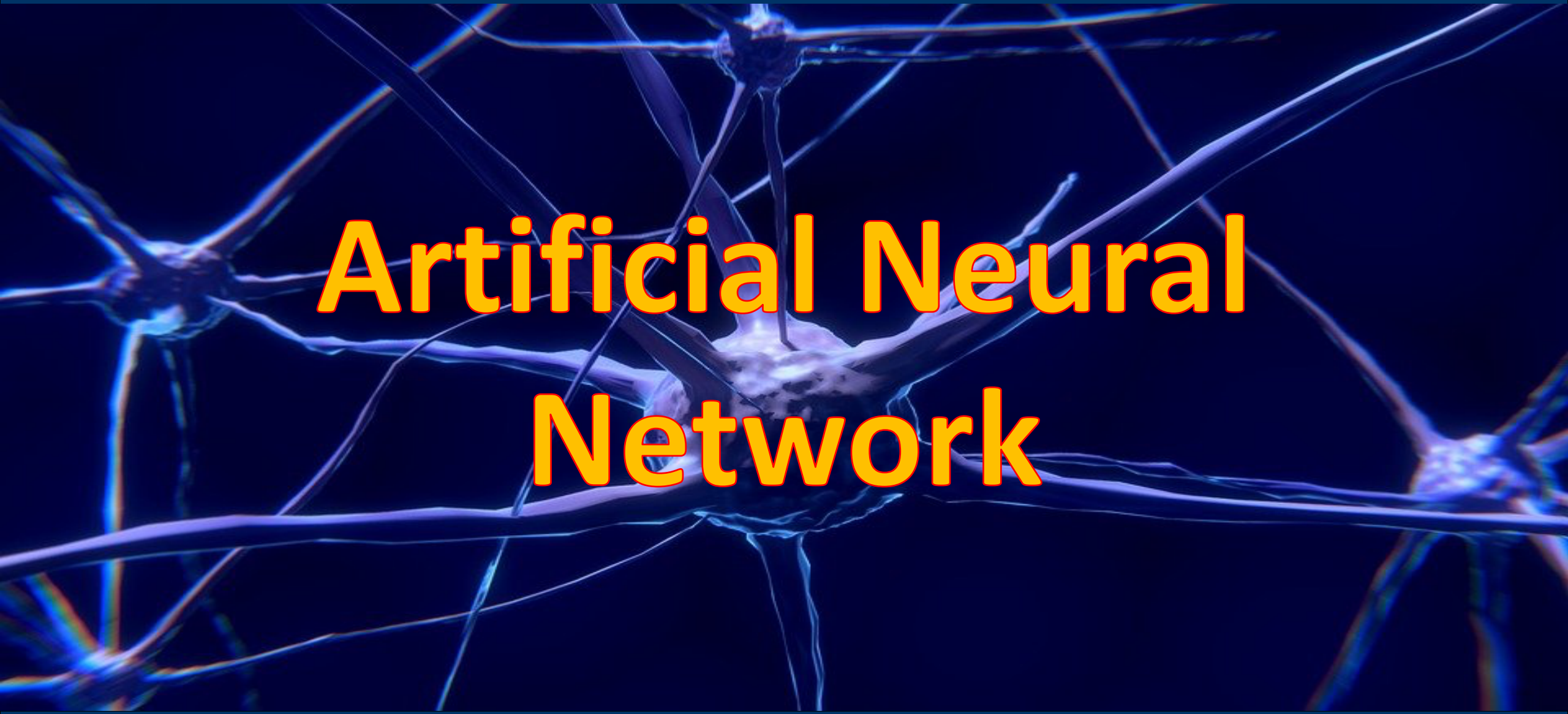
Advanced Machine Learning and Deep Learning

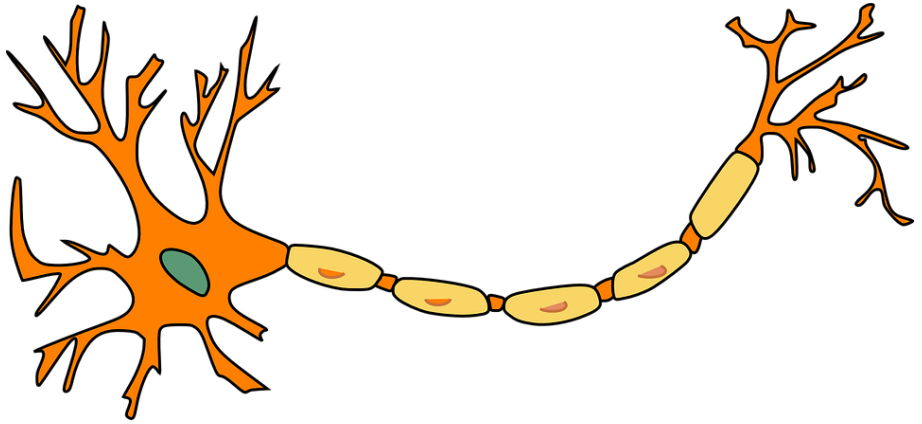
Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles

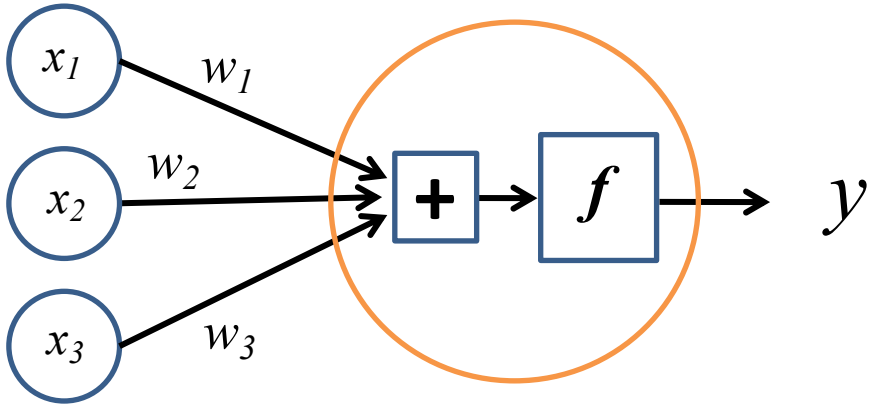

Artificial Neural
Network

Artificial Neural Networks

- **Artificial Neural Networks (ANNs)**, or simply called **Neural Networks** (**NN**) is a family of Machine Learning systems/models inspired by the human's nervous system, particularly the brain.
- **Neural Network** algorithms try to mimic the brain!
- **Artificial Neural Networks** are able to learn from data and generate models that get a number of inputs and map them to the desired outputs in an optimal way.

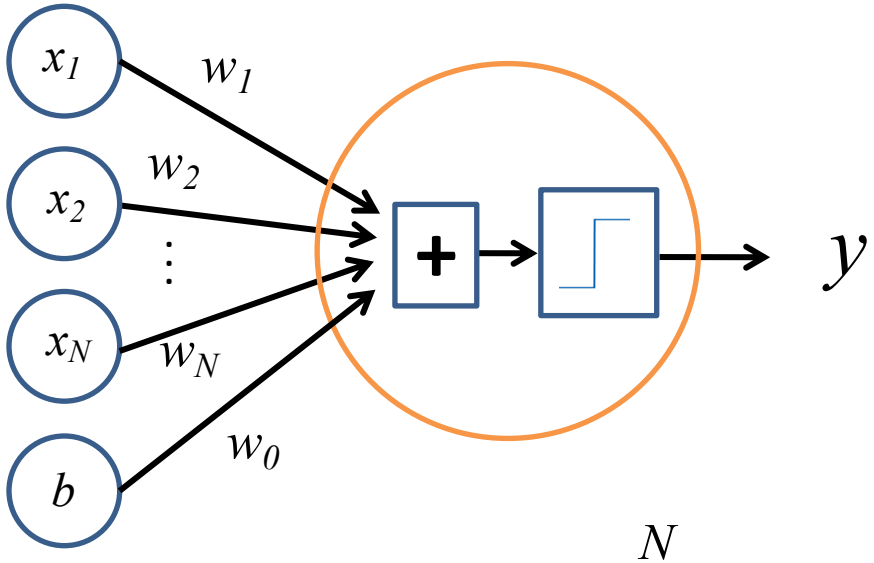
First Step in ANN: Simulating a Neuron

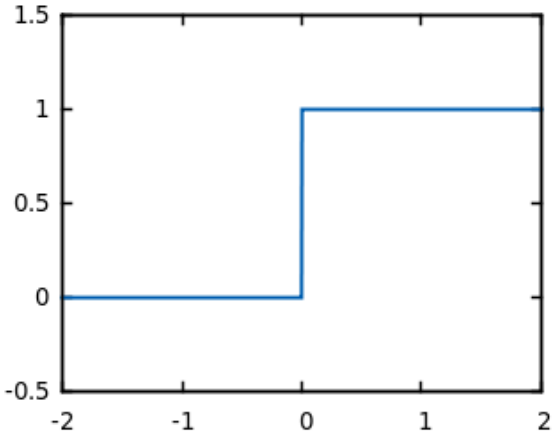




Neural Network Unit

- f should be a non-linear function. The simplest function that can be used is a ***step function***.



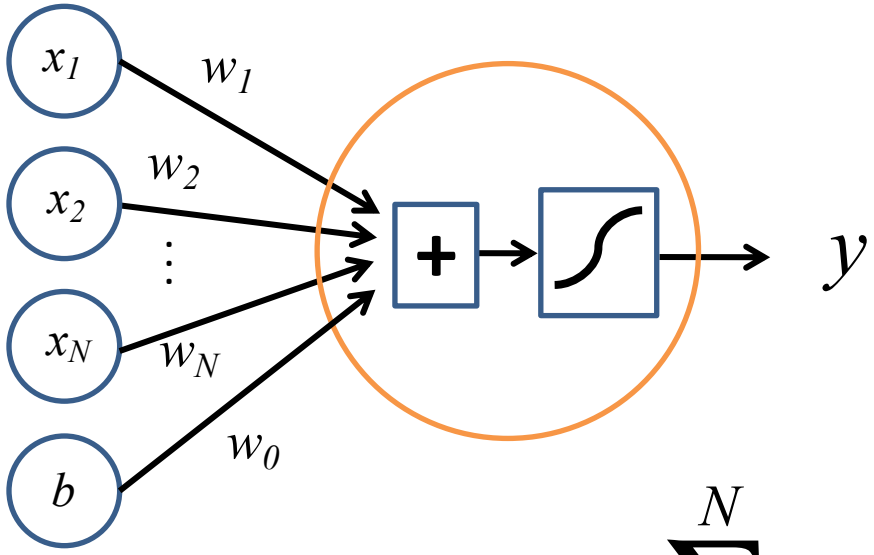


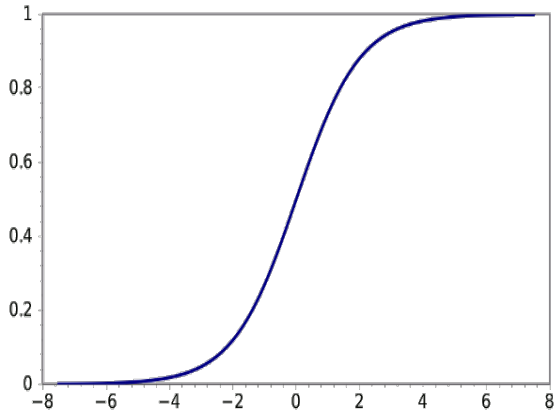
$y = f(x)$

$$y = f(w_0 b + \sum_{i=1}^{N} x_i w_i)$$

Neural Network Unit

- A popular choice for f is ***logistic sigmoid function***.



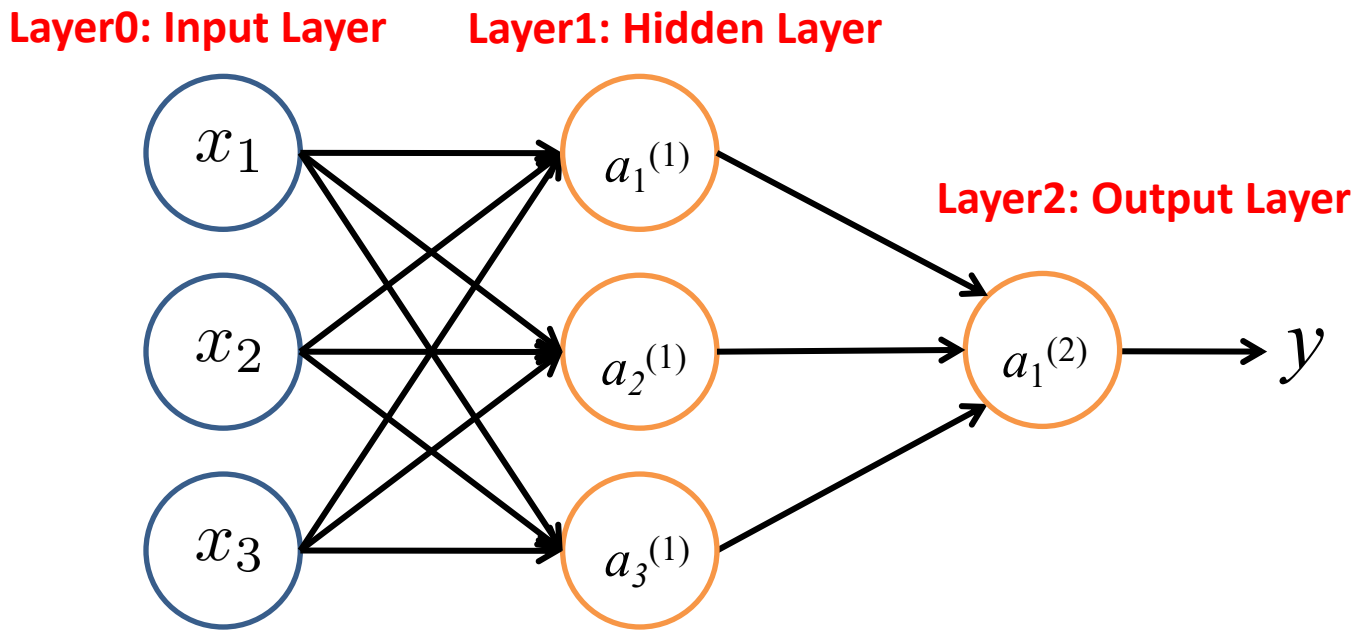


$y = g(x)$

$$y = g(w_0 b + \sum_{i=1}^{N} x_i w_i)$$

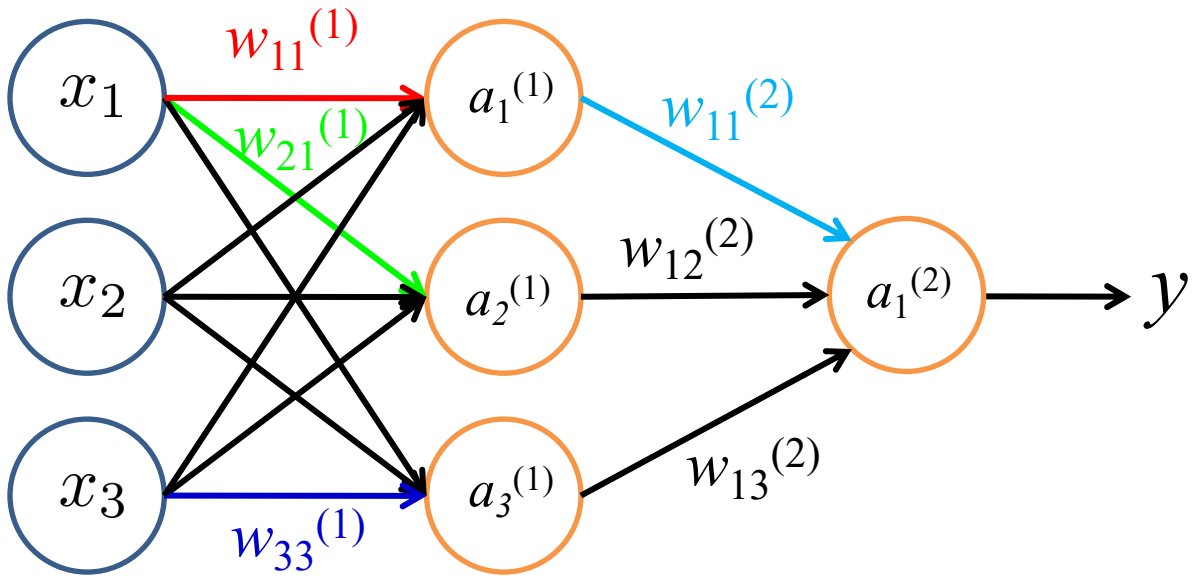
Neural Networks

- Similar to the human brain, an artificial neural network is formed by connecting multiple artificial neurons.
- The most common structure of connecting neurons into a network is by **layers**.
- The first layer is called **Input Layer**. The input layer only distribute the inputs and perform **no computation**.
- The last layer is called **Output Layer**.
- The middle layers are called **Hidden Layers**.

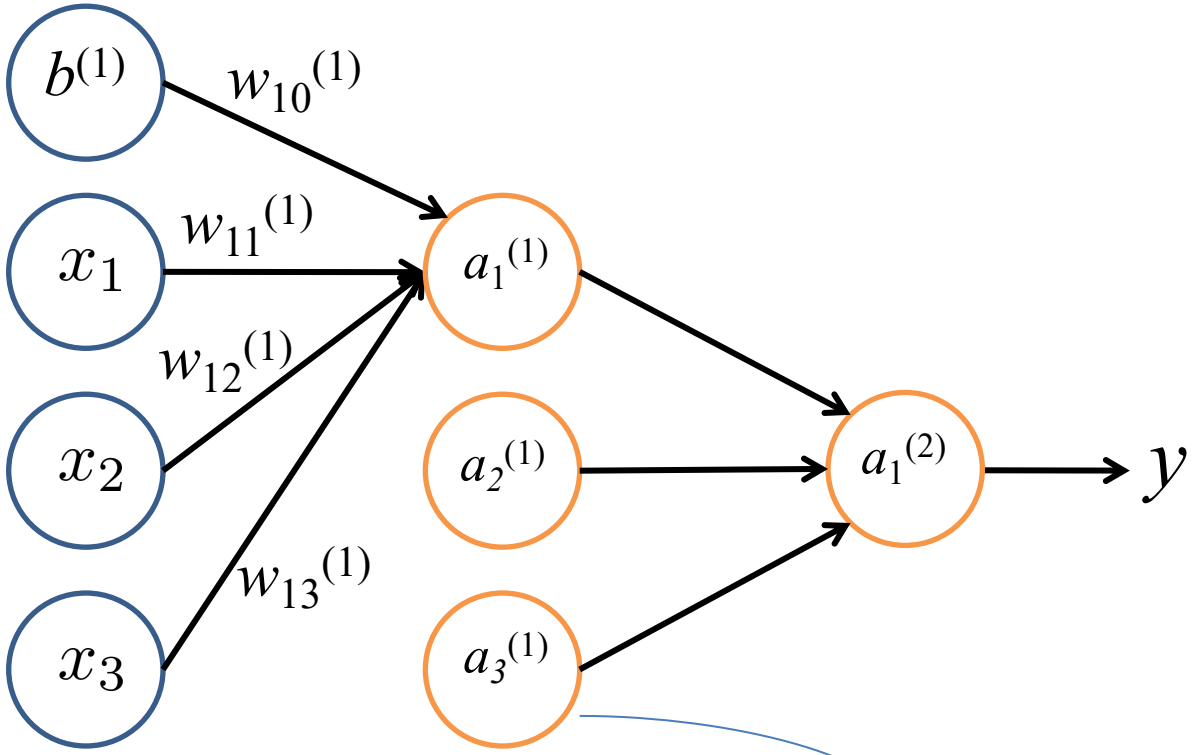


Notations

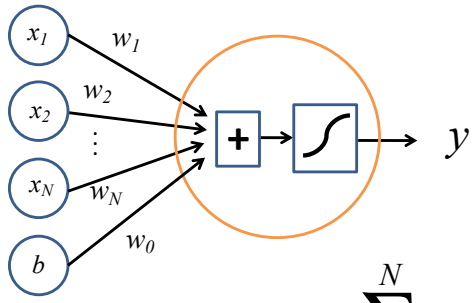
- $a_i^{(l)}$ = "***output of activation function***" of unit i in hidden layer l.
- $w_{ij}^{(l)}$ = "***weight***" of path from unit j in layer l-1 to unit i in layer l
- $W^{(l)}$ = "***matrix of weights***" from layer l-1 to layer l .
- $b^{(l)}$ = "***Bias***" in layer l .



The output of the first unit of hidden layer 1



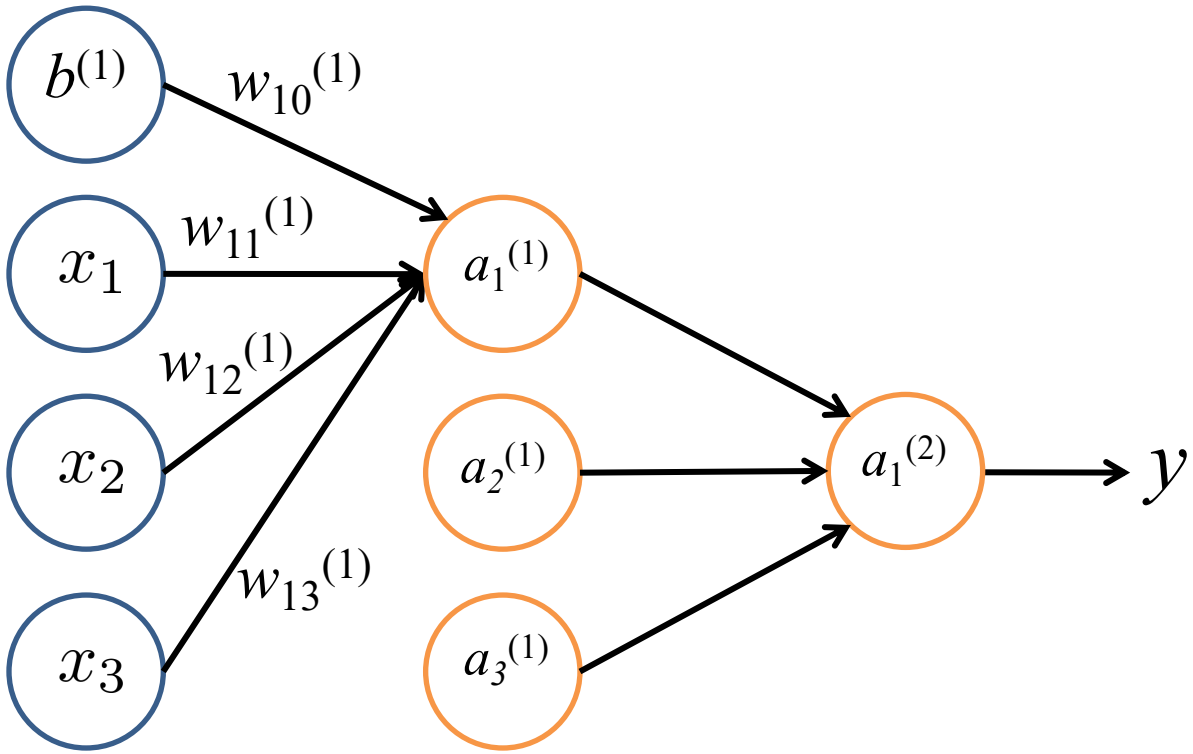
$a_1^{(1)} = ?$



$$y = g(w_0 b + \sum_{i=1}^{N} x_i w_i)$$

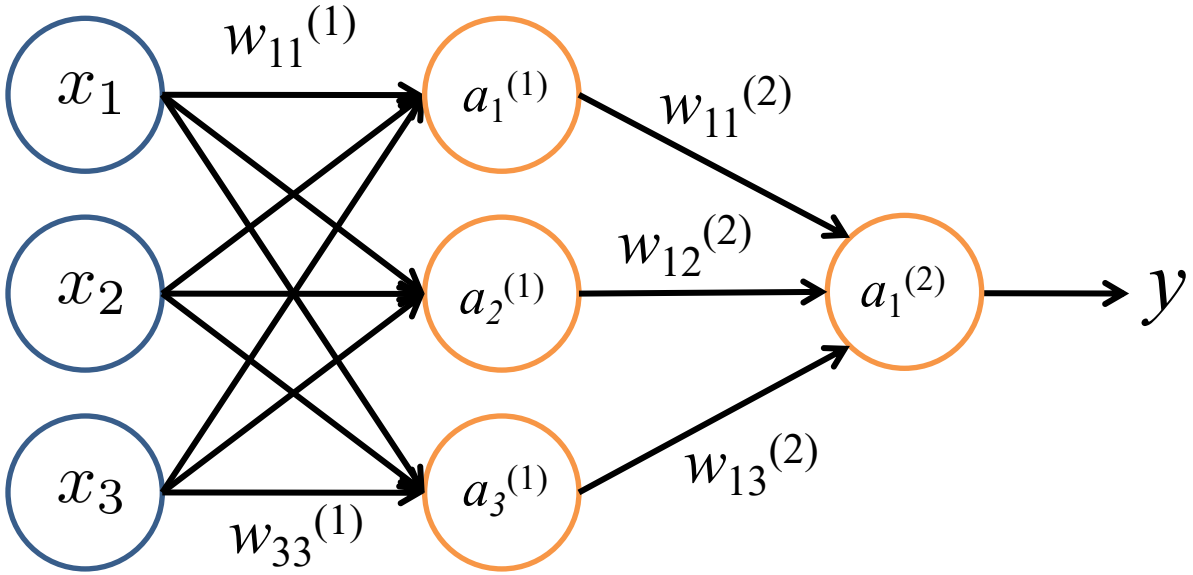
The output of the first unit of hidden layer 1

We usually hide the **bias** term to make the figures simpler. But, we should take it into account.



$$a_1^{(1)} = g\left(w_{10}^{(1)} b^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3\right)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$



$$a_1^{(1)} = g\left(w_{10}^{(1)} b^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3\right)$$

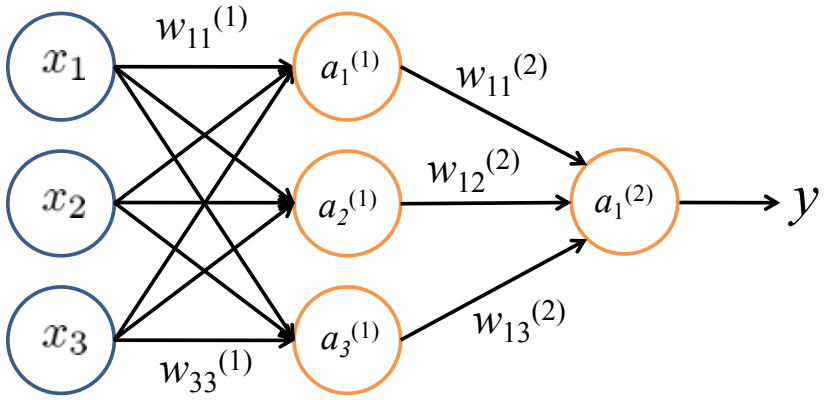
$$a_2^{(1)} = g\left(w_{20}^{(1)} b^{(1)} + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3\right)$$

$$a_3^{(1)} = g\left(w_{30}^{(1)} b^{(1)} + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3\right)$$

$$y = a_1^{(2)} = g\left(w_{10}^{(2)} b^{(2)} + w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + w_{13}^{(2)} a_3^{(1)}\right)$$

Vectorized Implementation of Forward Propagation

Forward propagation: Vectorized implementation



$$a_1^{(1)} = g\left(w_{10}^{(1)} b^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3\right)$$

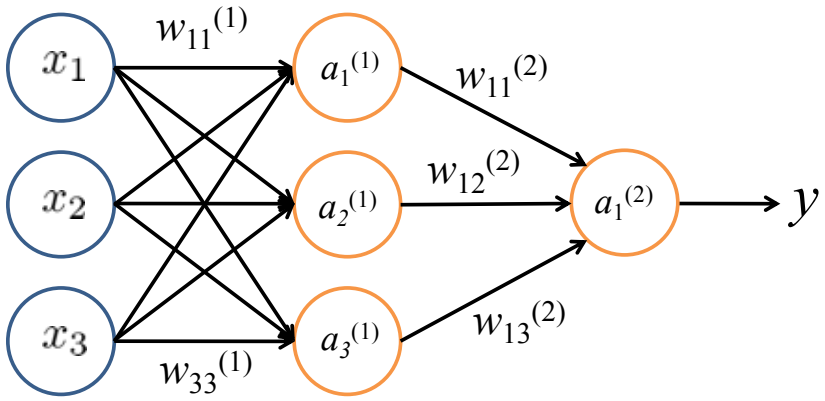
$$a_2^{(1)} = g\left(w_{20}^{(1)} b^{(1)} + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3\right)$$

$$a_3^{(1)} = g\left(w_{30}^{(1)} b^{(1)} + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3\right)$$

$$\boldsymbol{a}^{(1)} = \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} \qquad W^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{30}^{(1)} & w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} b \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{a}^{(1)} = g(\boldsymbol{W}^{(1)}\boldsymbol{x})$$

Forward propagation: Vectorized implementation



$$a_1^{(1)} = g\left(w_{10}^{(1)} b^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3\right)$$

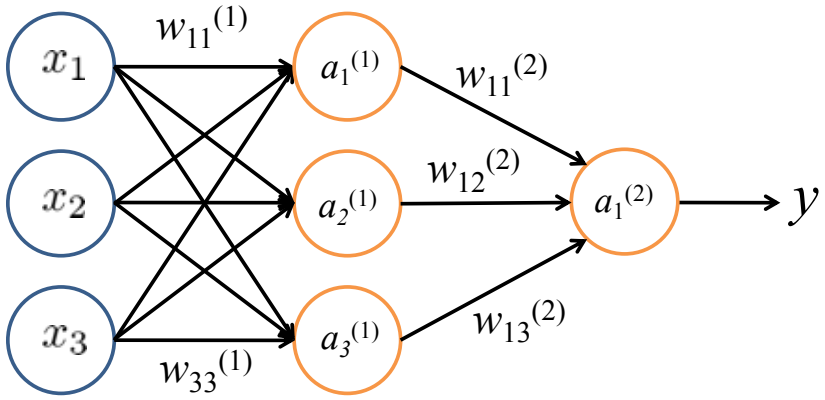
$$a_2^{(1)} = g\left(w_{20}^{(1)} b^{(1)} + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3\right)$$

$$a_3^{(1)} = g\left(w_{30}^{(1)} b^{(1)} + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3\right)$$

$$\boldsymbol{a}^{(1)} = \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} \qquad W^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{30}^{(1)} & w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} b \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{a}^{(1)} = g(\boldsymbol{W}^{(1)}\boldsymbol{x})$$

Forward propagation: Vectorized implementation



$$a_1^{(1)} = g\left(w_{10}^{(1)} b^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3\right)$$

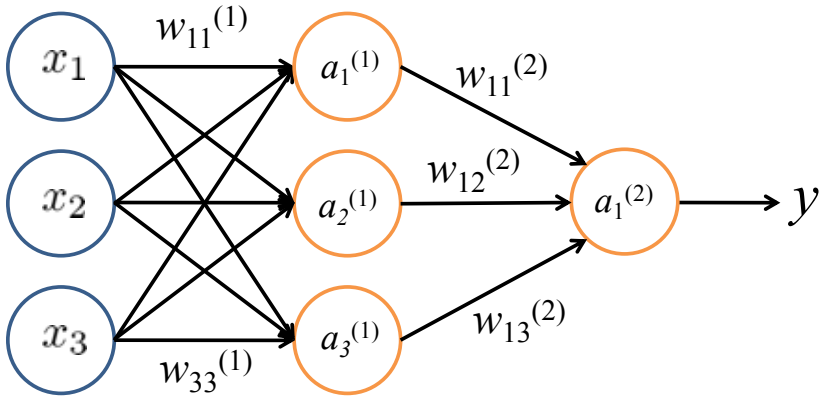
$$a_2^{(1)} = g\left(w_{20}^{(1)} b^{(1)} + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3\right)$$

$$a_3^{(1)} = g\left(w_{30}^{(1)} b^{(1)} + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3\right)$$

$$\boldsymbol{a}^{(1)} = \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} \qquad W^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{30}^{(1)} & w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} b \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{a}^{(1)} = g(\boldsymbol{W}^{(1)}\boldsymbol{x})$$

Forward propagation: Vectorized implementation



$$a_1^{(1)} = g\left(w_{10}^{(1)} b^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3\right)$$

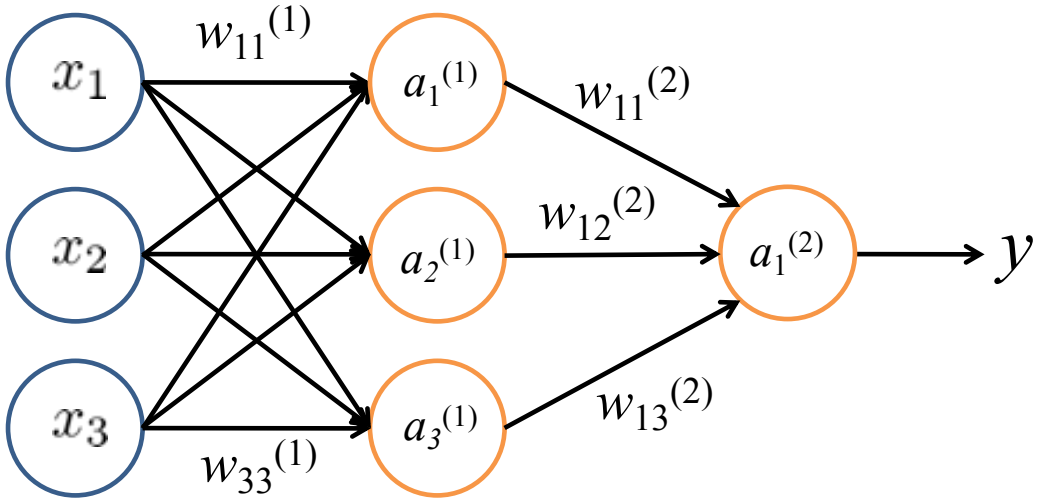
$$a_2^{(1)} = g\left(w_{20}^{(1)} b^{(1)} + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3\right)$$

$$a_3^{(1)} = g\left(w_{30}^{(1)} b^{(1)} + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3\right)$$

$$\boldsymbol{a}^{(1)} = \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} \qquad W^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{30}^{(1)} & w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} b \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{a}^{(1)} = g(\boldsymbol{W}^{(1)}\boldsymbol{x})$$

Forward propagation: Vectorized implementation



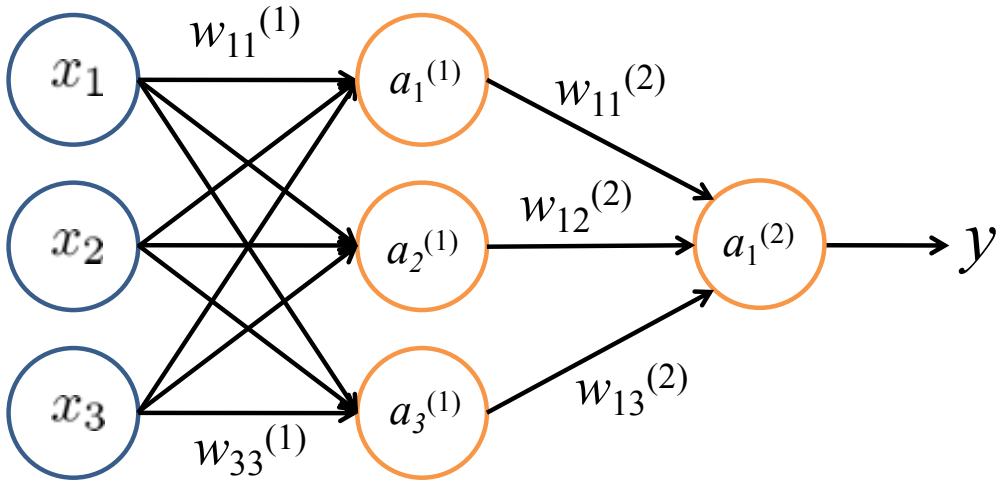
$$\boldsymbol{x} = \begin{bmatrix} b \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{a}^{(1)} = \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix}$$

Thus,

$$\boldsymbol{a}^{(1)} = g(\boldsymbol{W}^{(1)}\boldsymbol{x})$$

$$W^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{30}^{(1)} & w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix}$$

Forward propagation: Vectorized implementation

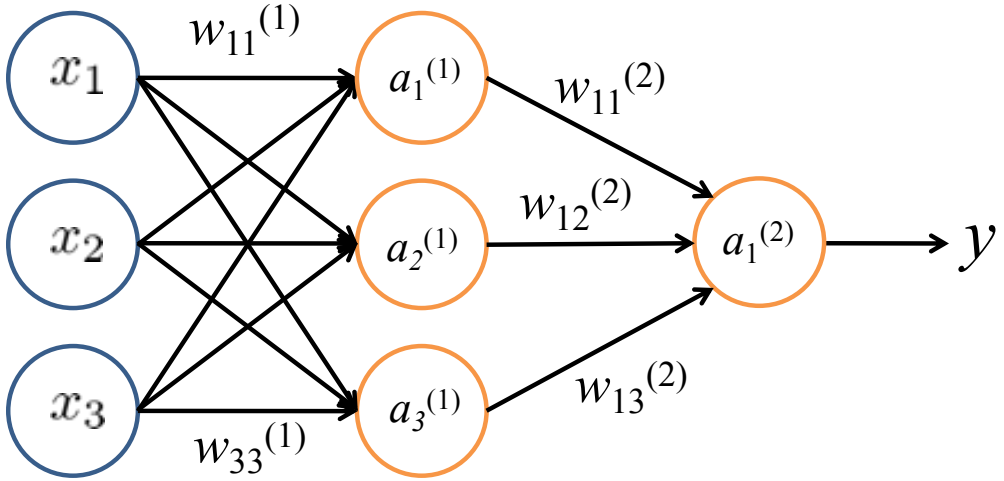


Similarly: $y = a_1^{(2)} = g\,(w_{10}^{(2)}\, b^{(2)} + w_{11}^{(2)}\, a_1^{(1)} + w_{12}^{(2)}\, a_2^{(1)} + w_{13}^{(2)}\, a_3^{(1)}\,)$

$$y = \boldsymbol{a}^{(2)} = g(\boldsymbol{W}^{(2)}\, \boldsymbol{a}^{(1)})$$

Note: in last equation, we have to add $a_0^{(1)} = b^{(2)}$ to take into account the bias of second layer.

Forward propagation: Vectorized implementation



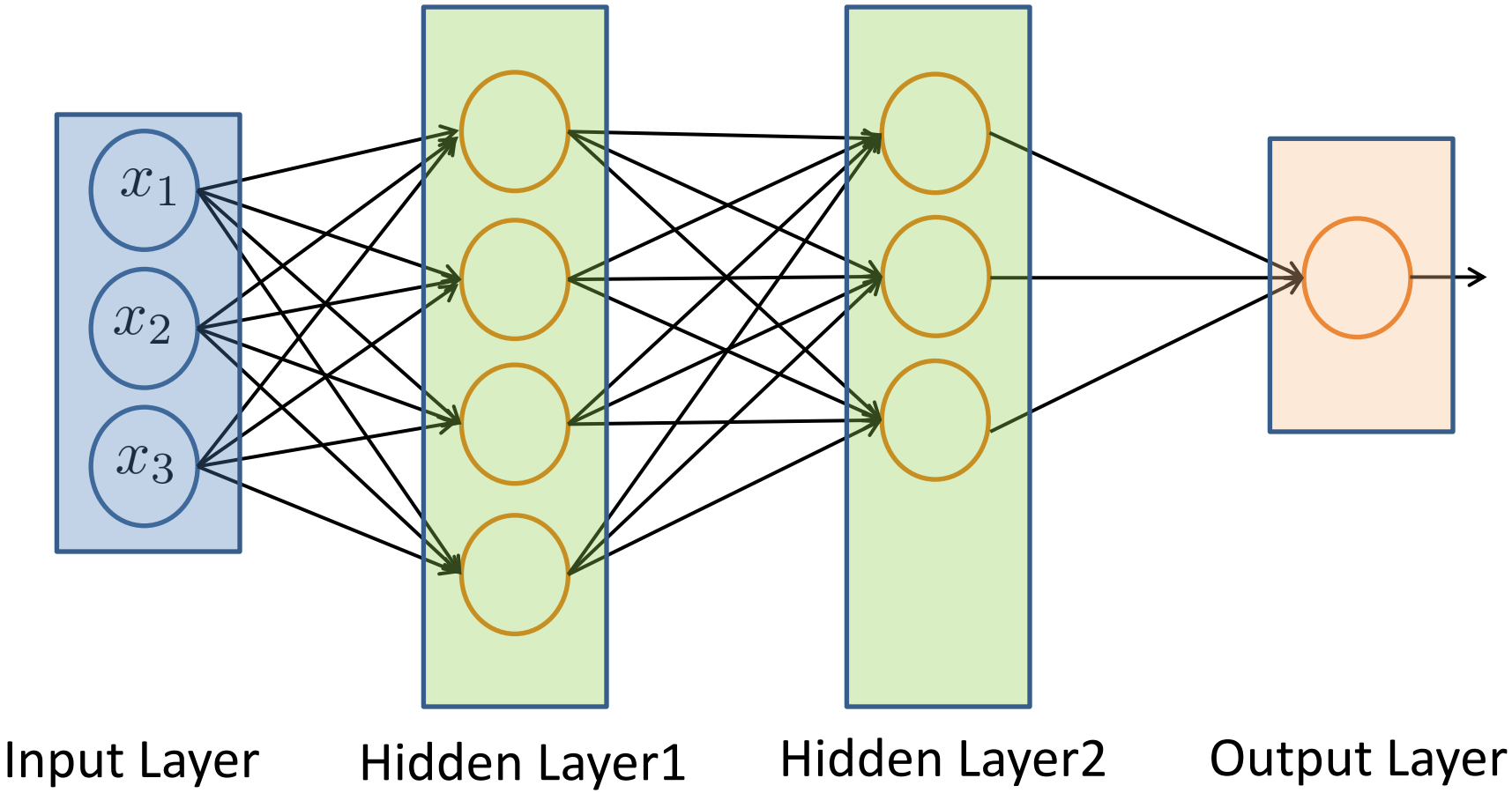
Similarly,

$$\boldsymbol{a}^{(1)} = g(\boldsymbol{W}^{(1)}\boldsymbol{x})$$

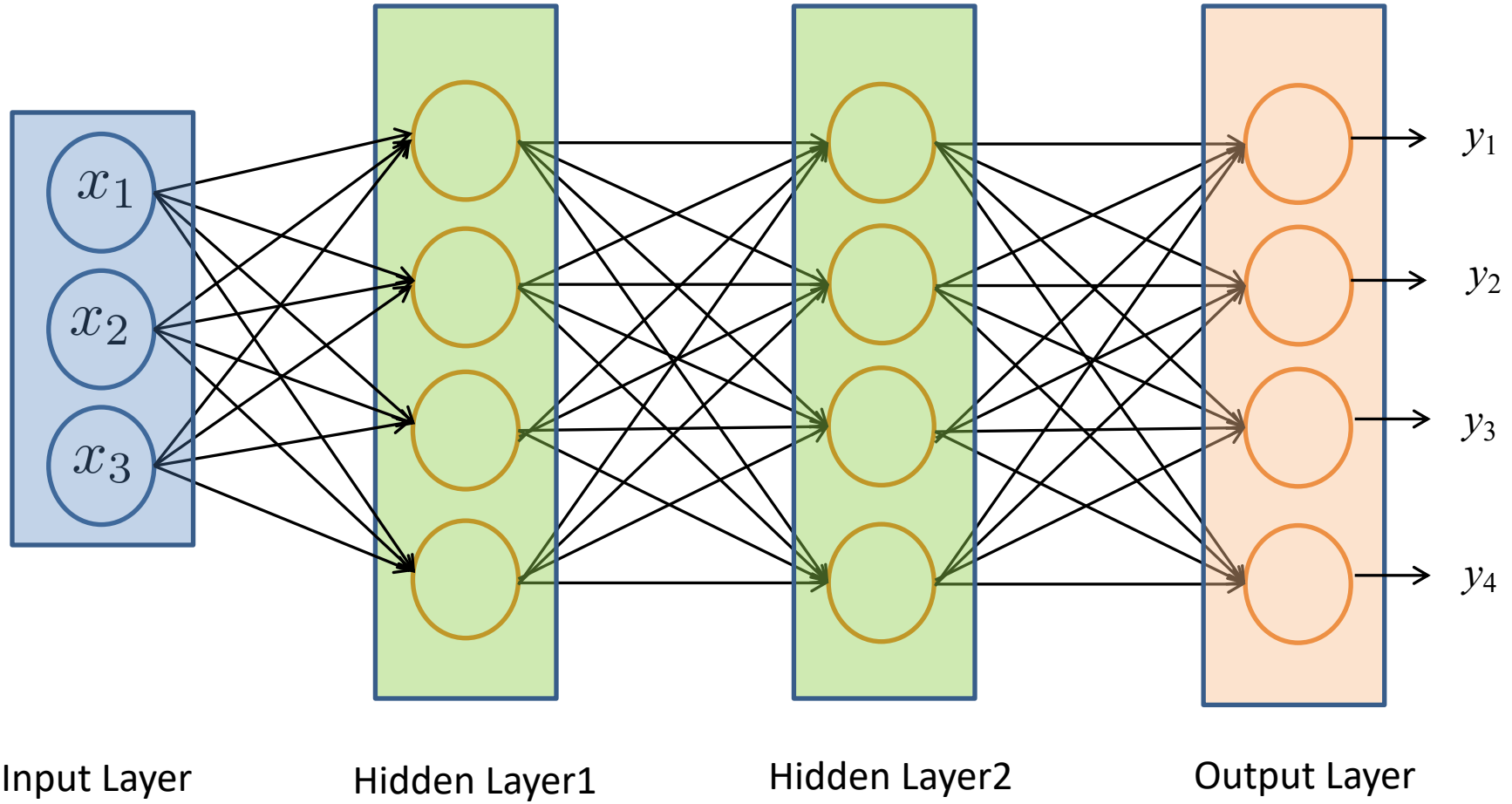
$$y = \boldsymbol{a}^{(2)} = g(\boldsymbol{W}^{(2)}\,\boldsymbol{a}^{(1)})$$

Note: in last equation, we have to add $a_0^{(1)} = b^{(2)}$ to take into account the bias of second layer.

Example: A network with more than 1 hidden layer!



Example: A Multiclass Network!



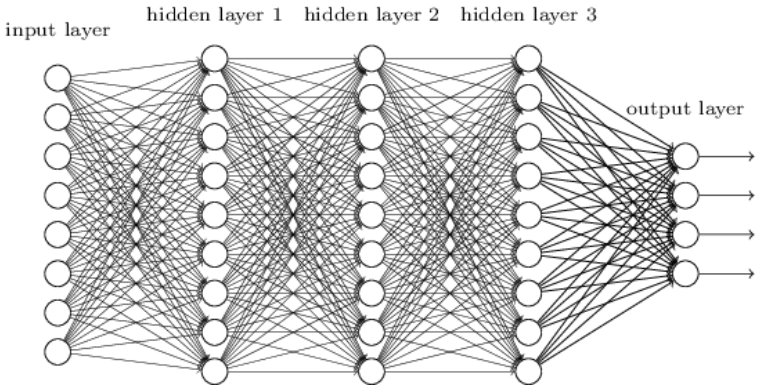
Example: A Multiclass Network!

Cat

Dog

Horse

Bird



$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Cat Dog Horse Bird

Example: A Multiclass Network!

Cat

Dog

Horse

Bird

Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

$$y^{(i)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

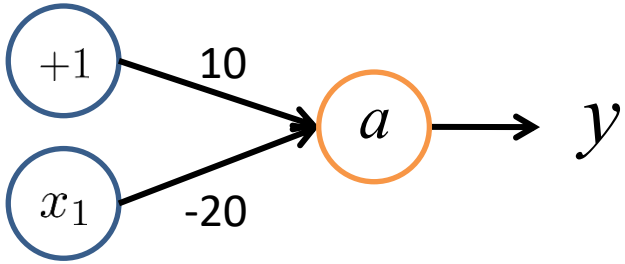
Cat Dog Horse Bird

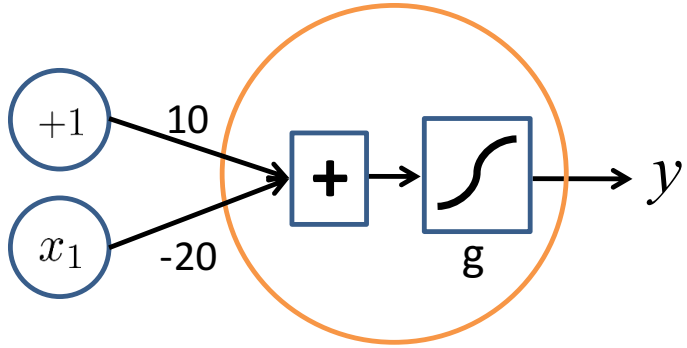
Muffin or Puppy???

[Image Ref]: www.boredpanda.com

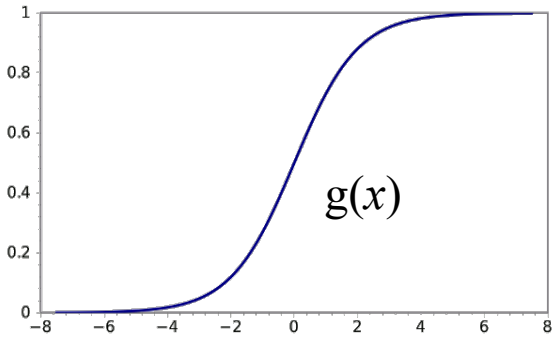
One Simple Application:
Building Logical Operators Using ANN

Negation (Logical NOT)





$$y = g(10\text{-}20x_1)$$



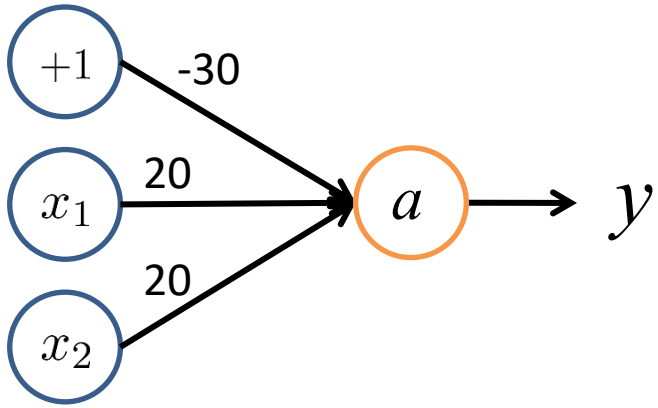
x_1=0 ➔ $y = g(10) \approx 1$

x_1=1 ➔ $y = g(\text{-}10) \approx 0$

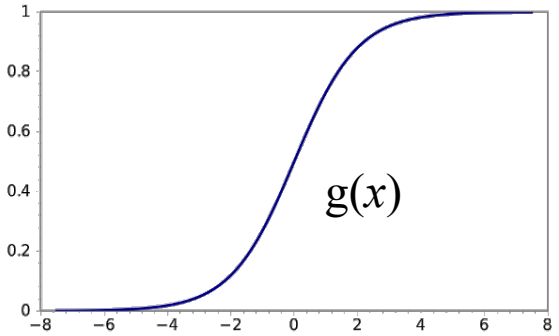
x_1	y
0	1
1	0

[Example Ref]: Andrew Ng, Machine Learning, Stanford University

Conjunction (Logical AND)



$$y = g(20x_1+20x_2-30)$$



$x_1=0, x_2=0 \rightarrow y = g(-30) \approx 0$

$x_1=1, x_2=0 \rightarrow y = g(-10) \approx 0$

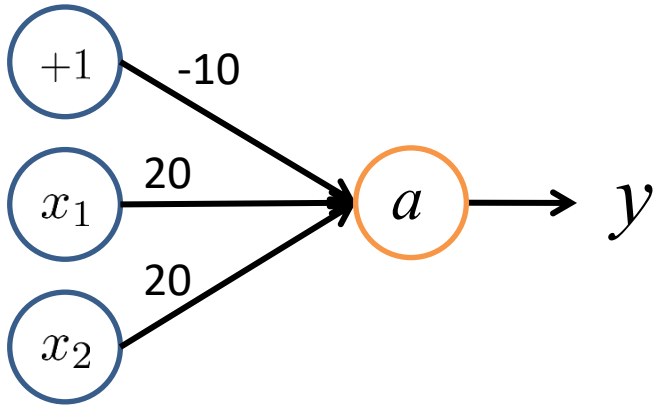
$x_1=0, x_2=1 \rightarrow y = g(-10) \approx 0$

$x_1=1, x_2=1 \rightarrow y = g(+10) \approx 1$

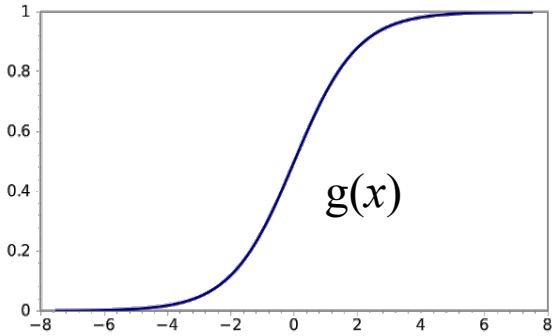
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

[Example Ref]: Andrew Ng, Machine Learning, Stanford University

Disjunction (Logical OR)



$$y = g(20x_1+20x_2-10)$$



$x_1=0, x_2=0 \rightarrow y = g(-10) \approx 0$

$x_1=1, x_2=0 \rightarrow y = g(+10) \approx 1$

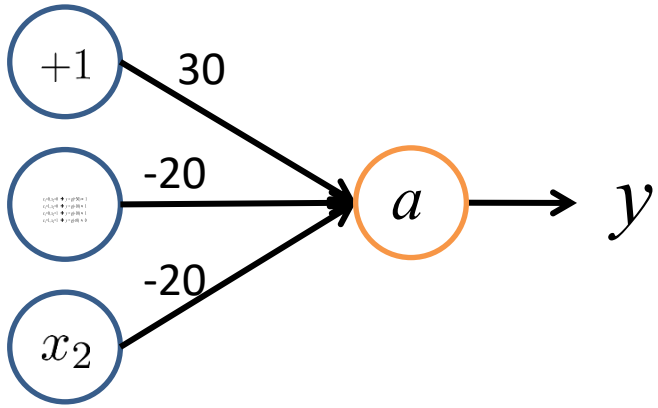
$x_1=0, x_2=1 \rightarrow y = g(+10) \approx 1$

$x_1=1, x_2=1 \rightarrow y = g(+30) \approx 1$

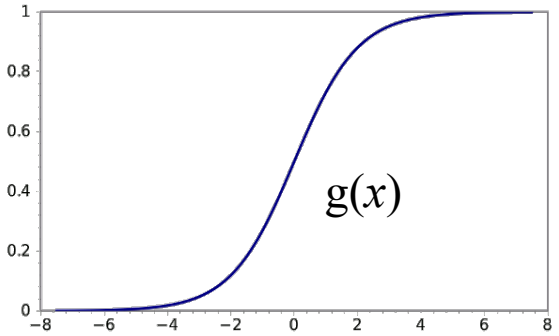
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

[Example Ref]: Andrew Ng, Machine Learning, Stanford University

Disjunction (Logical NAND = NOT(AND))



$$y = g(-20x_1 - 20x_2 + 30)$$



$x_1=0, x_2=0 \rightarrow y = g(+30) \approx 1$

$x_1=1, x_2=0 \rightarrow y = g(+10) \approx 1$

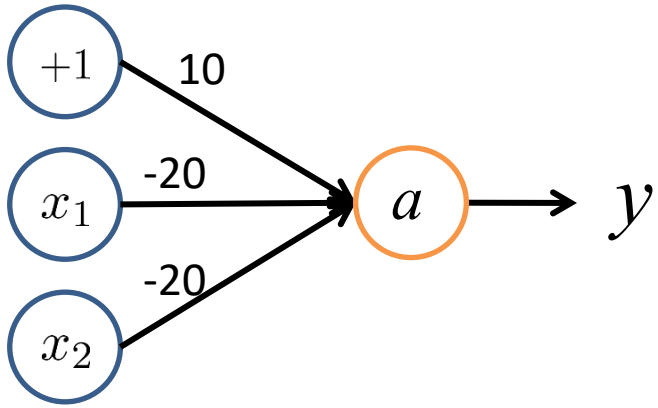
$x_1=0, x_2=1 \rightarrow y = g(+10) \approx 1$

$x_1=1, x_2=1 \rightarrow y = g(-10) \approx 0$

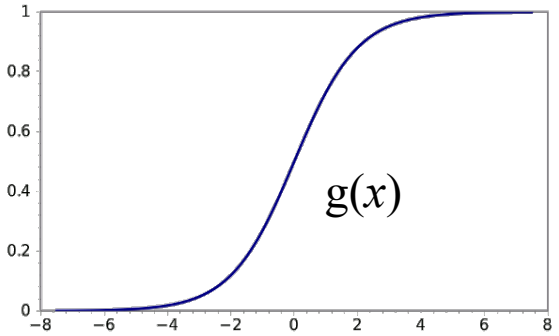
x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

[Example Ref]: Andrew Ng, Machine Learning, Stanford University

Joint Denial (Logical NOR = NOT(OR))



$$y = g(-20x_1 - 20x_2 + 10)$$



$x_1=0, x_2=0 \rightarrow y = g(+10) \approx 1$

$x_1=1, x_2=0 \rightarrow y = g(-10) \approx 0$

$x_1=0, x_2=1 \rightarrow y = g(-10) \approx 0$

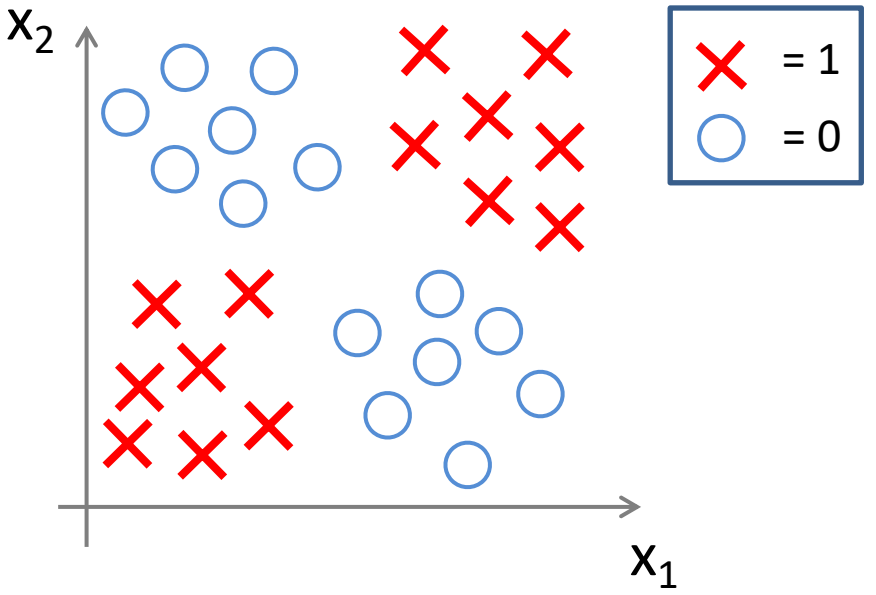
$x_1=1, x_2=1 \rightarrow y = g(-30) \approx 0$

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

[Example Ref]: Andrew Ng, Machine Learning, Stanford University

A Simple Classification Example

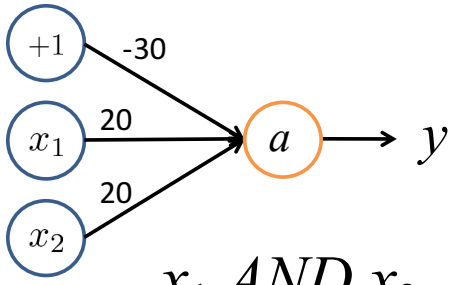
- In this figure, can we perform classification using a linear model?
- Can we take advantage of a logical operator for classification?
- ***Answer***: $y = x_1 \text{ XNOR } x_2$



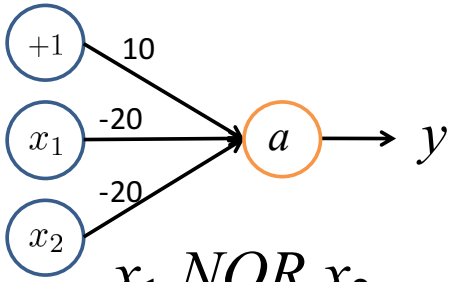
$$y = x_1 \text{ XNOR } x_2$$
$$= (x_1 \text{ AND } x_2) \text{ OR } (x_1 \text{ NOR } x_2)$$

$x_1 \approx$	$x_2 \approx$	y
0	0	1
0	1	0
1	0	0
1	1	1

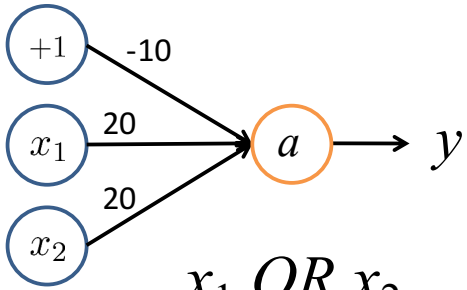
[Example Ref]: Andrew Ng, Machine Learning, Stanford University



x_1 *AND* x_2

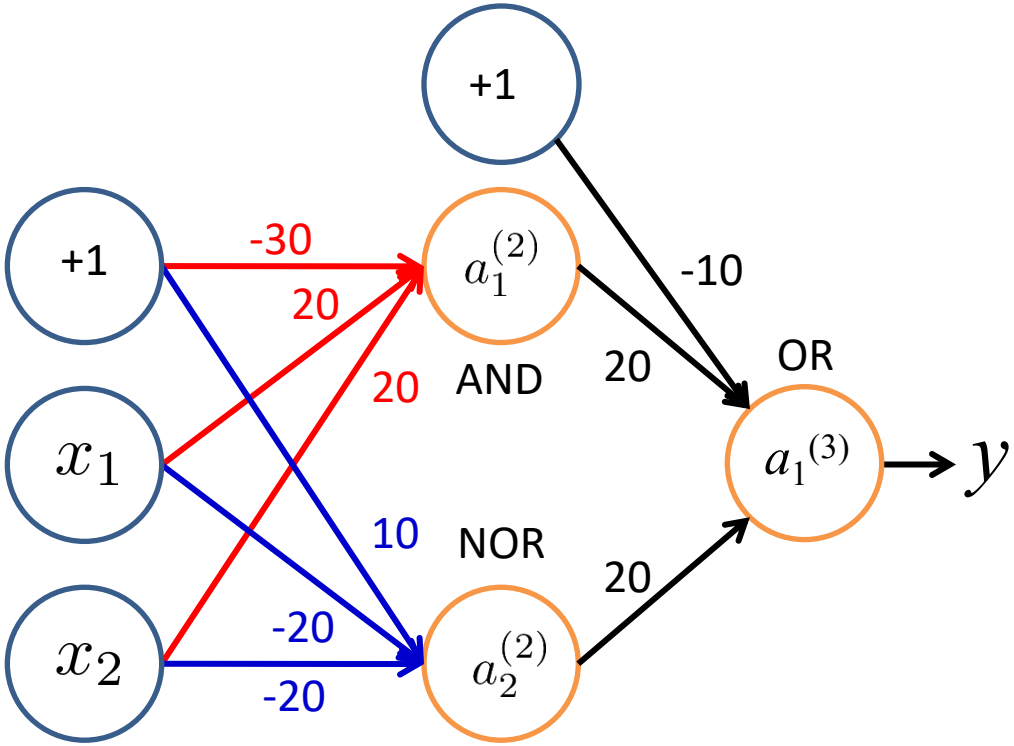


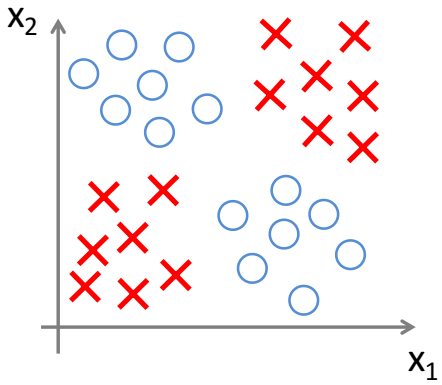
x_1 *NOR* x_2



x_1 *OR* x_2

Answer: $y = (x_1 \text{ XNOR } x_2) = (x_1 \text{ AND } x_2) \text{ OR } (x_1 \text{ NOR } x_2)$





x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	y
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Thank You!

Questions?