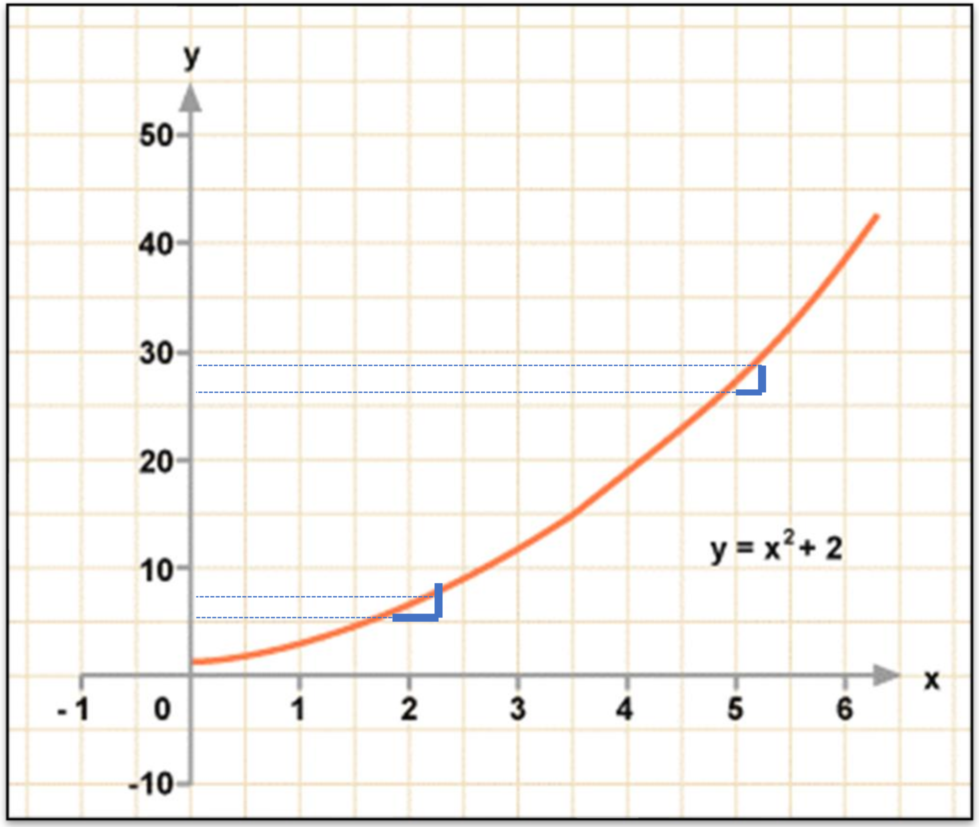
Gradient Descent

Derivatives



$y=f(x)=x^2+2$

$\frac{df(x)}{dx} = 2x$

x=2 f(x)=6

x=2.0001 f(x)=6.00040001

Slope at x=2 is .0004/.0001= 4

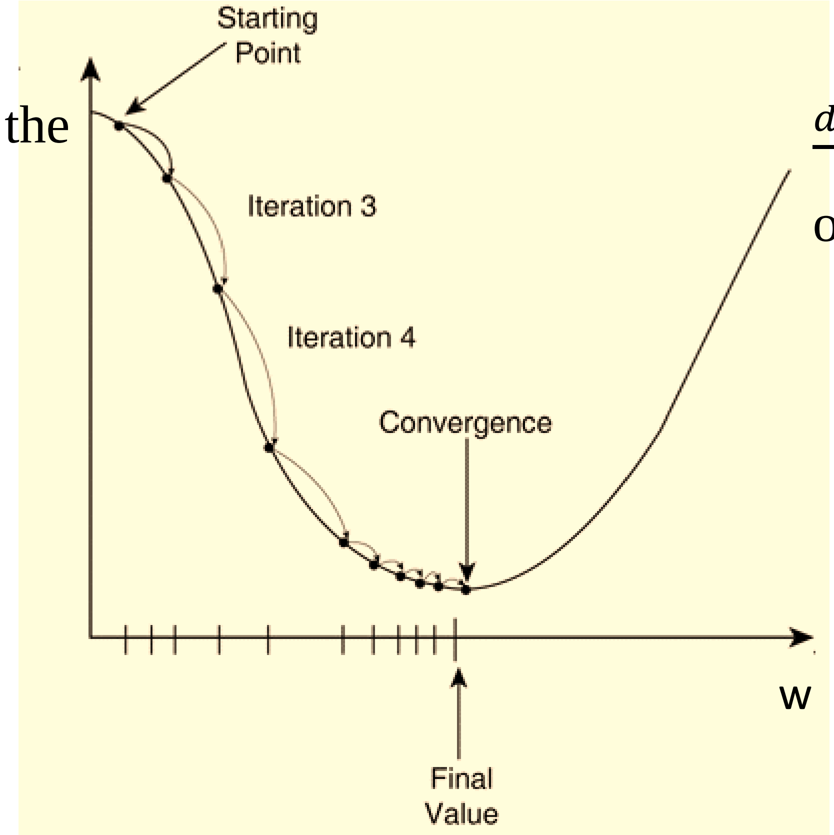
x=5 f(x) =27

x=5.0001 f(x) = 27.00100001

Slope at a =5 is .0010/.0001= 10

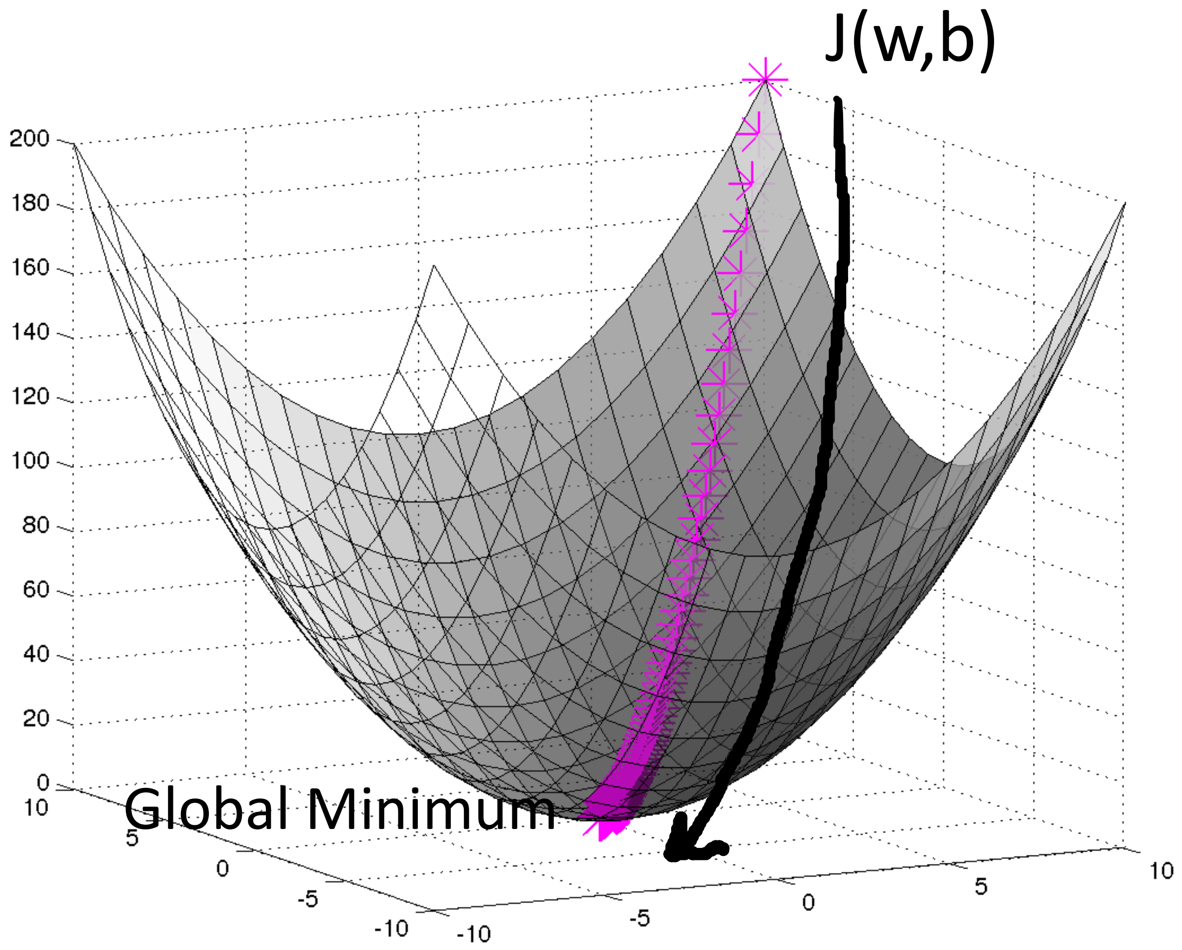
Gradient Descent single dimension

$\frac{dj(w)}{dw}$ is less than the optimum value



$\frac{dj(w)}{dw}$ is more than the optimum value

Gradient Descent



How the weights will change

This process will repeat until we find or reach near the global minimum.

α is the learning rate which will decide that how fast or slow we are going towards the global minima. How big or small steps we want to take. Both have their own limitations and advantages.

Derivatives in logistic regression

Original Value of y=1

$x_1=5$ $w_1=0.5$

$b=1$ $\quad z=w_1x_1+w_2x_2+b \rightarrow \hat{y}=a=\sigma(z) \rightarrow L(a,y)$

$x_2=3$ $w_2=0.5$

$z=0.5*5+0.5*3+1=5$ $\quad a=0.99307 \quad \approx 0.0023$

$dz=\frac{dL(a,y)}{dz}=\frac{dL}{da}\frac{da}{dz}=(\frac{-y}{a}+\frac{1-y}{1-a})*a(1-a)=a-y=-0.00693$

$da=\frac{dL(a,y)}{da}=\frac{-y}{a}+\frac{1-y}{1-a}=-1.006978$

$dw_1=\frac{dL(a,y)}{dw_1}=x_1.dz=5*-0.00695=-0.03475$ $\quad dw_2=\frac{dL(a,y)}{dw_2}=x2.dz=3*-0.00695=-0.02085$ $\quad db=dz$

$\alpha=0.01$

$w_1=w_1-\alpha dw_1=0.5-(0.01*-0.03475)=0.503475$

$w_2=w_2-\alpha dw_2=0.5-(0.01*-0.02085)=0.502085$

$b=b-\alpha db=1-(0.01)*(-0.00693)=1.000693$

Logistic Regression on whole training set

J=0, dw_i=0,db=0

for i=1 to m

$z^{(i)}=w^T x^{(i)}+b$

$a^{(i)}=\sigma z^{(i)}$

J+=-[$y^{(i)}\log a^{(i)} + (1-y^{(i)}) \log (1- a^{(i)})$]

$dz^{(i)}=a^{(i)}-y^{(i)}$

for k=1 to n #no of features

dw_k+=$x_k^{(i)}dz^{(i)}$

db+=$dz^{(i)}$

J/=m

db/=m

for i=1 to n

dw_i/=m

$w_i=w_i-\alpha\, dwi$

b=b-αdb

Vectorized

for iteration (epoch) in range (1000)

$Z=w^TX+b$ (In python np.dot(w.T,X)+b)

$A=\sigma Z$

$dZ=A-Y$

$dw=\frac{1}{m}XdZ^T$

$db=\frac{1}{m}$ np.sum(dz)

$w=w-\alpha\, dw$

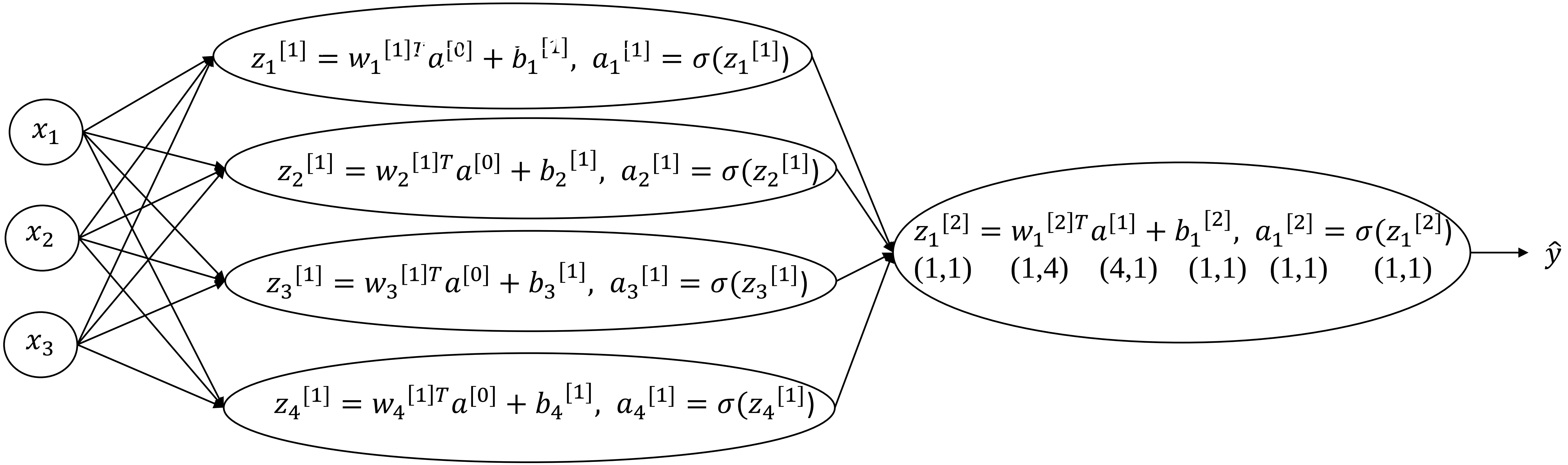
$b=b-\alpha db$

Neural Network Representation

2 layer neural network

$a_i{}^l \quad a^{layer}_{node\ in\ layer}$

(4,1) (4,3) (3,1) (4,1) (4,1) (4,1)

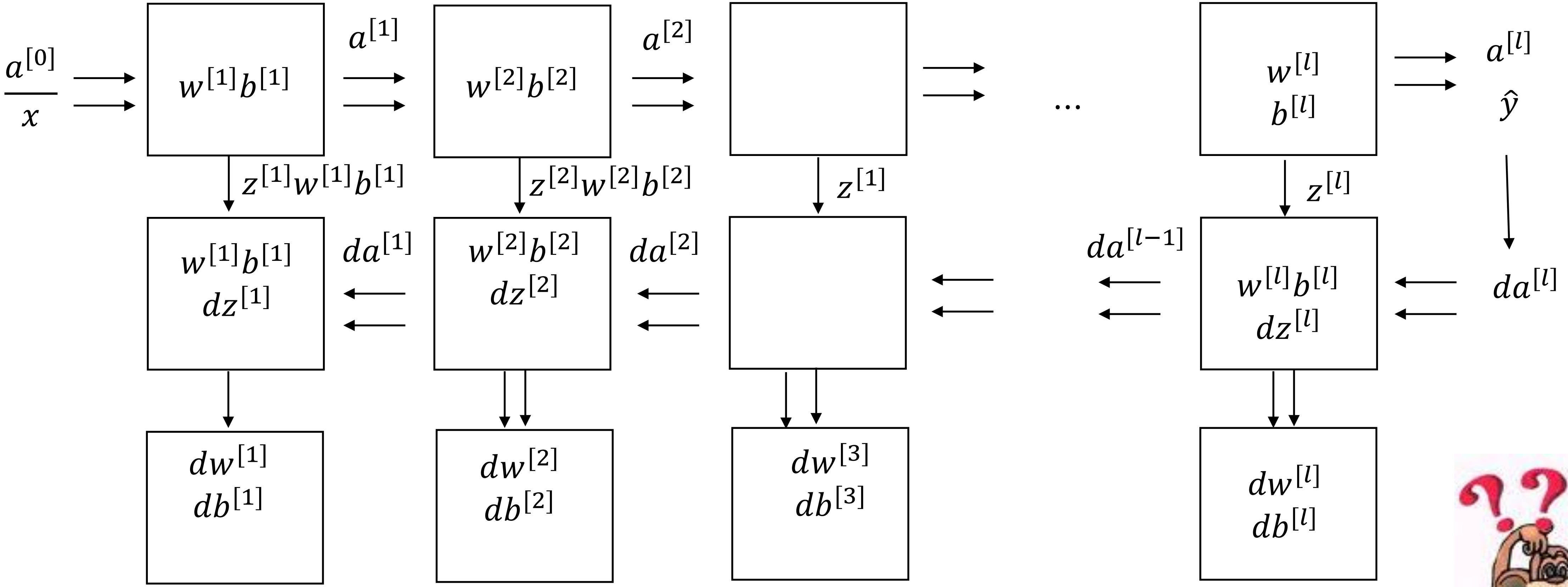


$a^{[0]}$ Layer 0
Input Layer

$a^{[1]}$ Layer 1 $w^{[1]}$ $b^{[1]}$ $z^{[1]}$
Hidden Layer

$a^{[2]}$ Layer 2 $w^{[2]}$ $b^{[2]}$ $z^{[2]}$
Output Layer

Forward and Backward Functions



$$w^{[l]} = w^{[l]} - \propto dw^{[l]}$$
$$b^{[l]} = b^{[l]} - \propto db^{[l]}$$

When to stop the training?