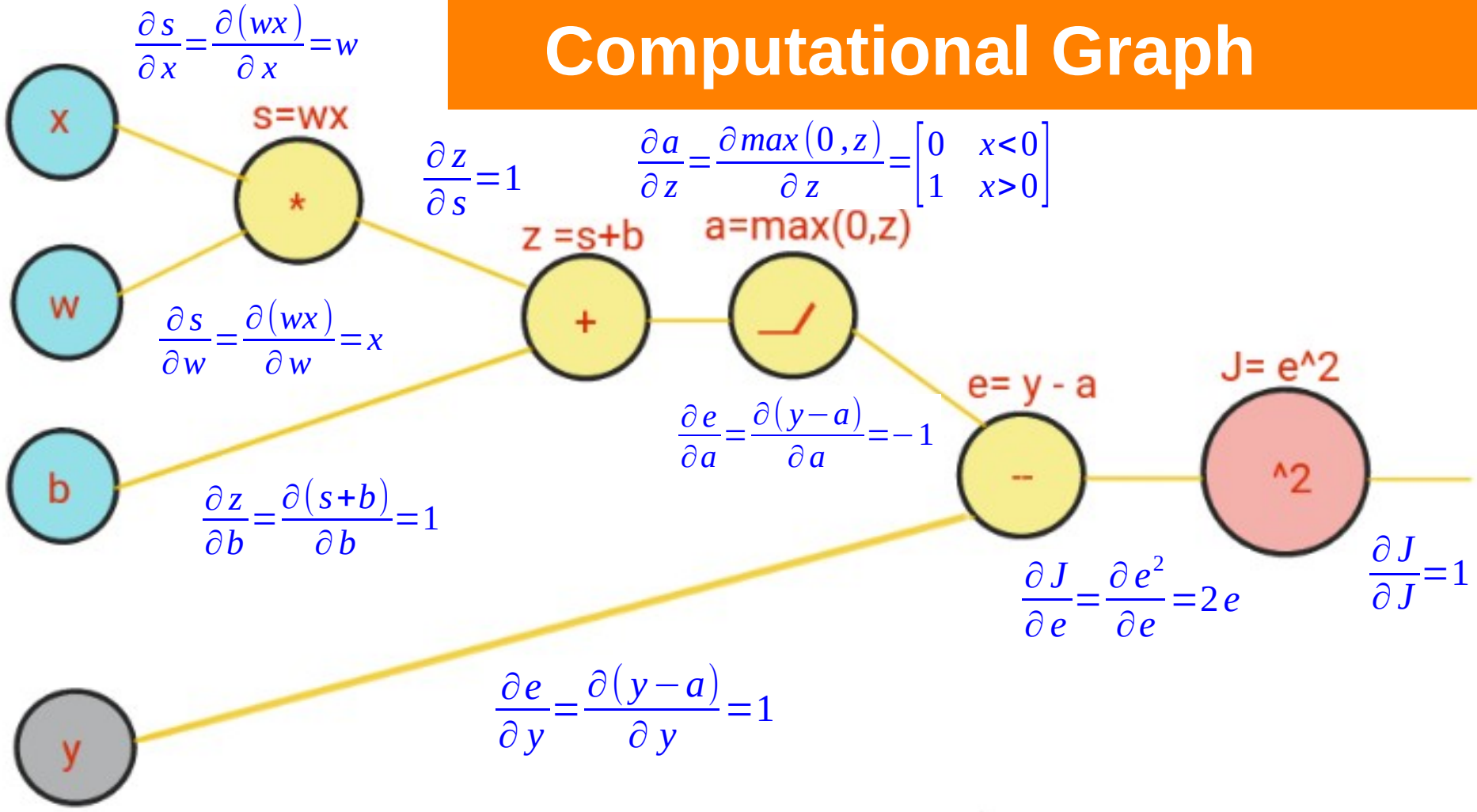
$$J\left(w, b\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - max\left(0, wx_i + b\right)\right)^2$$

# Computational Graph



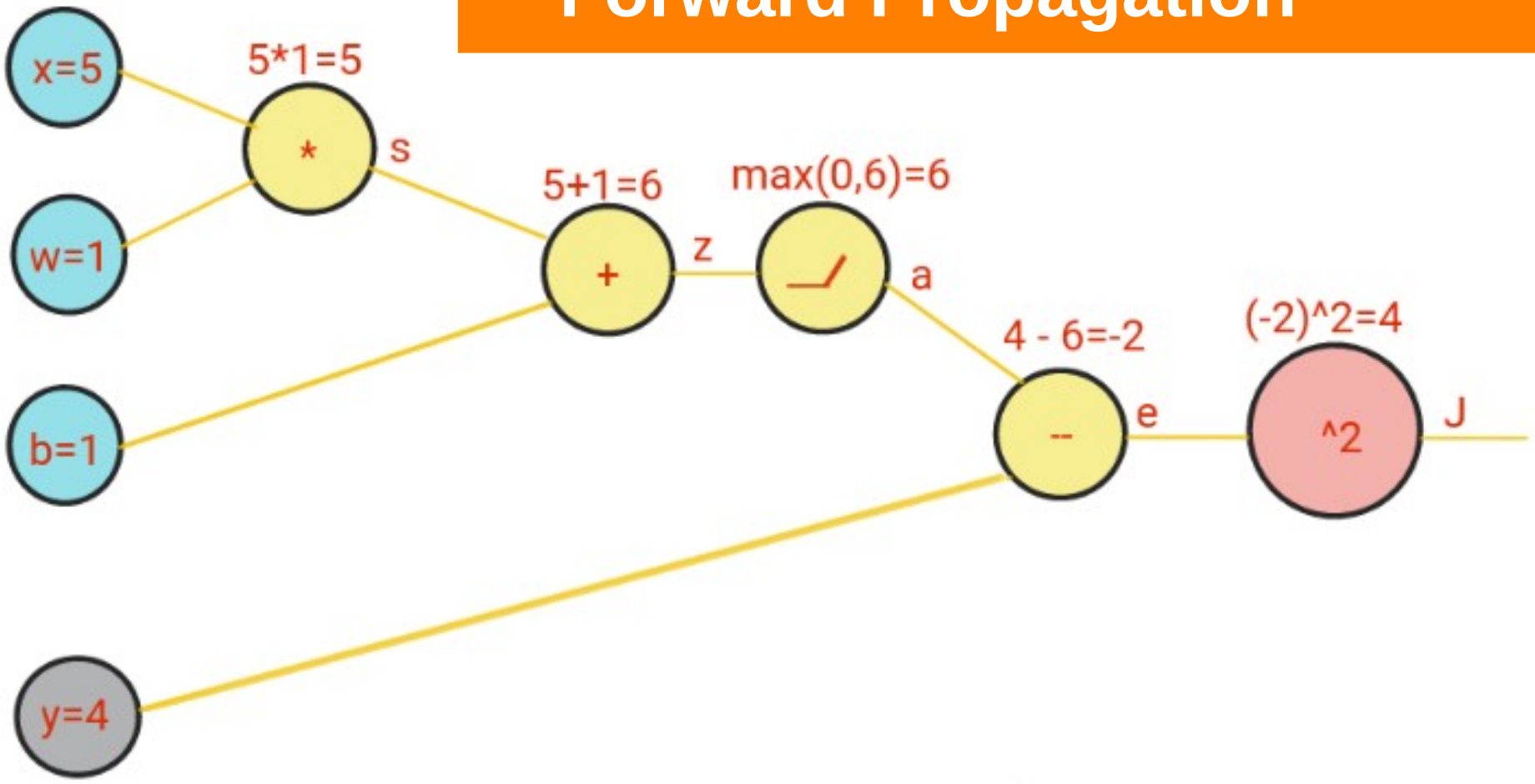

Refrence: https://cs231n.github.io/optimization-2/

# Computational Graph




Refrence: https://cs231n.github.io/optimization-2/

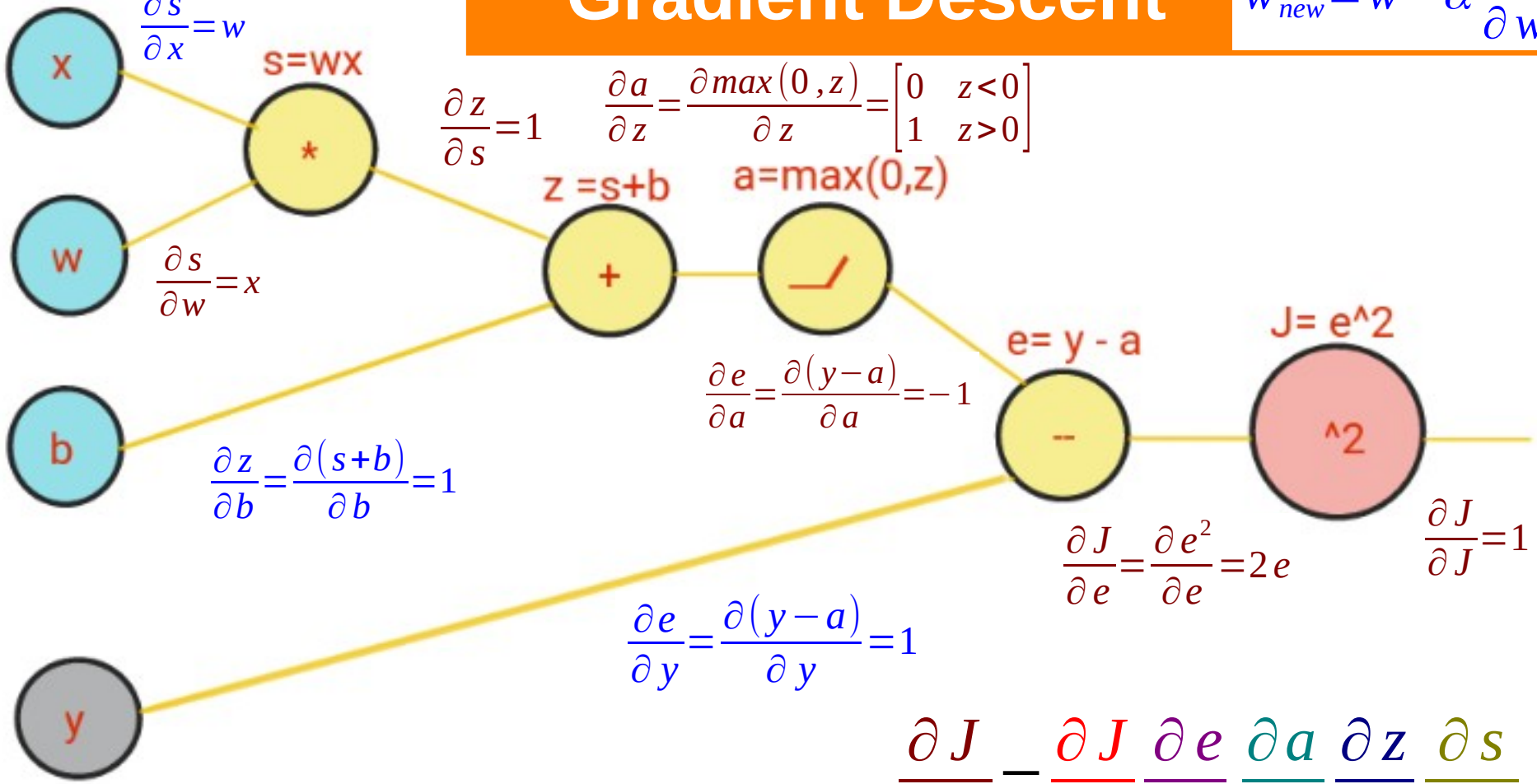
# Forward Propagation




Refrence: https://cs231n.github.io/optimization-2/

# Gradient Descent

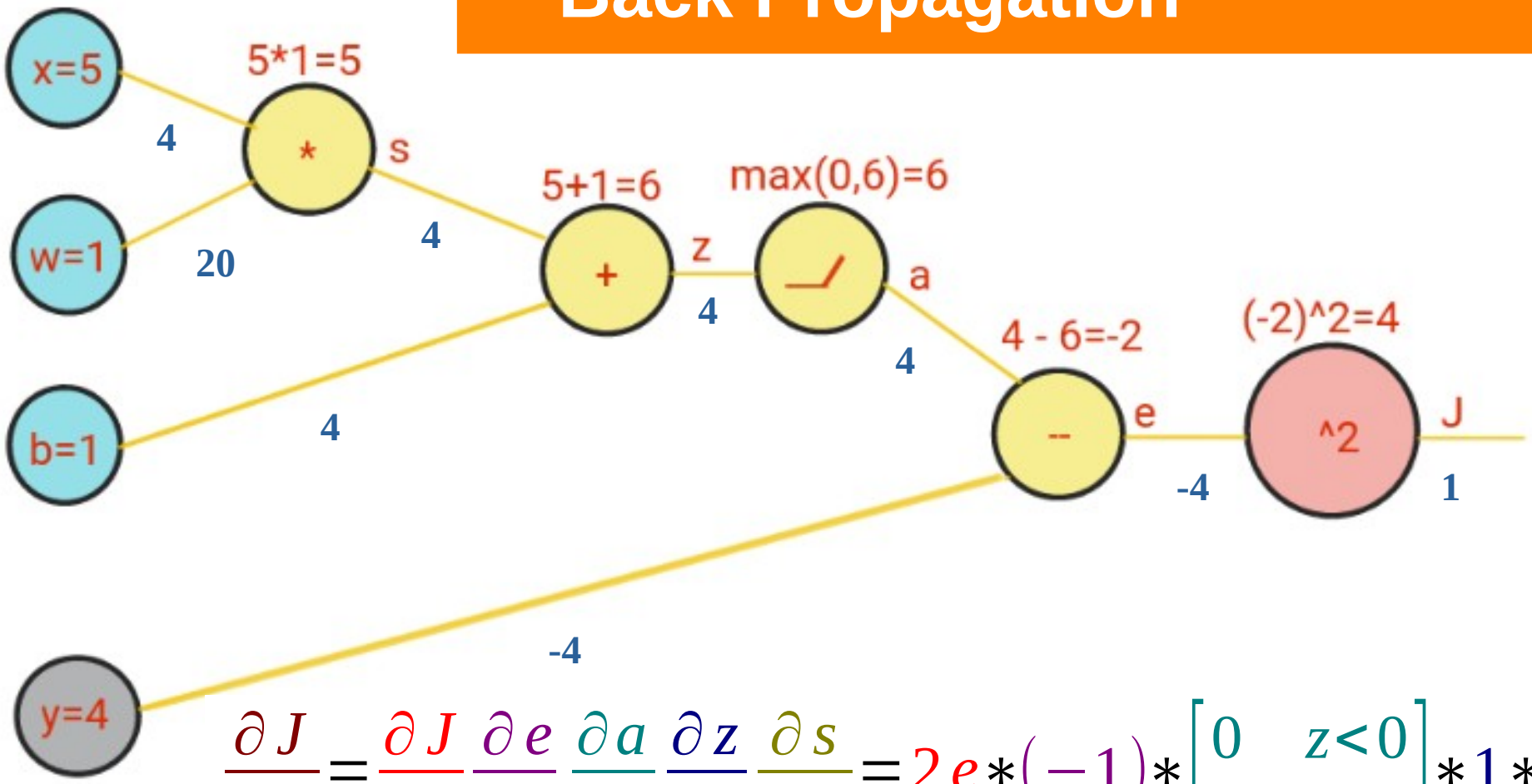
$$w_{new} = w - \alpha \frac{\partial J}{\partial w}$$




$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial w}$$

Refrence: https://cs231n.github.io/optimization-2/

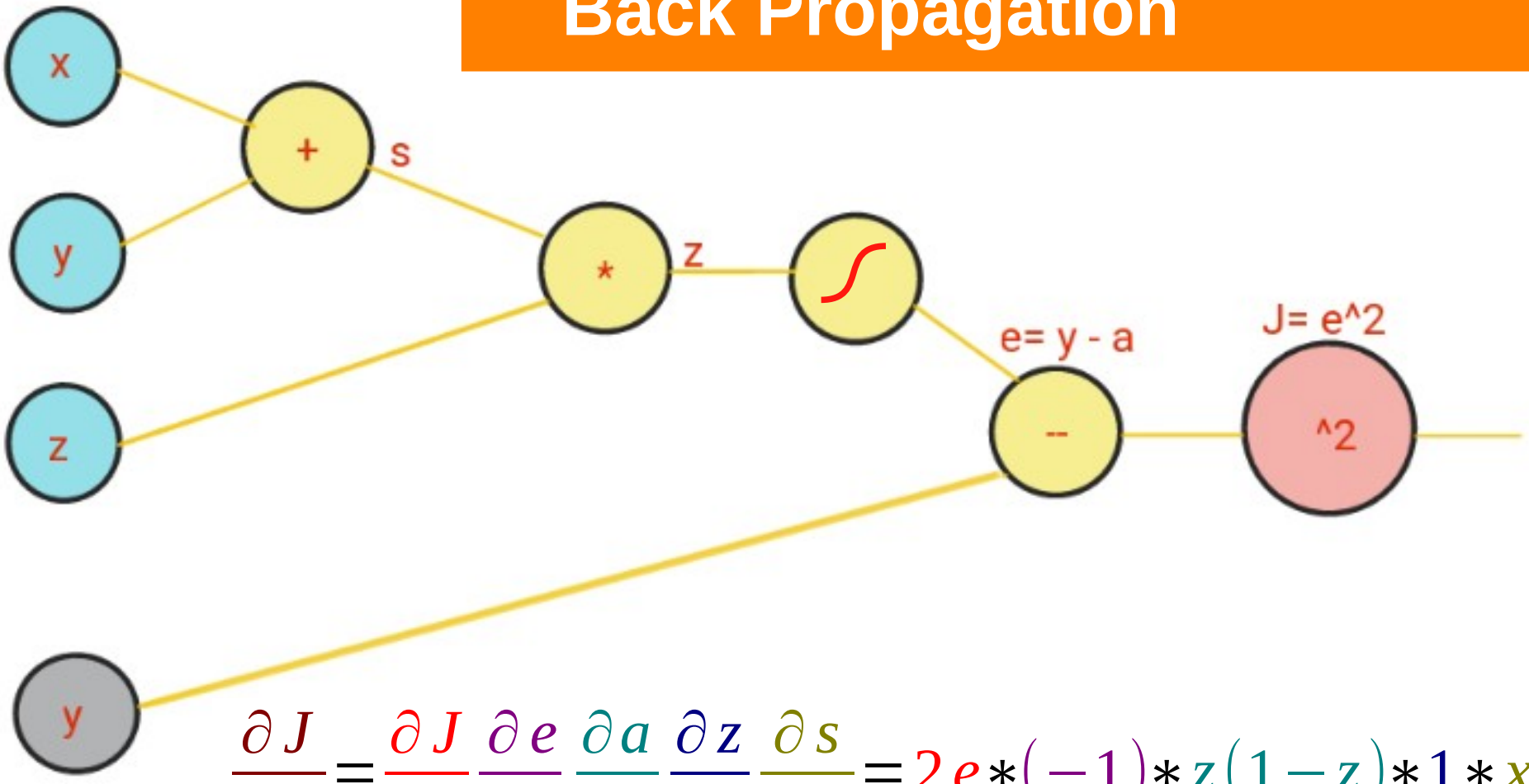
# Back Propagation




$$\frac{\partial J}{\partial w}=\frac{\partial J}{\partial e}\frac{\partial e}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial w}=2e*(-1)*\begin{bmatrix}0 & z<0\\ 1 & z>0\end{bmatrix}*1*x$$

Refrence: https://cs231n.github.io/optimization-2/

# Back Propagation




$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial w} = 2e*(-1)*z(1-z)*1*x$$

Refrence: https://cs231n.github.io/optimization-2/

# Gradient descent

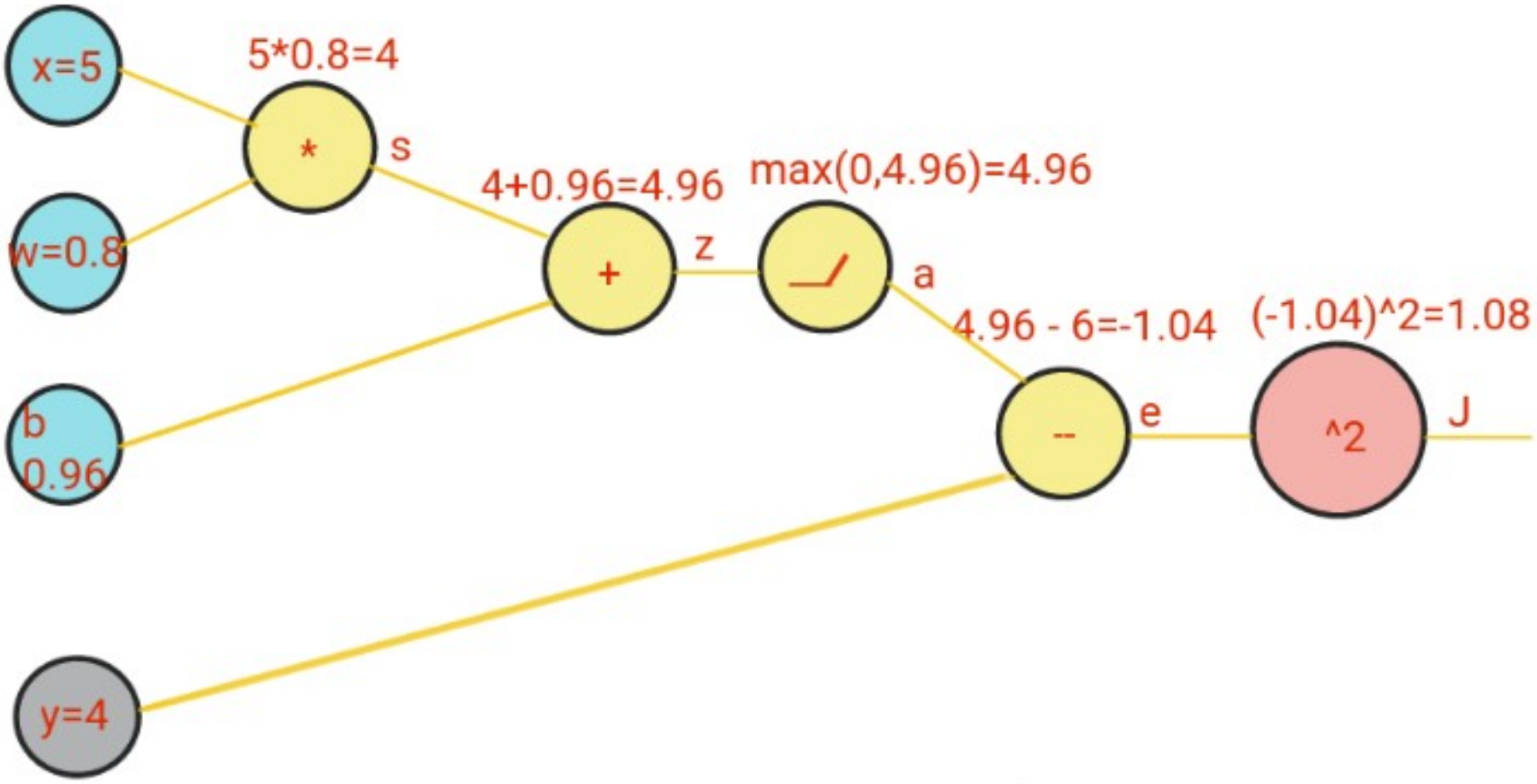
$$w_{new} = w - \alpha \frac{\partial J}{\partial w}$$

$$b_{new} = b - \alpha \frac{\partial J}{\partial b}$$

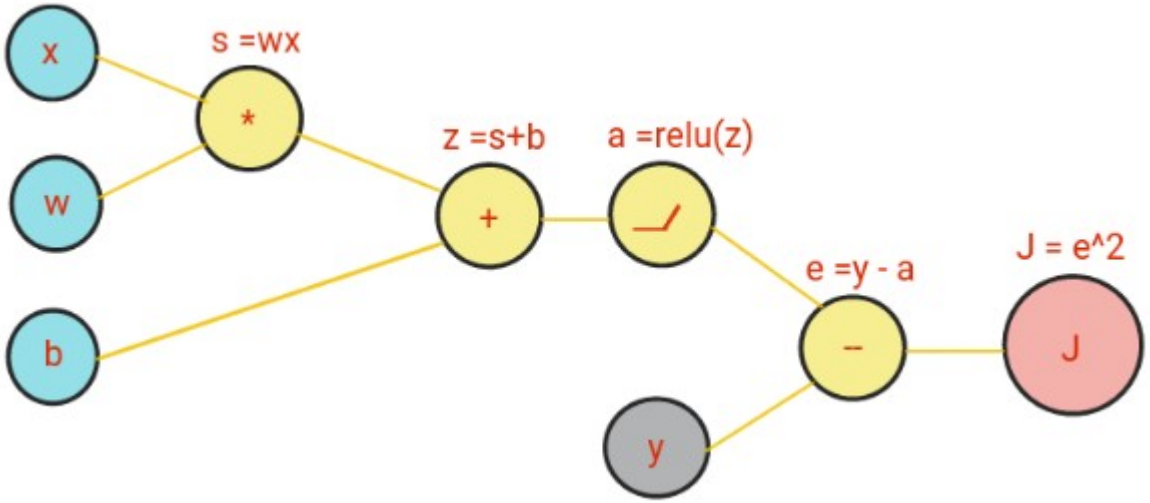
$$w_{new} = 1 - (0.01)20 = 0.8$$

$$b_{new} = 1 - (0.01)\,4 = 0.96$$

# Gradient descent



# ons with chain rule

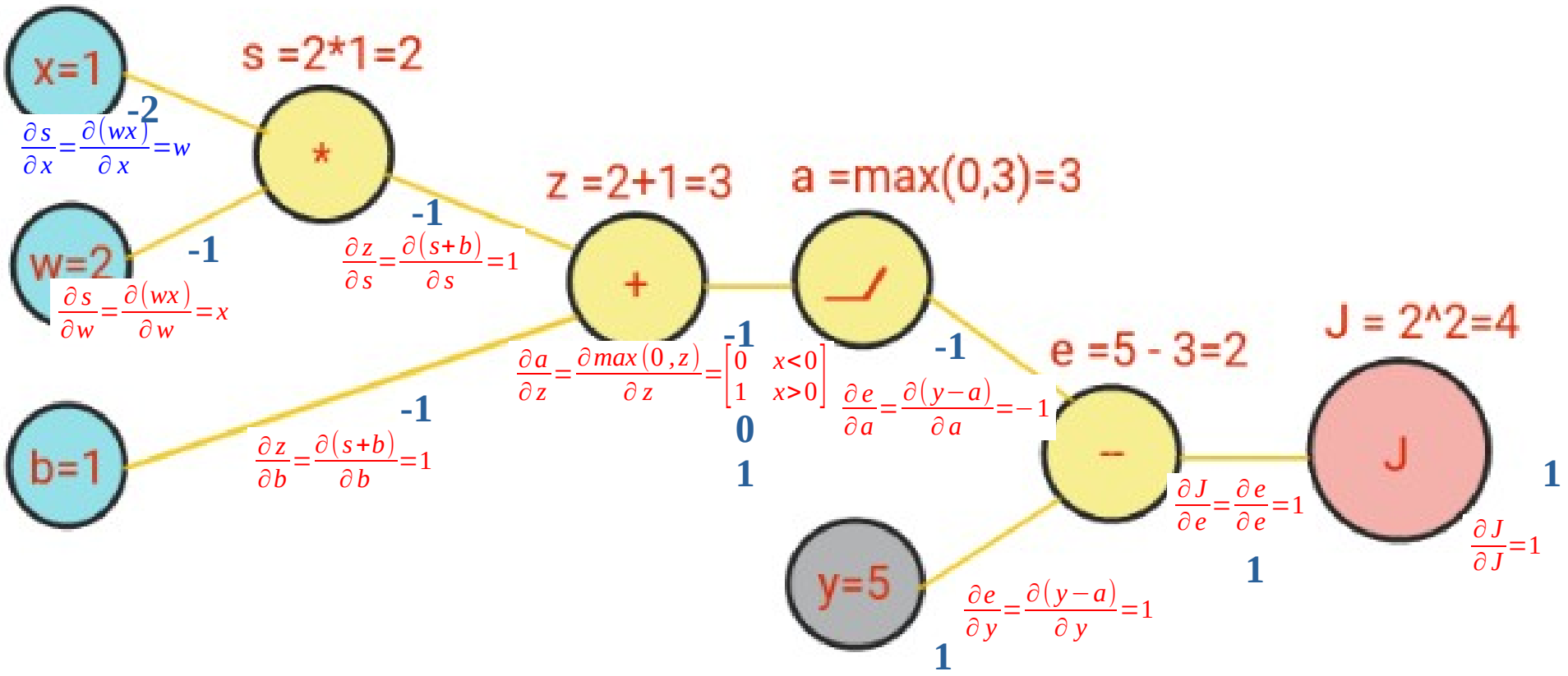


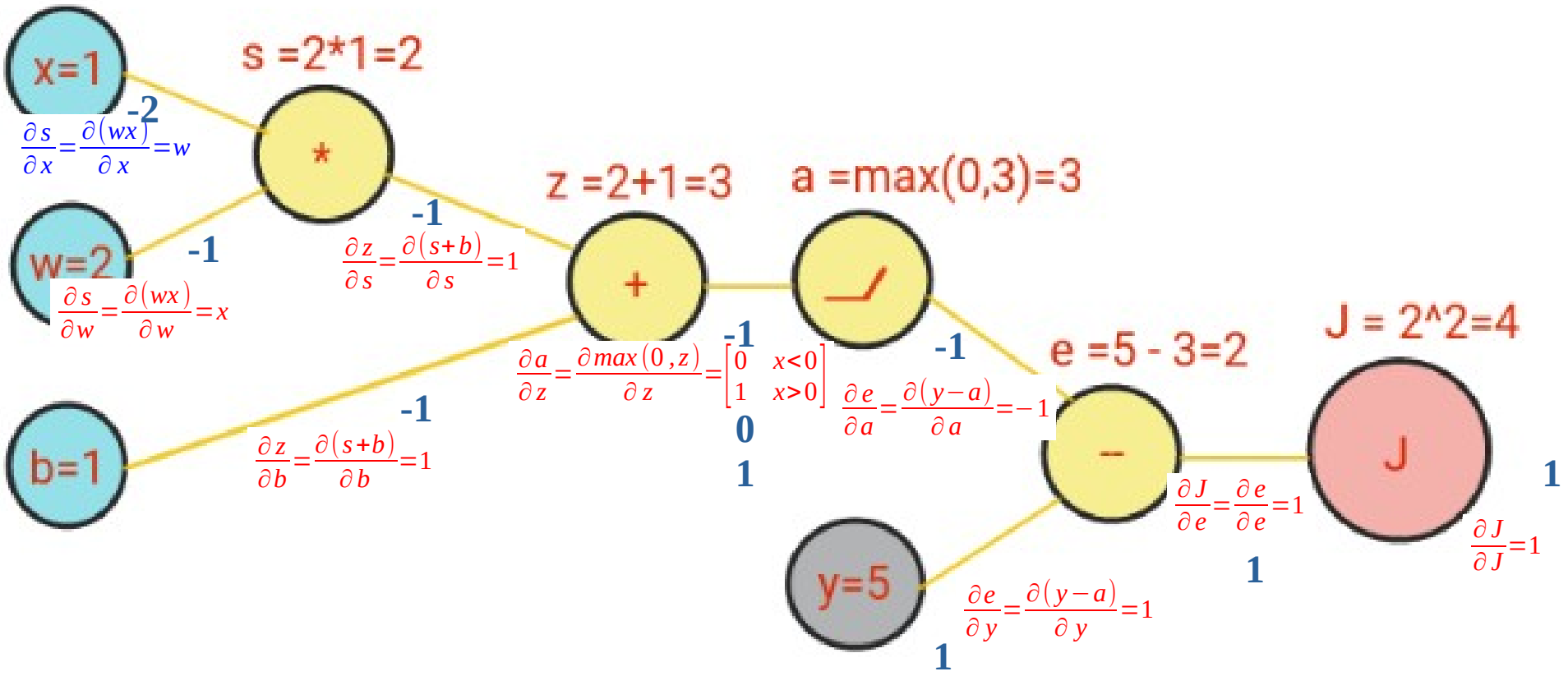

$$\frac{\partial J}{\partial w}=\frac{\partial J}{\partial e}\frac{\partial e}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial w}=2*(-1)*\begin{bmatrix}0 & x<0\\ 1 & x>0\end{bmatrix}*1*x$$

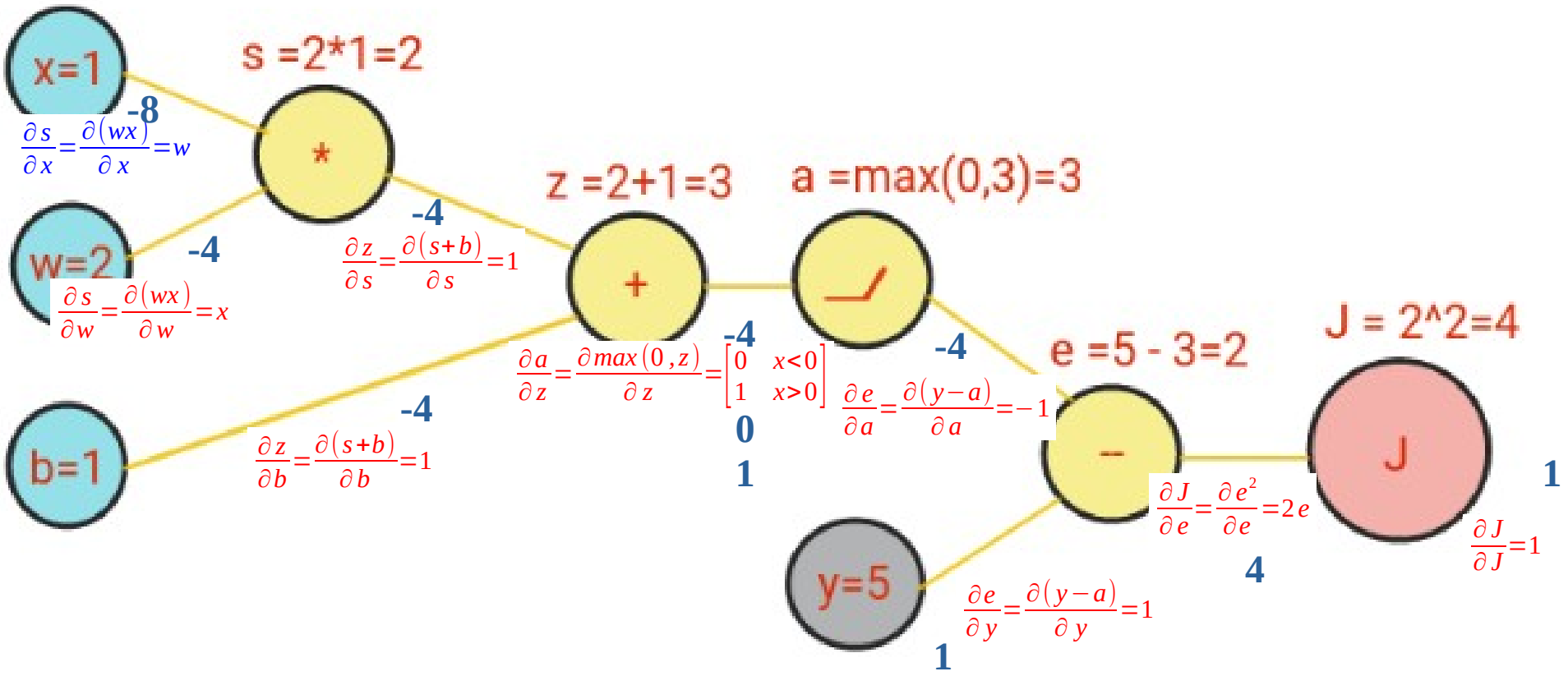
| | |
|---|---|
| $\frac{\partial s}{\partial w}=\frac{\partial (wx)}{\partial w}=x$ | $\frac{\partial s}{\partial x}=\frac{\partial (wx)}{\partial x}=w$ |
| $\frac{\partial z}{\partial s}=\frac{\partial (s+b)}{\partial s}=1$ | $\frac{\delta w_6}{\delta w_3}=\frac{\delta w_5+w_3}{\delta w_3}=1$ |
| $\frac{\partial a}{\partial z}=\frac{\partial max(0,z)}{\partial z}=\begin{bmatrix}0 & x<0\\ 1 & x>0\end{bmatrix}$ | $\frac{\partial e}{\partial a}=\frac{\partial (y-a)}{\partial a}=-1$ |
| $\frac{\delta w_8}{\delta w_4}=\frac{\delta w_4-w_7}{\delta w_4}=1$ | $\frac{\partial J}{\partial e}=\frac{\partial e^2}{\partial e}=2$ |

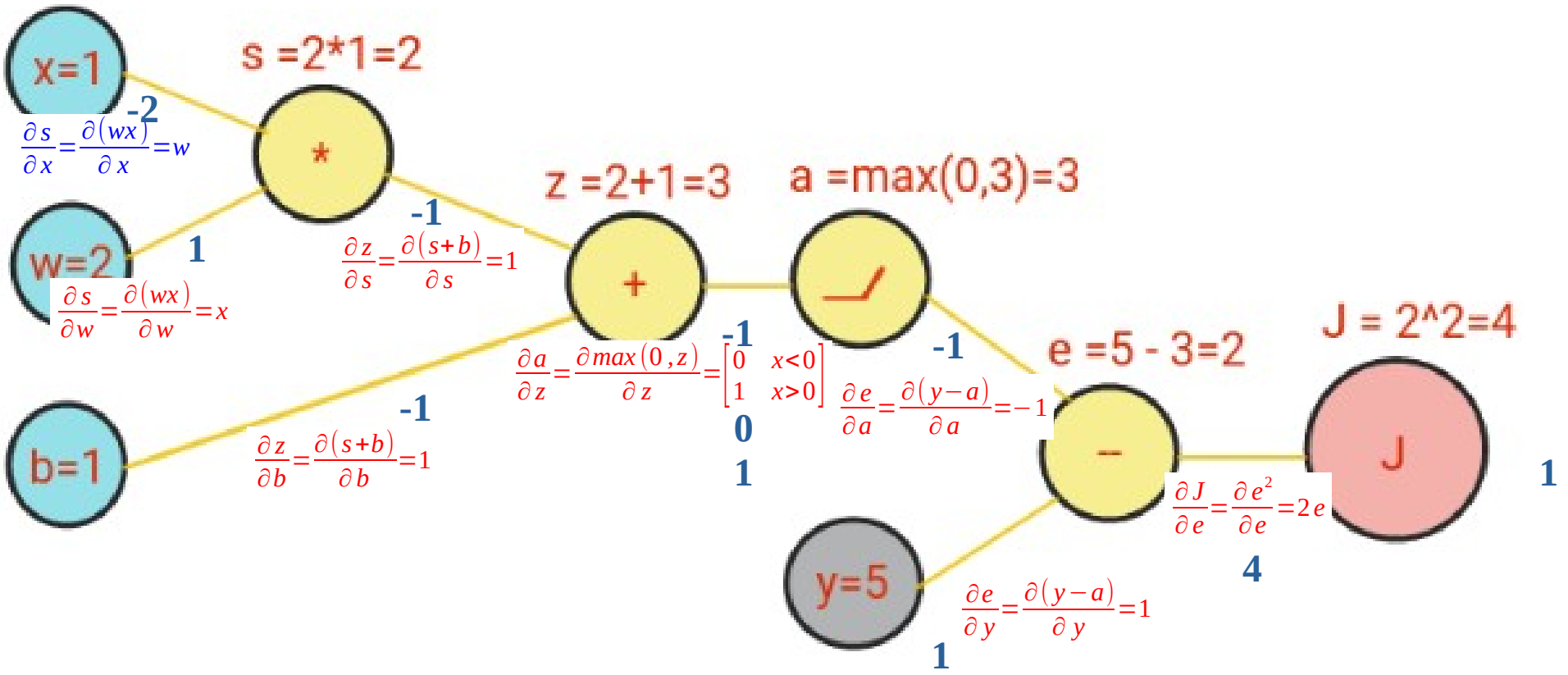
$$\frac{\partial J}{\partial w}=\frac{\partial J}{\partial e}\frac{\partial e}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial w}$$

ference: https://cs231n.github.io/optimization-2/

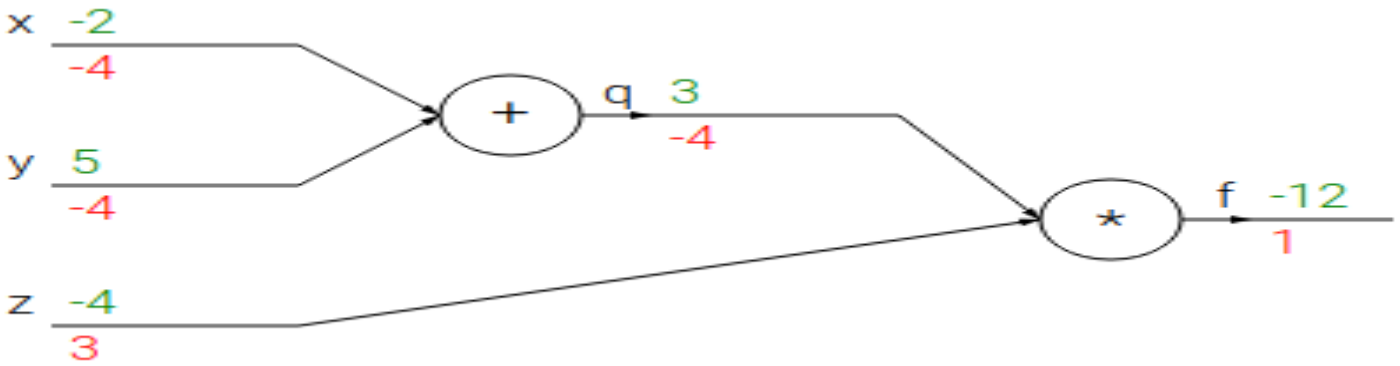

x=1
w=2
b=1
s =2*1=2
*
z =2+1=3
+
a =max(0,3)=3
e =5 - 3=2
−
J = 2^2=4
J
y=5
-2
-1
-1
-1
-1
0
1
1
1
1
$\frac{\partial s}{\partial x}=\frac{\partial(wx)}{\partial x}=w$
$\frac{\partial s}{\partial w}=\frac{\partial(wx)}{\partial w}=x$
$\frac{\partial z}{\partial s}=\frac{\partial(s+b)}{\partial s}=1$
$\frac{\partial z}{\partial b}=\frac{\partial(s+b)}{\partial b}=1$
$\frac{\partial a}{\partial z}=\frac{\partial max(0,z)}{\partial z}=\begin{bmatrix}0 & x<0\\ 1 & x>0\end{bmatrix}$
$\frac{\partial e}{\partial a}=\frac{\partial(y-a)}{\partial a}=-1$
$\frac{\partial e}{\partial y}=\frac{\partial(y-a)}{\partial y}=1$
$\frac{\partial J}{\partial e}=\frac{\partial e}{\partial e}=1$
$\frac{\partial J}{\partial J}=1$


x=1
w=2
b=1
s =2*1=2
*
z =2+1=3
+
a =max(0,3)=3
e =5 - 3=2
−
J = 2^2=4
J
y=5
-2
-1
-1
-1
-1
0
1
1
1
1
$\frac{\partial s}{\partial x}=\frac{\partial(wx)}{\partial x}=w$
$\frac{\partial s}{\partial w}=\frac{\partial(wx)}{\partial w}=x$
$\frac{\partial z}{\partial s}=\frac{\partial(s+b)}{\partial s}=1$
$\frac{\partial z}{\partial b}=\frac{\partial(s+b)}{\partial b}=1$
$\frac{\partial a}{\partial z}=\frac{\partial max(0,z)}{\partial z}=\begin{bmatrix}0 & x<0\\ 1 & x>0\end{bmatrix}$
$\frac{\partial e}{\partial a}=\frac{\partial(y-a)}{\partial a}=-1$
$\frac{\partial e}{\partial y}=\frac{\partial(y-a)}{\partial y}=1$
$\frac{\partial J}{\partial e}=\frac{\partial e}{\partial e}=1$
$\frac{\partial J}{\partial J}=1$


x=1
w=2
b=1
s =2*1=2
*
z =2+1=3
+
a =max(0,3)=3
e =5 - 3=2
−
J = 2^2=4
J
y=5
-8
-4
-4
-4
-4
0
1
1
4
1
$\frac{\partial s}{\partial x}=\frac{\partial(wx)}{\partial x}=w$
$\frac{\partial s}{\partial w}=\frac{\partial(wx)}{\partial w}=x$
$\frac{\partial z}{\partial s}=\frac{\partial(s+b)}{\partial s}=1$
$\frac{\partial z}{\partial b}=\frac{\partial(s+b)}{\partial b}=1$
$\frac{\partial a}{\partial z}=\frac{\partial max(0,z)}{\partial z}=\begin{bmatrix}0 & x<0\\ 1 & x>0\end{bmatrix}$
$\frac{\partial e}{\partial a}=\frac{\partial(y-a)}{\partial a}=-1$
$\frac{\partial e}{\partial y}=\frac{\partial(y-a)}{\partial y}=1$
$\frac{\partial J}{\partial e}=\frac{\partial e^2}{\partial e}=2e$
$\frac{\partial J}{\partial J}=1$


x=1
w=2
b=1
y=5
s =2*1=2
z =2+1=3
a =max(0,3)=3
e =5 - 3=2
J = 2^2=4
*
+
J
$\frac{\partial s}{\partial x}=\frac{\partial(wx)}{\partial x}=w$
-2
$\frac{\partial s}{\partial w}=\frac{\partial(wx)}{\partial w}=x$
1
$\frac{\partial z}{\partial s}=\frac{\partial(s+b)}{\partial s}=1$
-1
$\frac{\partial z}{\partial b}=\frac{\partial(s+b)}{\partial b}=1$
-1
$\frac{\partial a}{\partial z}=\frac{\partial max(0,z)}{\partial z}=\begin{bmatrix}0 & x<0\\ 1 & x>0\end{bmatrix}$
0
1
-1
$\frac{\partial e}{\partial a}=\frac{\partial(y-a)}{\partial a}=-1$
$\frac{\partial e}{\partial y}=\frac{\partial(y-a)}{\partial y}=1$
1
$\frac{\partial J}{\partial e}=\frac{\partial e^2}{\partial e}=2e$
4
1

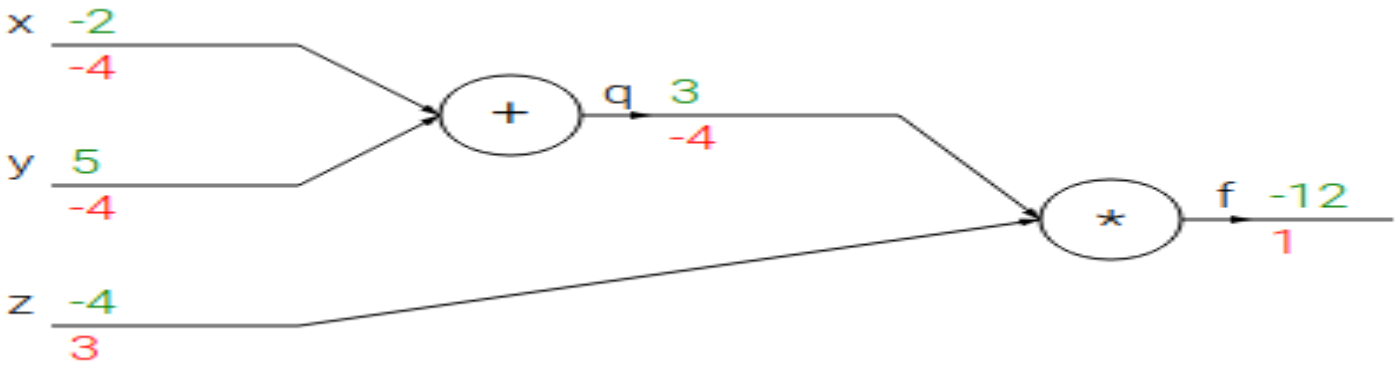
# Backprop-Compound expressions with chain rule




- The add operation received inputs [-2, 5] and computed output 3.
- Since the gate is computing the addition operation, its local gradient for both of its inputs is +1.
- The rest of the graph computed the final value, which is -12.

Refrence: https://cs231n.github.io/optimization-2/

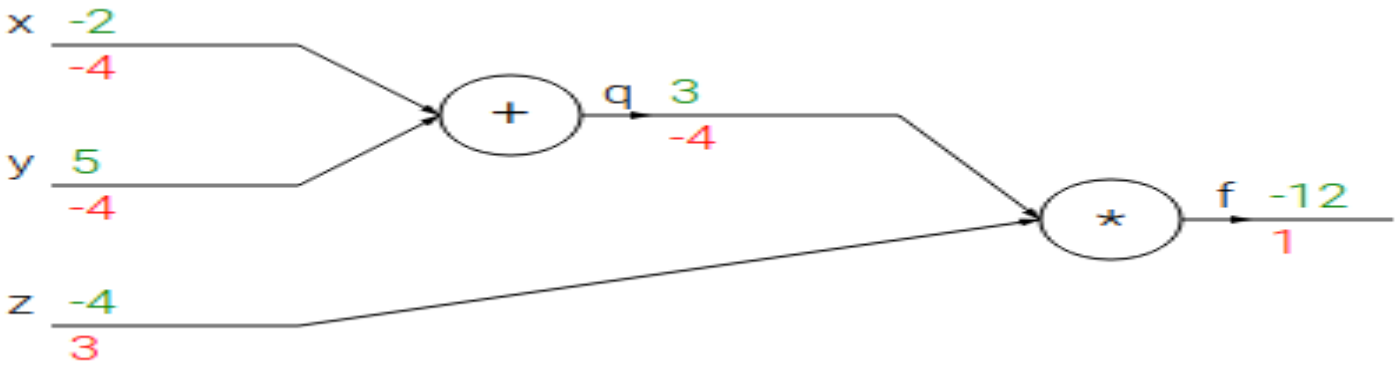
# Backprop-Compound expressions with chain rule




- During the backward pass in which the chain rule is applied recursively backwards through the computational graph
- The add operation (which is an input to the multiply operation) learns that the gradient for its output was -4.

Refrence: https://cs231n.github.io/optimization-2/

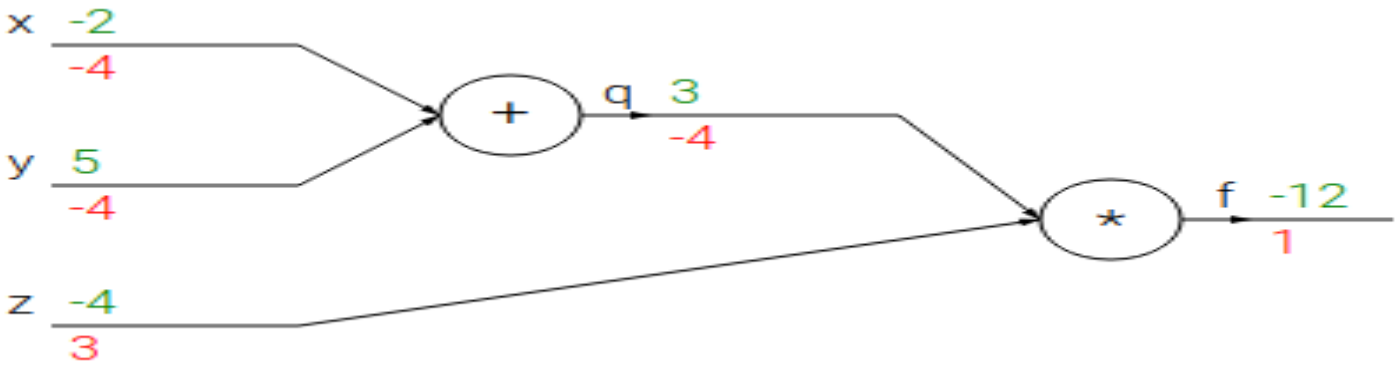
# Backprop-Compound expressions with chain rule




- Assume we want to maimixe the value of graph then the output of the add operation to be lower (due to negative sign), and with a force of 4.
- To continue the recurrence and to chain the gradient, the add operation takes that gradient and multiplies it to all of the local gradients for its inputs (making the gradient on both x and y 1 * -4 = -4).

**Refrence: https://cs231n.github.io/optimization-2/**

# Backprop-Compound expressions with chain rule




- Notice that this has the desired effect
- If x,y were to decrease (responding to their negative gradient) then the add gate's output would decrease, which in turn makes the multiply gate's output increase.

Refrence: https://cs231n.github.io/optimization-2/

# Gradient descent

- Becaus $\|\nabla J\|^2 \geq 0$ this guarantees that $\Delta J \leq 0$
- Means J will always decrease, never increase, if we change θ according to the $\Delta \theta = -\alpha \nabla J$

$$\Delta w = -\alpha \frac{\partial J}{\partial w} \qquad \Delta b = -\alpha \frac{\partial J}{\partial b}$$

$$w_{new} = w + \Delta w = w - \alpha \frac{\partial J}{\partial w} \qquad b_{new} = b + \Delta b = b - \alpha \frac{\partial J}{\partial b}$$

Reference: Michael A. Nielsen, “Neural Networks and Deep Learning” Determination Press 2015

# Gradient-Based Optimization

- Lets take simple example $f(x,y)=2x^2+y^2$
- The partial derivative $\frac{\partial f(x,y)}{\partial x}$ measures how f changes as only the variable x increases.
- The gradient of f is the vector containing all of the partial derivatives
- The vector of partial derivatives $\nabla f=\begin{bmatrix}\frac{\partial f(x,y)}{\partial x}\\ \frac{\partial f(x,y)}{\partial y}\end{bmatrix}=\begin{bmatrix}4x\\ 2y\end{bmatrix}$

*The derivatives tell you:*
*They indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:*

$$\Delta z=4x.\Delta x+2y.\Delta y$$

# Gradient-Based Optimization

- Here $f(x,y)=2x^2+y^2$ $f(x,y)=z$

$$\nabla f=\begin{bmatrix}4x\\2y\end{bmatrix} \qquad \Delta z=4x.\Delta x+2y.\Delta y$$

$$(0.5,0.5)=0.75$$

$$(0.6,0.5)=0.97 \qquad \Delta z=4(0.5).(0.1)=0.2$$

$$(0.6,0.6)=1.08 \qquad \Delta z=2(0.5).(0.1)=0.1$$

$$\Delta z=0.2+0.1=0.3$$

$$z=z+\Delta z=0.75+0.3=1.05$$

# Gradient-Based Optimization

- Consider a simple addition function of two numbers

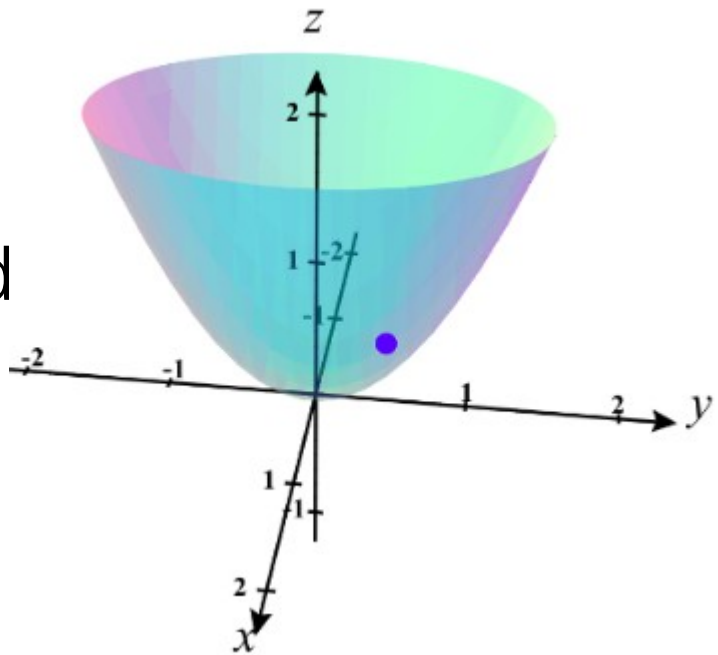
$$f(x,y)=x+y \qquad \frac{\partial f(x,y)}{\partial x}=\frac{\partial f}{\partial x}=1 \qquad \frac{\partial f(x,y)}{\partial y}=\frac{\partial f}{\partial y}=1$$

- That is, the derivative on both x,y is 1 regardless of what the values of x,y are.
- This makes sense, since increasing either x,y would increase the output of f, and the rate of that increase would be independent of what the actual values of x,y are (unlike the case of multiplication above).

Refrence: https://cs231n.github.io/optimization-2/

# Gradient-Multiariate function

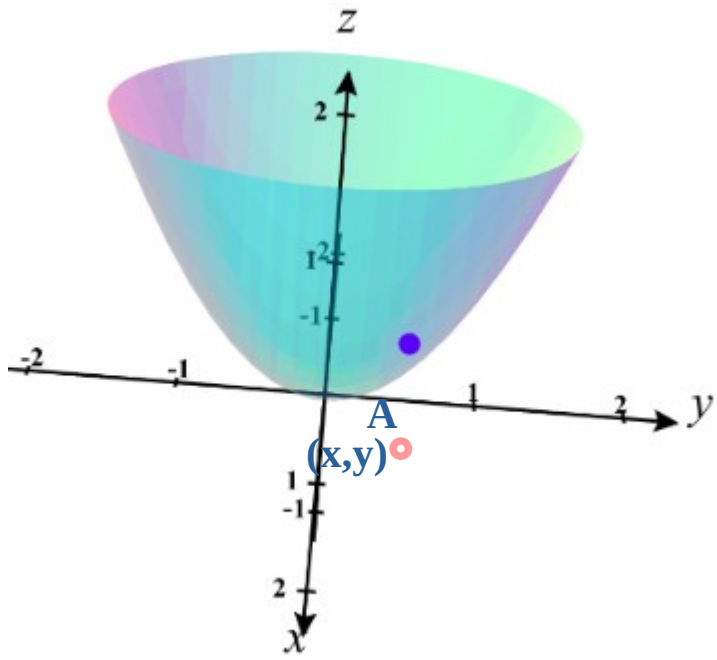
- Let z = f(x, y) is multivariate function
- function z = f(x, y) of two variables, whose graph will be a surface.
- Imagine we change x to x + Δx and y to y + Δy with Δx and Δy very small.
- What is the corresponding change in z?




Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Multiariate function

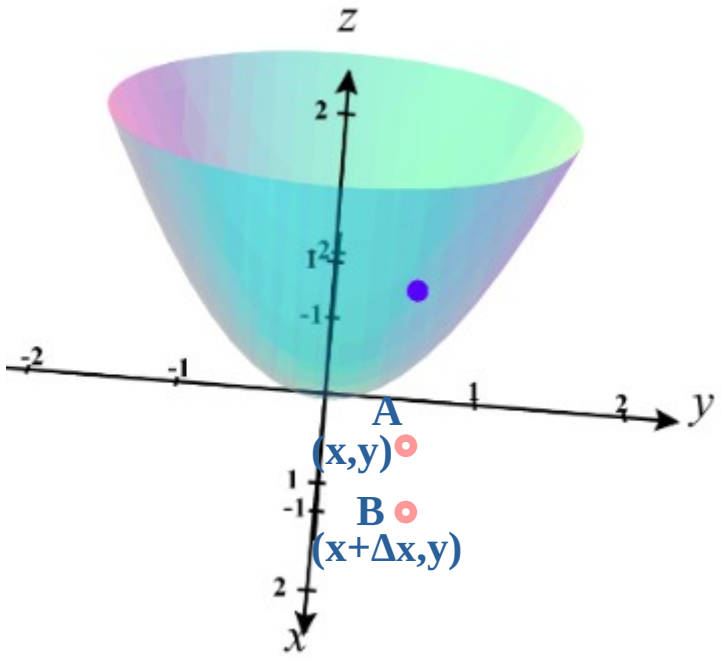
- In the (x, y) plane let A be the point with coordinates (x, y)




Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Multiariate function

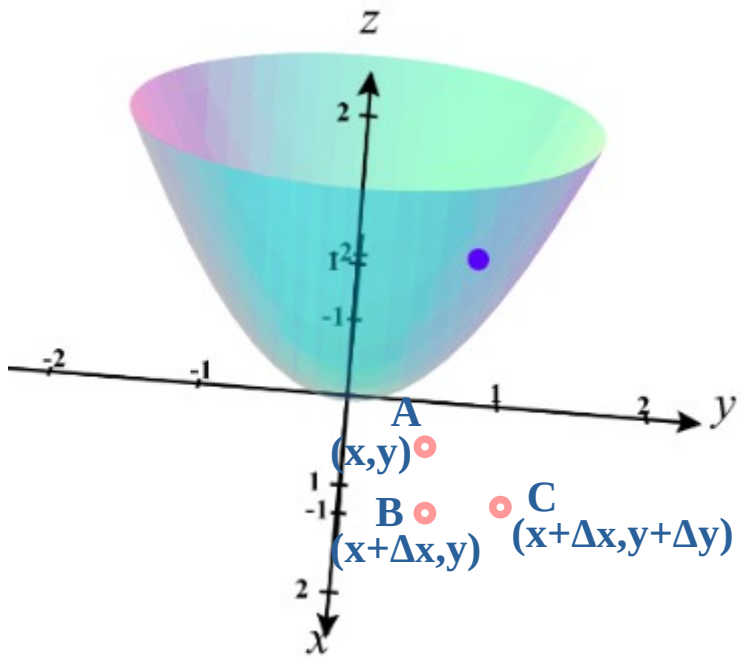
- In the (x, y) plane let A be the point with coordinates (x, y)
- let B be the point with coordinates (x + Δx, y)




Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Multiariate function

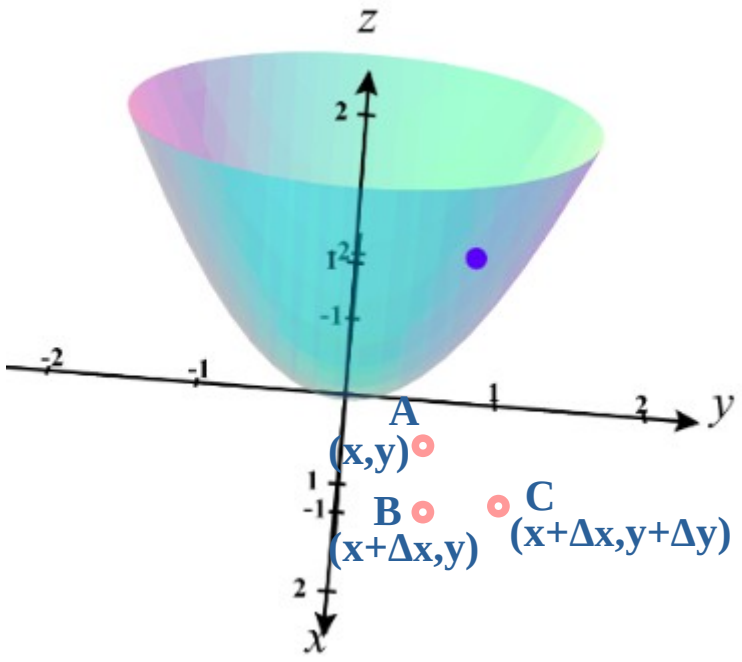
- In the (x, y) plane let A be the point with coordinates (x, y)
- let B be the point with coordinates (x + Δx, y)
- and C the point with coordinates (x + Δx, y + Δy).




Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Multiariate function

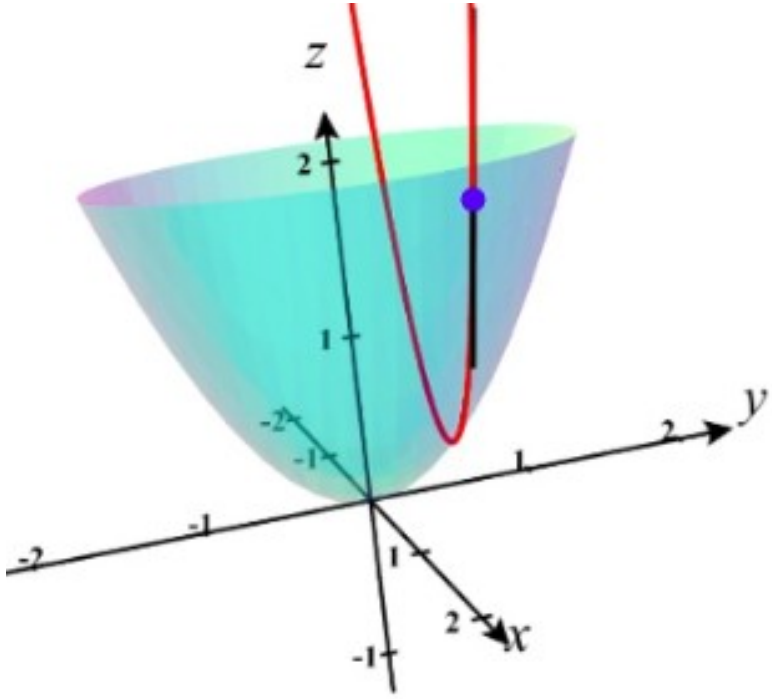
- The overall change in height, Δz, from A to C is given by
- Δz = (change in height A to B)+ (change in height B to C)




Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Multiariate function

- In calculating the change in height from A to B we are travelling across the surface from A to B along a curve in which y is held fixed
- change in height A to B ≈ $\frac{\partial z}{\partial x}\Delta x$

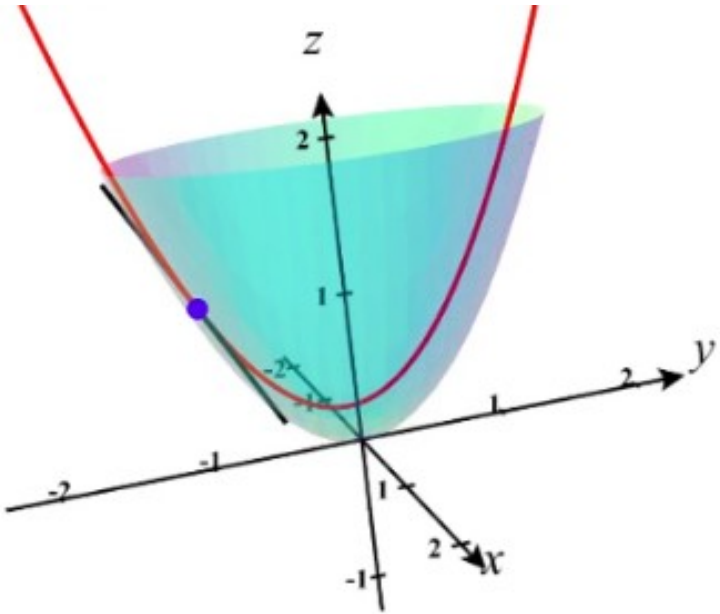



$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Multiariate function

- In calculating the change in height from B to C we are travelling across the surface from B to C along a curve in which x is held fixed
- change in height B to C $\approx \frac{\partial z}{\partial y} \Delta y$

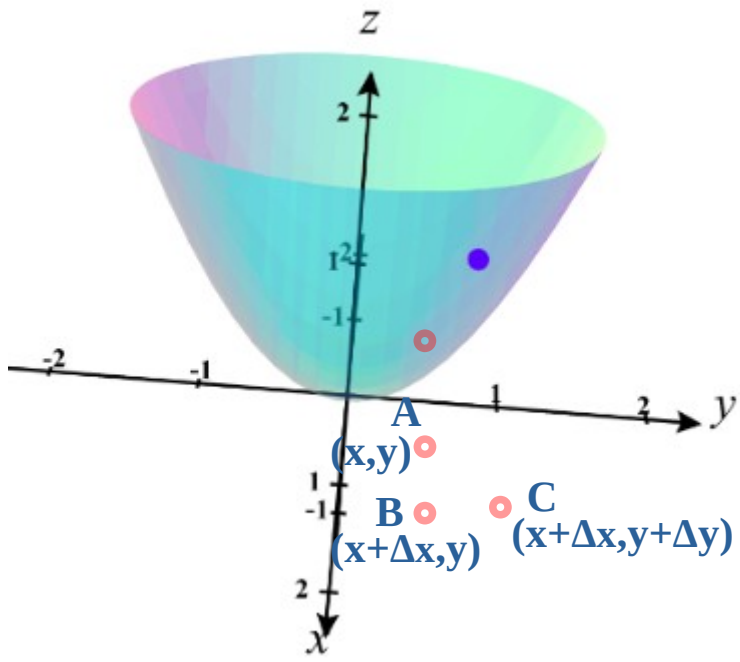



Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf

# Gradient-Multiariate function

- The overall change in height, Δz, from A to C is given by
- Δz = (change in height A to B)+ (change in height B to C)
- Therefore

$$\Delta z = \frac{\partial z}{\partial x}\Delta x + \frac{\partial z}{\partial y}\Delta y$$




Reference: http://personal.maths.surrey.ac.uk/st/S.Zelik/teach/calculus/partial_derivatives.pdf