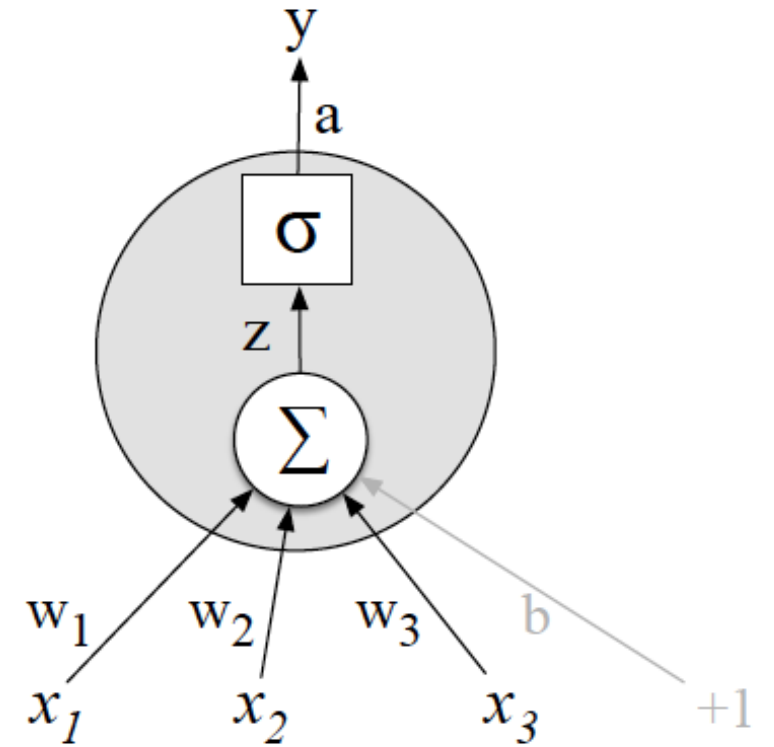


NEURAL LANGUAGE MODELS



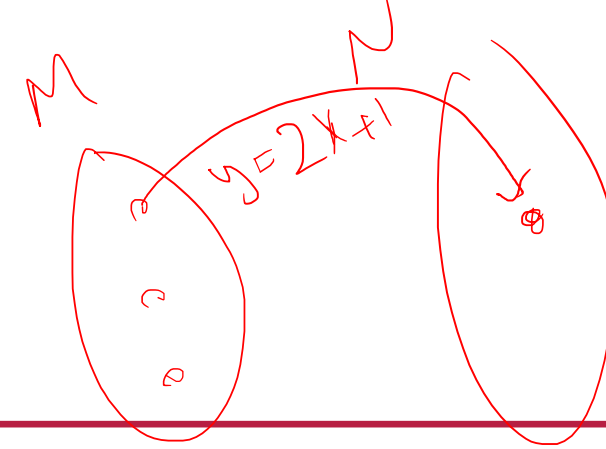
Backgrounds

Computation \leftrightarrow Function

Computational Intelligence \leftrightarrow Learning the function



The image contains two horizontal red lines. The top line is a solid straight line. The bottom line is also a solid straight line. Between these two lines, there are two lines of text. The first line is 'Computation \leftrightarrow Function'. The second line is 'Computational Intelligence \leftrightarrow Learning the function'. There are hand-drawn red annotations: a curved arrow pointing from 'Function' down to 'Learning the function', and another curved arrow pointing from 'Computational Intelligence' up to 'Computation'. Additionally, there are red wavy underlines under 'Computational Intelligence' and 'Learning the function'.



Function

$$f: M \rightarrow N$$

Transferring elements of source space (M) to target space (N)

$$f: \mathbb{R} \rightarrow \mathbb{R} : f(x) = y = x \times 2 + 1$$

f is known

f has two parameters ($a=2$, $b=1$)

Function

$$f: M \rightarrow N$$

Transferring elements of source space (M) to target space (N)

$$f: \mathbb{R} \rightarrow \mathbb{R} : \{(x, f(x)) \mid (1,1), (2,4), (3,9), \dots\}$$

$$f(x) = ?$$

f is unknown!

Function

$$f: M \rightarrow N$$

Transferring elements of source space (M) to target space (N)

$$f: \mathbb{R} \rightarrow \mathbb{R} : \{(x, f(x)) \mid (1,1), (2,4), (3,9), \dots\}$$

$$f(x) = ?$$

f is unknown! Might not exist!

Function

$$f: M \rightarrow N$$

Transferring elements of source space (M) to target space (N)

$$f: \mathbb{R} \rightarrow \mathbb{R} : \{(x, f(x)) \mid (1,1), (2,4), (3,9), \dots\}$$

$$f(x) = ?$$

f is unknown! If exists, there may be more than one!

Function

$$f: M \rightarrow N$$

Transferring elements of source space (M) to target space (N)

$$f: \mathbb{R} \rightarrow \mathbb{R} : \{(x, f(x)) \mid (1,1), (2,4), (3,9), \dots\}$$

$$f(x) = ?$$

f is unknown! Even more than one exists, we may not find one!

Function

$$f: M \rightarrow N$$

Transferring elements of source space (M) to target space (N)

$$f: \mathbb{R} \rightarrow \mathbb{R} : \{(x, f(x)) \mid (1,1), (2,4), (3,9), \dots\}$$

$$f(x) = ?$$

f is unknown! We assume at least one exists, we try to find it!

Transformation

$$f: M \rightarrow N$$

Transferring points (vectors) of source space (M) to target space (N)

$$[T][X] = [Y]; X \in M, Y \in N$$

T is a matrix that includes the parameters of f

Transformation (Linear Algebra)

$$f: M \rightarrow N$$

Transferring points (vectors) of source space (M) to target space (N)

$$\begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = [2x + 1]$$
$$f(x) = 2x + 1$$

or $2x + 1$

Transformation (Linear Algebra)

$$f: M \rightarrow N$$

Transferring points (vectors) of source space (M) to target space (N)

$$(AB)^T = B^T A^T \rightarrow$$

$$\begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = [2x + 1]$$

$$f(x) = 2x + 1$$

Transformation (Linear Algebra)

$$f: M \rightarrow N$$

Transferring points (vectors) of source space (M) to target space (N)

$$\underbrace{[x \ 1]}_{\text{row vector}} \underbrace{\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix}}_{\text{column vector}} = \{(x, f(x)) \mid (1, 1), (2, 4), (3, 9), \dots\}$$

$f(x) = ?$

Handwritten calculations:

$$\begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \rightarrow 4$$
$$\begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \rightarrow 9$$

Neural Network

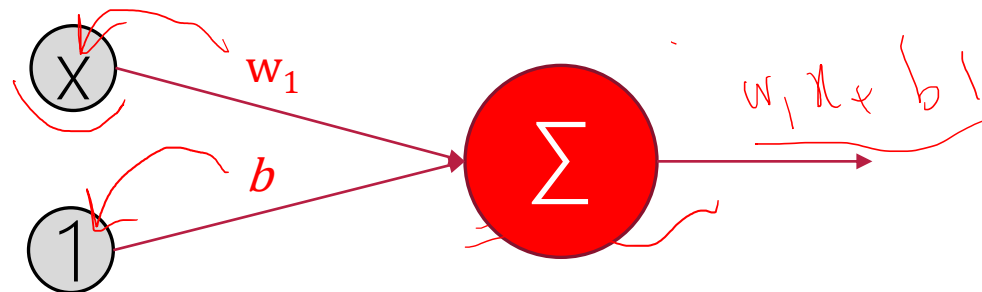
$$f: M \rightarrow N$$

Perceptron

Transferring points (vectors) of source space (M) to target space (N)

$$\begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ b \end{bmatrix} = y$$

$f(x) = y$



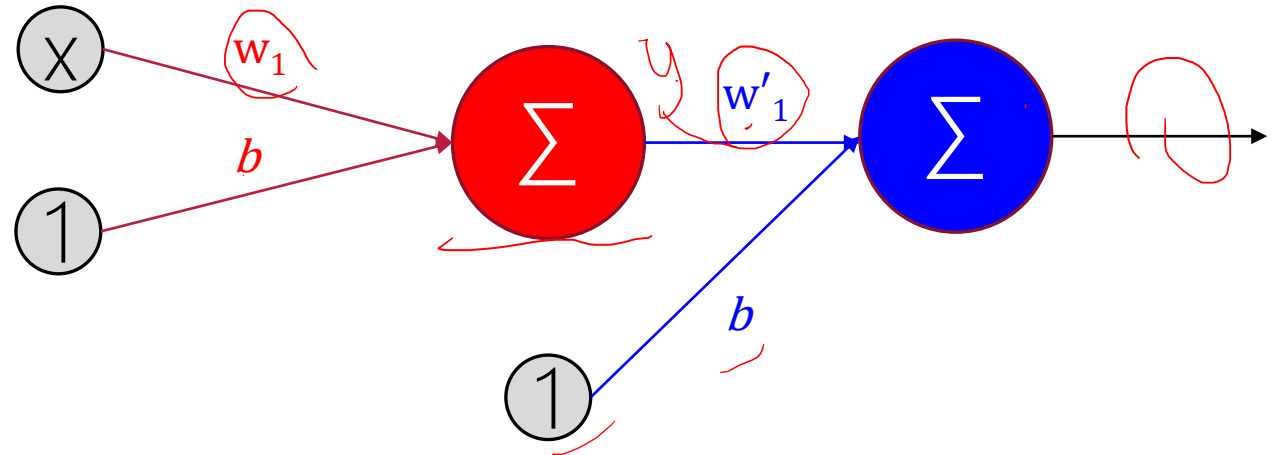
Neural Network

$$g: (f: M \rightarrow N) \rightarrow O$$

Transferring points (vectors) of source space (M) to target space (O)

$$\begin{bmatrix} [x \ 1] \begin{bmatrix} w_1 \\ b \end{bmatrix} \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} w'_1 \\ b \end{bmatrix} = z$$

$$(g \circ f)(x) = g(f(x)) = z$$

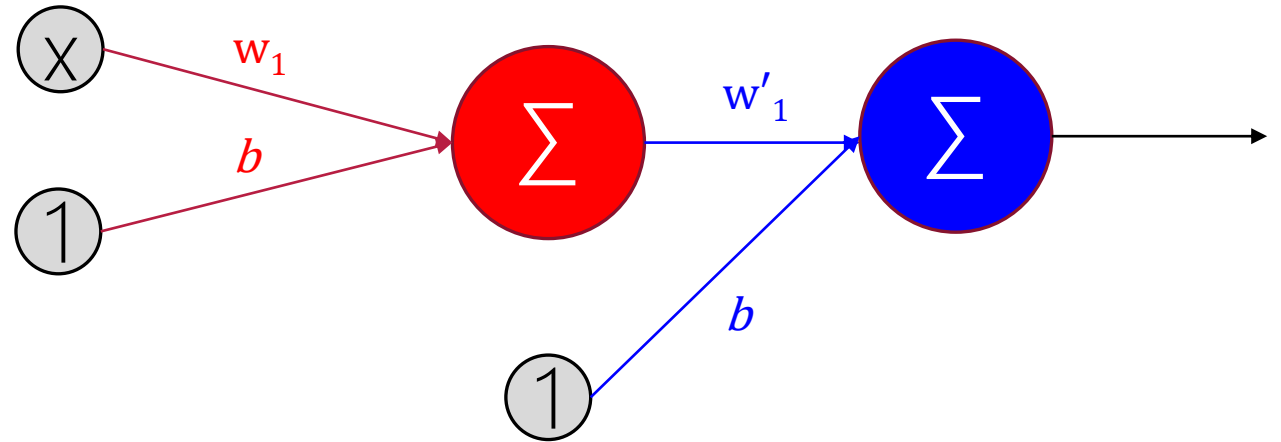


Neural Network

$$g: (f: M \rightarrow N) \rightarrow O$$

Transferring points (vectors) of source space (M) to target space (O)

$$\begin{bmatrix} y & 1 \end{bmatrix} \begin{bmatrix} w'_1 \\ b \end{bmatrix} = z$$
$$g(f(x)) = g(y) = z$$



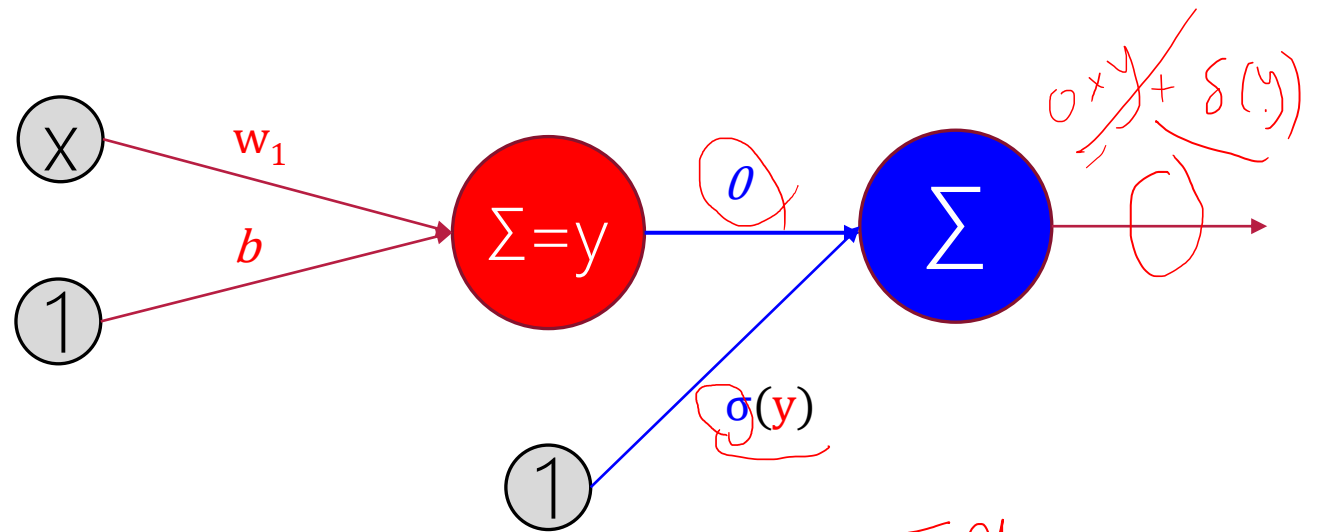
Neural Network

$$g: (f: M \rightarrow N) \rightarrow O$$

Transferring points (vectors) of source space (M) to target space (O)

$$\begin{bmatrix} y & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \sigma(y) \end{bmatrix} = y$$

$$g(f(x)) = g(y) = z$$



$$\begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ b \end{bmatrix} \begin{bmatrix} 0 \\ \sigma(x \times w_1 + b) \end{bmatrix} = z$$

Neural Network

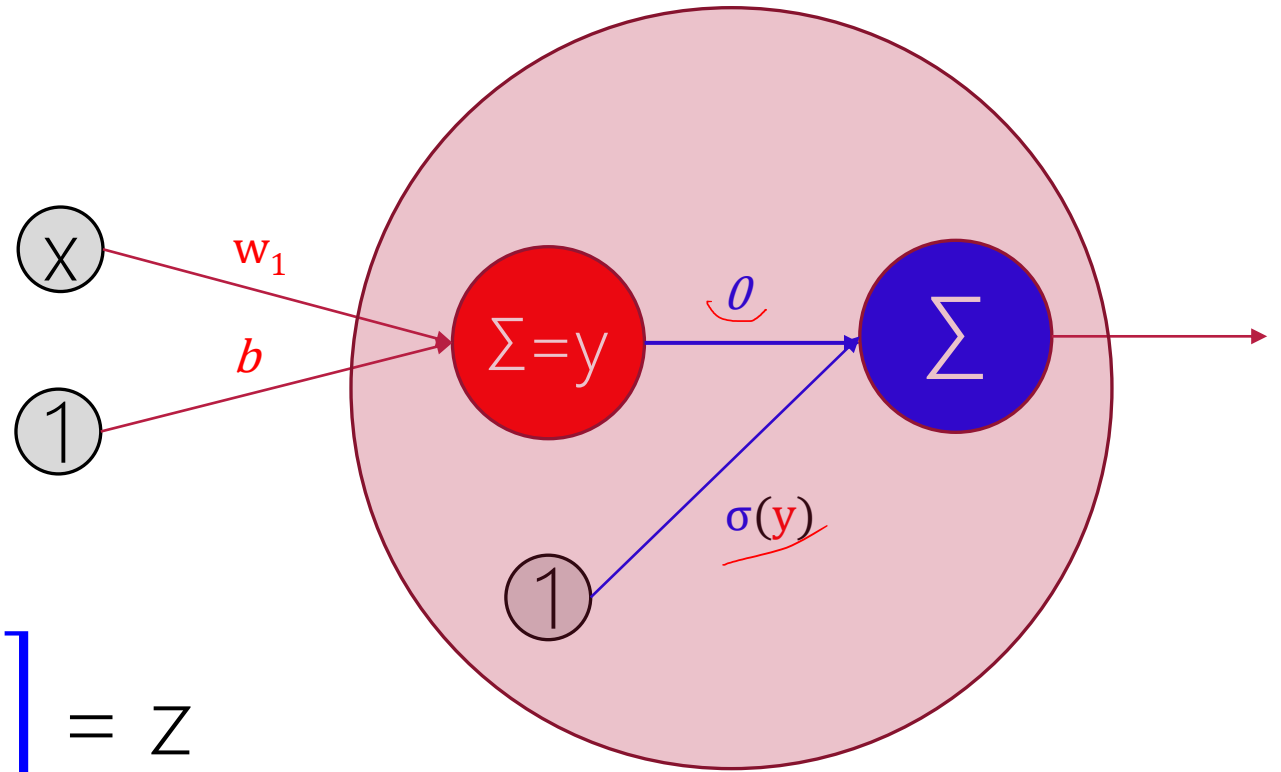
$$g: (f: M \rightarrow N) \rightarrow O$$

Transferring points (vectors) of source space (M) to target space (O)

$$\begin{bmatrix} y & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \sigma(y) \end{bmatrix} = y$$

$$g(f(x)) = g(y) = z$$

$$\begin{bmatrix} [x & 1] \begin{bmatrix} w_1 \\ b \end{bmatrix} & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \sigma(x \times w_1 + b) \end{bmatrix} = z$$

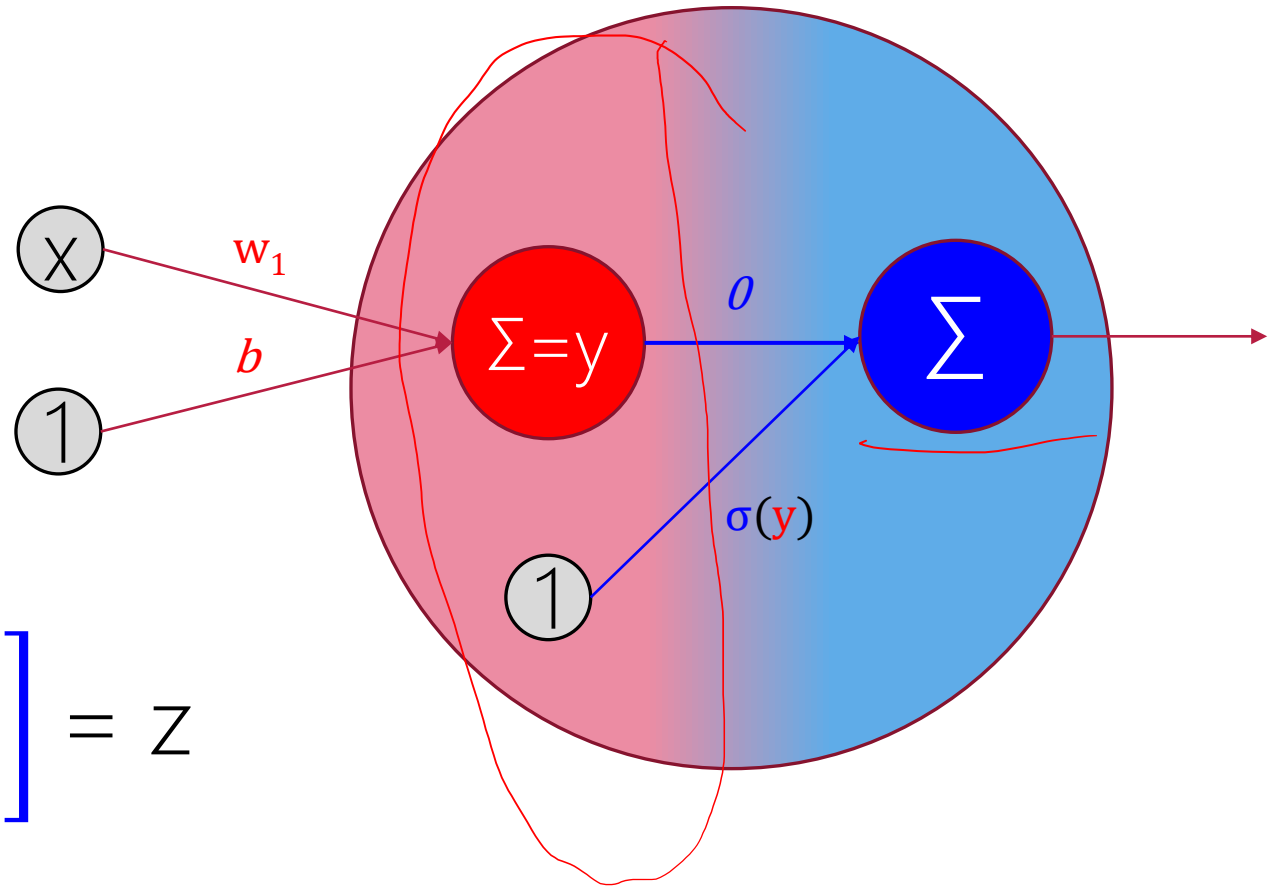


Neural Network

$$h: M \rightarrow O$$

Transferring points (vectors) of source space (M) to target space (O)

$$h(x) = \sigma(x \times w_1 + b)$$



$$\begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ b \end{bmatrix} + 1 \cdot o = z$$

Neural Network

$$h: M \rightarrow O$$

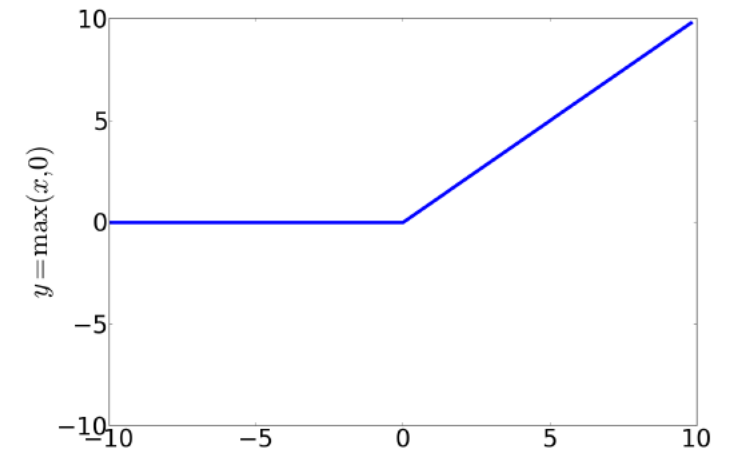
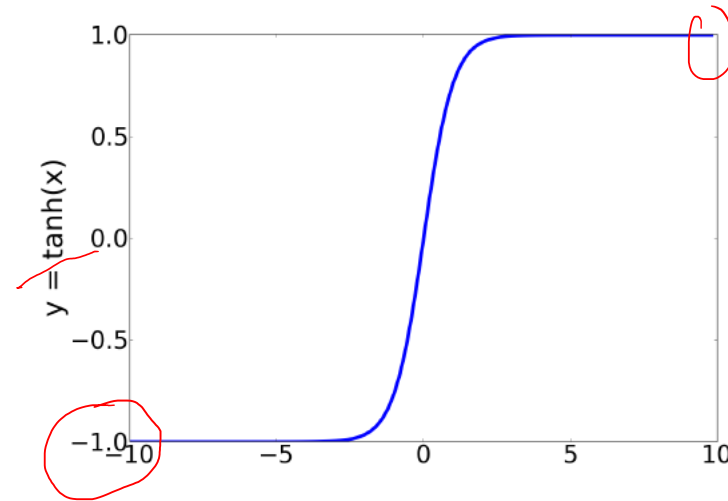
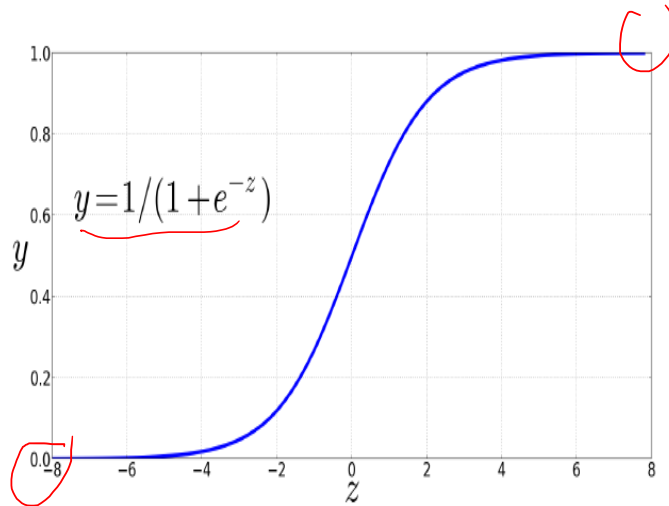
Transferring points (vectors) of source space (M) to target space (O)

$$h(x) = \sigma(x \times w_1 + b)$$

σ is a known *non*-linear function: Sigmoid, ReLU, Tanh

$$\begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ b \end{bmatrix} = \sigma(x \times w_1 + b)$$

Neural Network: Activation Function σ



Neural Network

$$h: M=\{0,1\}^2 \rightarrow O=\{0,1\}$$

$$M = \{(0,0), (0,1), (1,0), (1,1)\}$$

$$O = \{0, 1\}$$

$$h = \{((0,0), 0), ((0,1), 0), ((1,0), 0), ((1,1), 1)\}$$

AND

$$\begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \sigma(x_1 \times w_1 + x_2 \times w_2 + b) = z$$

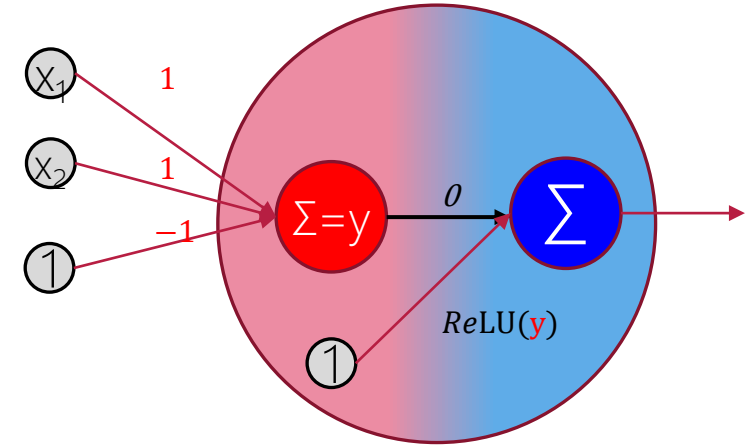
Neural Network

$$AND: M = \{0,1\}^2 \rightarrow O = \{0,1\}$$

$$M = \{(0,0), (0,1), (1,0), (1,1)\}$$

$$O = \{0, 1\}$$

$$h = \{((0,0), 0), ((0,1), 0), ((1,0), 0), ((1,1), 1)\}$$



$$\begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = 0 + 0 - 1 = -1$$

$$\left[\max(0, -1) \right] = 0$$

Handwritten annotations include red arrows pointing from the input values to the weights, and red text indicating the calculation: $0 + 0 - 1 = -1$ and $\max(0, -1) = 0$.

God (Oracle) told the weights

Neural Network

$$OR: M = \{0,1\}^2 \rightarrow O = \{0,1\}$$

$$M = \{(0,0), (0,1), (1,0), (1,1)\}$$

$$O = \{0, 1\}$$

$$h = \{((0,0), 0), ((0,1), 1), ((1,0), 1), ((1,1), 1)\}$$

$$\begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \sigma(x_1 \times w_1 + x_2 \times w_2 + b) = z$$

Handwritten annotations in red:

- A circle around the input vector $[x_1 \ x_2 \ 1]$ with the label "2.0."
- A circle around the weight vector $\begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$ with the label "1.0."
- A circle around the sigmoid function σ with the label "sig mat".
- A circle around the weight w_1 with an arrow pointing to it from the label w_1 .
- A circle around the weight w_2 with an arrow pointing to it from the label w_2 .
- A circle around the bias b with an arrow pointing to it from the label b .

Ask God (Oracle)

Neural Network

$$XOR: M = \{0,1\}^2 \rightarrow O = \{0,1\}$$

$$M = \{(0,0), (0,1), (1,0), (1,1)\}$$

$$O = \{0, 1\}$$

$$h = \{((\underline{0,0}), \underline{0}), ((\underline{0,1}), 1), ((\underline{1,0}), 1), ((\underline{1,1}), 0)\}$$

$$[[x_1 \ x_2 \ 1] \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{b} \end{bmatrix}]^T \begin{bmatrix} 0 \\ \sigma(x_1 \times \mathbf{w}_1 + x_2 \times \mathbf{w}_2 + \mathbf{b}) \end{bmatrix} = z$$

Even God (Oracle) cannot tell

Neural Network

$$XOR: M=\{0,1\}^2 \rightarrow O=\{0,1,2\}^2 \rightarrow P=\{0,1\}$$

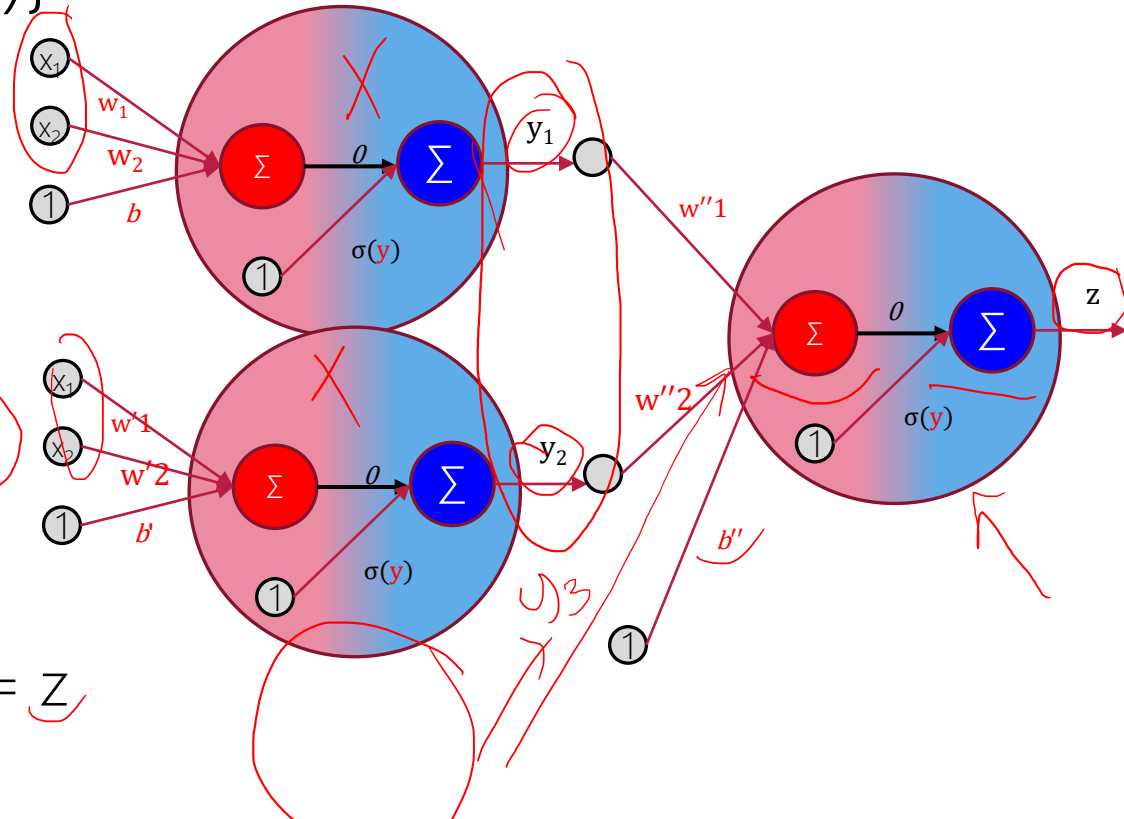
$$M = \{(0,0), (0,1), (1,0), (1,1)\} \rightarrow O = \{(0,0), (0,1), \dots, (2,2)\} \rightarrow P = \{0, 1\}$$

$$h = \{((0,0), 0), ((0,1), 1), ((1,0), 1), ((1,1), 0)\}$$

$$[[x_1 \ x_2 \ 1] \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}]^0 \left[\sigma(x_1 \times w_1 + x_2 \times w_2 + b) \right] = y_1$$

$$[[x_1 \ x_2 \ 1] \begin{bmatrix} w'_1 \\ w'_2 \\ b' \end{bmatrix}]^0 \left[\sigma(x_1 \times w'_1 + x_2 \times w'_2 + b') \right] = y_2$$

$$[[y_1 \ y_2 \ 1] \begin{bmatrix} w''_1 \\ w''_2 \\ b'' \end{bmatrix}]^0 \left[\sigma(x_1 \times w''_1 + x_2 \times w''_2 + b'') \right] = z$$



Neural Network

$$XOR: M=\{0,1\}^2 \rightarrow O=\{0,1,2\}^2 \rightarrow P=\{0,1\}$$

$$M = \{(0,0), (0,1), (1,0), (1,1)\} \rightarrow O = \{(0,0), (0,1), \dots, (2,2)\} \rightarrow P = \{0, 1\}$$

$$h = \{((0,0), 0), ((0,1), 1), ((1,0), 1), ((1,1), 0)\}$$

$$\begin{aligned}
 & [[x_1 \ x_2 \ 1] \begin{bmatrix} w_1 & w'_1 \\ w_2 & w'_2 \\ b & b' \end{bmatrix} 1] = \\
 & [x_1 \times w_1 + x_2 \times w_2 + b \quad x_1 \times w'_1 + x_2 \times w'_2 + b' \quad 1] \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \\
 & [\underbrace{\sigma(x_1 \times w_1 + x_2 \times w_2 + b)}_{\alpha} \quad \underbrace{\sigma(x_1 \times w'_1 + x_2 \times w'_2 + b')}_{\beta} \quad 1] \begin{bmatrix} w''_1 \\ w''_2 \\ b'' \end{bmatrix} 1] \left[\sigma(\alpha \times w''_1 + \beta \times w''_2 + b'') \right] = z
 \end{aligned}$$

Neural Network

$$XOR: M=\{0,1\}^2 \rightarrow O=\{0,1,2\}^2 \rightarrow P=\{0,1\}$$

$$M = \{(0,0), (0,1), (1,0), (1,1)\} \rightarrow O = \{(0,0), (0,1), \dots, (2,2)\} \rightarrow P = \{0, 1\}$$

$$h = \{((0,0), 0), ((0,1), 1), ((1,0), 1), ((1,1), 0)\}$$

$g(f(x_1, x_2)) = z$

God (Oracle)

$$[[x_1 \ x_2 \ 1] \begin{bmatrix} w_1 & w'_1 \\ w_2 & w'_2 \\ b & b' \end{bmatrix} 1] = [x_1 \ x_2 \ 1] \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & -1 \end{bmatrix}$$

$$[(x_1 \times 1 + x_2 \times 1 + 0) \ (x_1 \times 1 + x_2 \times 1 + -1) \ 1] \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \text{ReLU}(x_1 + x_2) & \text{ReLU}(x_1 + x_2 - 1) \end{bmatrix} =$$

$$[\text{ReLU}(x_1 + x_2) \ \text{ReLU}(x_1 + x_2 - 1) \ 1] \begin{bmatrix} w''_1 \\ w''_2 \\ b'' \end{bmatrix} 1] = [\text{ReLU}(x_1 + x_2) \ \text{ReLU}(x_1 + x_2 - 1) \ 1] \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix} 1]$$

Neural Network

$$XOR: M = \{0,1\}^2 \rightarrow O = \{0,1,2\}^2 \rightarrow P = \{0,1\}$$

$$M = \{(0,0), (0,1), (1,0), (1,1)\} \rightarrow O = \{(0,0), (0,1), \dots, (2,2)\} \rightarrow P = \{0, 1\}$$

$$(0,0) \rightarrow (0,0) \rightarrow (0)$$

$$(0,1) \rightarrow (1,0) \rightarrow (1)$$

$$(1,0) \rightarrow (1,0) \rightarrow (1)$$

$$(1,1) \rightarrow (2,1) \rightarrow (0)$$

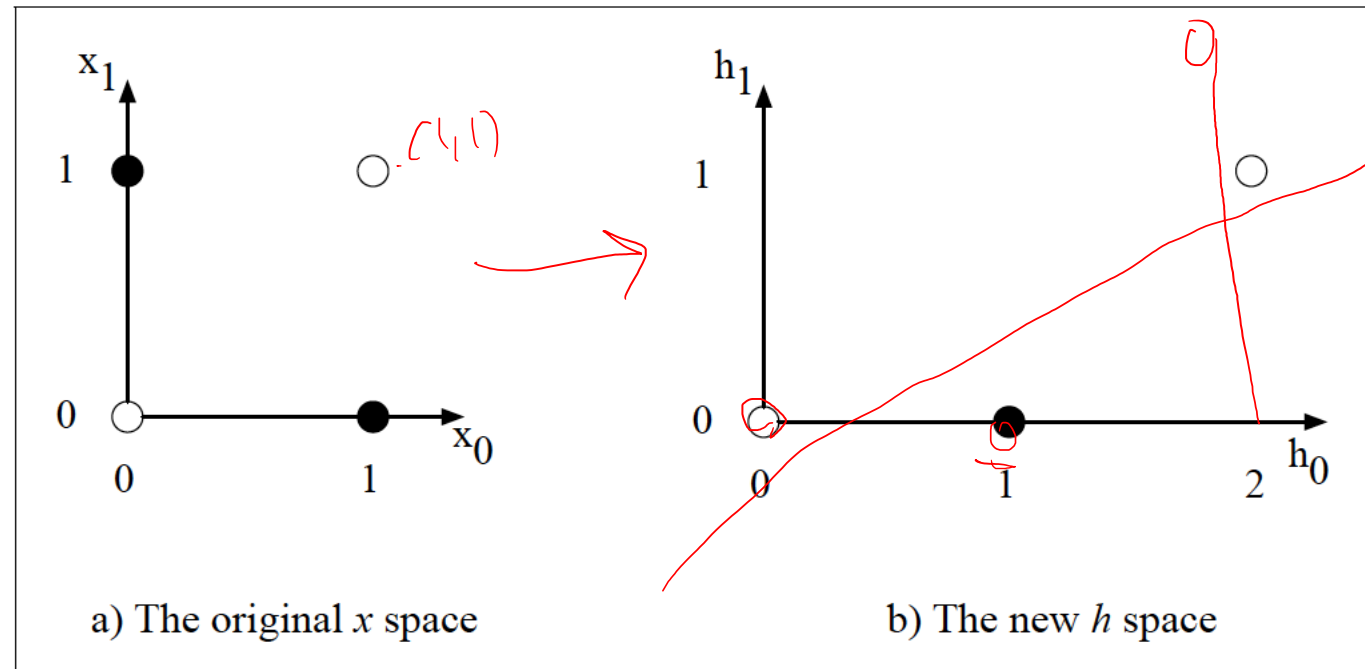


Figure 7.7 The hidden layer forming a new representation of the input. Here is the representation of the hidden layer, h , compared to the original input representation x . Notice that the input point $[0\ 1]$ has been collapsed with the input point $[1\ 0]$, making it possible to linearly separate the positive and negative cases of XOR. After [Goodfellow et al. \(2016\)](#).

https://www.facebook.com/watch/?extid=NS-UNK-UNK-UNK-IOS_GK0T-GKIC&v=473881334464404

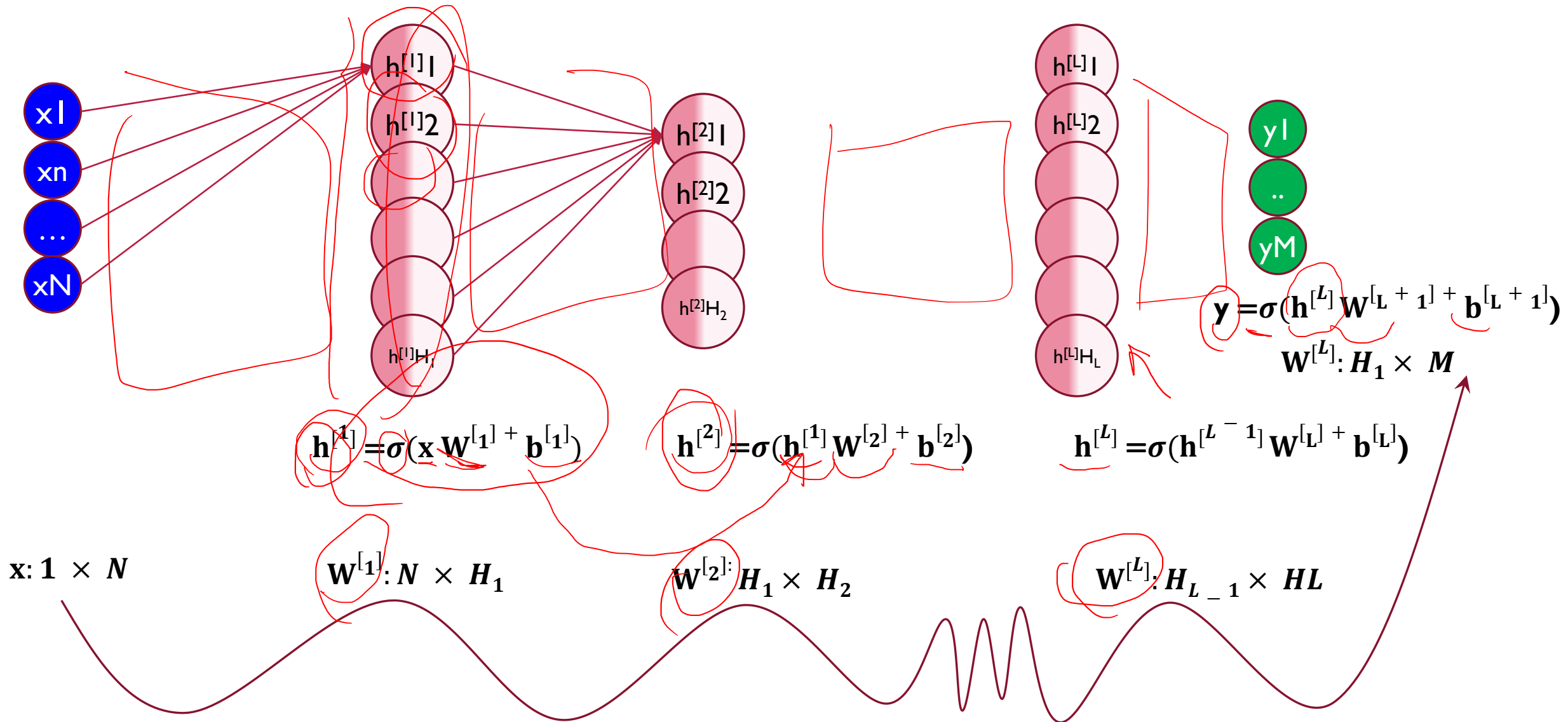
Universal Approximation Theorem

<http://neuralnetworksanddeeplearning.com/chap4.html>

Be careful, it shows the existence (power) of neural nets, but it does not show which architecture is the function!

How many transformation (#layers)?
To what intermediate space (#nodes)?

Neural Network: Feed-forward



Neural Network: Feed-forward Train

The only known is the data (input: X , output: Y)
Given the input point X , the net should land it to Y .

If the net should land it to $f(X) = Y' \rightarrow$ error!
Correct the net to make Y' close to Y .

How?

x_1
 x_n
...
 x_N

y_1
..
 y_M



Neural Network: Feed-forward Train

How?

By minimizing the overall error.

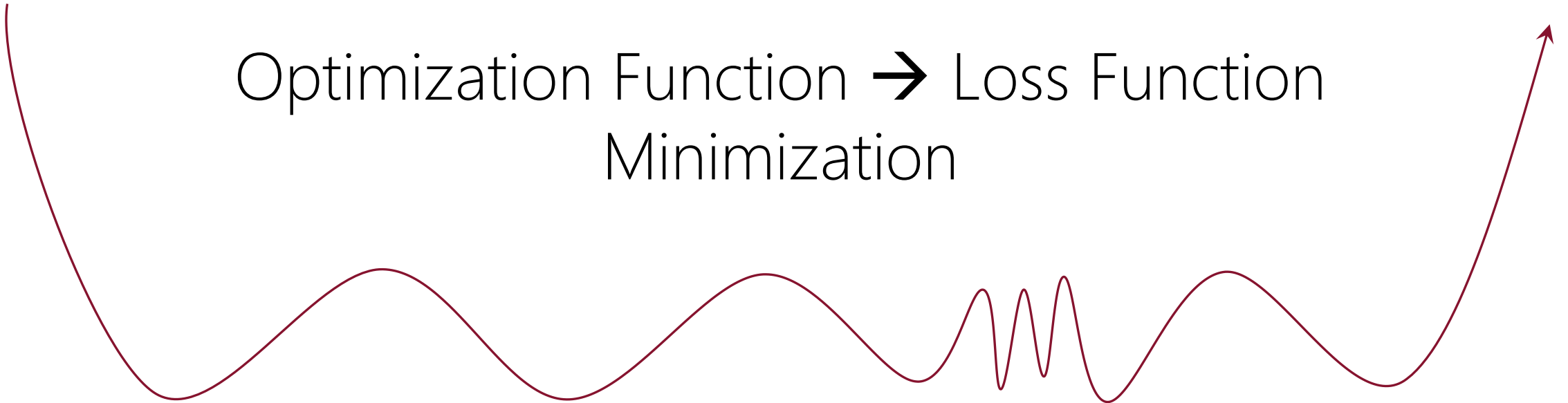
What is error?

$e = \text{distance}(Y, f(X)); \text{ for all } (X, Y)$

x_1
 x_n
...
 x_N

y_1
..
 y_M

Optimization Function \rightarrow Loss Function
Minimization



Neural Network: Gradient

How?

By minimizing the overall error.

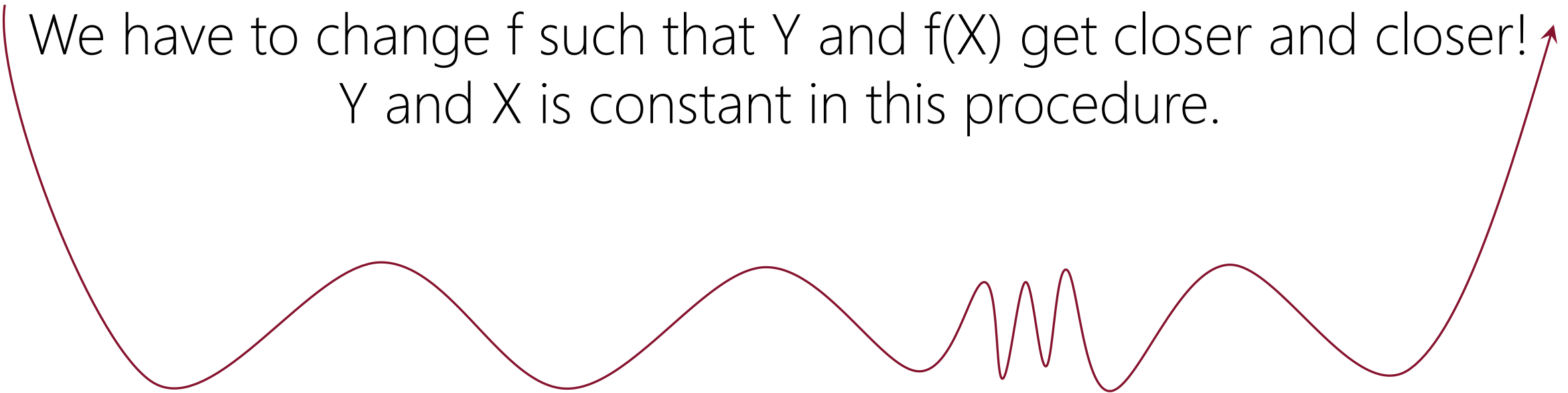
What is error?

$e = \text{distance}(Y, f(X));$ for all (X, Y)

x_1
 x_n
...
 x_N

y_1
..
 y_M

We have to change f such that Y and $f(X)$ get closer and closer!
 Y and X is constant in this procedure.



Neural Network: Gradient

How?

By minimizing the overall error.

What is error?

$e = \text{distance}(Y, f(X));$ for all (X, Y)

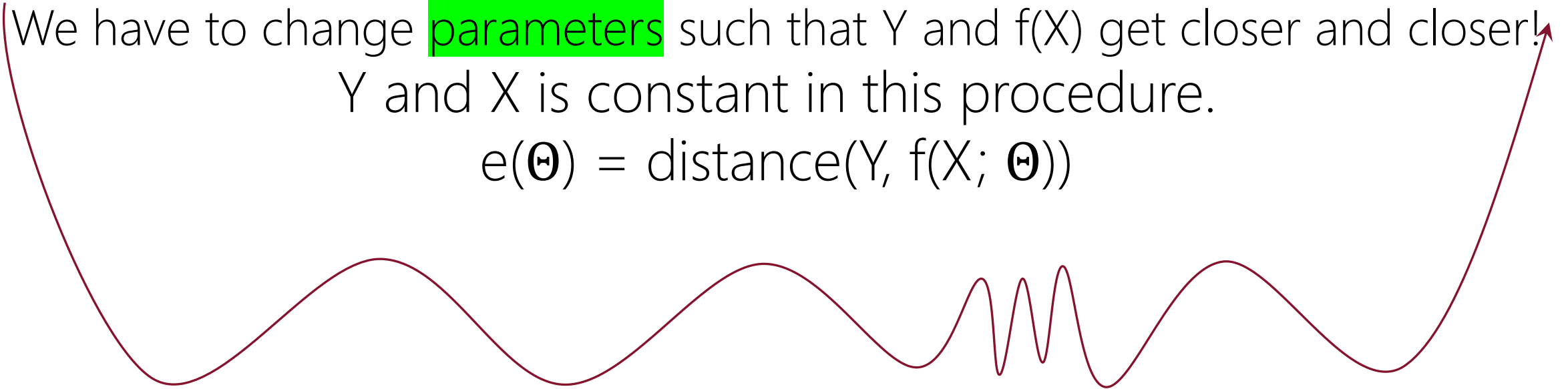
x_1
 x_n
...
 x_N

y_1
..
 y_M

We have to change **parameters** such that Y and $f(X)$ get closer and closer!

Y and X is constant in this procedure.

$$e(\Theta) = \text{distance}(Y, f(X; \Theta))$$



Neural Network: Gradient

$$f(x) = 3x+1 \rightarrow f(x; \Theta) = f(x; [3,1]) = 3x+1$$

$$f(x; [w_1, w_2]) = w_1 x + w_2$$

$$(x, y) = (2, 1)$$

$$f(2; [w_1, w_2]) = w_1 \cdot 2 + w_2$$

$$e = e(w_1, w_2) = e(\Theta) = |f(2; [w_1, w_2]) - 1| = w_1 \cdot 2 + w_2 - 1$$

How to change w_1 and w_2 to reduce the error?

x_1
 x_n
...
 x_N

y_1
..
 y_M

Neural Network: Gradient Descent

$$f(x) = 3x+1 \rightarrow f(x; \Theta) = f(x; [3, 1]) = 3x+1$$

$$f(x; [w_1, w_2]) = w_1 x + w_2$$

$$(x, y) = (2, 1)$$

$$f(2; [w_1, w_2]) = w_1 \cdot 2 + w_2$$

$$e = e(w_1, w_2) = e(\Theta) = |f(2; [w_1, w_2]) - 1| = w_1 \cdot 2 + w_2 - 1$$

$$w_1 = w_1 - \eta \frac{\partial e}{\partial w_1} = w_1 - 0.1 \times 2 = w_1 - 0.2$$

$$w_2 = w_2 - \eta \frac{\partial e}{\partial w_2} = w_2 - 0.1 \times 1 = w_2 - 0.1$$

x1
xn
...
xN

yl
..
yM

Neural Network: Gradient Descent

$$f(x; \Theta) = f(x; [2.8, 0.9]) = 2.8x + 0.9 =$$

$$f(x; \Theta) = f(x; [2.6, 0.8]) = 2.6x + 0.8 =$$

$$f(x; \Theta) = f(x; [2.4, 0.7]) = 2.4x + 0.7 =$$

...

x_1

x_n

...

x_N

y_1

..

y_M

Neural Network: Gradient Descent

Convex Error Function

Many Parameters (all matrices' elements)

x_1
 x_n
...
 x_N

y_1
..
 y_M

Backpropagation!

Computation Graph

Forward Pass

Backward Differentiation.