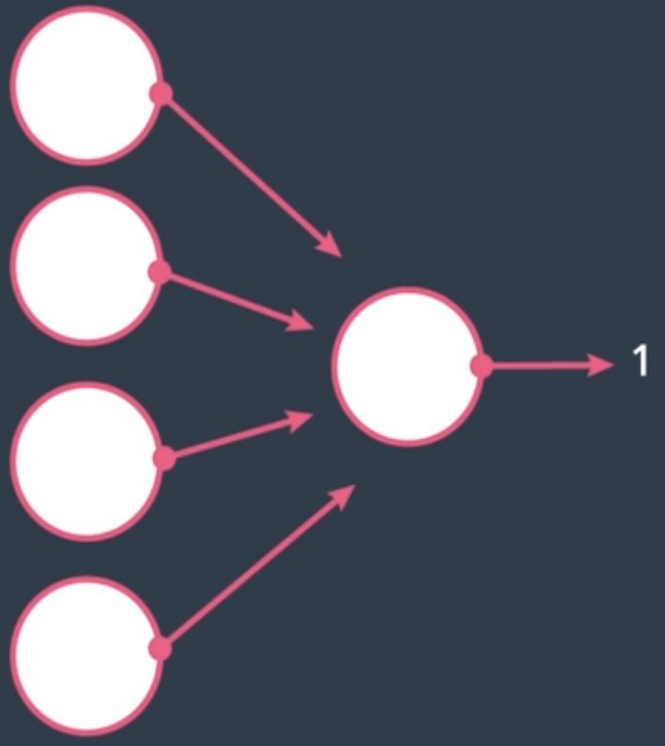
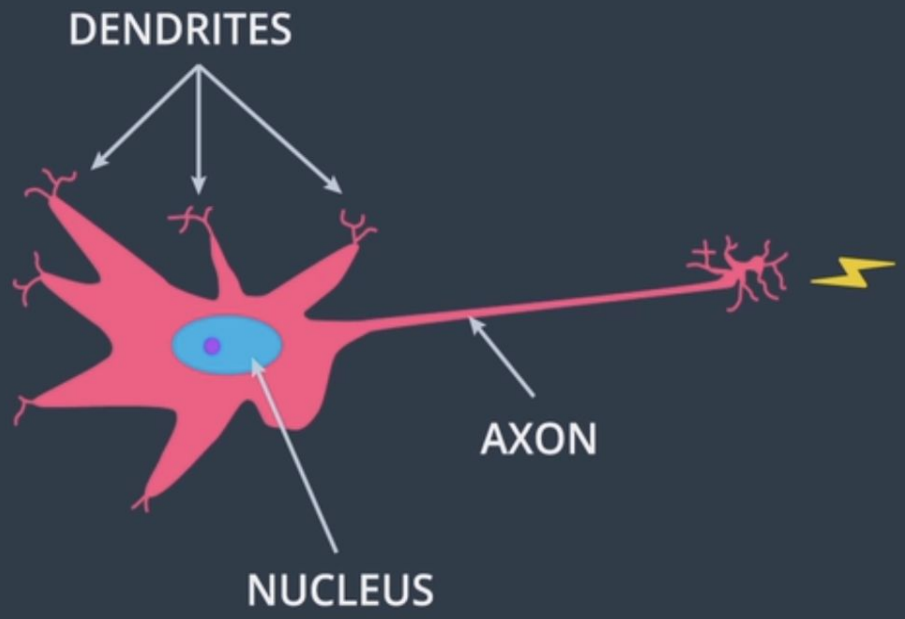
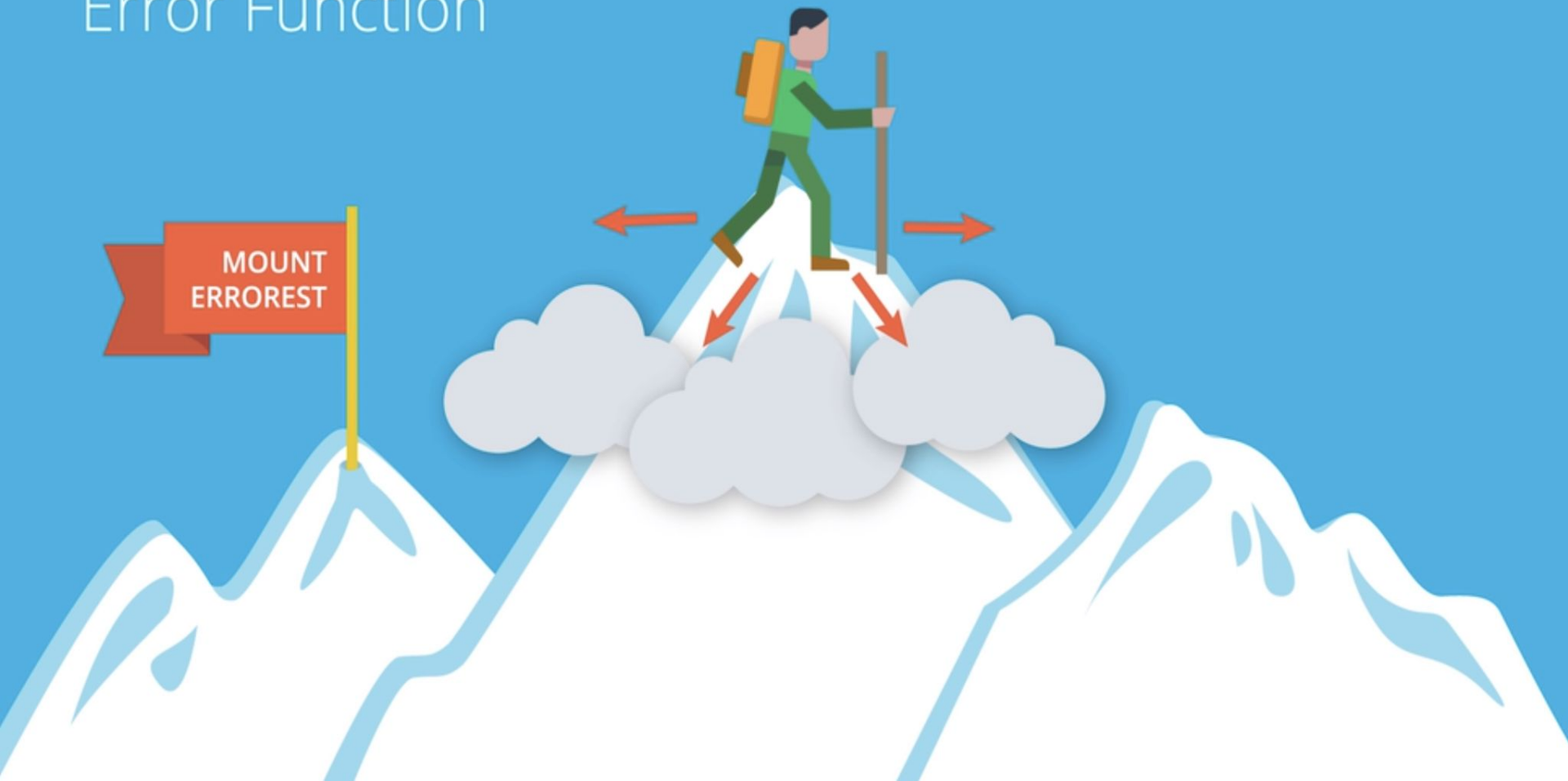
Perceptron





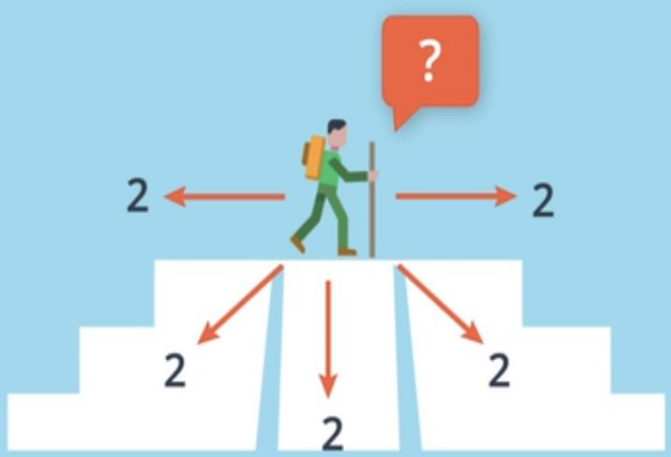
Error Function



Error Function

Error Function

Discrete vs Continuous



DISCRETE

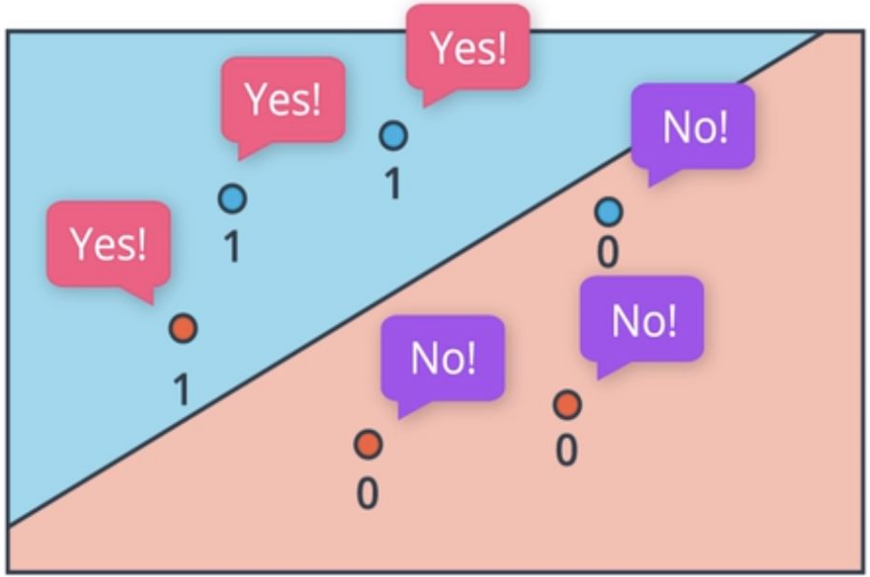
CONTINUOUS

Predictions

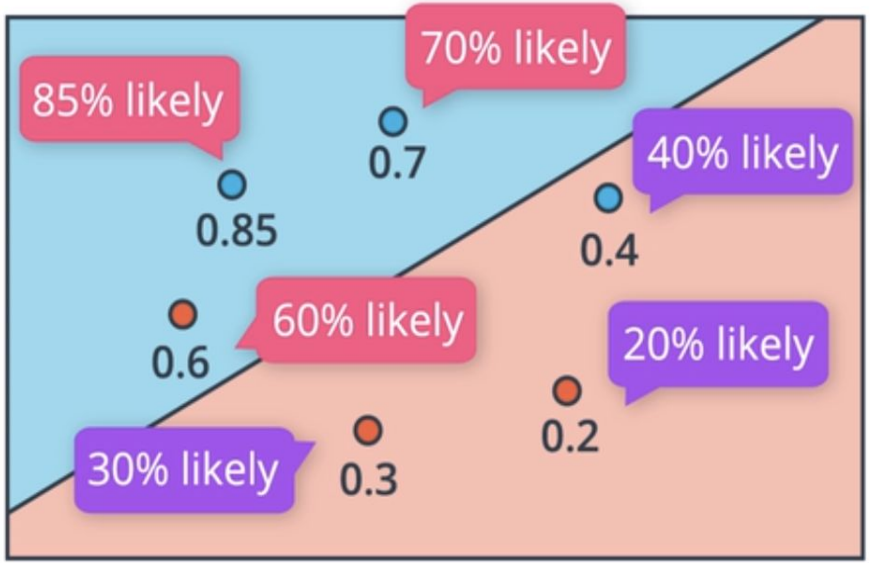
DISCRETE

CONTINUOUS

Predictions

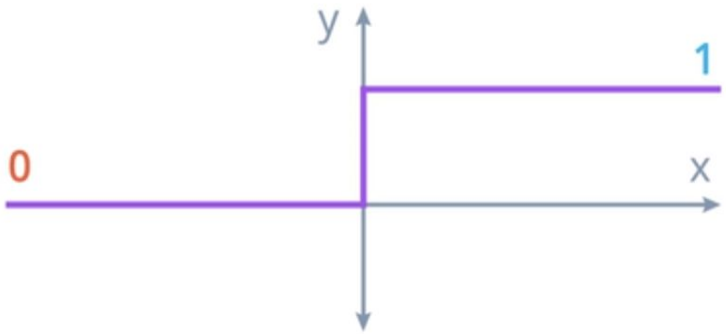


DISCRETE



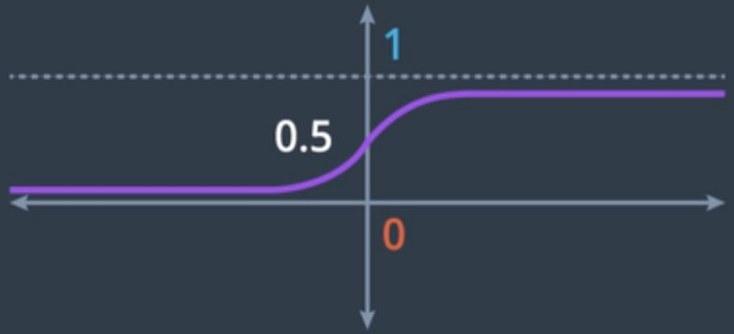
CONTINUOUS

Activation Functions



DISCRETE:
Step Function

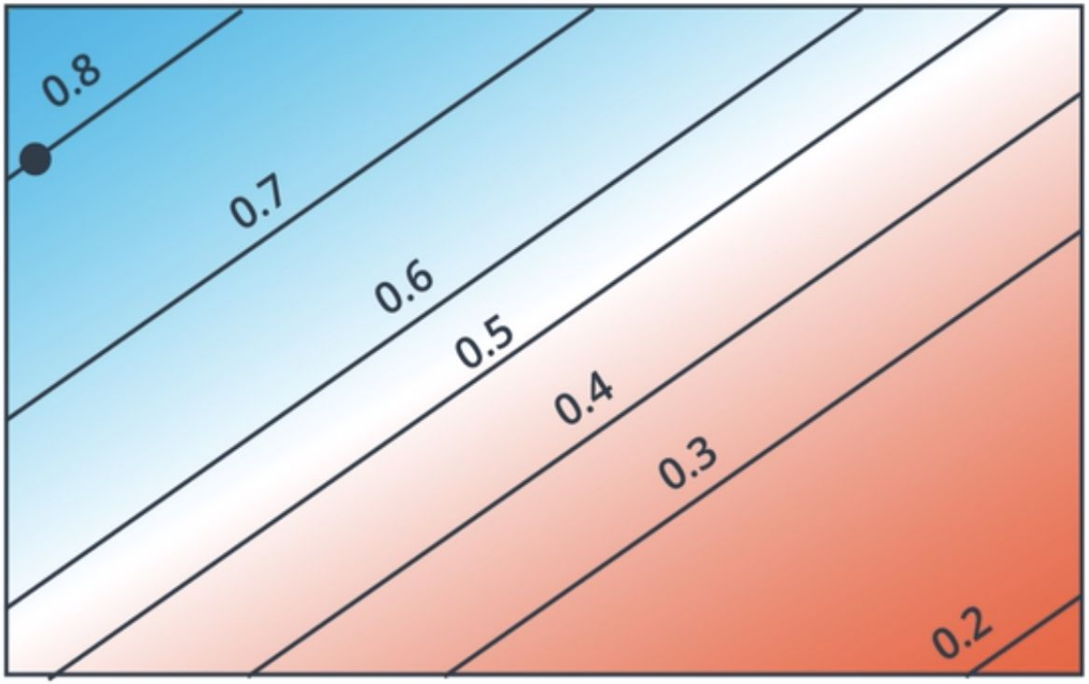
$$y = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$



CONTINUOUS:
Sigmoid Function

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

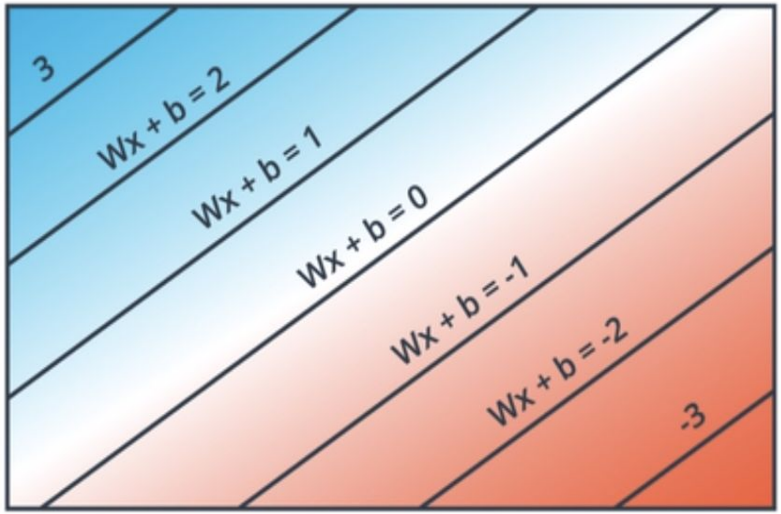
Predictions



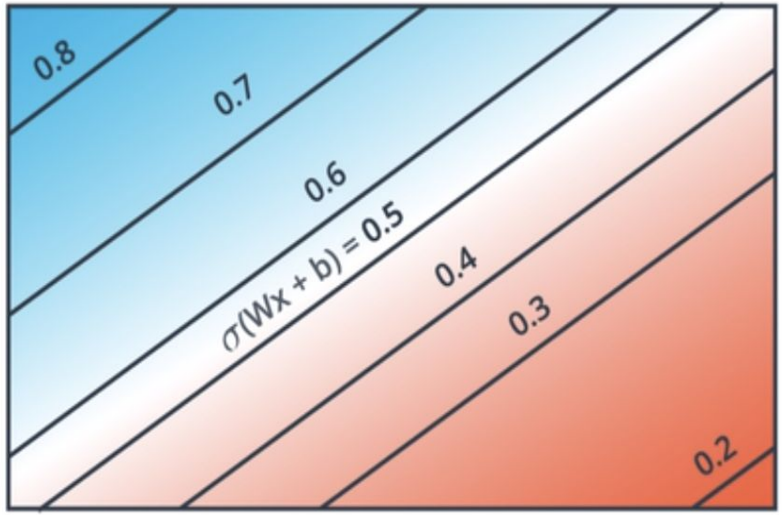
P(BLUE) = 0.8

P(RED) = 0.2

Predictions



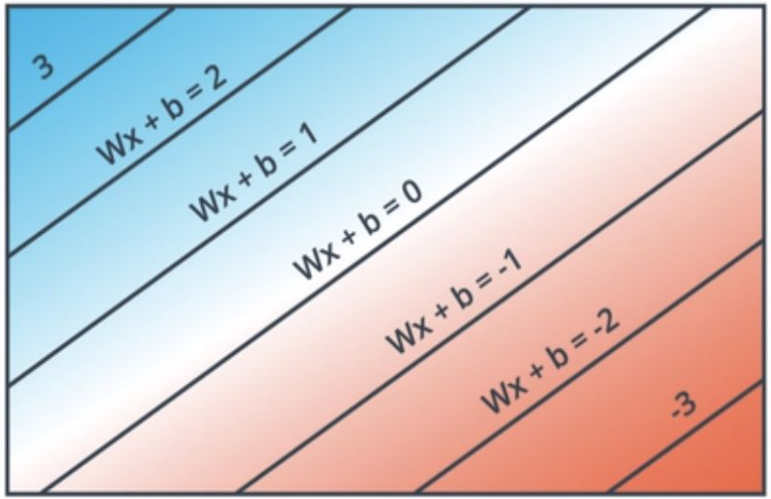
σ



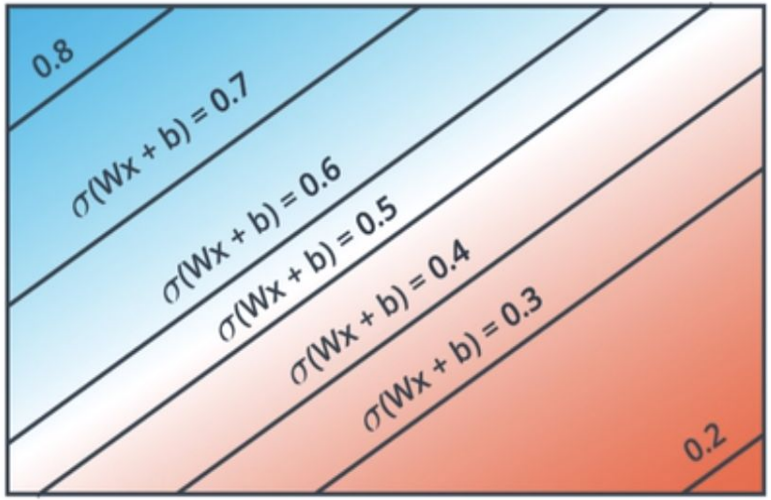
$Wx + b$

$\hat{y} = \sigma(Wx + b)$

Predictions



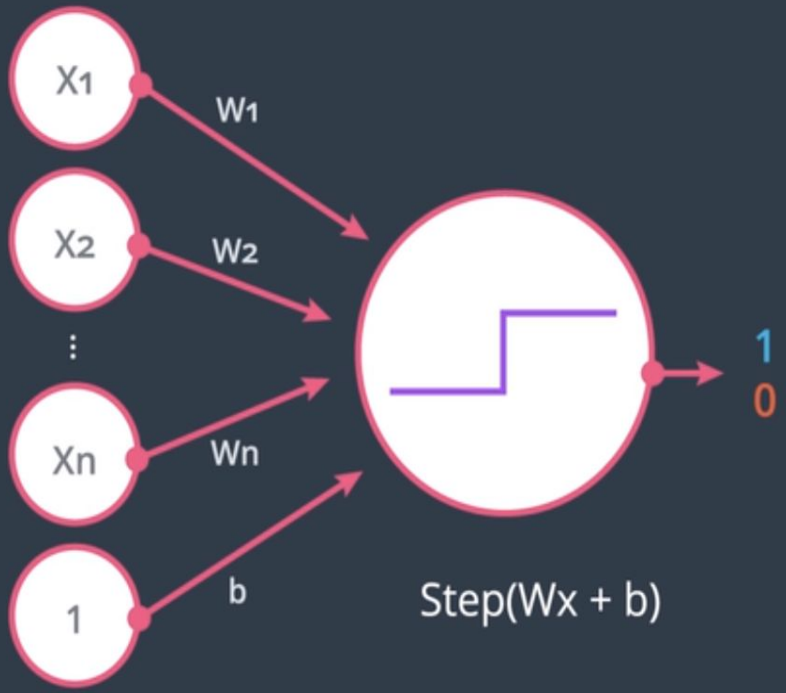
σ

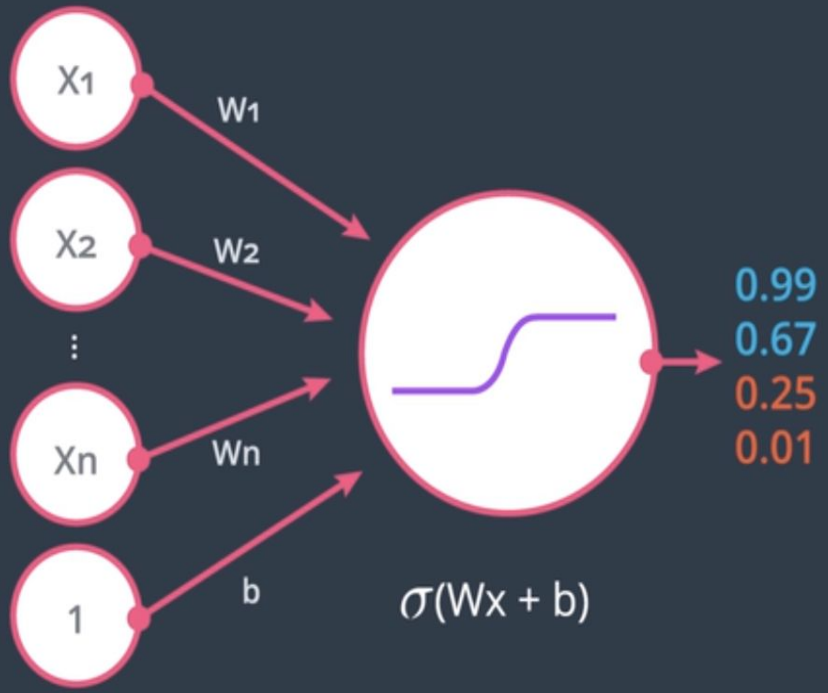


$Wx + b$

$\hat{y} = \sigma(Wx + b)$

Perceptron


X1
W1
X2
W2
Xn
Wn
1
b
1
0
Step(Wx + b)


X1
W1
X2
W2
Xn
Wn
1
b
0.99
0.67
0.25
0.01
σ(Wx + b)

Classification Problem

P(gift) = 0.8

P(no gift) = 0.2

Score(gift) =
Linear Function

P(gift) =
σ(Score)

Classification Problem

P(duck) =
0.67

Score = 2

P(beaver) =
0.24

Score = 1

P(walrus) =
0.09

Score = 0

Classification Problem

2

$$\frac{2}{2 + 1 + 0}$$

1

$$\frac{1}{2 + 1 + 0}$$

0

$$\frac{0}{2 + 1 + 0}$$

Problem:
Negative Numbers?

$$\frac{1}{1 + 0 + (-1)}$$

QUIZ

What function turns every number into positive?

- ○ sin
- ○ log
- ○ cos
- ○ exp

Classification Problem

2

$$\frac{2}{2 + 1 + 0}$$

1

$$\frac{1}{2 + 1 + 0}$$

0

$$\frac{0}{2 + 1 + 0}$$

Problem:
Negative Numbers?

$$\frac{1}{1 + 0 + (-1)}$$

QUIZ

What function turns every number into positive?

- ○ sin
- ○ log
- ○ cos
- ● exp

Classification Problem

2

$$\frac{2}{2+1+0}$$

$$\frac{e^2}{e^2+e^1+e^0} = \mathbf{0.67}$$

P(duck)

1

$$\frac{1}{2+1+0}$$

$$\frac{e^1}{e^2+e^1+e^0} = \mathbf{0.24}$$

P(beaver)

0

$$\frac{0}{2+1+0}$$

$$\frac{e^0}{e^2+e^1+e^0} = \mathbf{0.09}$$

P(walrus)

Softmax Function

LINEAR FUNCTION SCORES:

$Z_1, \ldots, Z_n$

$$P(\text{class } i) = \frac{e^{Z_i}}{e^{Z_1} + \ldots + e^{Z_n}}$$

QUESTION

Is Softmax for n=2 values the same as the sigmoid function?

One-Hot Encoding

GIFT?	VARIABLE
	1
	1
	0
	1
	0

ANIMAL

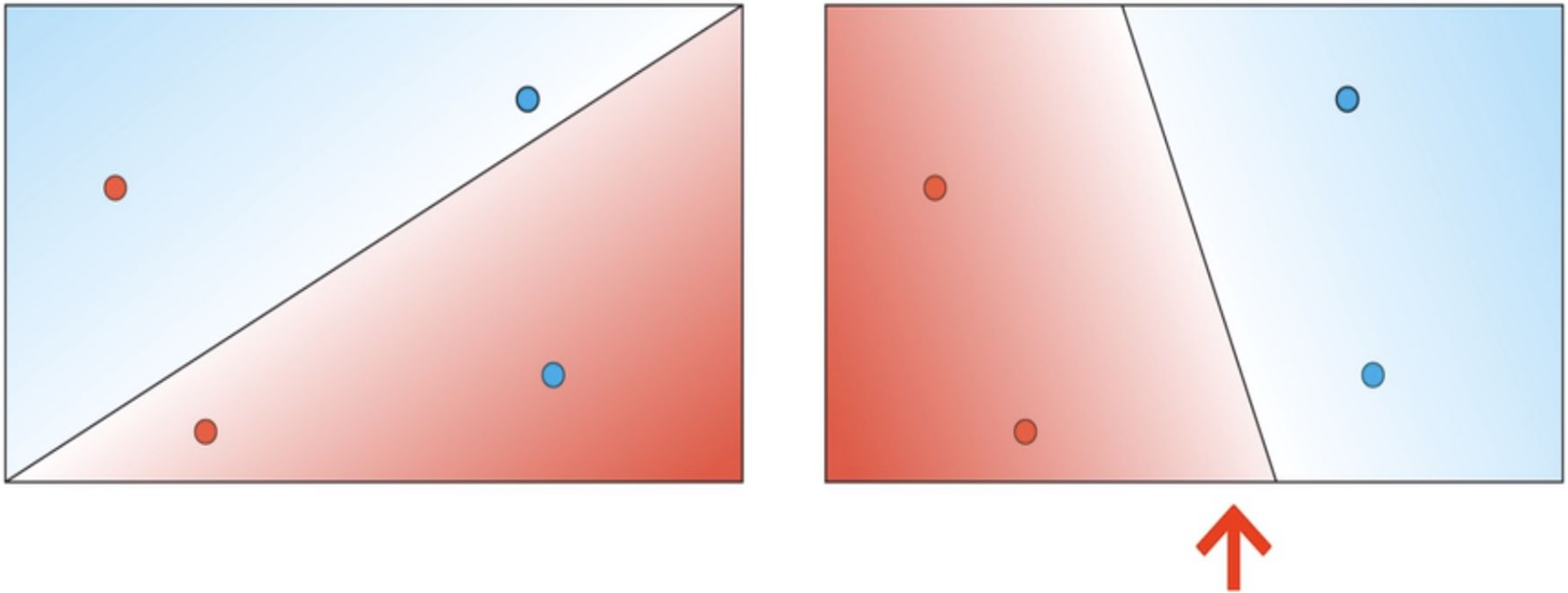



One-Hot Encoding

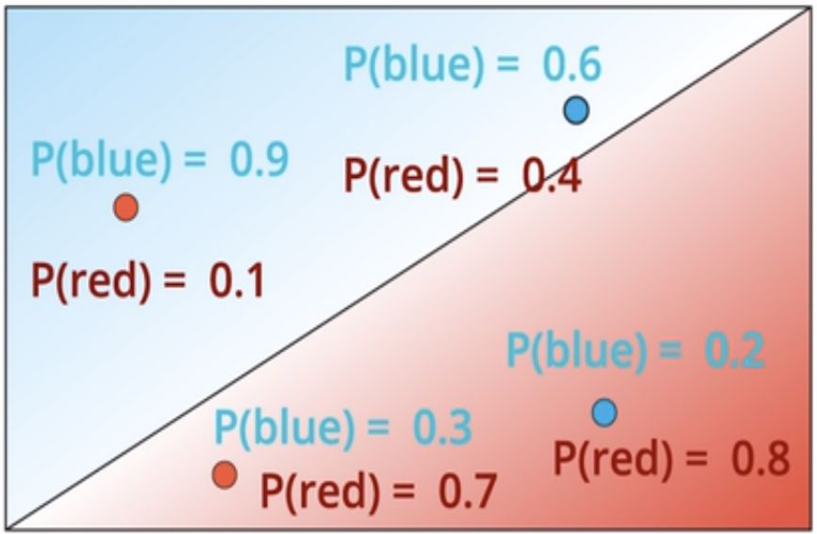
ANIMAL	VALUE
	?
	?
	?
	?
	?

ANIMAL	DUCK?	BEAVER?	WALRUS?
	1	0	0
	0	1	0
	1	0	0
	0	0	1
	0	1	0

Probability



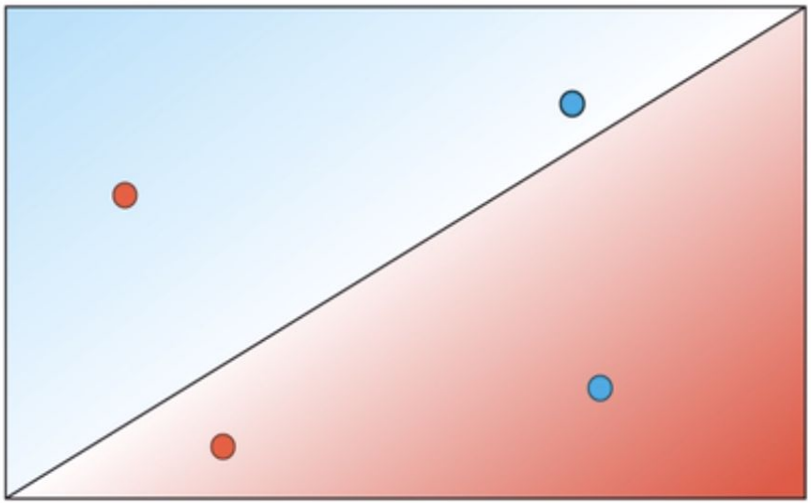
Probability



$\hat{y} = \sigma(Wx+b)$

$P(\text{blue}) = \sigma(Wx+b)$

Probability



$\hat{y} = \sigma(Wx+b)$

$P(\text{blue}) = \sigma(Wx+b)$

P(red) = 0.1

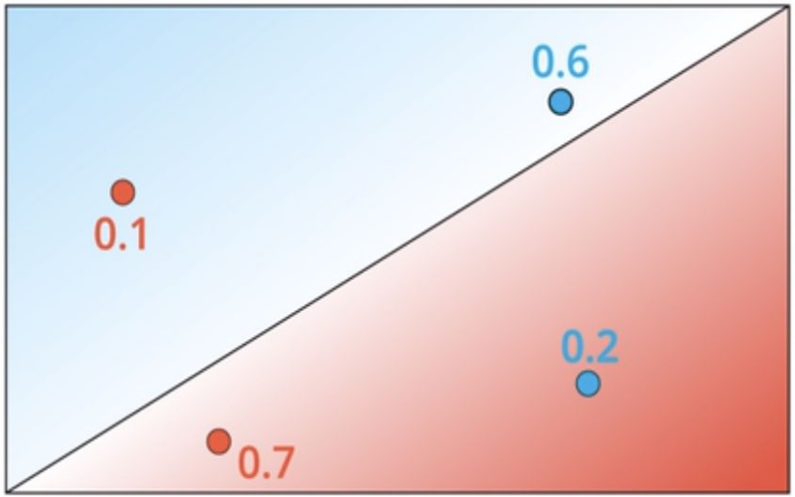
P(blue) = 0.6

P(red) = 0.7

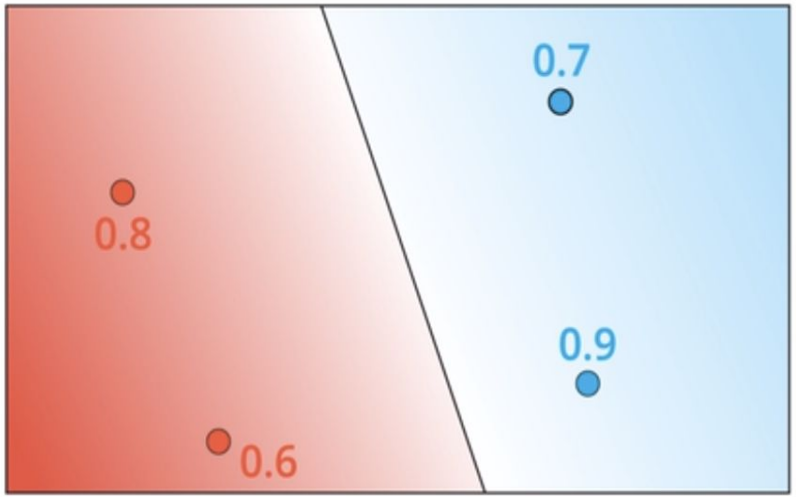
x P(blue) = 0.2

P(all) = 0.0084

Probability

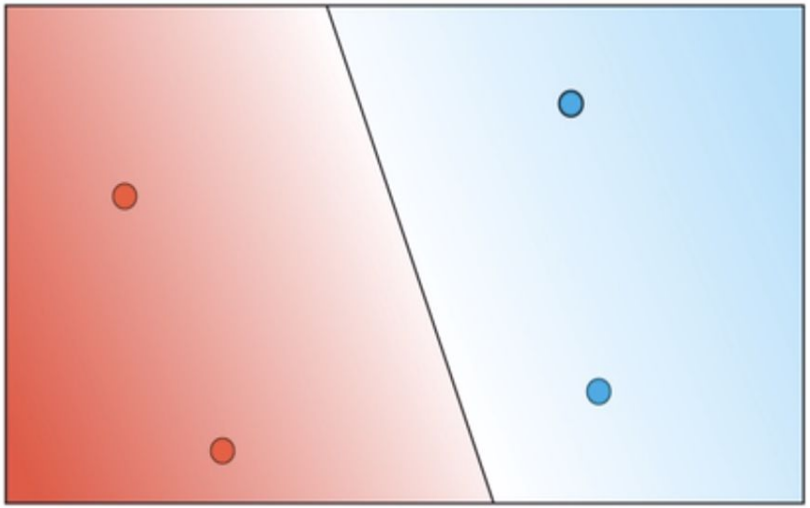


0.6 * 0.2 * 0.1 * 0.7 = 0.0084



0.7 * 0.9 * 0.8 * 0.6 = 0.3024

Probability

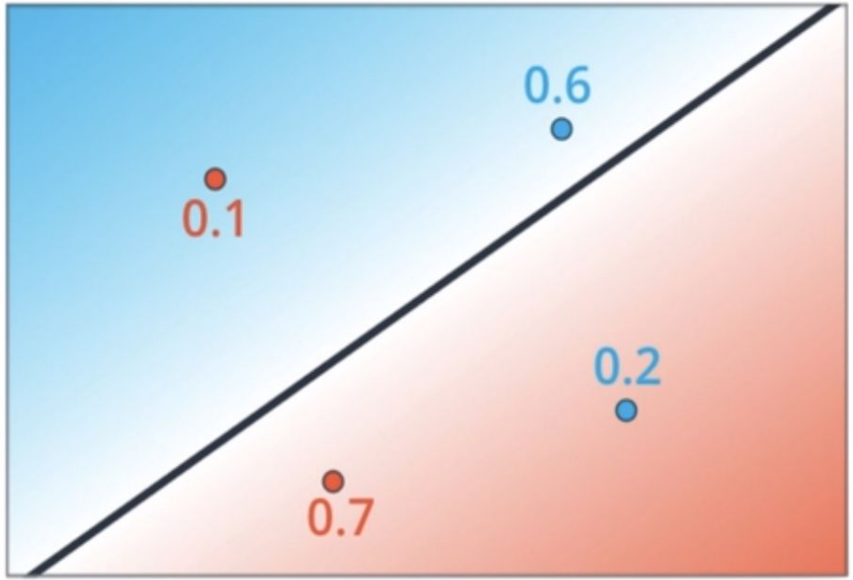


0.6 * 0.2 * 0.1 * 0.7 = 0.0084

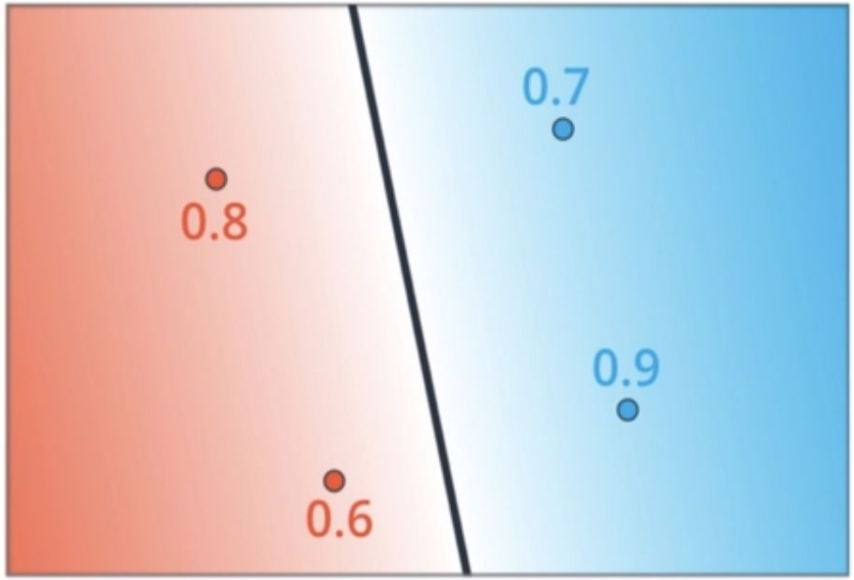
0.7 * 0.9 * 0.8 * 0.6 = 0.3024

Maximum Likelihood

Maximum Likelihood



0.6*0.2*0.1*0.7 = 0.0084



0.7*0.9*0.8*0.6 = 0.3024

Products

$0.6 * 0.2 * 0.1 * 0.7 = 0.0084$

$0.7 * 0.9 * 0.8 * 0.6 = 0.3024$

Quiz:
What function to use?

- sin ○
- cos ○
- log ●
- exp ○

$\log(ab) = \log(a) + \log(b)$

Products

$0.6 * 0.2 * 0.1 * 0.7 = 0.0084$

$\ln(0.6) + \ln(0.2) + \ln(0.1) + \ln(0.7)$

$-0.51 \quad -1.61 \quad -2.3 \quad -0.36$

$-\ln(0.6) - \ln(0.2) - \ln(0.1) - \ln(0.7)$

$0.51 \quad 1.61 \quad 2.3 \quad 0.36$

$0.7 * 0.9 * 0.8 * 0.6 = 0.3024$

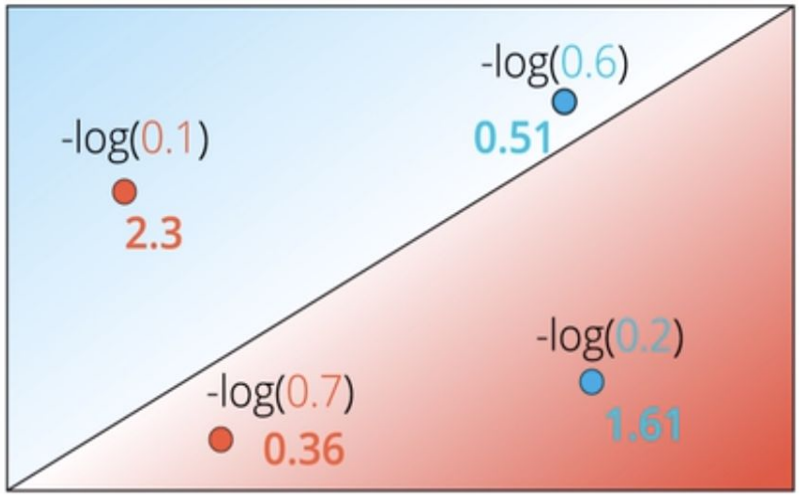
$\ln(0.7) + \ln(0.9) + \ln(0.8) + \ln(0.6)$

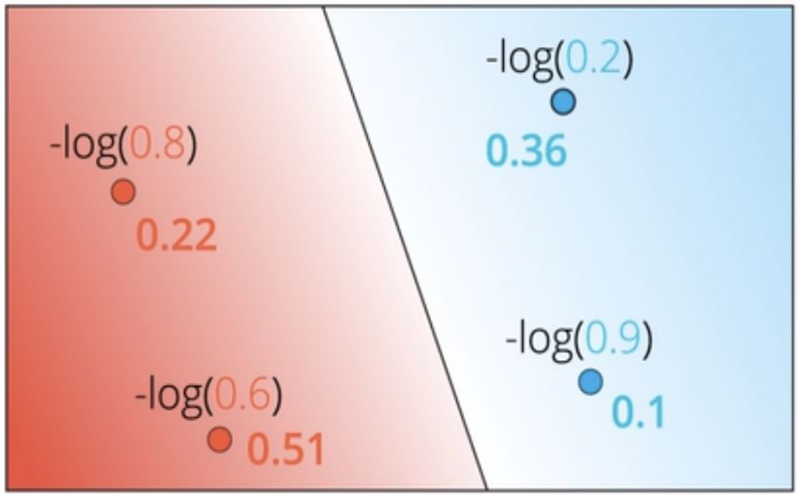
$-0.36 \quad -0.1 \quad -.22 \quad -0.51$

$-\ln(0.7) - \ln(0.9) - \ln(0.8) - \ln(0.6)$

$0.36 \quad 0.1 \quad .22 \quad 0.51$

Cross Entropy





0.6 * 0.2 * 0.1 * 0.7 = 0.0084

-log(0.6) - log(0.2) - log(0.1) - log(0.7) = 4.8

0.7 * 0.9 * 0.8 * 0.6 = 0.3024

-log(0.7) - log(0.9) - log(0.8) - log(0.6) = 1.2

Goal: Minimize the Cross Entropy

Cross-Entropy

Cross-Entropy

			
P(gift)	0.8	0.7	0.1
P(no gift)	0.2	0.3	0.9

Probability= 0.504

Cross-Entropy

			Probability	-ln(Probability)
 0.8	 0.7	 0.1	0.056	2.88
 0.8	 0.7	 0.9	0.504	0.69
 0.8	 0.3	 0.1	0.024	3.73
 0.2	 0.7	 0.1	0.014	4.27
 0.8	 0.3	 0.9	0.216	1.53
 0.2	 0.7	 0.9	0.126	2.07
 0.2	 0.3	 0.1	0.006	5.12
 0.2	 0.3	 0.9	0.054	2.92

Cross-Entropy

$p_1 = 0.8$

$p_2 = 0.7$

$p_3 = 0.1$

p_1 p_2 $1 - p_3$

$y_i = 1$ if present on box i

$y_1 = 1$ $y_2 = 1$ $y_3 = 0$

Cross-Entropy

$-\ln(0.8) - \ln(0.7) - \ln(0.9)$

$$\text{Cross-Entropy} = -\sum_{i=1}^{m} y_i \ln(p_i) + (1 - y_i)\ln(1 - p_i)$$

y_i $\times$ $(1 - y_i)$

$\text{CE}[(1,1,0), (0.8,0.7,0.1)] = 0.69$

$\text{CE}[(0,0,1), (0.8,0.7,0.1)] = 5.12$

Multi-Class Cross-Entropy

Multi-Class Cross-Entropy

ANIMAL	DOOR 1	DOOR 2	DOOR 3
	0.7	0.3	0.1
	0.2	0.4	0.5
	0.1	0.3	0.4
	1	1	1

Multi-Class Cross-Entropy

ANIMAL	DOOR 1	DOOR 2	DOOR 3
	0.7	0.3	0.1
	0.2	0.4	0.5
	0.1	0.3	0.4

$$P = 0.7 * 0.3 * 0.4 = 0.084$$

$$CE = -\ln(0.7) + -\ln(0.3) + -\ln(0.4) = 2.48$$

Multi-Class Cross-Entropy

ANIMAL	DOOR 1	DOOR 2	DOOR 3
	p_{11}	p_{12}	p_{13}
	p_{21}	p_{22}	p_{23}
	p_{31}	p_{32}	p_{33}

$y_{1j} = 1$ if  behind 

$y_{2j} = 1$ if  behind 

$y_{3j} = 1$ if  behind 

Multi-Class Cross-Entropy

ANIMAL	DOOR 1	DOOR 2	DOOR 3
	p_{11}	p_{12}	p_{13}
	p_{21}	p_{22}	p_{23}
	p_{31}	p_{32}	p_{33}

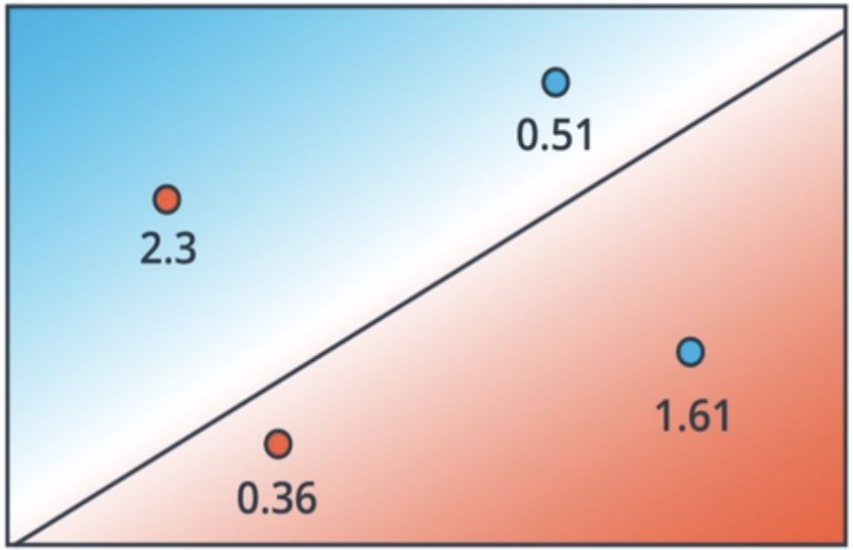
$y_{1j} = 1$ if  behind 

$y_{2j} = 1$ if  behind 

$y_{3j} = 1$ if  behind 

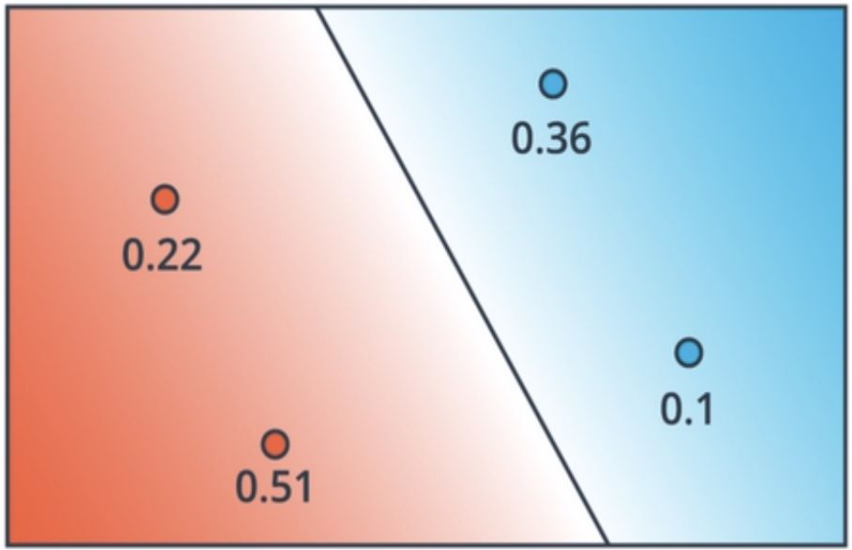
$$\text{Cross-Entropy} = -\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \ln(p_{ij})$$

Error Function



$$-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7) = 4.8$$

0.51 1.61 2.3 0.36



$$-\log(0.7) - \log(0.9) - \log(0.8) - \log(0.6) = 1.2$$

0.36 0.1 0.22 0.51

Error Function

$$\text{Error Function} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)(\ln(1-\hat{y}_i)) + y_i\ln(\hat{y}_i)$$

$$E(W,b) = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)(\ln(1-\sigma(Wx^{(i)}+b)) + y_i\ln(\sigma(Wx^{(i)}+b)$$

GOAL

Minimize Error Function

Error Function

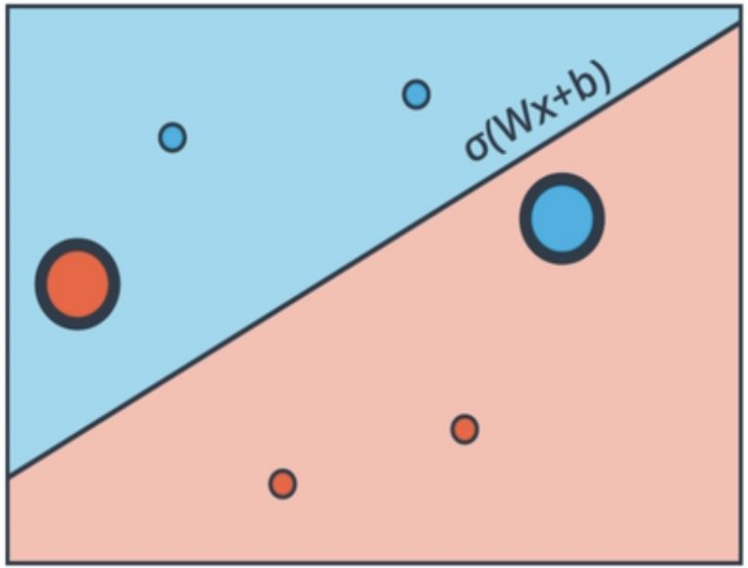
ERROR FUNCTION:

$$-\frac{1}{m}\sum_{i=1}^{m} y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)$$

ERROR FUNCTION:

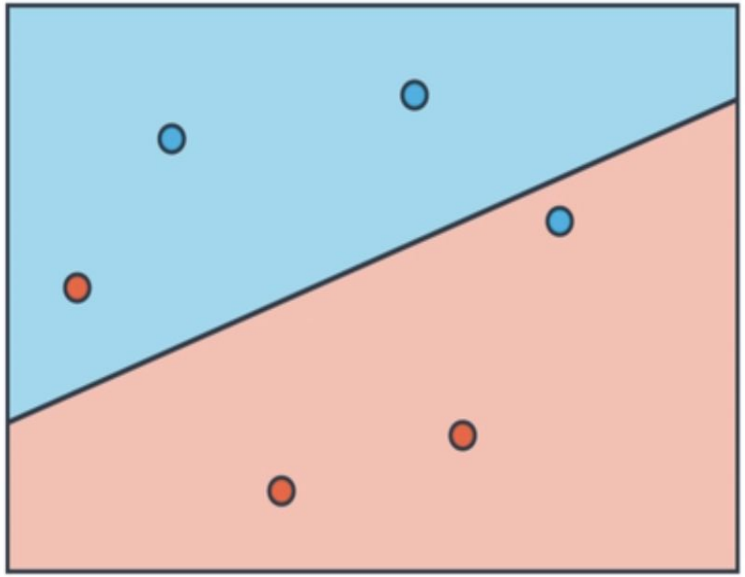
$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} \ln(\hat{y}_{ij})$$

Goal: Minimize Error Function



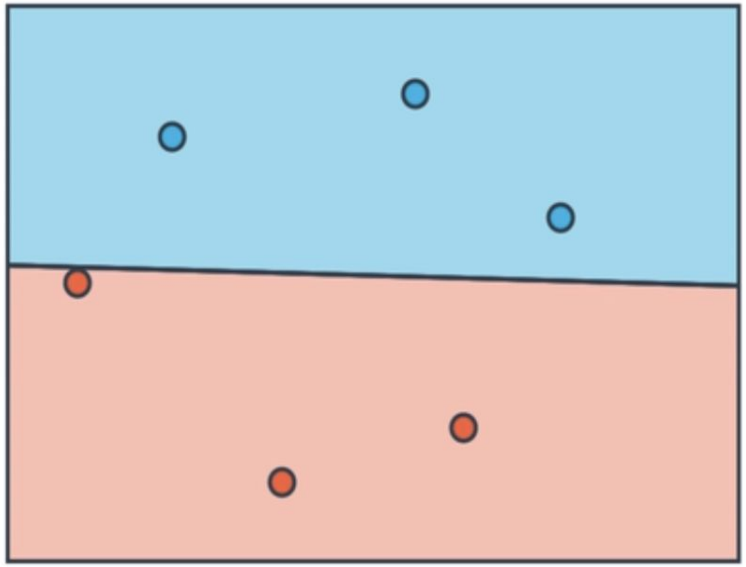
$$E(W,b) = -\frac{1}{m}\sum_{i=1}^{m} (1-y_i)\ln(1-\sigma(Wx^{(i)}+b)) + y_i \ln(\sigma(Wx^{(i)}+b))$$

Goal: Minimize Error Function



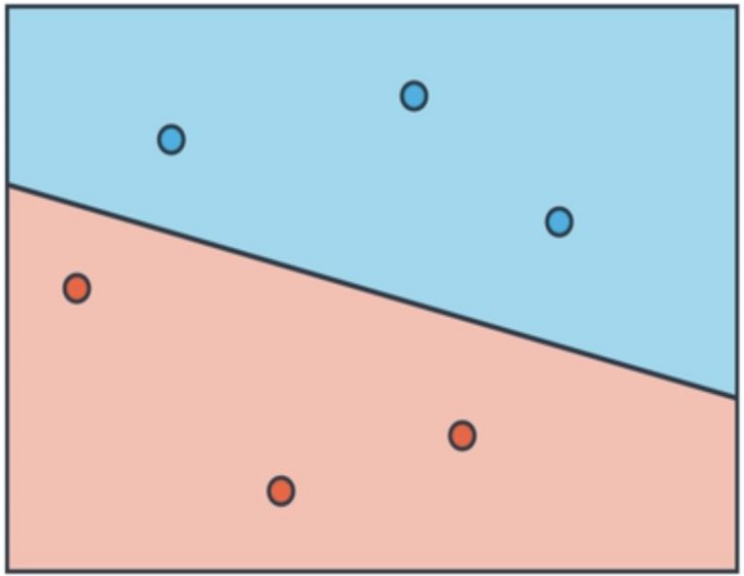
$$E(W,b) = -\frac{1}{m}\sum_{i=1}^{m} (1-y_i)\ln(1-\sigma(Wx^{(i)}+b)) + y_i\ln(\sigma(Wx^{(i)}+b))$$

Goal: Minimize Error Function

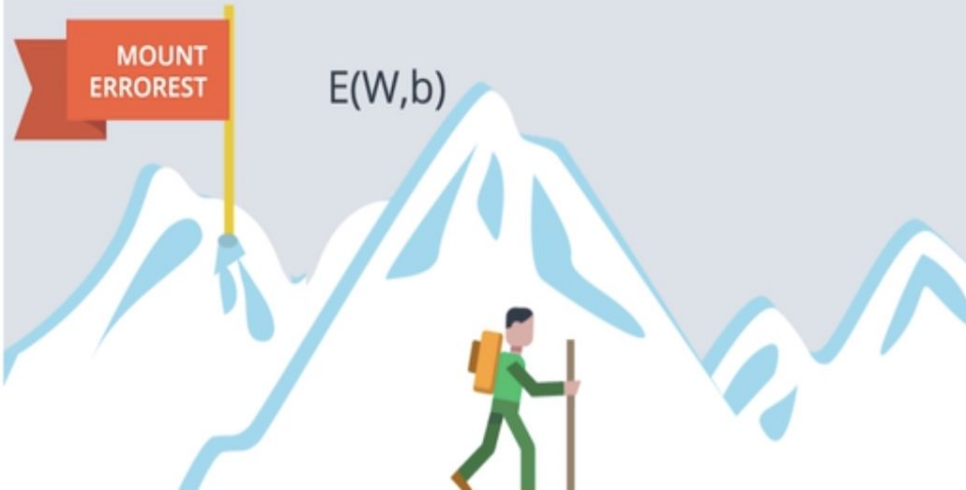


$$E(W,b) = -\frac{1}{m}\sum_{i=1}^{m} (1-y_i)\ln(1-\sigma(Wx^{(i)}+b)) + y_i\ln(\sigma(Wx^{(i)}+b))$$

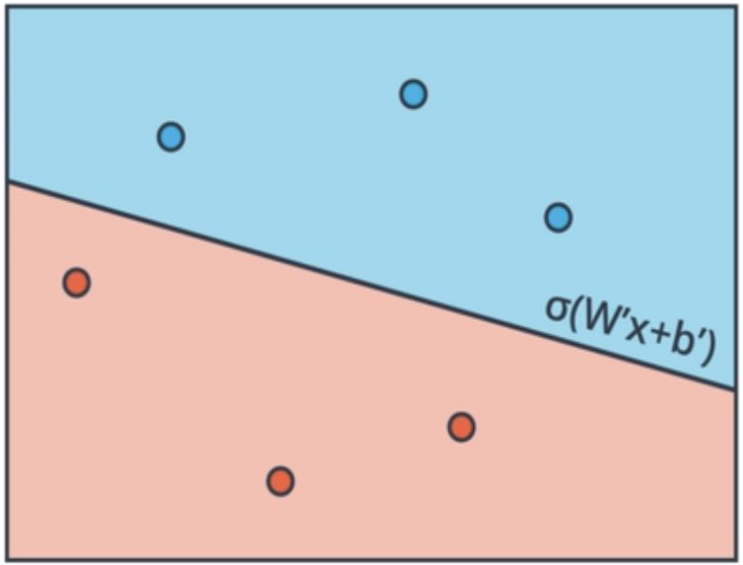
Goal: Minimize Error Function



$$E(W,b) = -\frac{1}{m}\sum_{i=1}^{m} (1-y_i)\ln(1-\sigma(Wx^{(i)}+b)) + y_i\ln(\sigma(Wx^{(i)}+b))$$

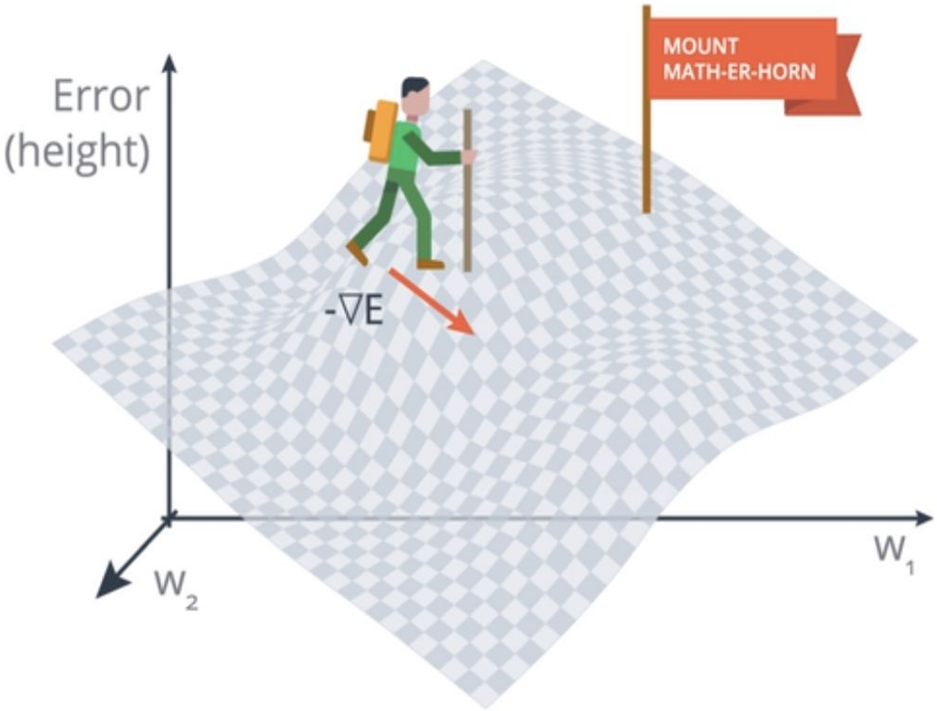


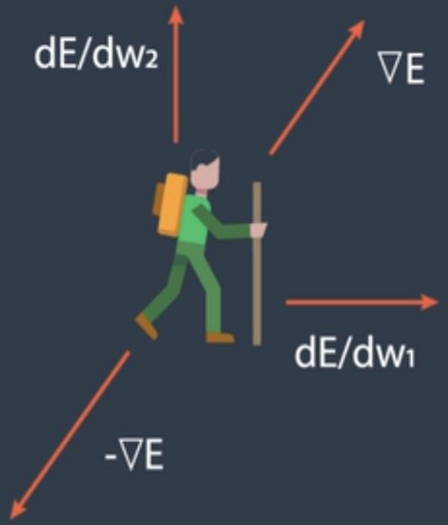
Goal: Minimize Error Function



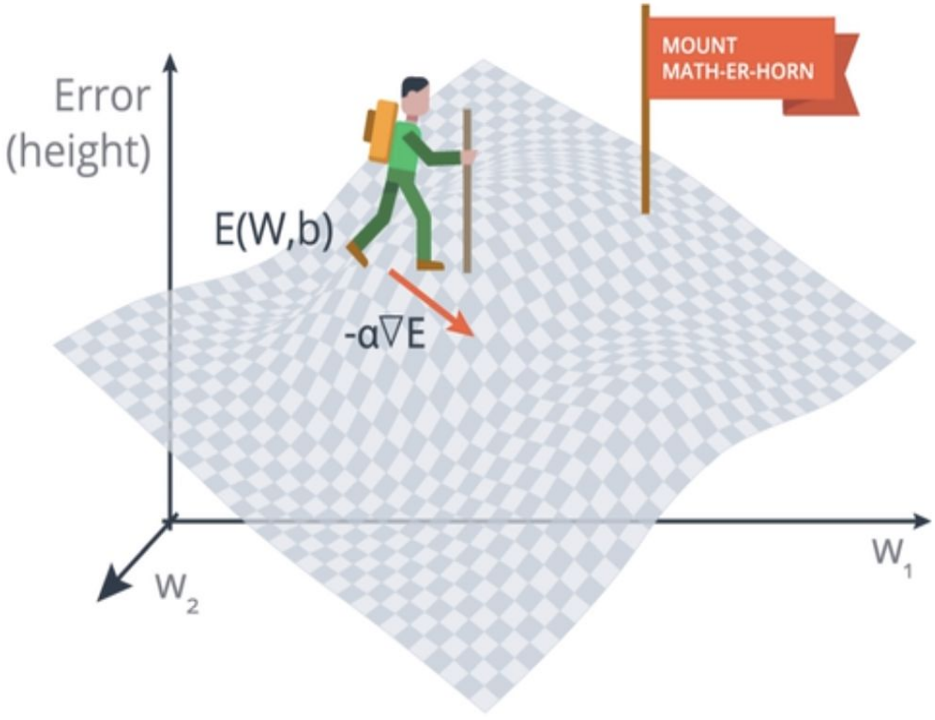
$$E(W,b) = -\frac{1}{m}\sum_{i=1}^{m} (1-y_i)\ln(1-\sigma(Wx^{(i)}+b)) + y_i\ln(\sigma(Wx^{(i)}+b))$$

Gradient Descent





Gradient Descent



$\hat{y} = \sigma(Wx+b)$ ← Bad

$\hat{y} = \sigma(w_1x_1 + \ldots + w_nx_n + b)$

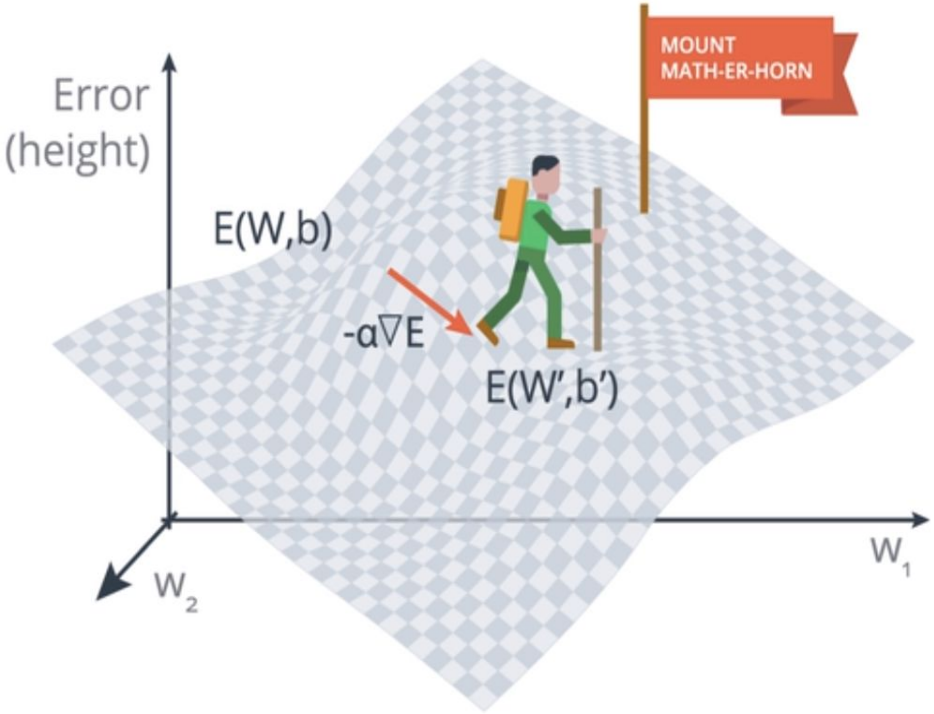
$\nabla E = (\partial E/\partial w_1, \ldots, \partial E/\partial w_n, \partial E/\partial b)$

$\alpha = 0.1$ (learning rate)

$w_i' \leftarrow w_i - \alpha \, \partial E/\partial w_i$

$b' \leftarrow b - \alpha \, \partial E/\partial b$

Gradient Descent



$\hat{y} = \sigma(Wx+b)$ ← Bad

$\hat{y} = \sigma(w_1x_1 + \ldots + w_nx_n + b)$

$\nabla E = (\partial E/\partial w_1, \ldots, \partial E/\partial w_n, \partial E/\partial b)$

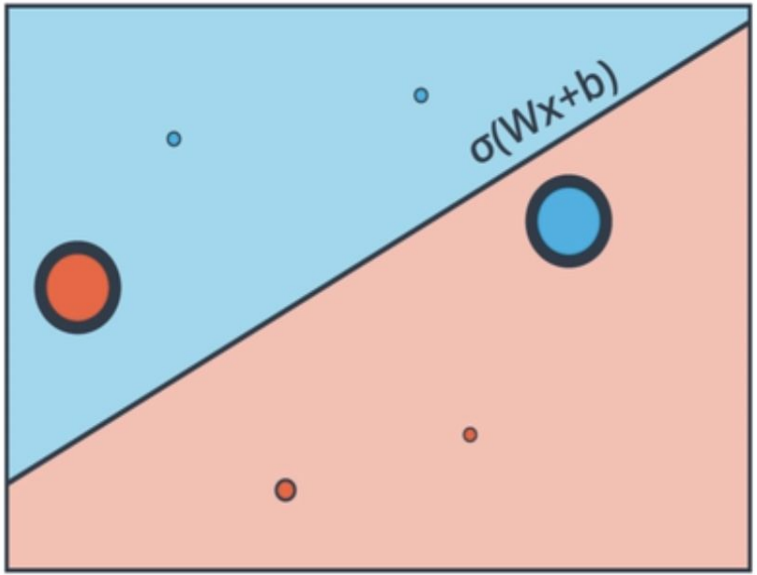
$\alpha = 0.1$ (learning rate)

$w_i' \leftarrow w_i - \alpha \, \partial E/\partial w_i$

$b' \leftarrow b - \alpha \, \partial E/\partial b$

$\hat{y} = \sigma(W'x+b')$

Gradient Descent Algorithm



1. Start with random weights:

   $w_1, \ldots, w_n, b$

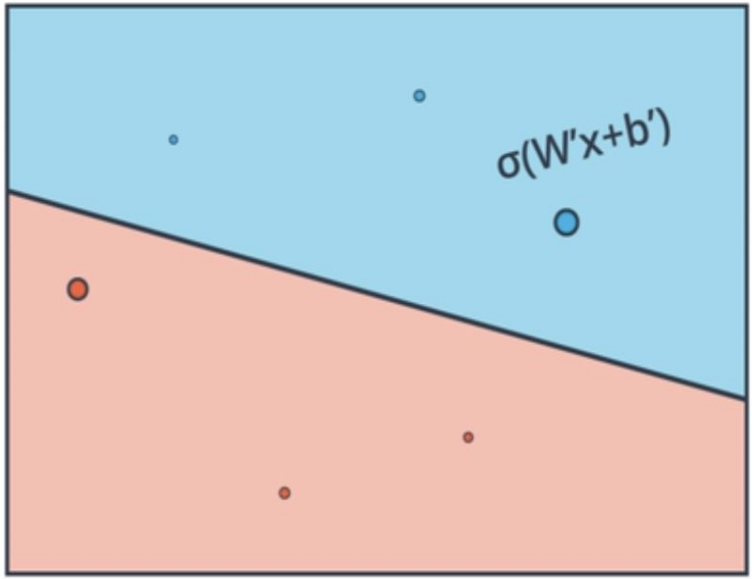
2. For every point $(x_1, \ldots, x_n)$:

   2.1. For i =1 . . . n

      2.1.1. Update $w_i' \longleftarrow w_i - \alpha \, \partial E/\partial w_i$

      2.1.2. Update $b' \longleftarrow b - \alpha \, \partial E/\partial b$

Gradient Descent Algorithm



1. Start with random weights:

   $w_1, \ldots, w_n, b$

2. For every point $(x_1, \ldots, x_n)$:

   2.1. For i =1 . . . n

      2.1.1. Update $w_i' \longleftarrow w_i - \alpha (\hat{y}-y)x_i$

      2.1.2. Update $b' \longleftarrow b - \alpha (\hat{y}-y)$

3. Repeat until error is small

Perceptron Algorithm!!!

Perceptron vs Gradient Descent

GRADIENT DESCENT ALGORITHM:

Change
w_i to $w_i + \alpha(y-\hat{y})x_i$

PERCEPTRON ALGORITHM:

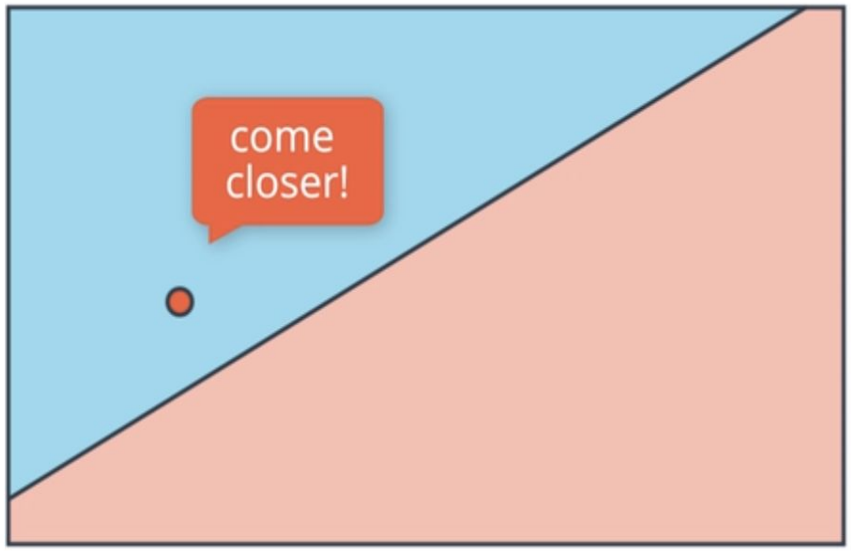
If x is missclassified:

Change w_i to $\begin{cases} w_i + \alpha x_i \text{ if positive} \\ w_i - \alpha x_i \text{ if negative} \end{cases}$

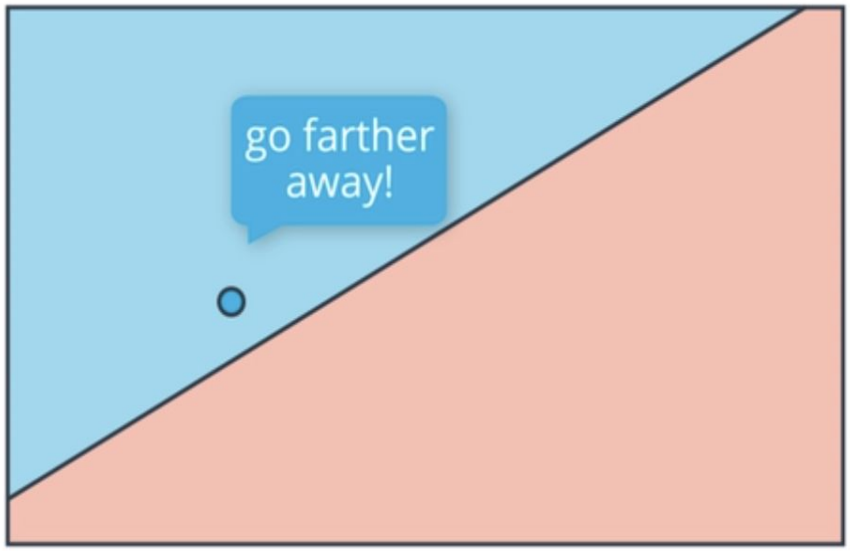
If correctly classified: $y-\hat{y}=0$

If missclassified: $\begin{cases} y-\hat{y} = 1 \text{ if positive} \\ y-\hat{y} = -1 \text{ if negative} \end{cases}$

Gradient Descent Algorithm

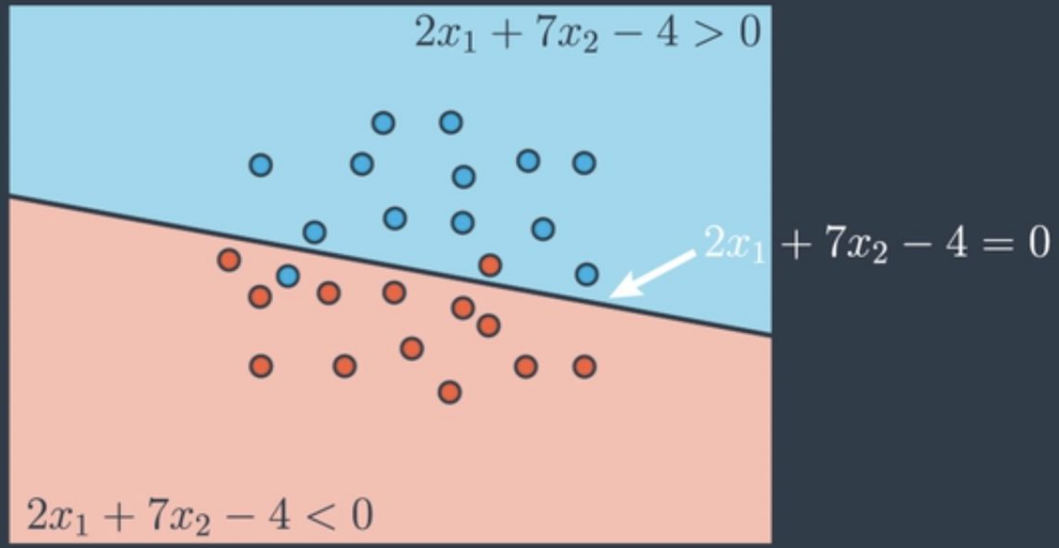


INCORRECTLY CLASSIFIED

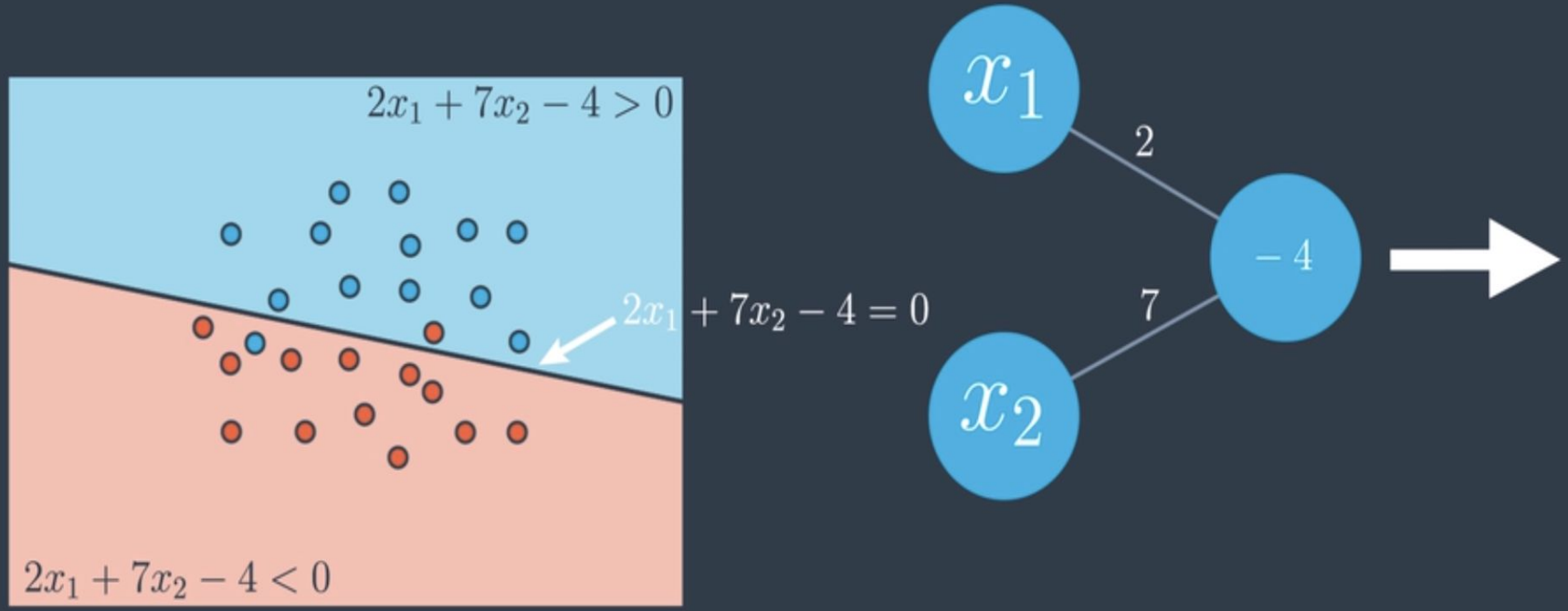


CORRECTLY CLASSIFIED

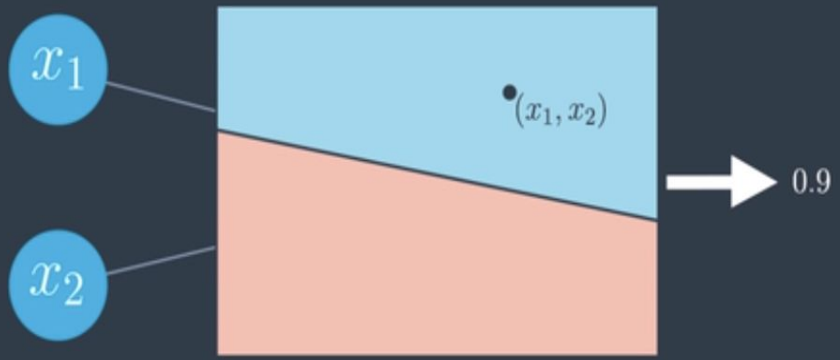
Perceptron

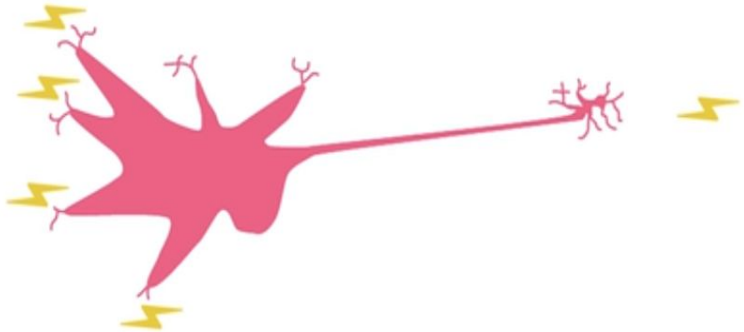


Perceptron



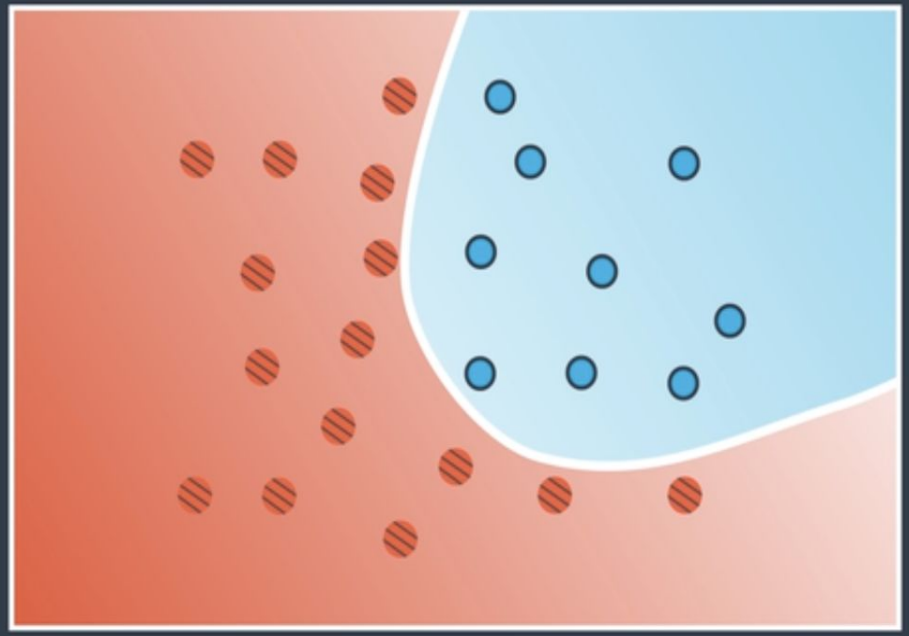
Perceptron



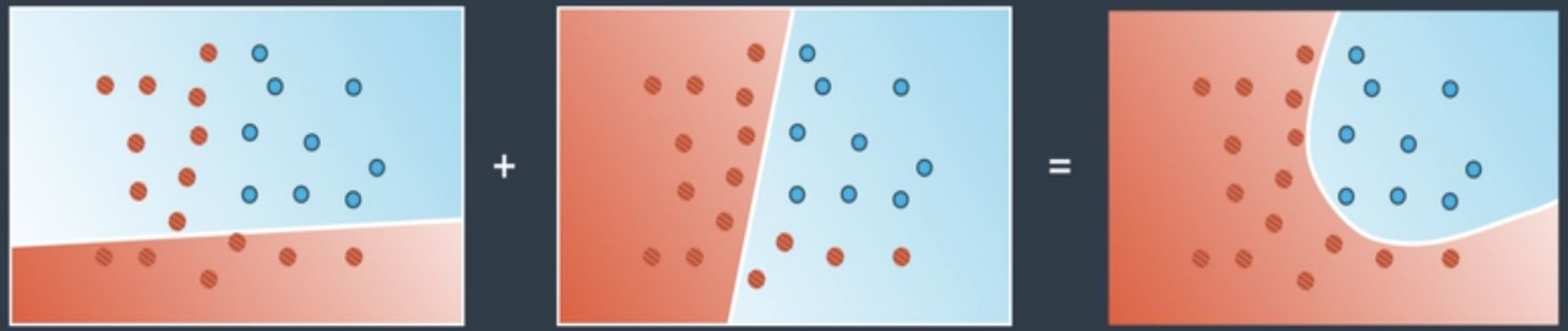


Acceptance at a University

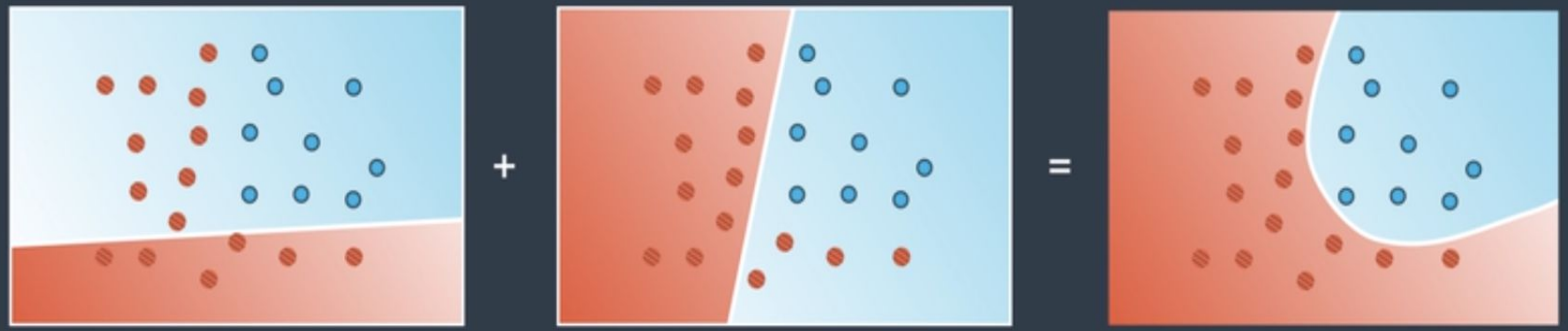
Non-Linear Regions



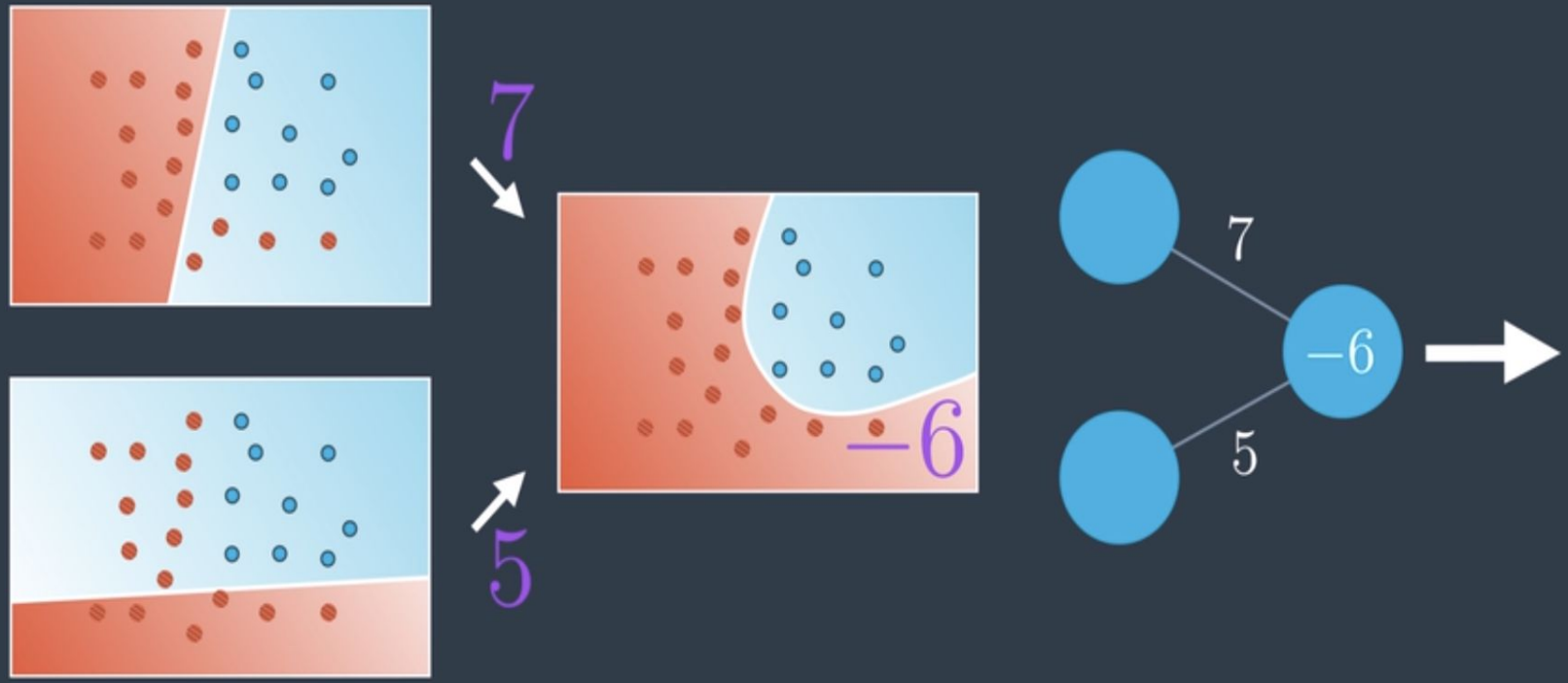
Combining Regions



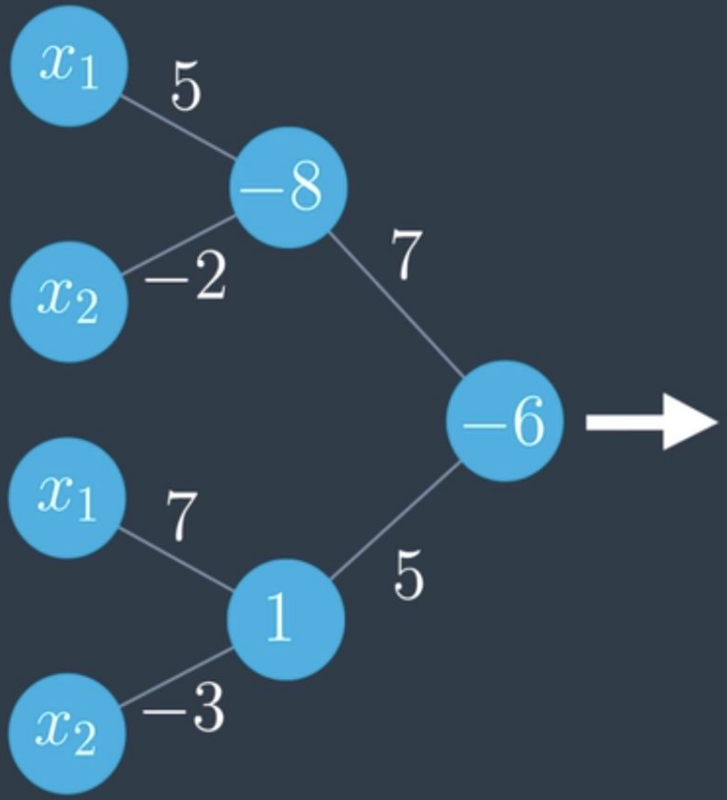
Combining Regions

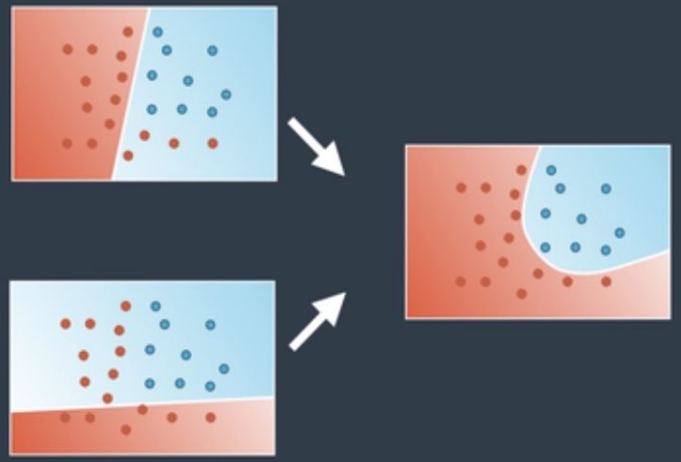


Neural Network

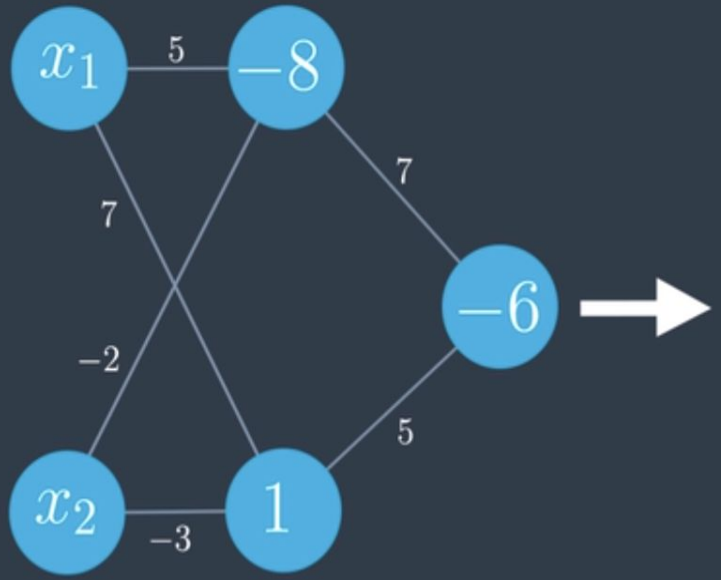


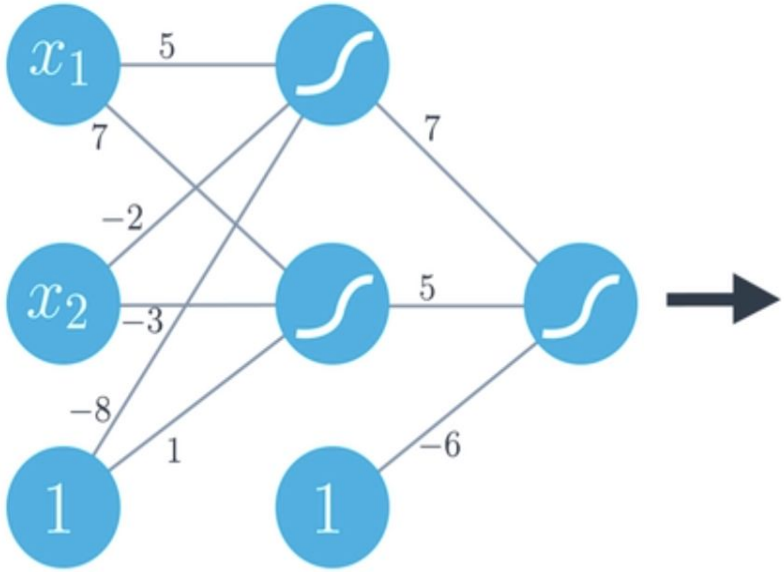
Neural Network



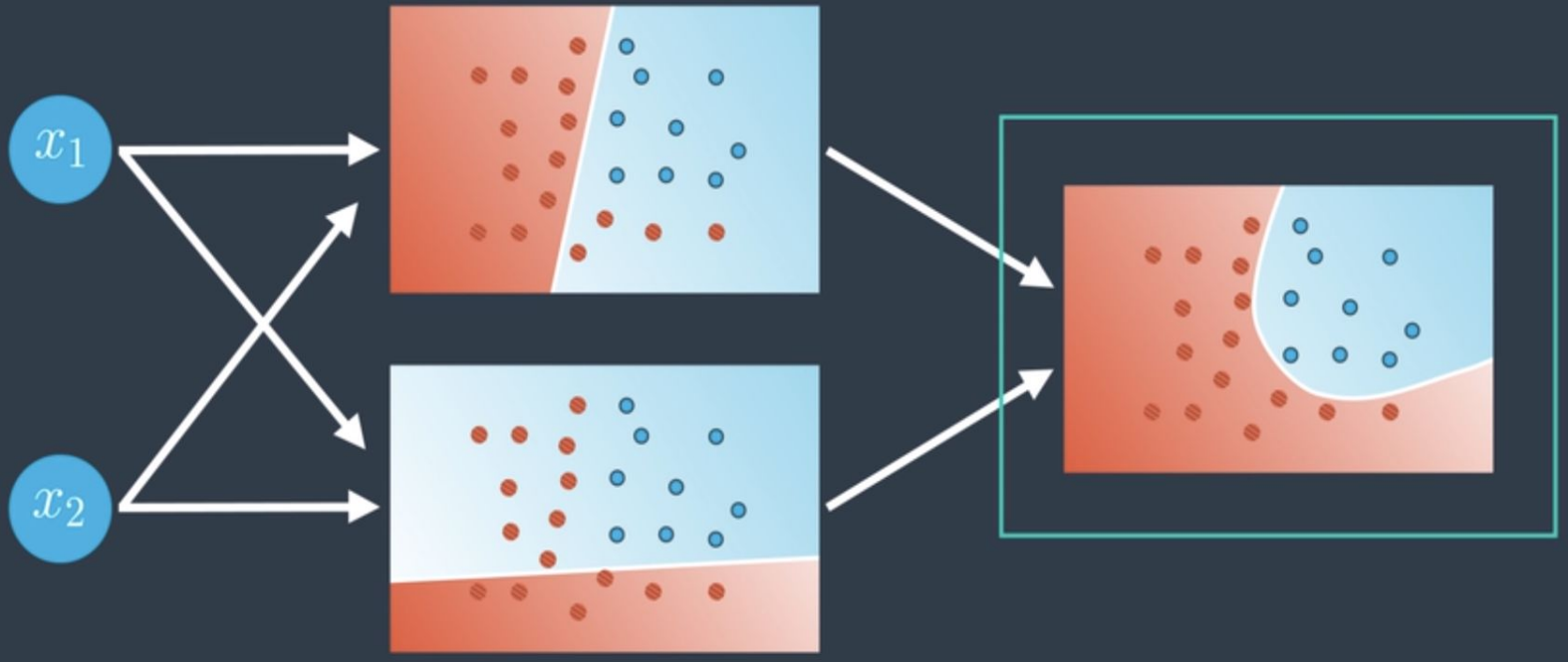


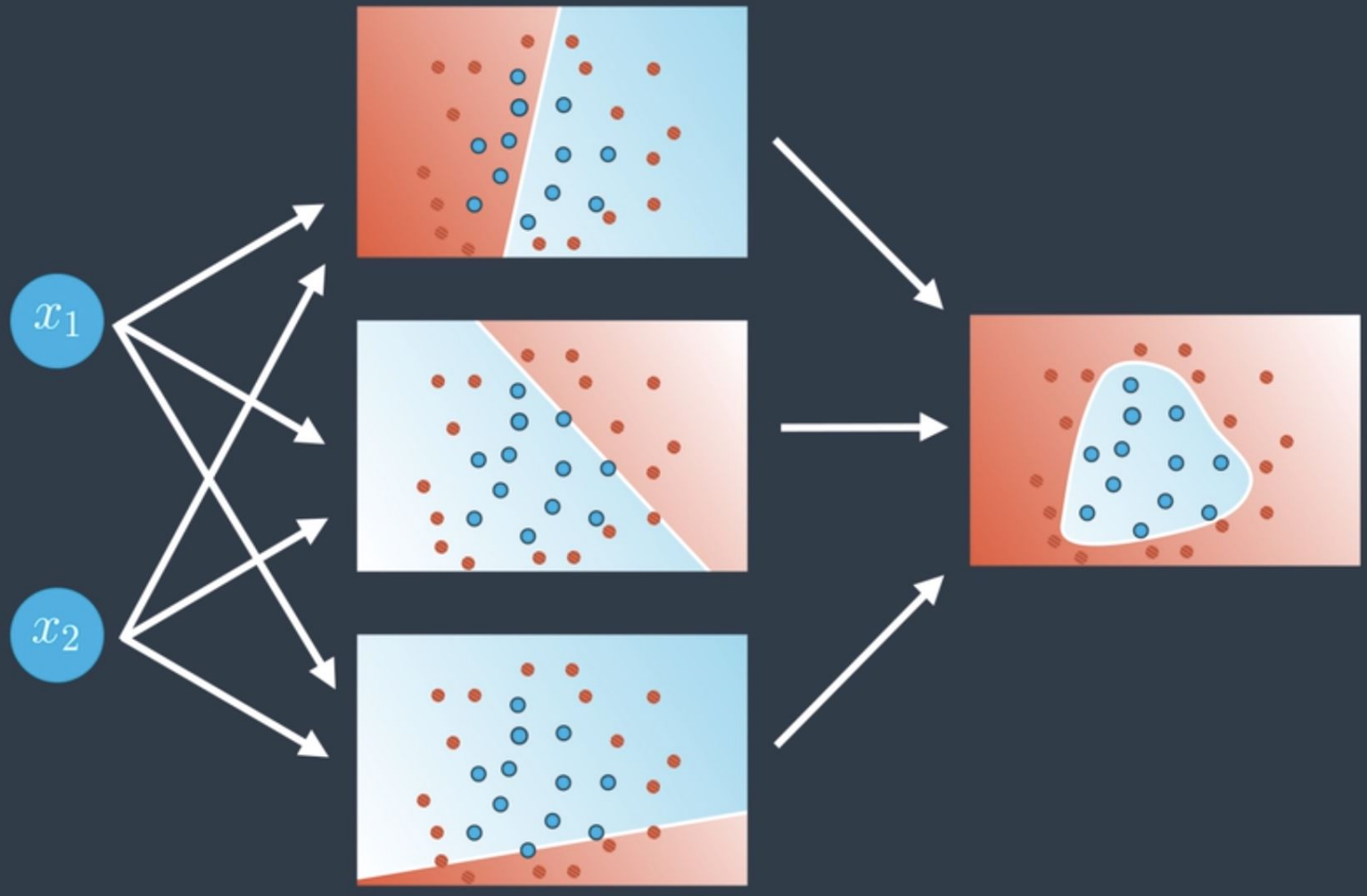
Neural Network



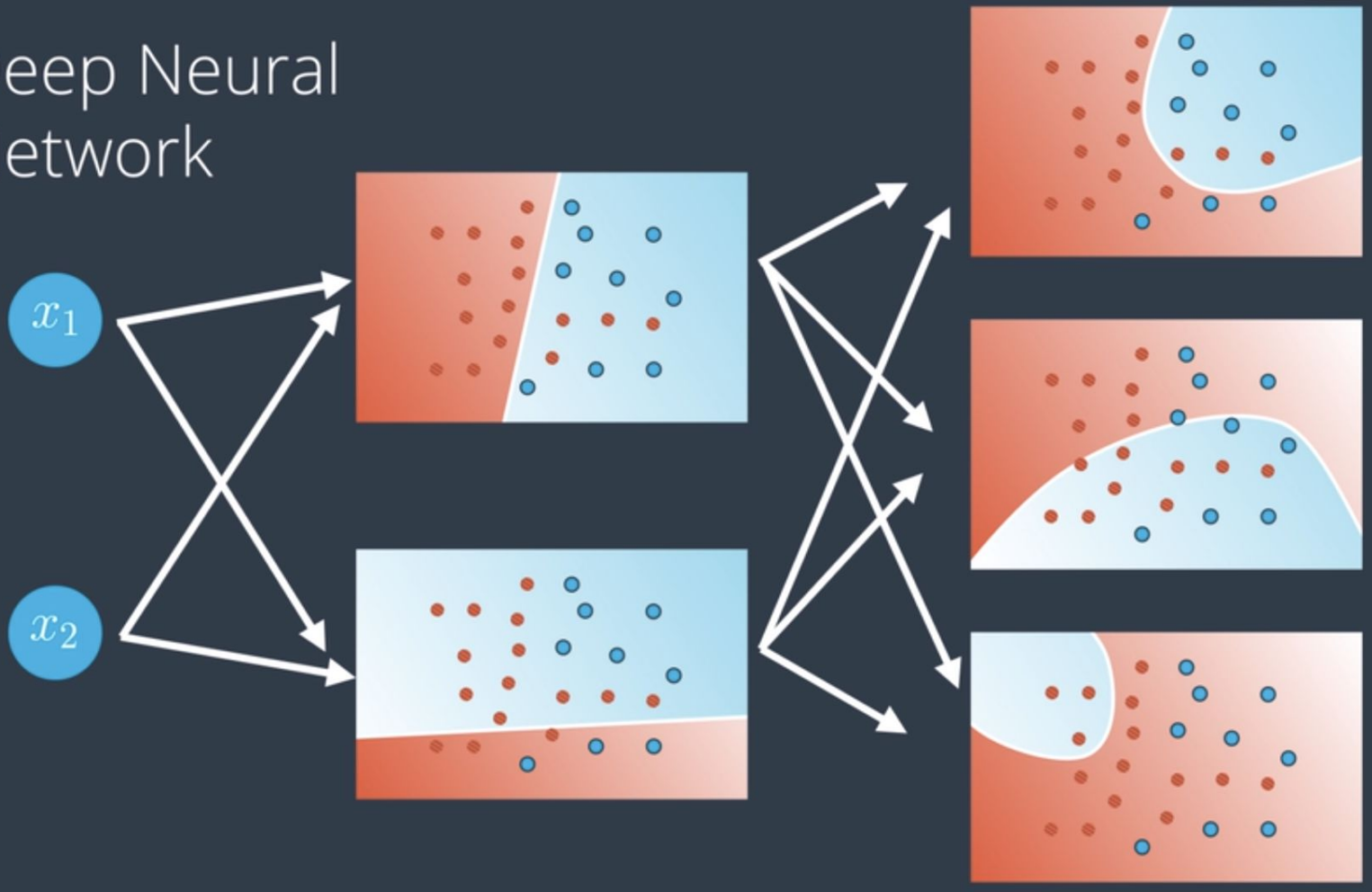


Neural Network

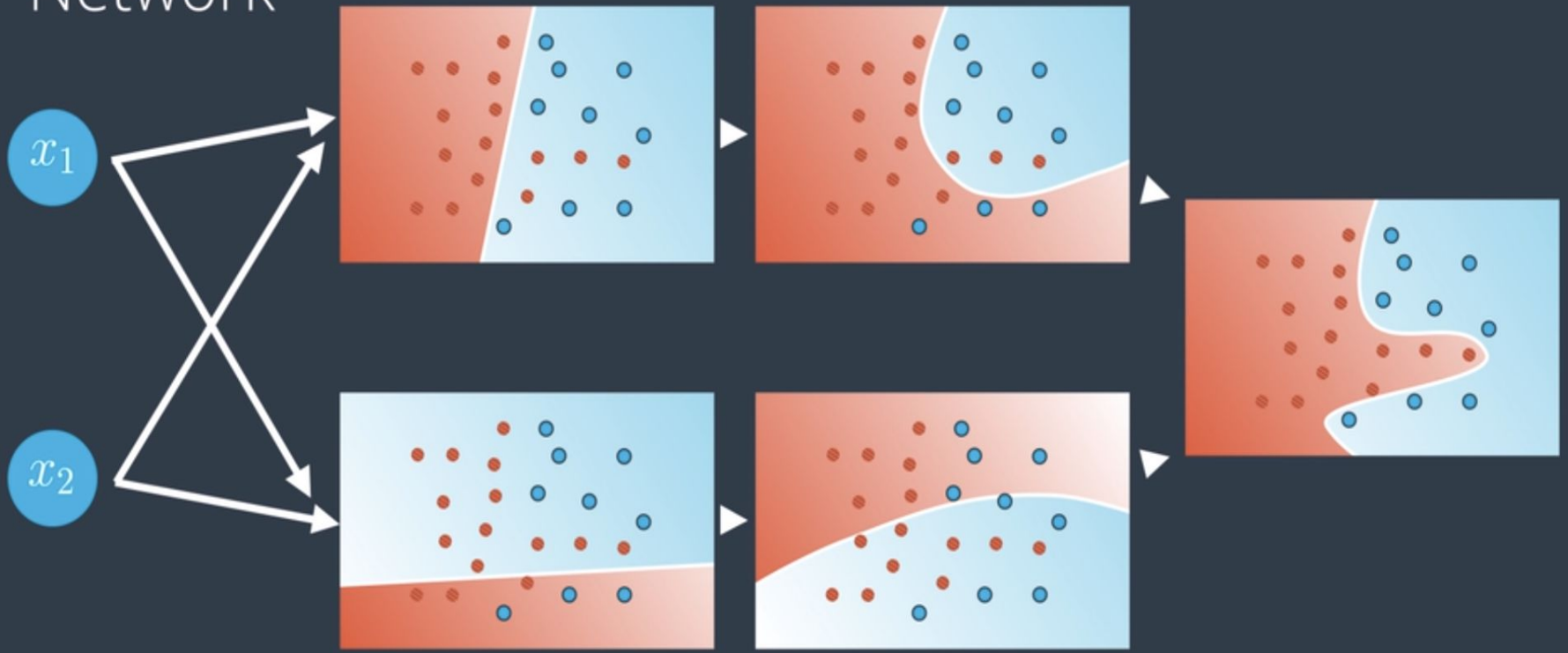



x_1
x_2

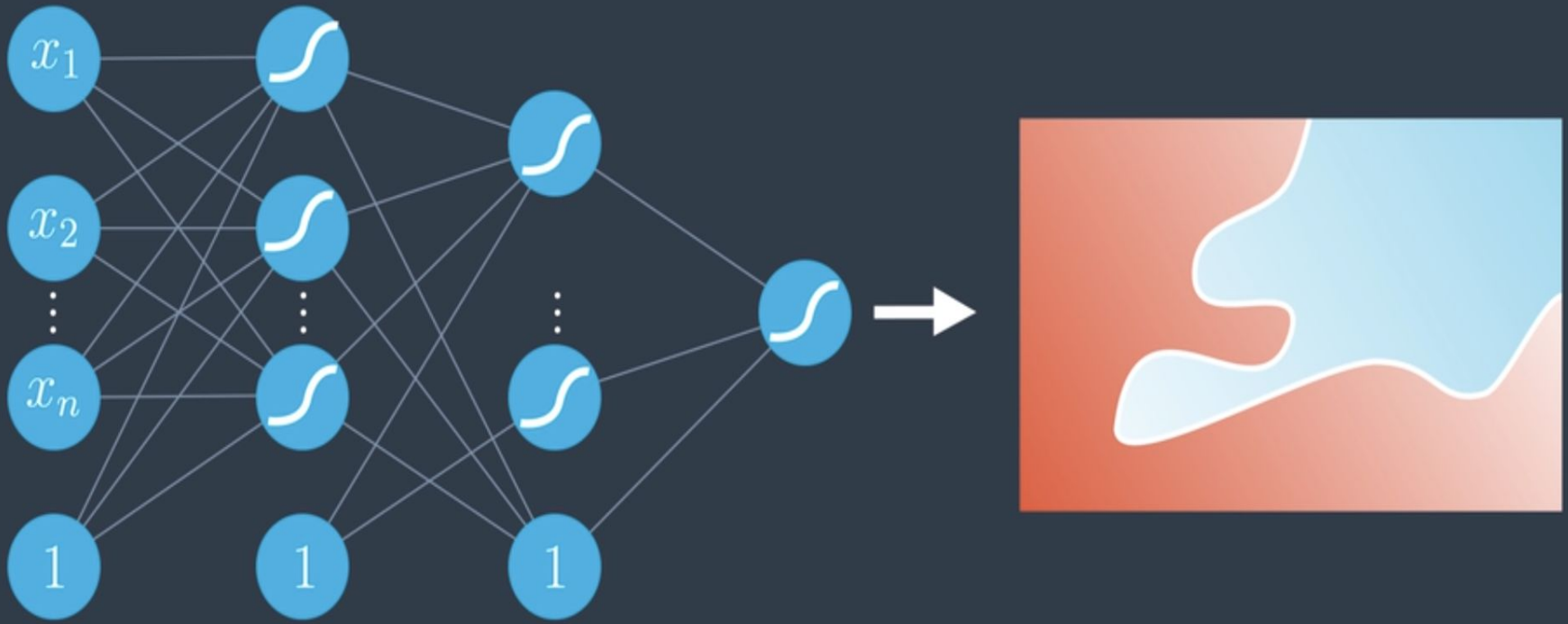
Deep Neural Network



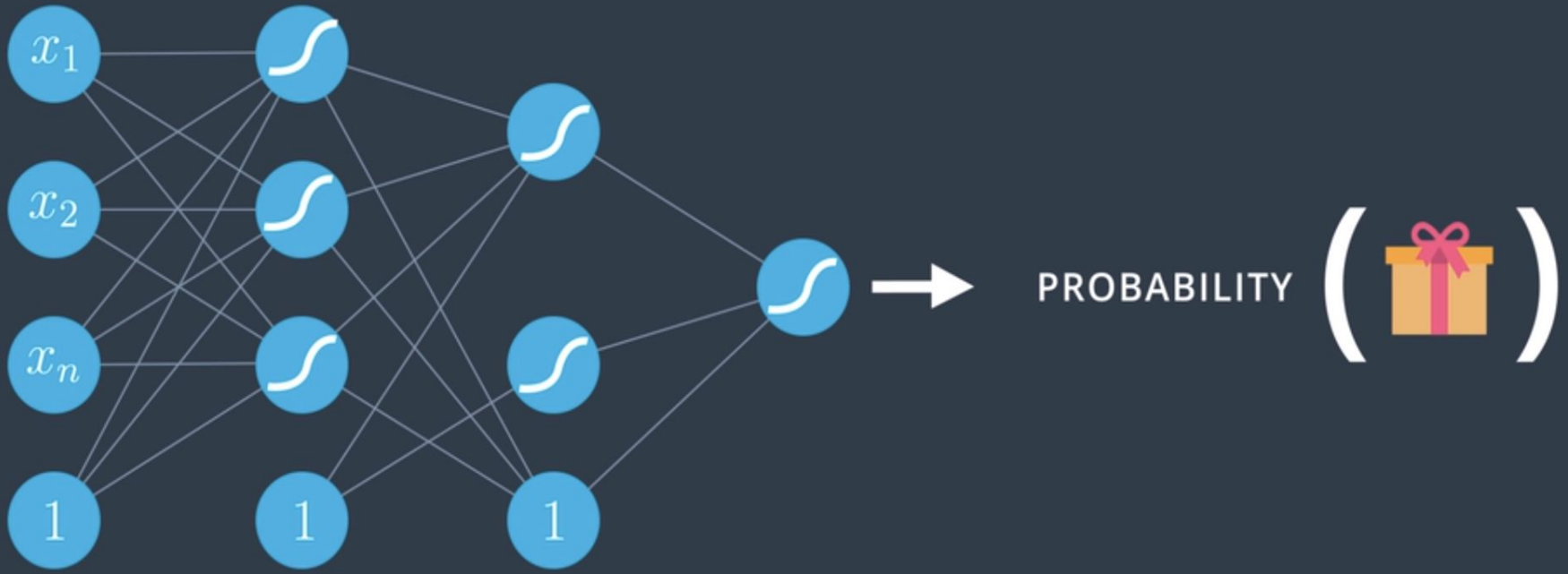
Deep Neural Network



Neural Network



Binary Classification



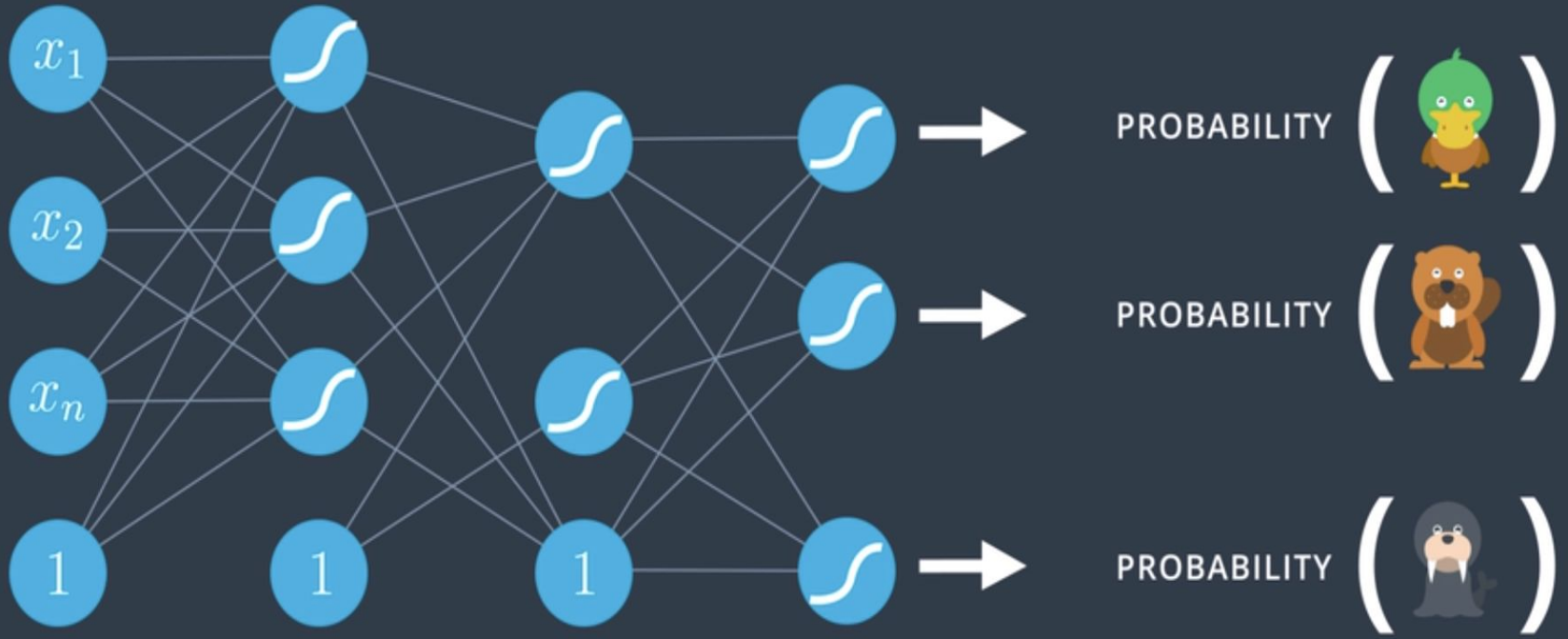
Multi-Class Classification

PROBABILITY ()

PROBABILITY ()

PROBABILITY ()

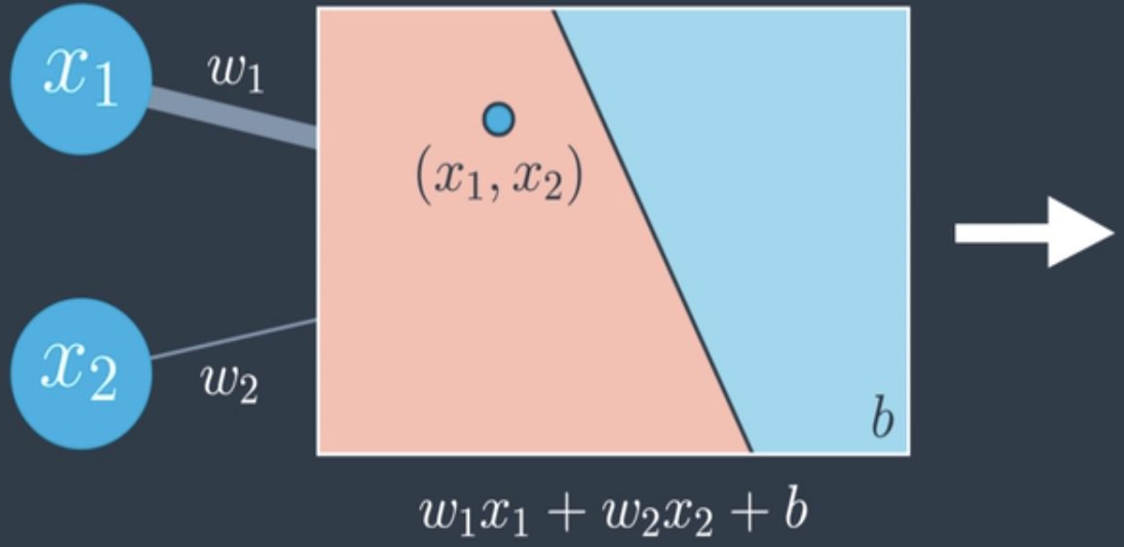
Neural Network



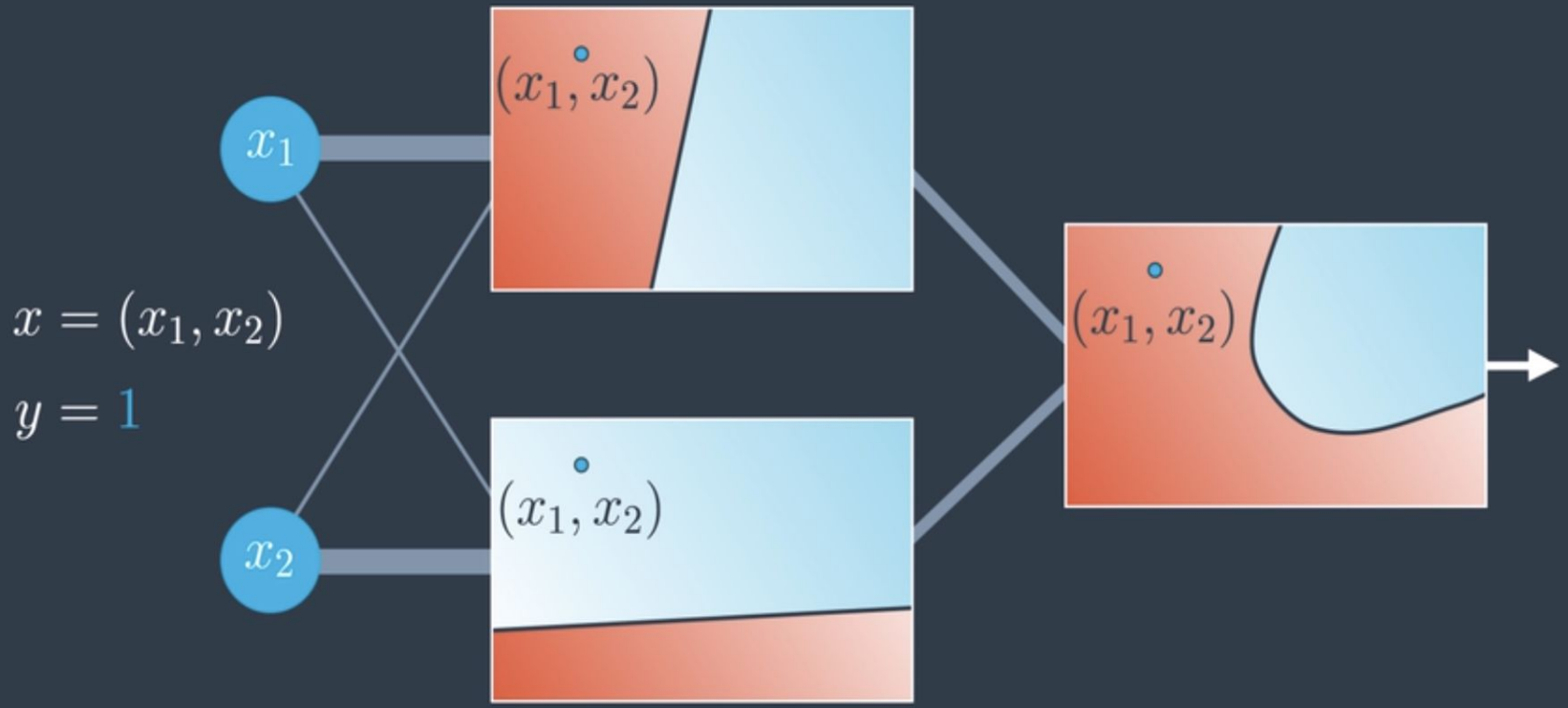
Perceptron

$x = (x_1, x_2)$

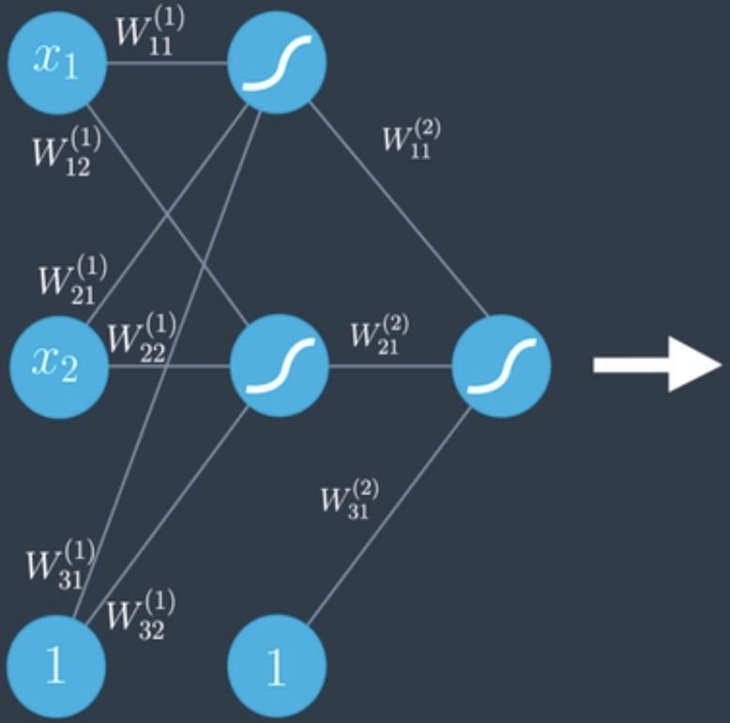
$y = 1$



Neural Network



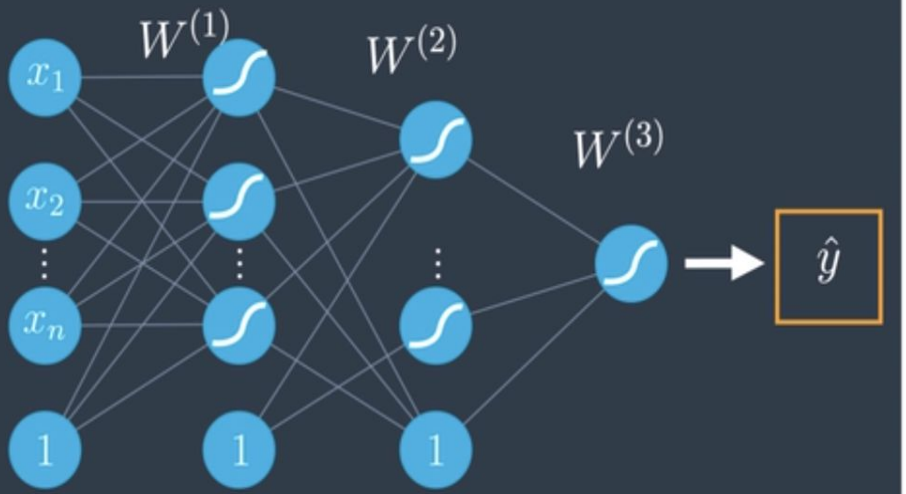
Feedforward



$$\hat{y} = \sigma \begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix} \sigma \begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

$$\hat{y} = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

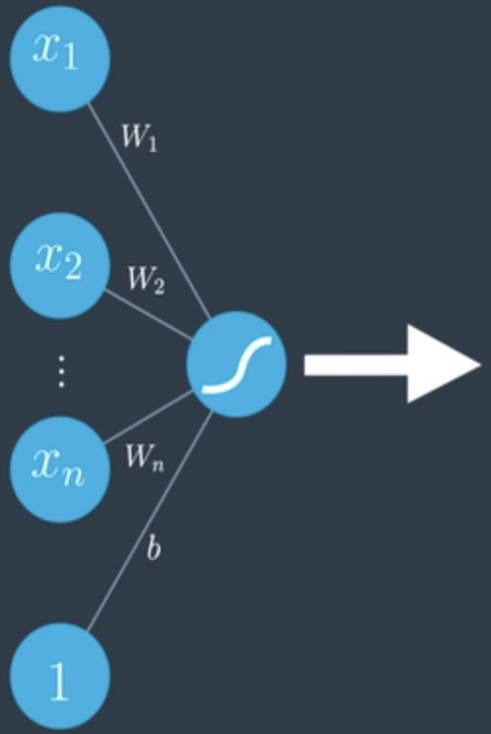
Multi-layer Perceptron



PREDICTION

$$\hat{y} = \sigma \circ W^{(3)} \circ \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

Perceptron

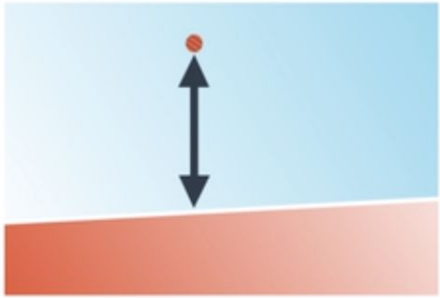


PREDICTION

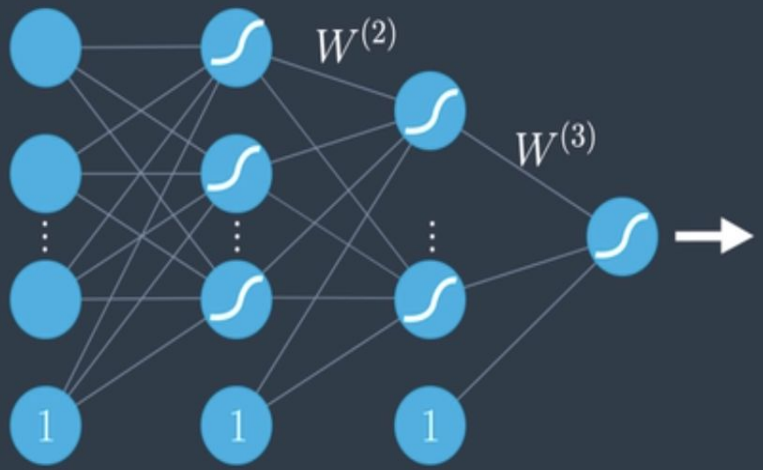
$$\hat{y} = \sigma(Wx + b)$$

ERROR FUNCTION

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$$



Multi-layer Perceptron

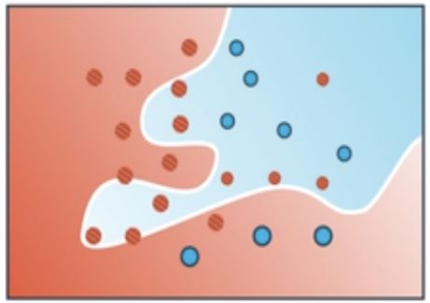


PREDICTION

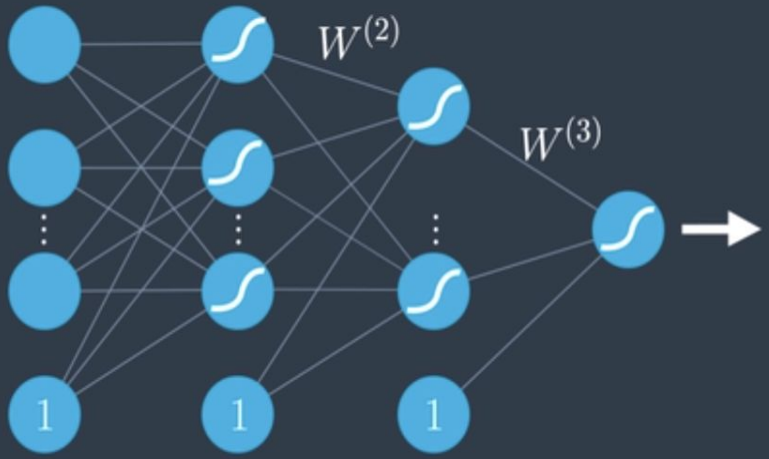
$$\hat{y} = \sigma \circ W^{(3)} \circ \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

ERROR FUNCTION

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$$



Multi-layer Perceptron

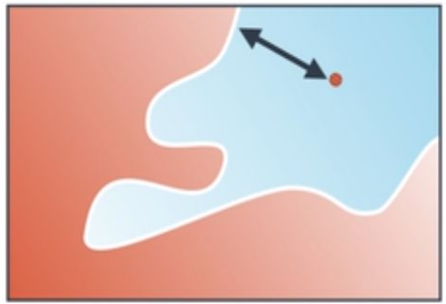


PREDICTION

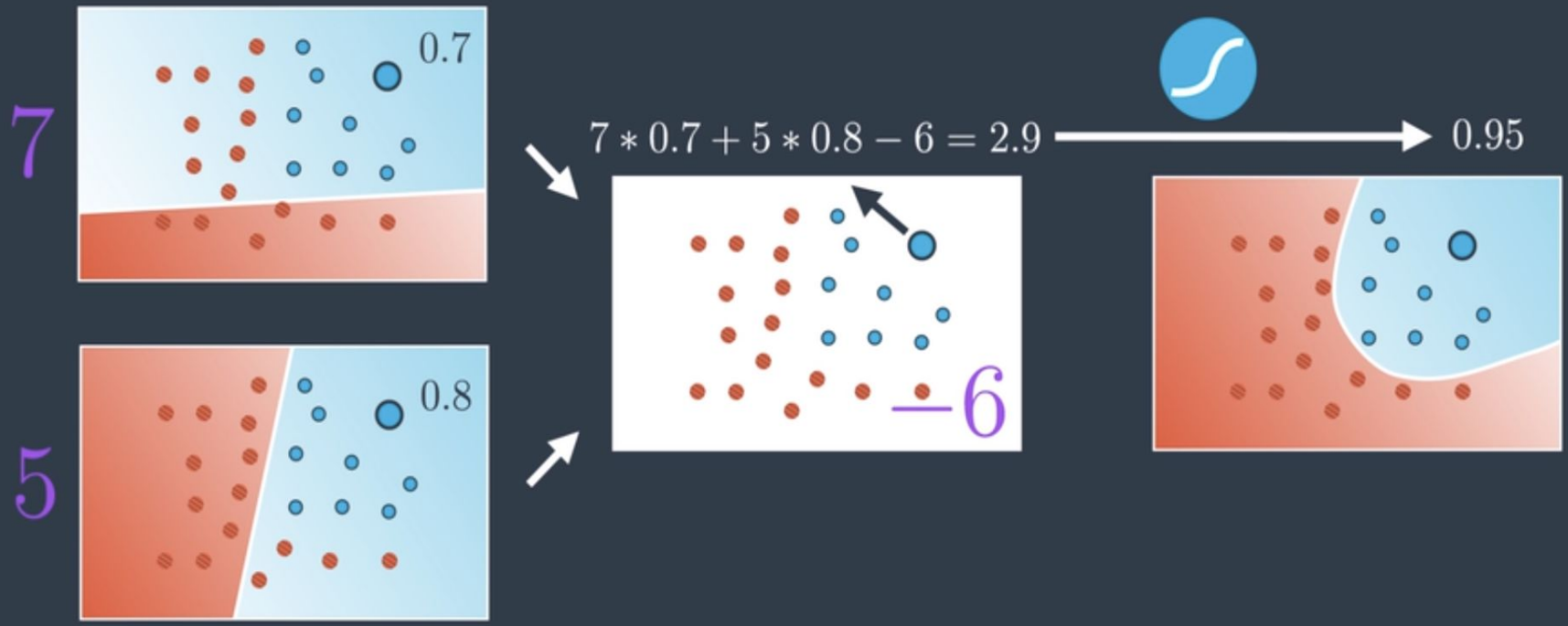
$$\hat{y} = \sigma \circ W^{(3)} \circ \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

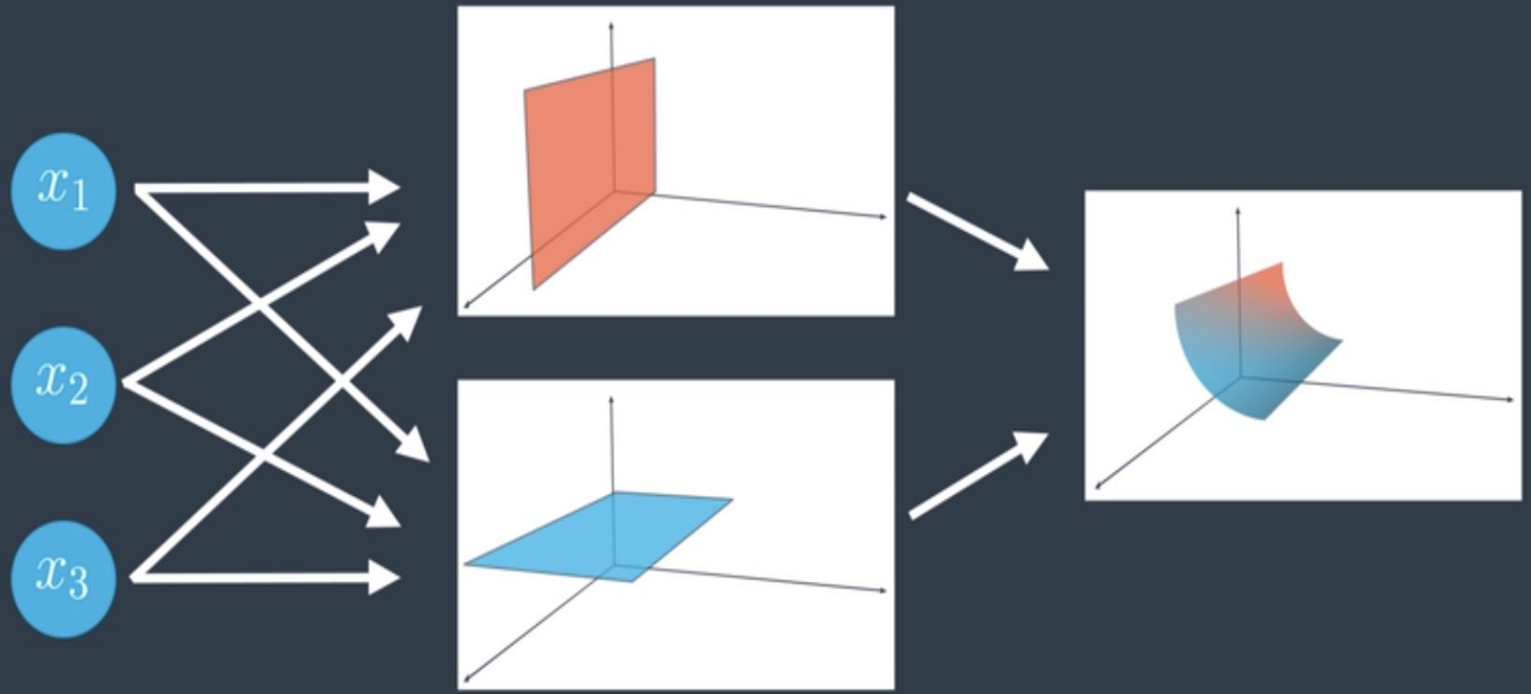
ERROR FUNCTION

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$$

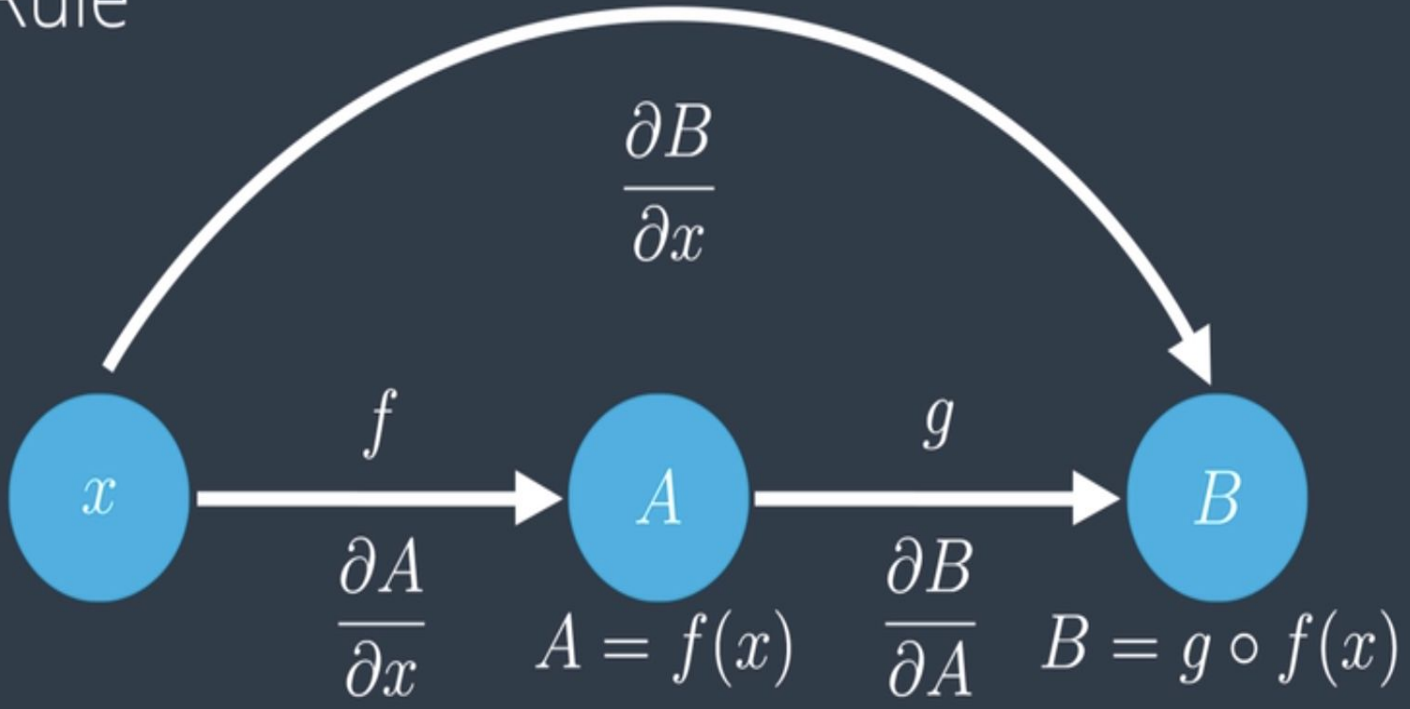


Neural Network

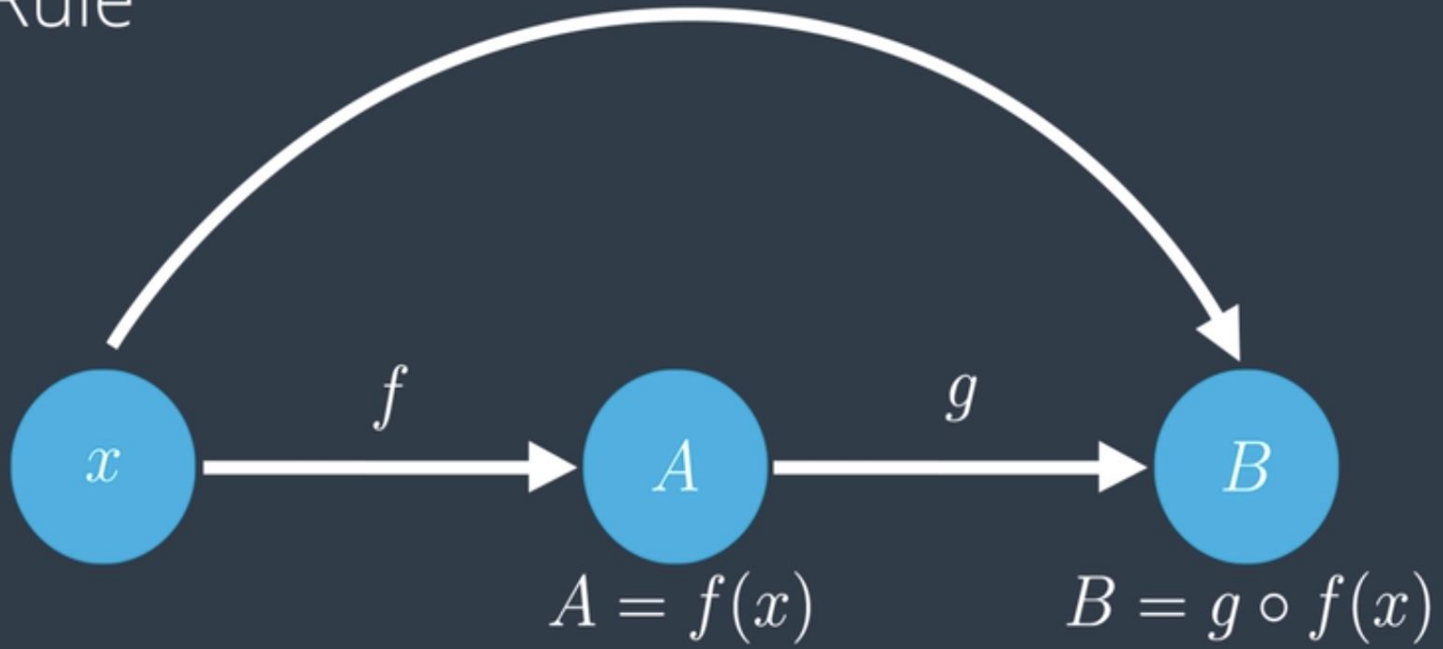



x_1
x_2
x_3

Chain Rule

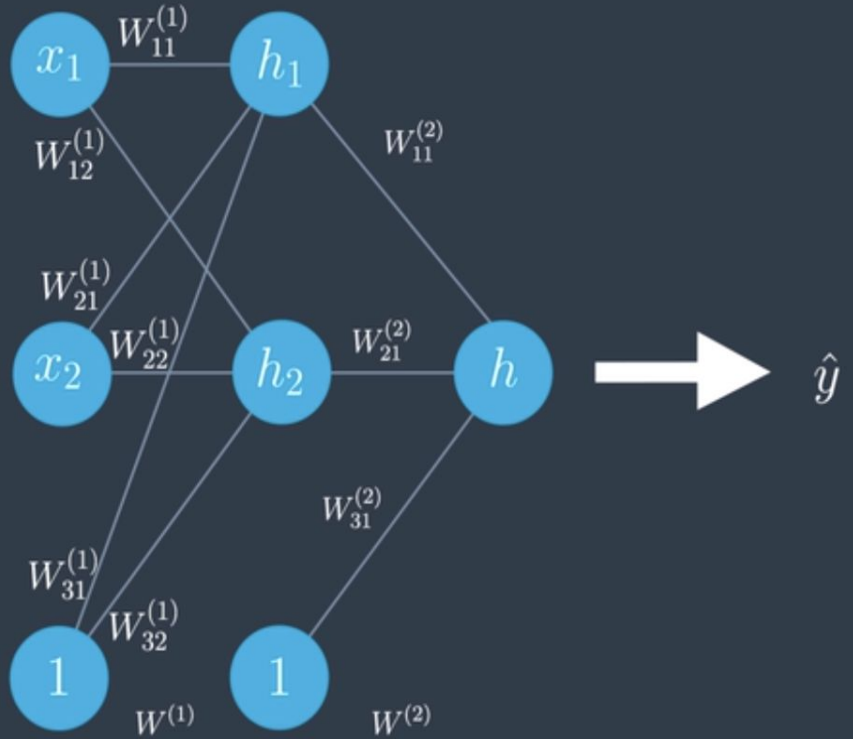

$\frac{\partial B}{\partial x}$
x
f
A
g
B
$\frac{\partial A}{\partial x}$
$A = f(x)$
$\frac{\partial B}{\partial A}$
$B = g \circ f(x)$

Chain Rule



$$\frac{\partial B}{\partial x} = \frac{\partial B}{\partial A}\frac{\partial A}{\partial x}$$

Feedforward



$$h_1 = W_{11}^{(1)} x_1 + W_{21}^{(1)} x_2 + W_{31}^{(1)}$$

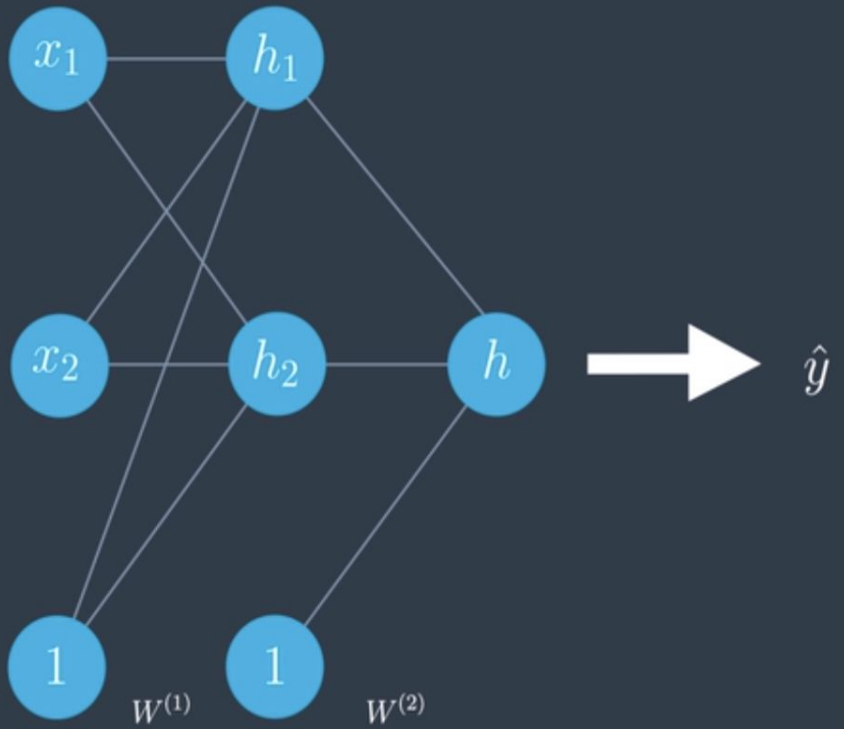
$$h_2 = W_{12}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{32}^{(1)}$$

$$h = W_{11}^{(2)} \sigma(h_1) + W_{21}^{(2)} \sigma(h_2) + W_{31}^{(2)}$$

$$\hat{y} = \sigma(h)$$

$$\hat{y} = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

Backpropagation



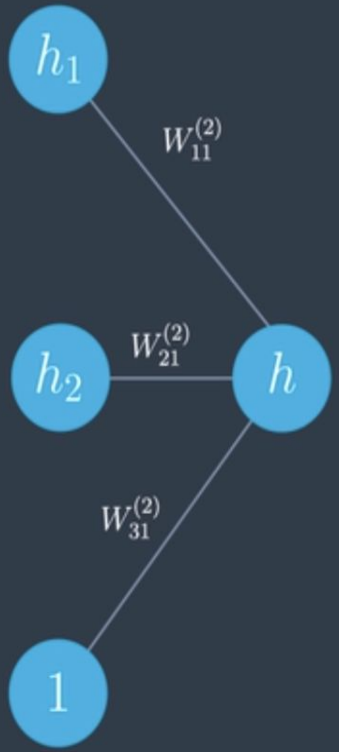
$$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$$

$$E(W) = E(W_{11}^{(1)}, W_{12}^{(1)}, ..., W_{31}^{(2)})$$

$$\bigtriangledown E = (\frac{\partial E}{\partial W_{11}^{(1)}}, ..., \frac{\partial E}{\partial W_{31}^{(2)}})$$

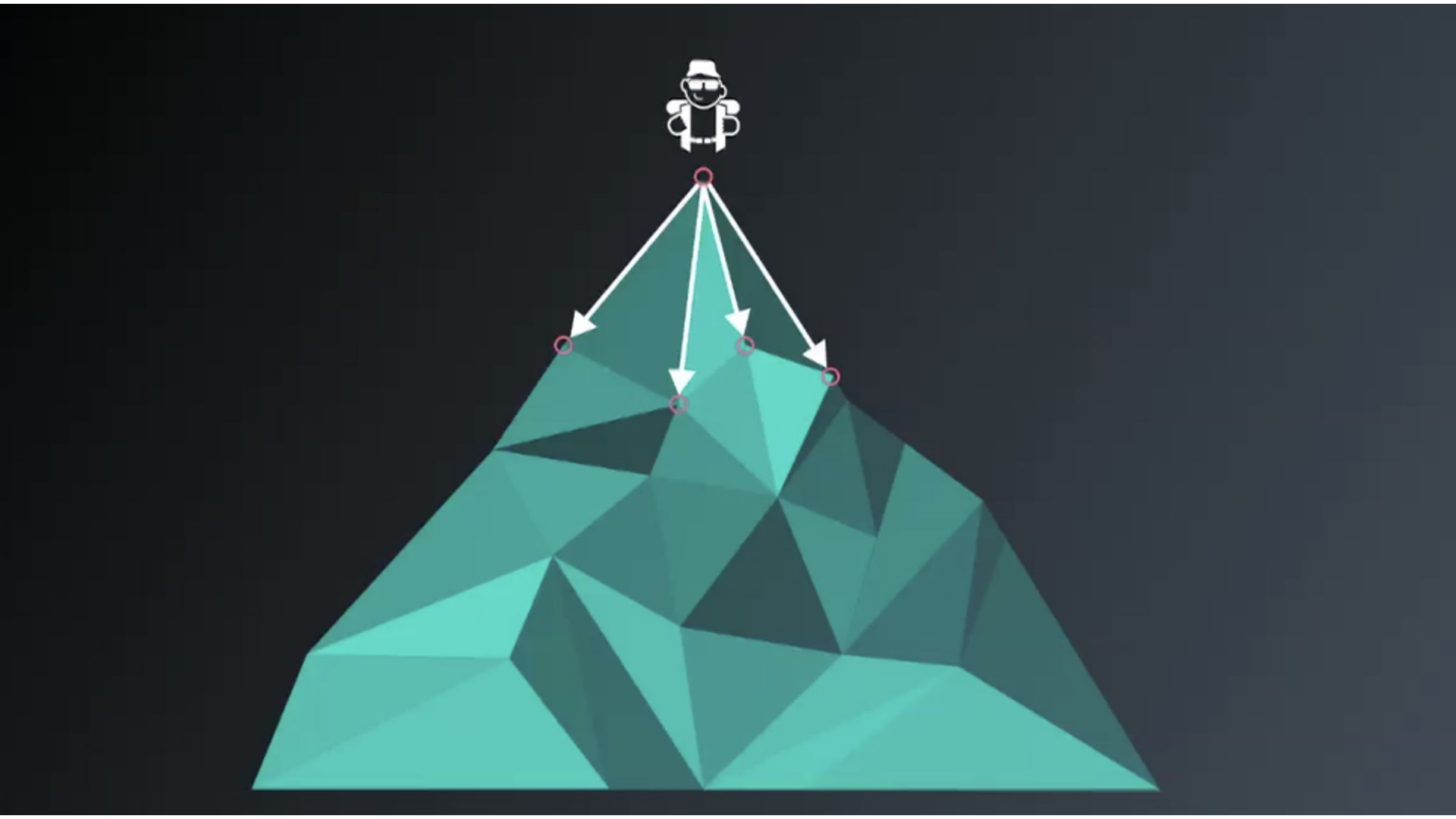
$$\frac{\partial E}{\partial W_{11}^{(1)}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{\partial h_1} \frac{\partial h_1}{\partial W_{11}^{(1)}}$$

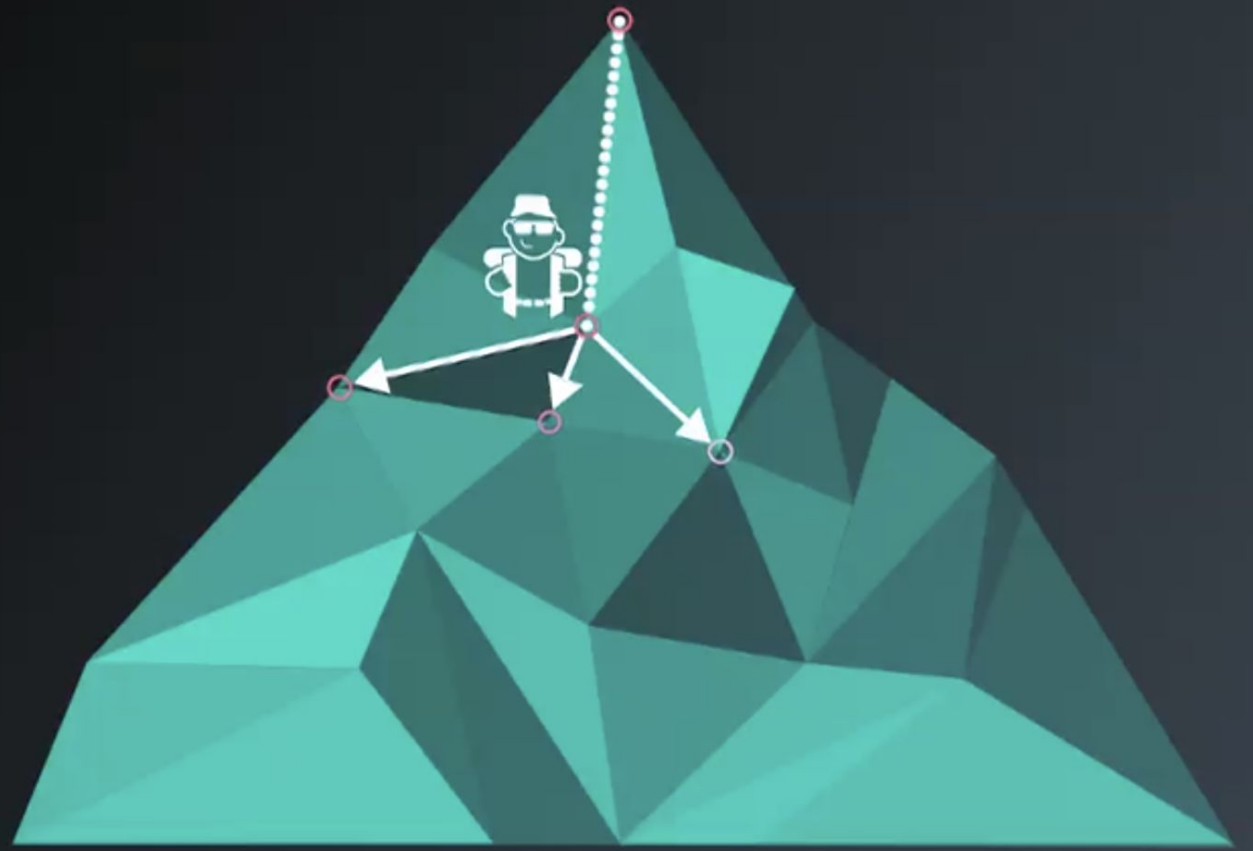
Backpropagation

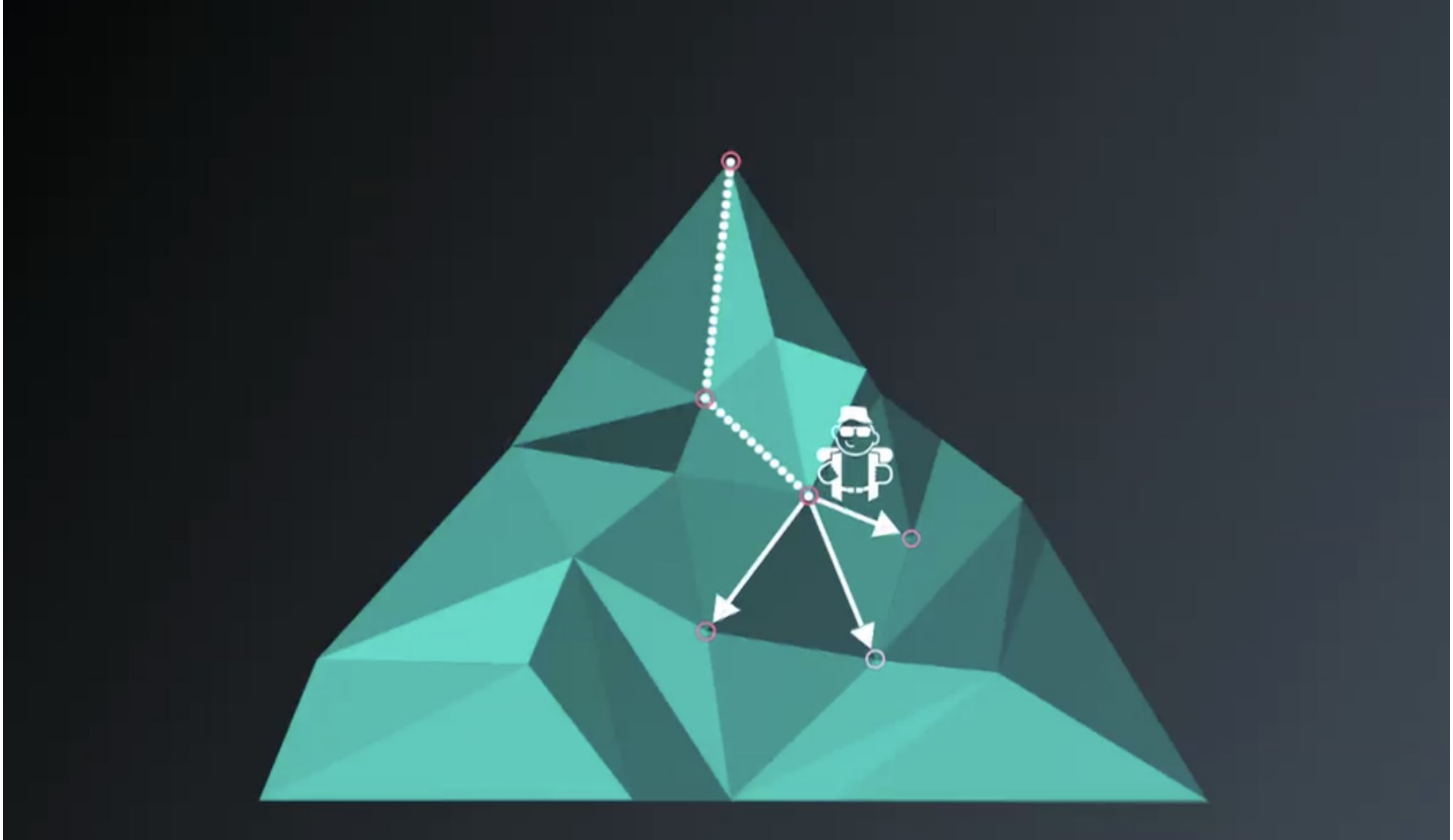


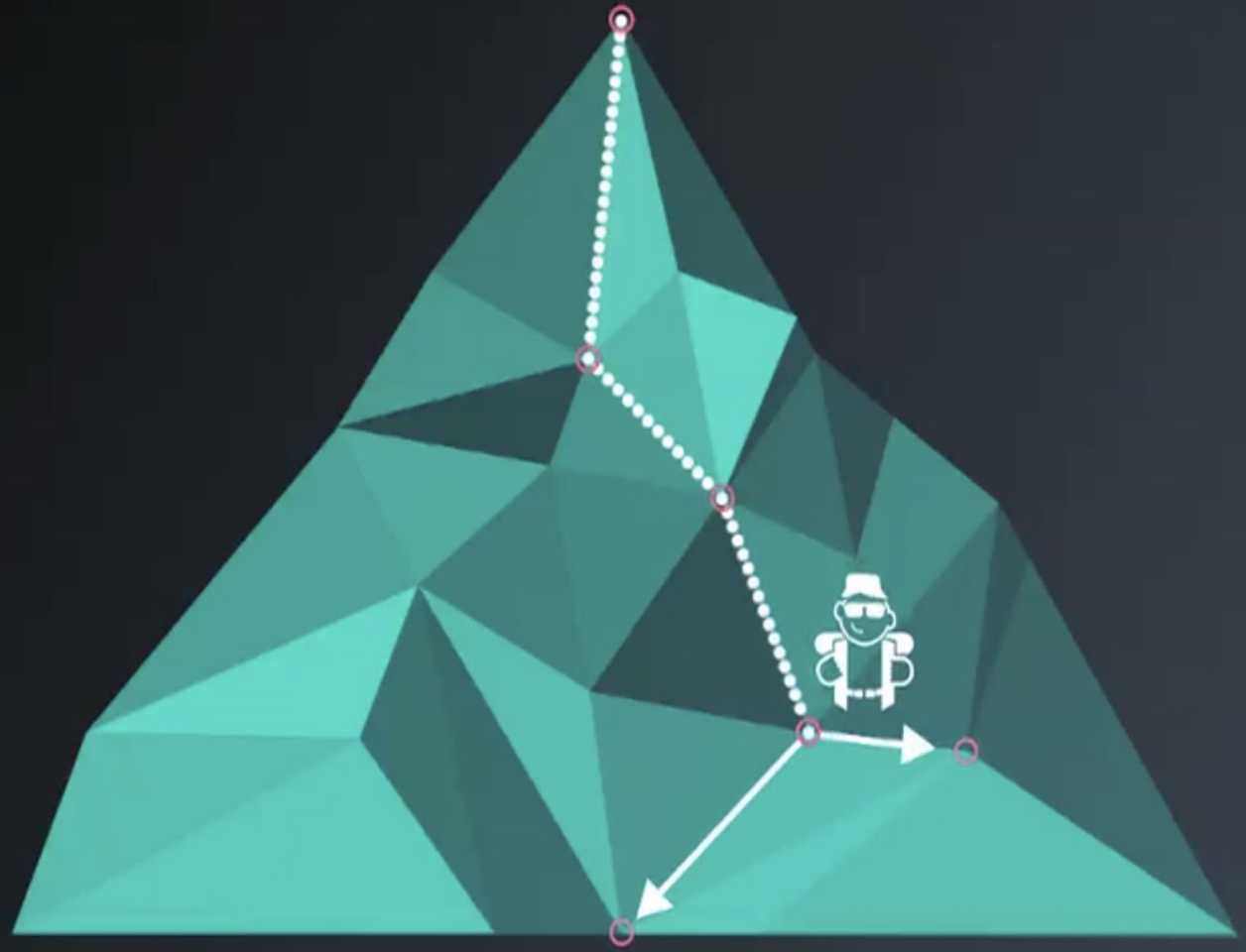
$$h = W_{11}^{(2)}\sigma(h_1) + W_{21}^{(2)}\sigma(h_2) + W_{31}^{(2)}$$

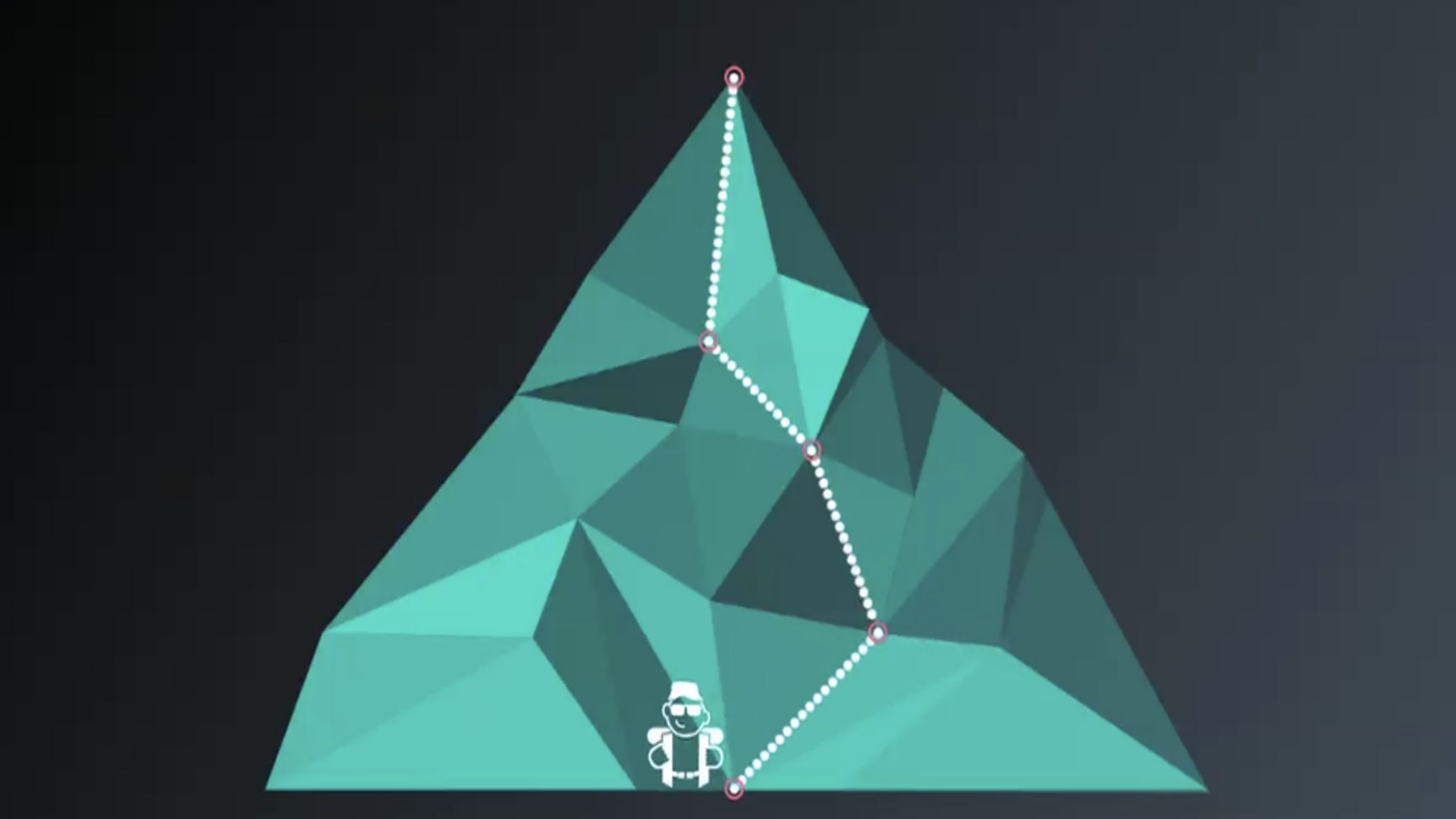
$$\frac{\partial h}{\partial h_1} = W_{11}^{(2)}\sigma(h_1)[1 - \sigma(h_1)]$$

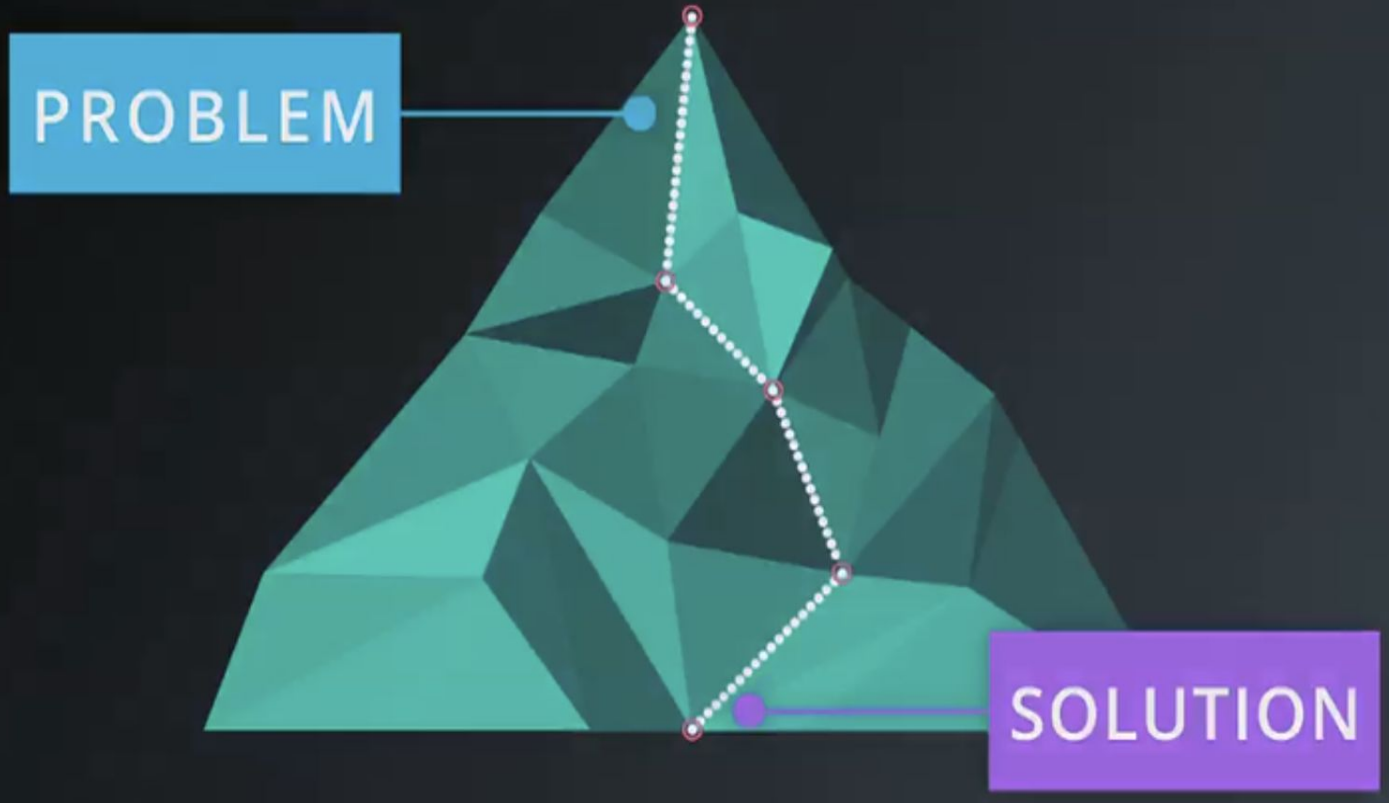


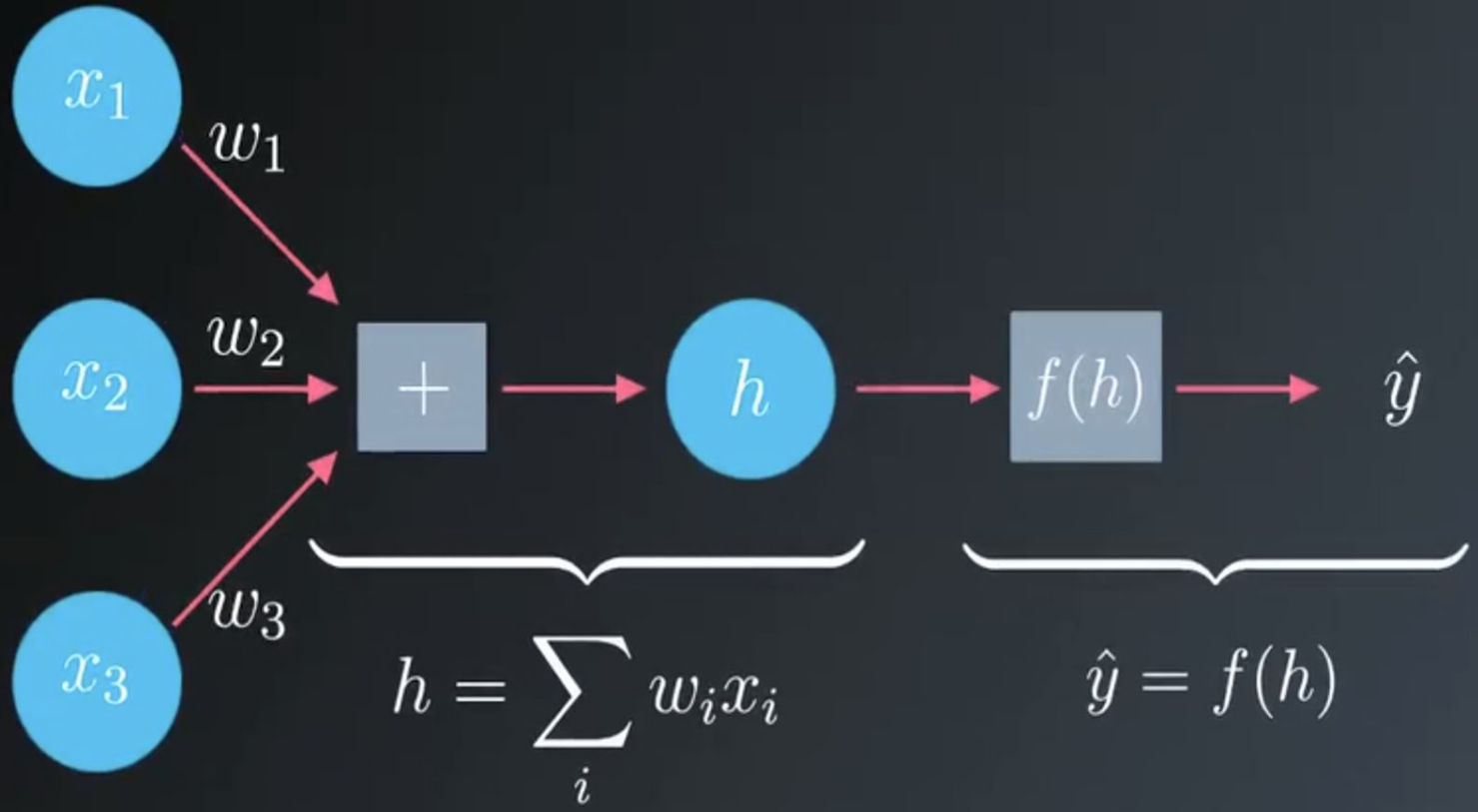









PROBLEM
SOLUTION


x_1
w_1
x_2
w_2
x_3
w_3
+
h
$f(h)$
$\hat{y}$
$h = \sum_i w_i x_i$
$\hat{y} = f(h)$

- THE SUM OF THE SQUARED ERRORS (SSE)

$$E = \frac{1}{2} \sum_{\mu} (y^{\mu} - \hat{y}^{\mu})^2$$

$$E = \frac{1}{2}\sum_{\mu}(y^{\mu} - \hat{y}^{\mu})^2$$

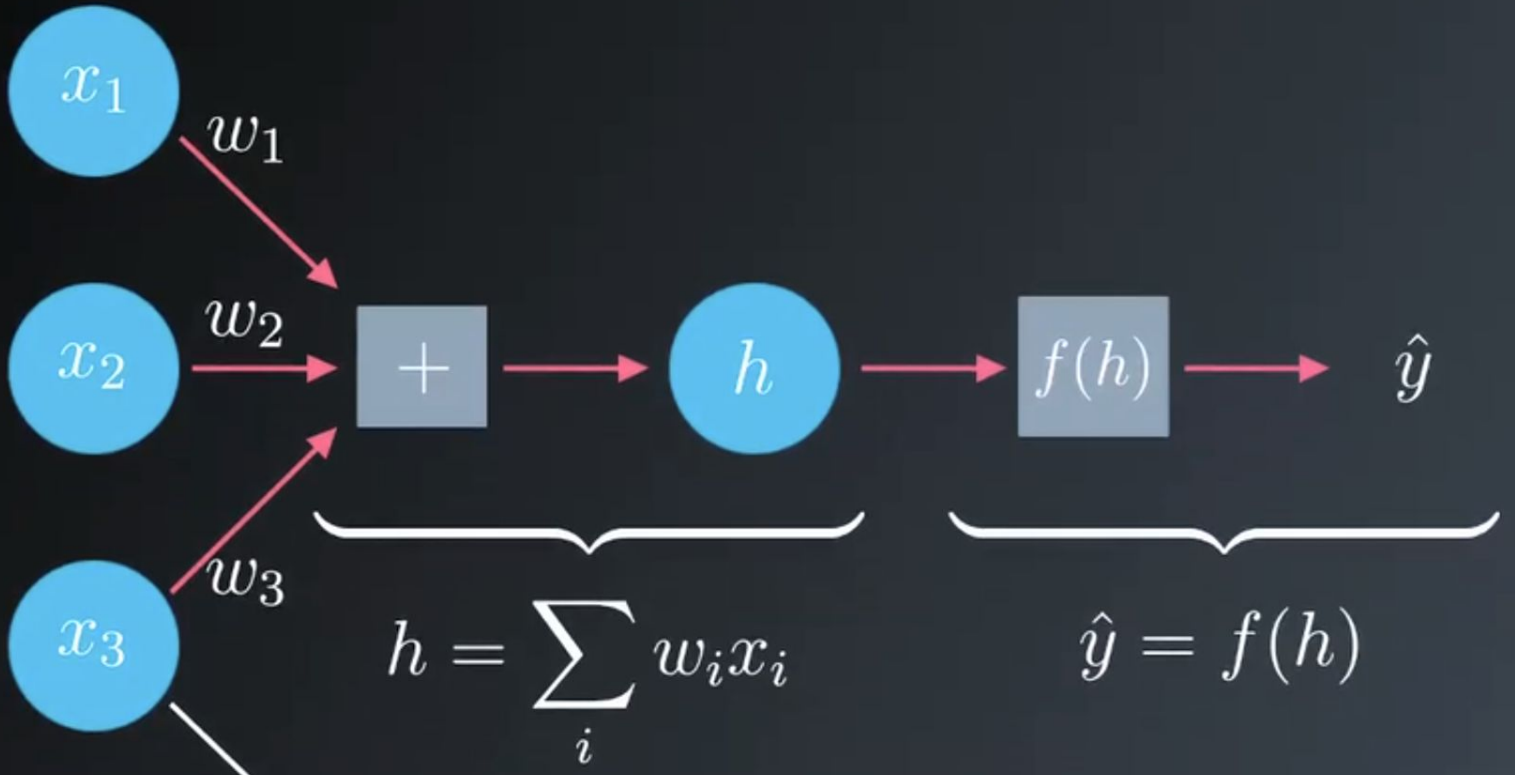
$$= \frac{1}{2}\sum_{\mu}(y^{\mu} - f(\sum_{i} w_i x_i^{\mu}))^2$$

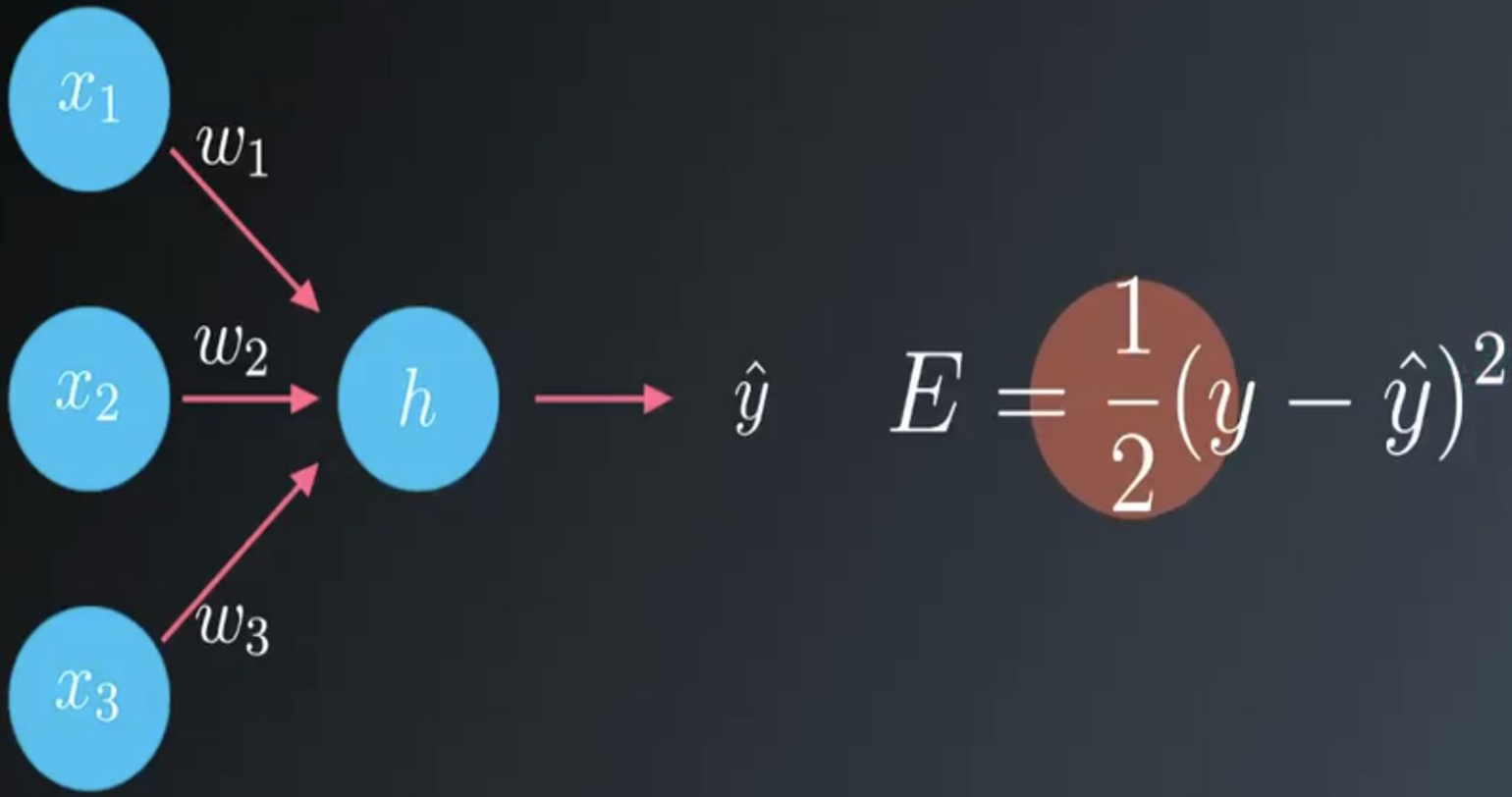
$$E = \frac{1}{2}\sum_{\mu}(y^{\mu} - \hat{y}^{\mu})^2$$

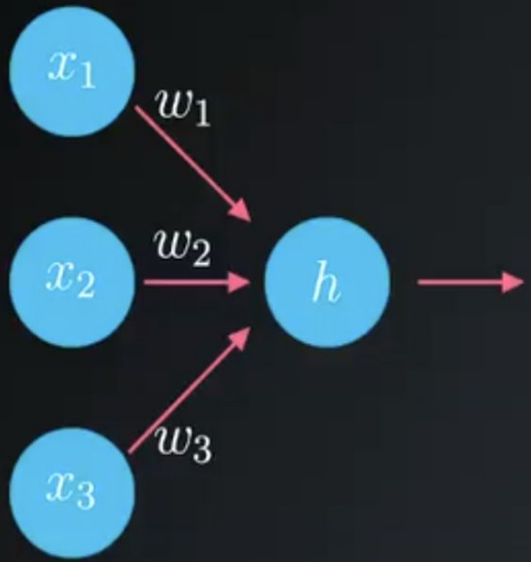
$$= \frac{1}{2}\sum_{\mu}(y^{\mu} - f(\sum_{i} w_i x_i^{\mu}))^2$$

DATA RECORDS

$$x = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 \\ x_1^2 & x_2^2 & x_3^2 \\ x_1^3 & x_2^3 & x_3^3 \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{bmatrix} \qquad y = \begin{bmatrix} y^1 \\ y^2 \\ y^3 \\ \dots \\ \dots \end{bmatrix}$$


x_1
w_1
x_2
w_2
$+$
h
$f(h)$
$\hat{y}$
w_3
x_3
$h = \sum_i w_i x_i$
$\hat{y} = f(h)$
ONLY ONE DATA RECORD,
ONLY ONE OUTPUT UNIT


x_1
w_1
x_2
w_2
h
$\hat{y}$
$E = \frac{1}{2}(y - \hat{y})^2$
x_3
w_3

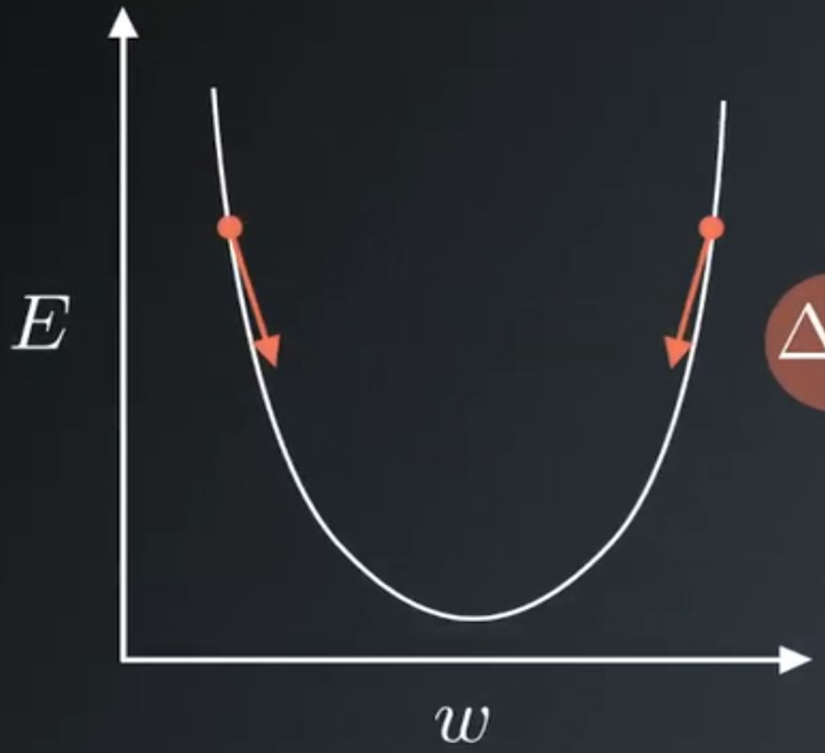


$$E = \frac{1}{2}(y - \hat{y})^2$$

$$= \frac{1}{2}(y - f(\sum_i w_i x_i))^2$$

WEIGHTS LET US TUNE
THE NETWORK

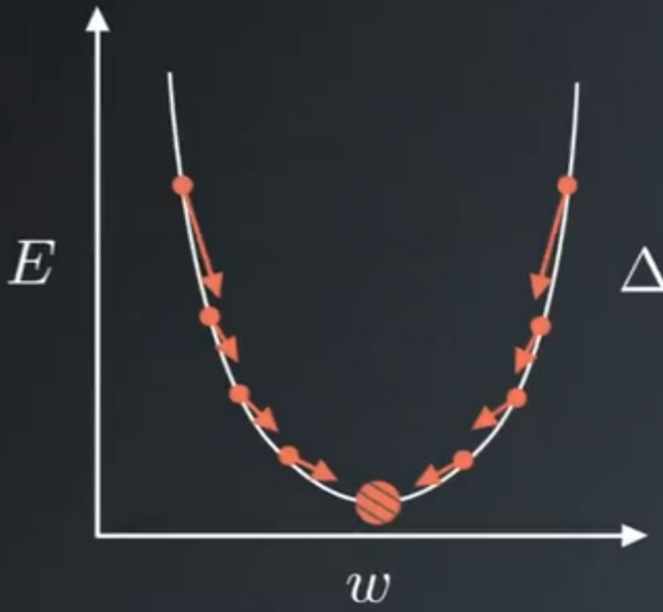
$$E = \frac{1}{2}(y - f(\sum w_i x_i))^2$$



$$\Delta w = -gradient$$

GRADIENT DESCENT

$$E = \frac{1}{2}(y - f(\sum w_i x_i))^2$$



$$\Delta w = -gradient$$

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i \propto -\frac{\partial E}{\partial w_i} \longrightarrow \text{THE GRADIENT}$$

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i \propto -\frac{\partial E}{\partial w_i}$$ → THE GRADIENT

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

LEARNING RATE

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}(y - \hat{y})^2$$

$$= \frac{\partial}{\partial w_i}\frac{1}{2}(y - \hat{y}(w_i))^2$$

- CHAIN RULE REFRESHER

$$\frac{\partial}{\partial z} p(q(z)) = \frac{\partial p}{\partial q} \frac{\partial q}{\partial z}$$

- CHAIN RULE REFRESHER

$$\frac{\partial}{\partial z} p(q(z)) = \frac{\partial p}{\partial q} \frac{\partial q}{\partial z}$$

$$q = (y - \hat{y}(w_i)) \qquad p = \frac{1}{2} q(w_i)^2$$

$$\frac{\partial E}{\partial w_i} = (y - \hat{y}) \frac{\partial}{\partial w_i} (y - \hat{y})$$

$$\hat{y} = f(h) \quad \text{where} \quad h = \sum_i w_i x_i$$

$$\frac{\partial E}{\partial w_i} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial w_i}$$

$$= -(y - \hat{y}) f'(h) \frac{\partial}{\partial w_i} \sum w_i x_i$$

$$\frac{\partial}{\partial w_i} \sum_i w_i x_i$$

$$= \frac{\partial}{\partial w_1} [w_1 x_1 + w_2 x_2 + \ldots + w_n x_n]$$

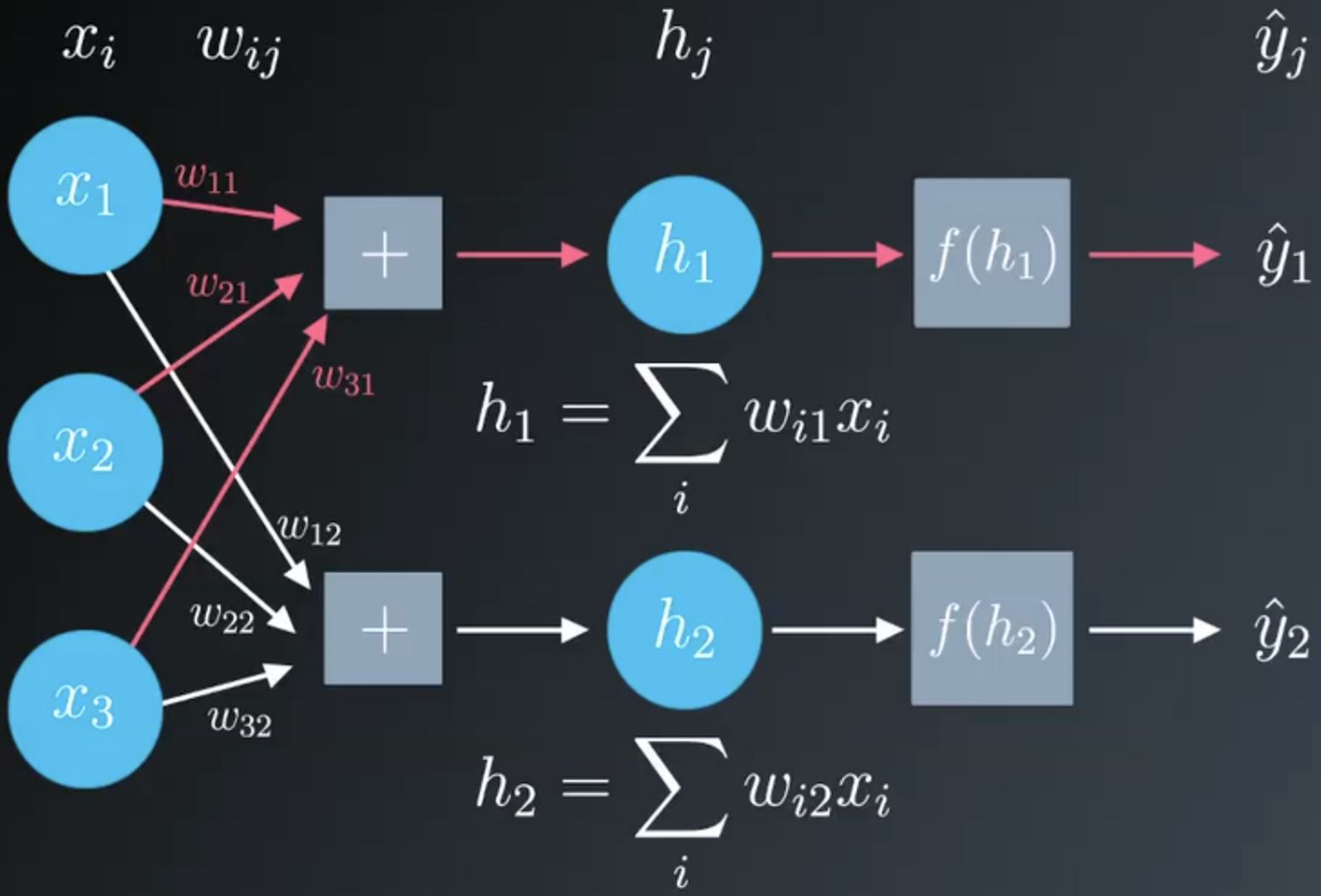
$$= x_1 + 0 + 0 + 0 + \ldots$$

$$\frac{\partial}{\partial w_i} \sum_i w_i x_i = x_i$$

- DEFINE AN “ERROR TERM”

$$\delta = (y - \hat{y}) f'(h)$$

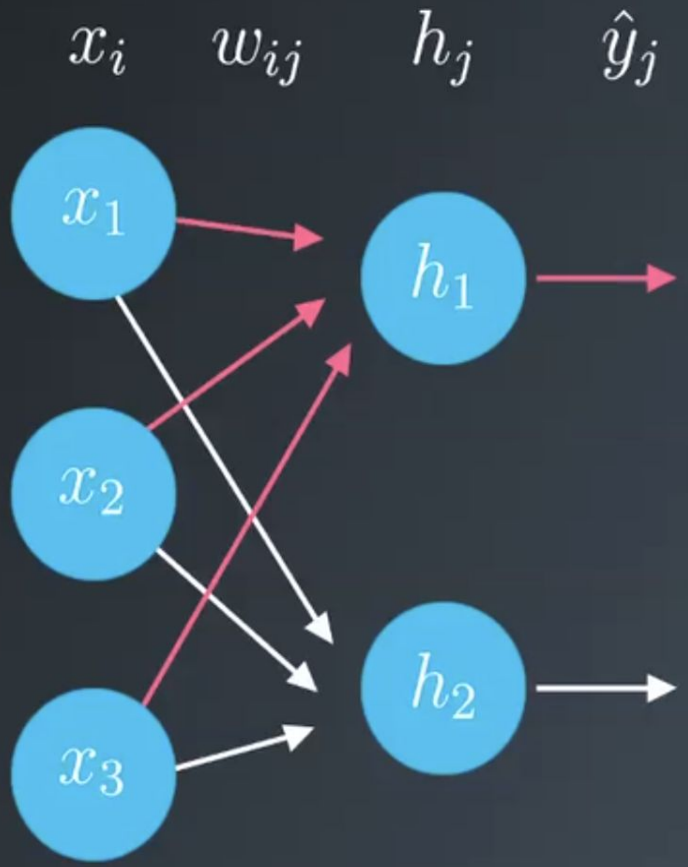
$$w_i = w_i + \eta \delta x_i$$

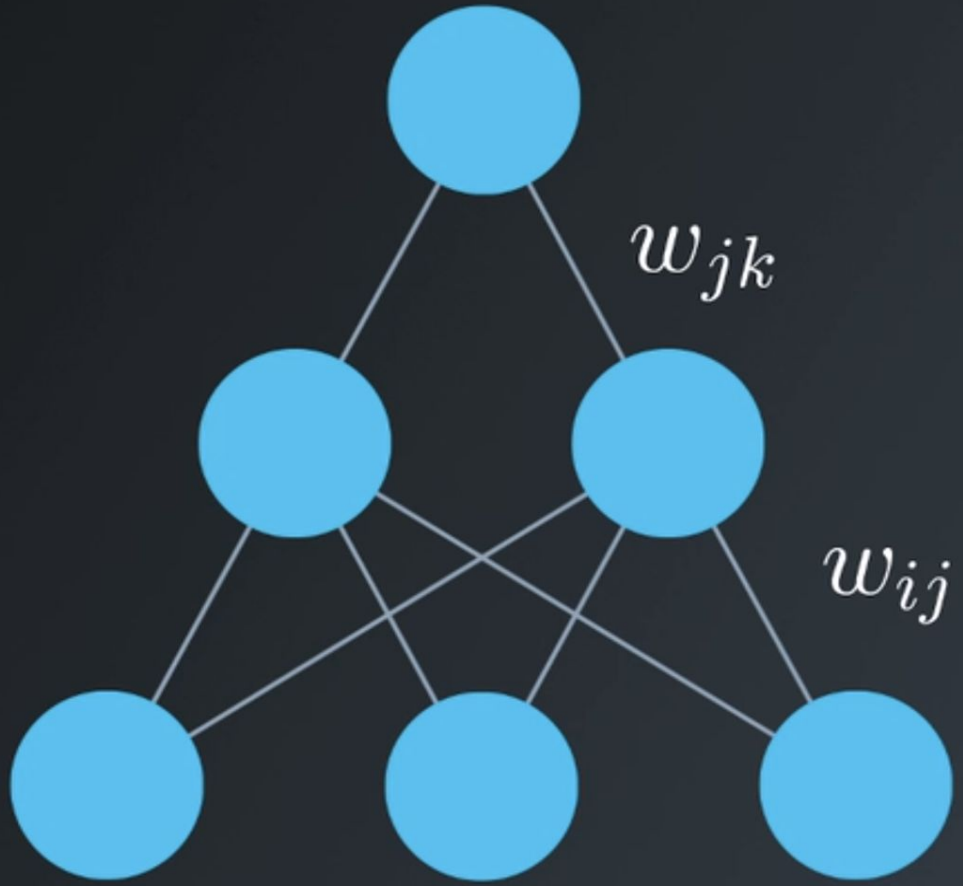

x_i
w_{ij}
h_j
$\hat{y}_j$
x_1
x_2
x_3
w_{11}
w_{21}
w_{31}
w_{12}
w_{22}
w_{32}
+
+
h_1
h_2
$f(h_1)$
$f(h_2)$
$\hat{y}_1$
$\hat{y}_2$
$h_1 = \sum_i w_{i1} x_i$
$h_2 = \sum_i w_{i2} x_i$

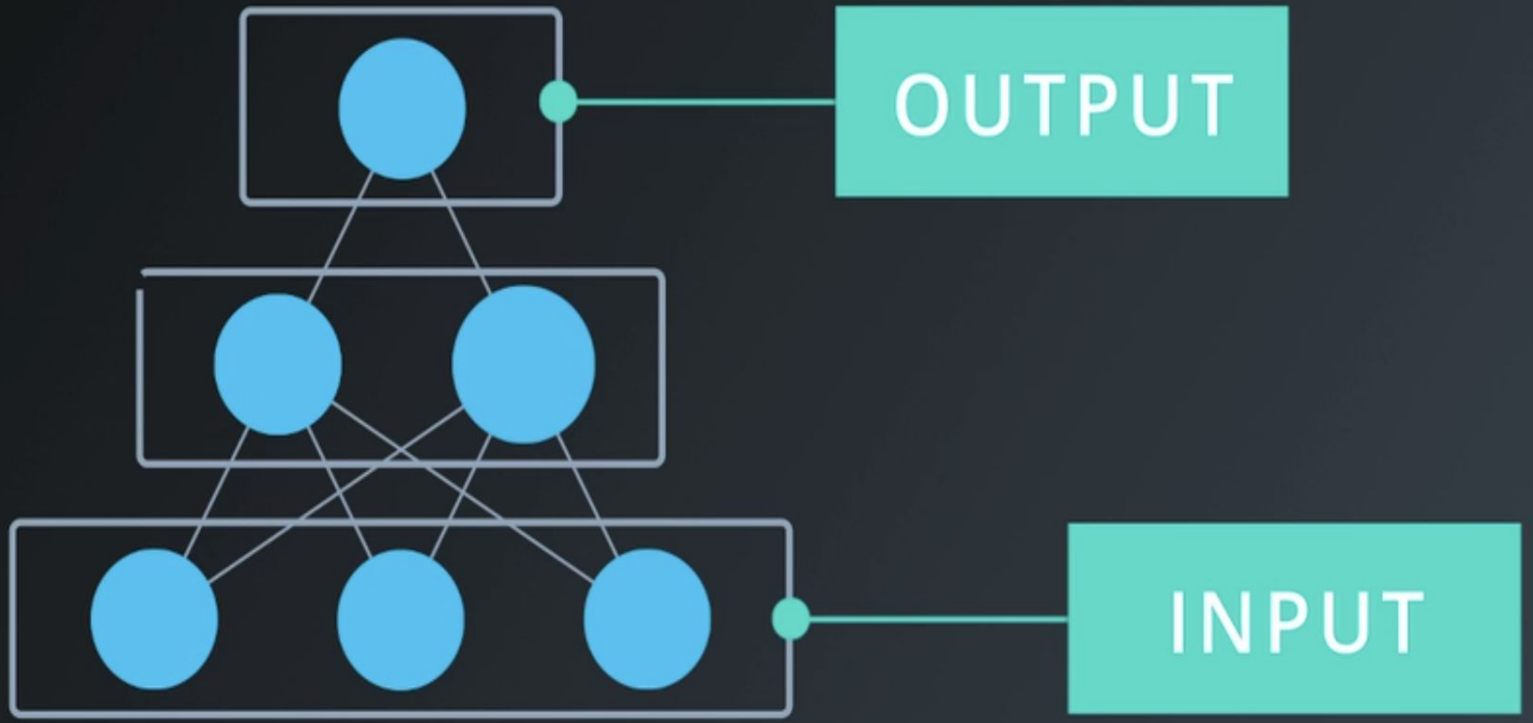
IF MULTIPLE OUTPUT UNITS:

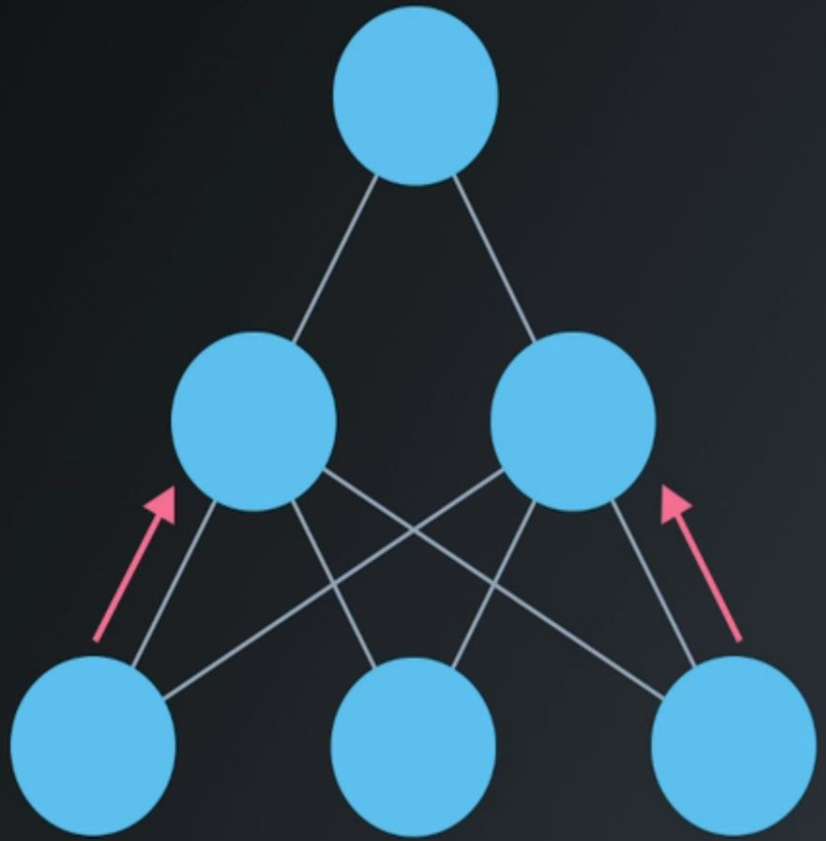
$$\delta_j = (y_j - \hat{y}_j) f'(h_j)$$

$$\Delta w_{ij} = \eta \delta_j x_i$$



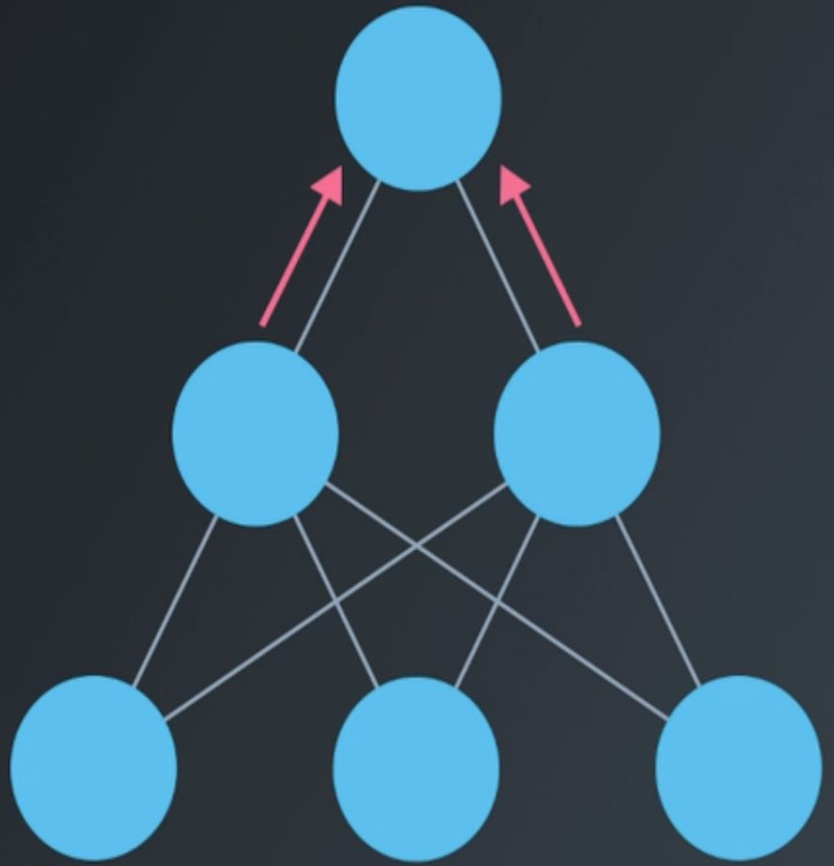

w_{jk}
w_{ij}

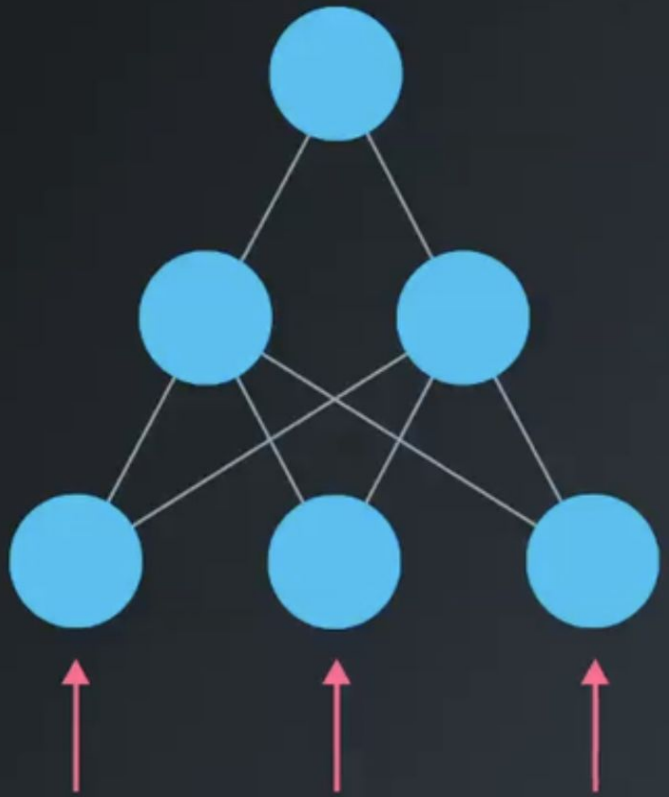

OUTPUT
INPUT

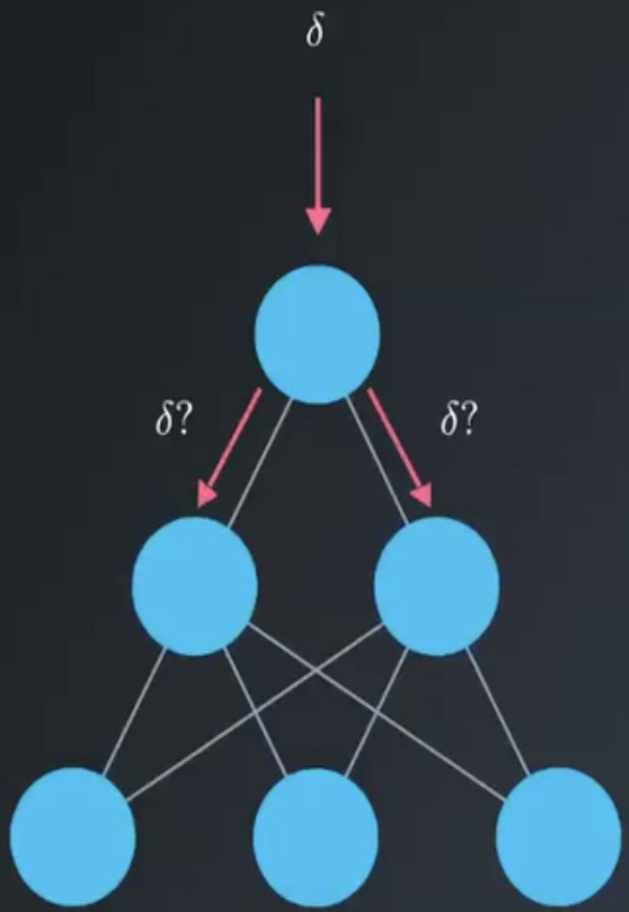


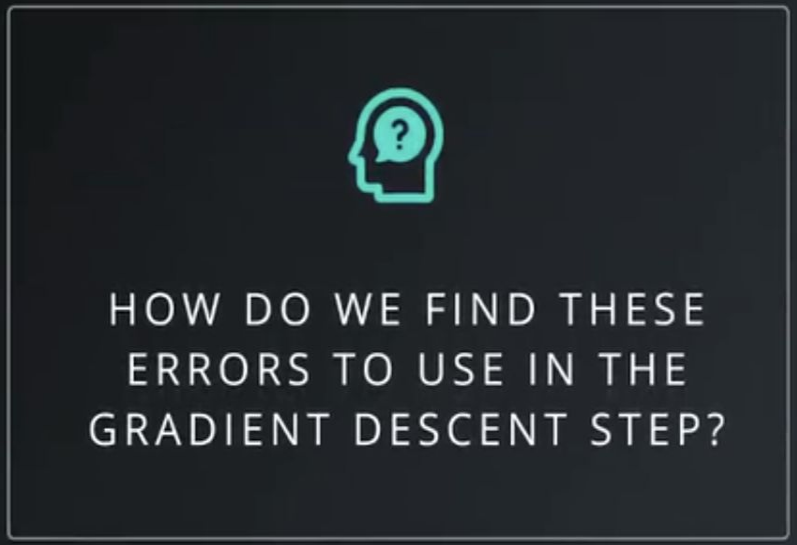
$$h_j = \sum_i w_{ij} * x_i + b_j$$

$$a_j = sigmoid(h_j)$$

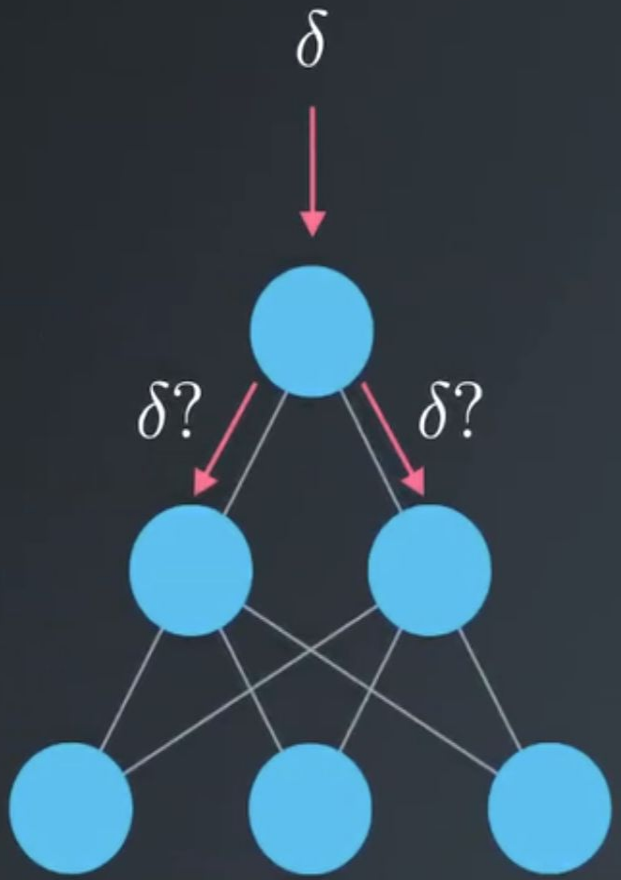


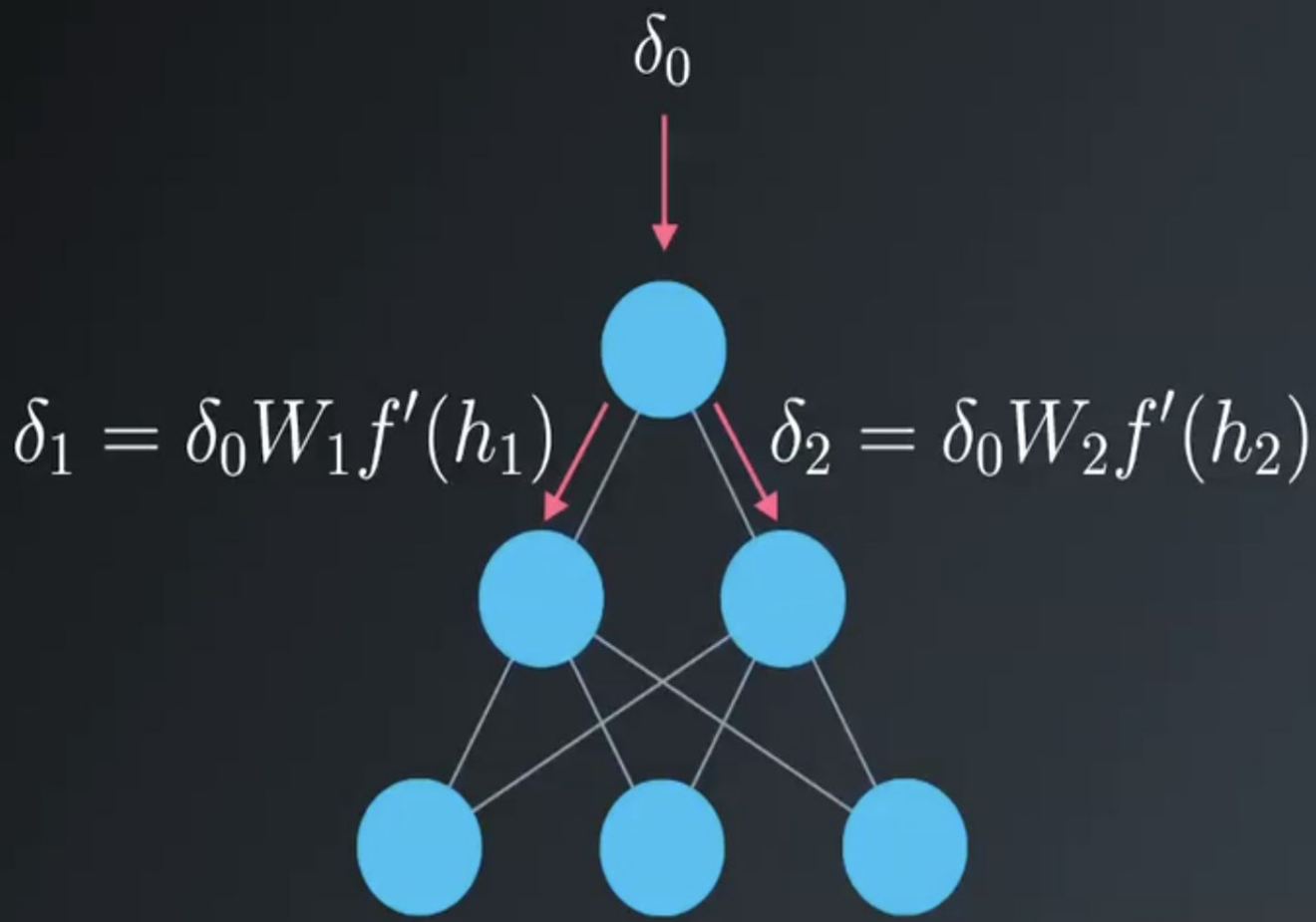

INPUT

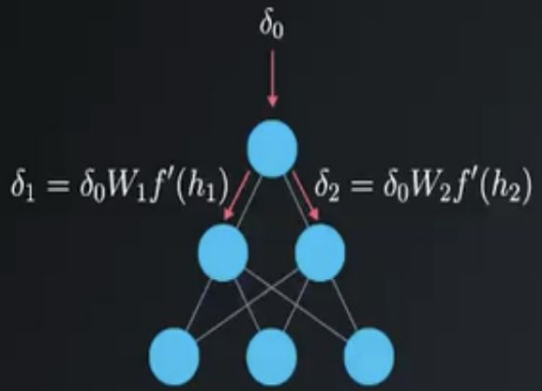

δ
$\delta?$
$\delta?$



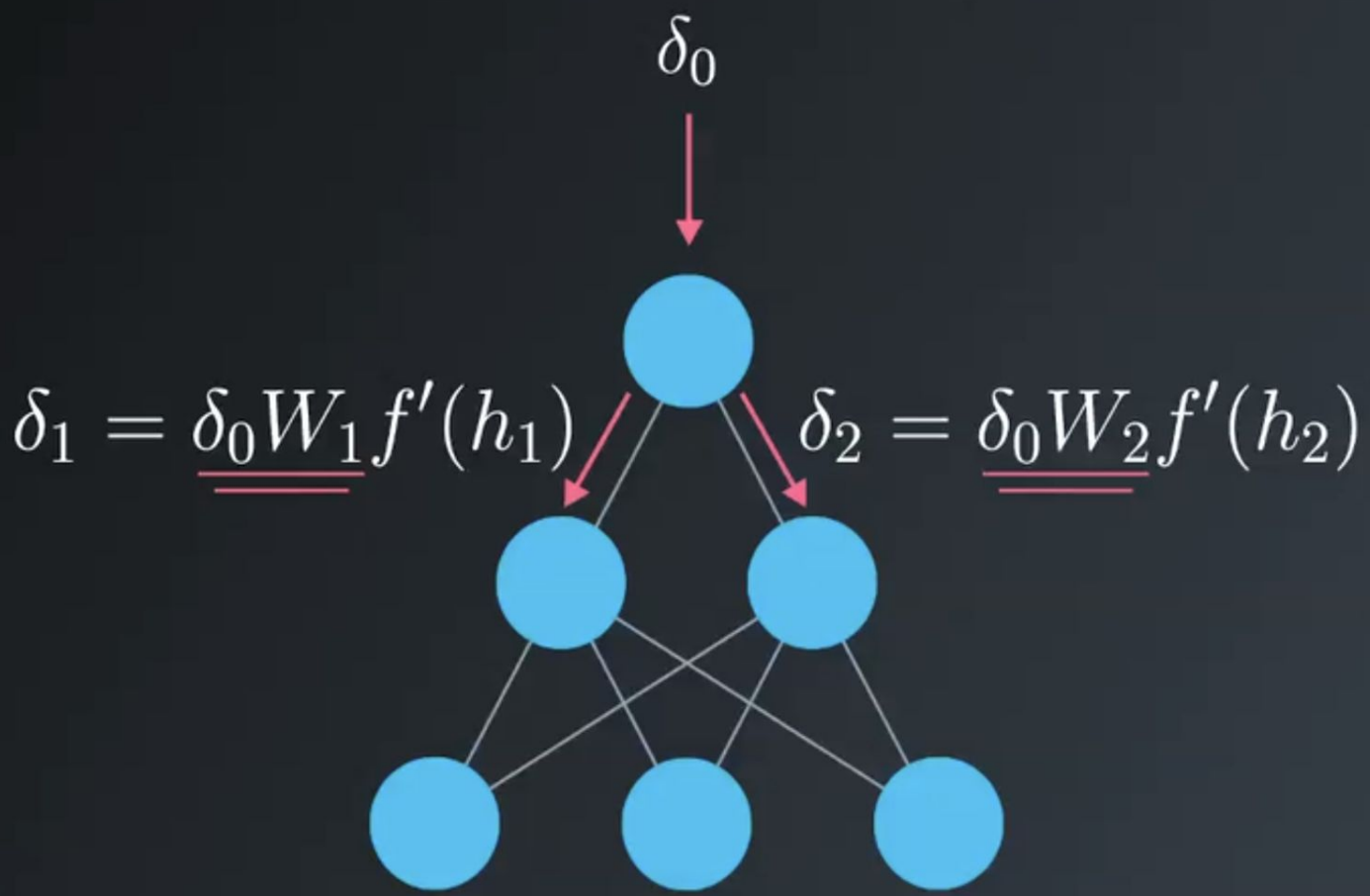
HOW DO WE FIND THESE ERRORS TO USE IN THE GRADIENT DESCENT STEP?

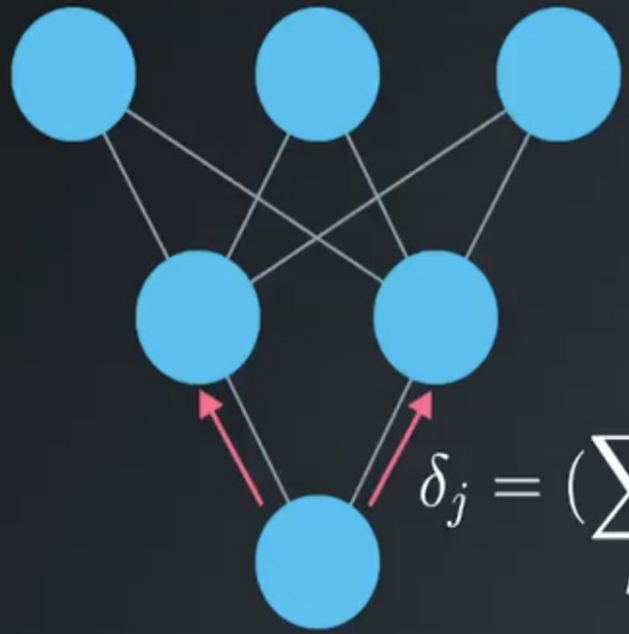



δ_0
$\delta_1 = \delta_0 W_1 f'(h_1)$
$\delta_2 = \delta_0 W_2 f'(h_2)$



ERROR FOR UNITS IS PROPORTIONAL TO THE ERROR IN THE OUTPUT LAYER TIMES THE WEIGHT BETWEEN THE UNITS

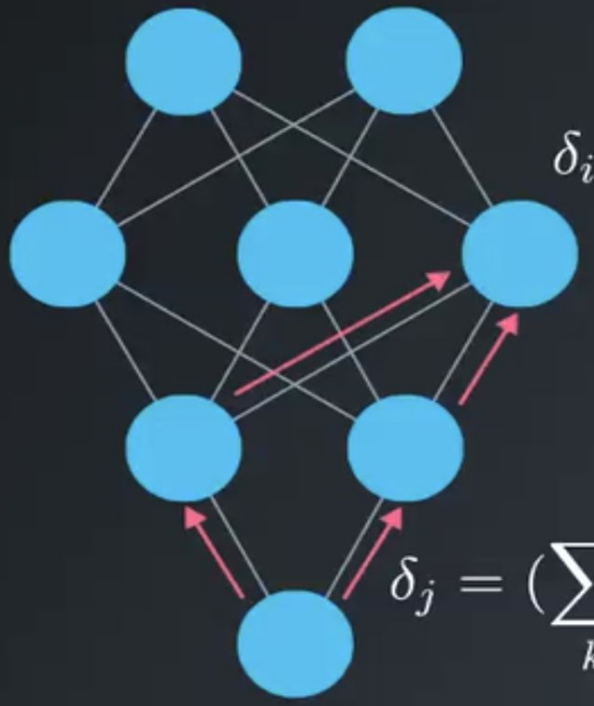

δ_0
$\delta_1 = \delta_0 W_1 f'(h_1)$
$\delta_2 = \delta_0 W_2 f'(h_2)$



$$\delta_j = (\sum_k w_{jk}\delta_k) f'(h_j)$$

- BACKPROPAGATION

δ_k



$$\delta_i = (\sum_j w_{ij}\delta_j) f'(h_i)$$

$$\delta_j = (\sum_k w_{jk}\delta_k) f'(h_j)$$

$$\delta_k$$

Backpropagation

IS FUNDAMENTAL TO HOW
NEURAL NETWORKS LEARN