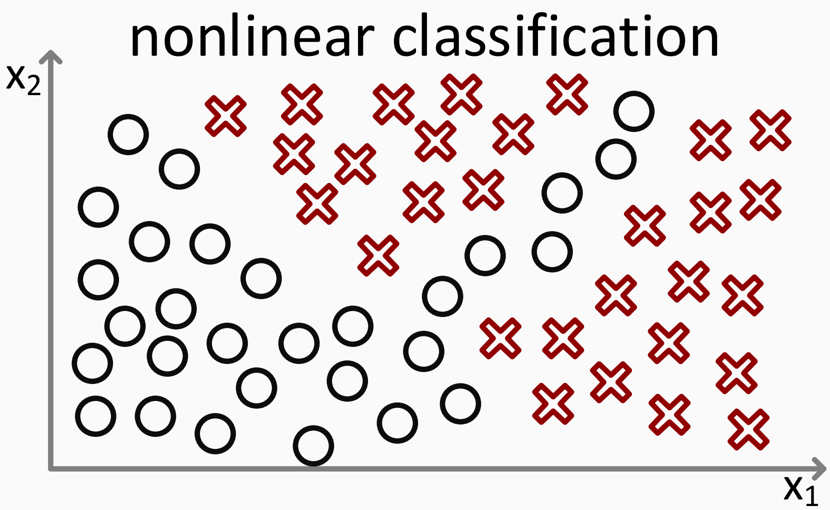
artificial neural networks ❖ representation

motivations

nonlinear hypothesis

with only two features as illustrated below, a traditional logistic regression classification method would suffice to fit the data accurately:



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \cdots)$$

assuming multiple features like housing data below, if $n = 100$, there would be $\approx$ 5000 features; this number grows exponentially with each additional feature.

the quadratic function above becomes computational features:

an option would be to take a subset of the features, but this would not be capable of predicting more intricate decision boundaries outside of basic ellipses etc. an additional option would be to use multi polynomial features but this would also result in very large amounts of required feature space.

x_1 = size
x_2 = # bedrooms
x_3 = # floors
x_4 = age
...
x_{100}

image recognition purposes are a function of this algorithm

this method uses pixel intensities as a function of n. (50 x 50 pixel images → 2500 pixels n = 2500) (n = 7500 if using rgb)

with quadratic features (x_i * x_j): $\approx$ 3 million features

Suppose you are learning to recognize cars from 100×100 pixel images (grayscale, not RGB). Let the features be pixel intensity values. If you train logistic regression including all the quadratic terms ($x_i x_j$) as features, about how many features will you have?

50 million (5x10^7)

neurons and the brain

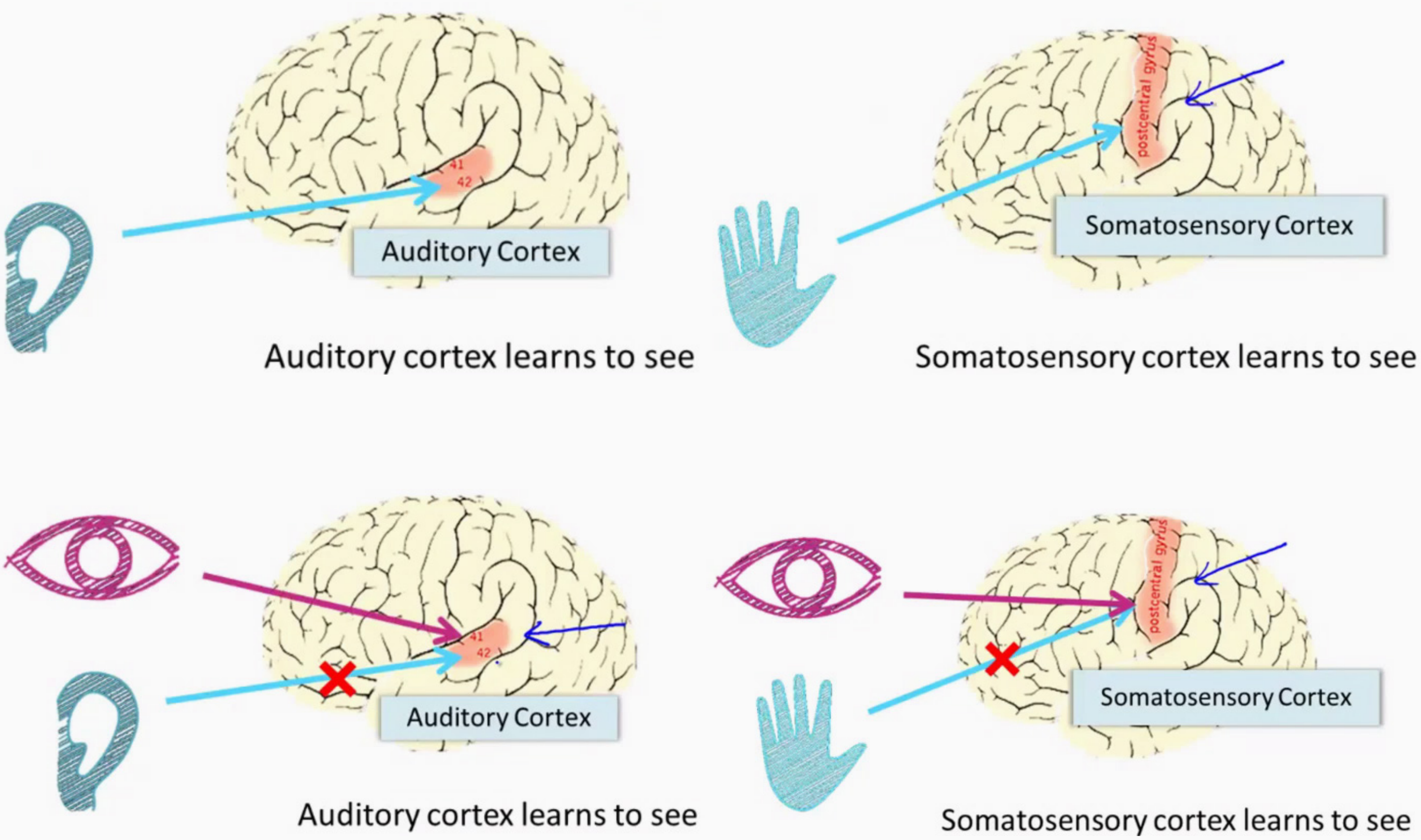
neural networks originated from algorithms that attempt to emulate the human brain. neural networks were used widely in the 80s and 90s; popularity diminished in the late 1990s.

a recent resurgence into artificial intelligence has curated state-of-the-art techniques for many applications with massive success.

the theory of the brain operating of neuron activity claims the lack of many learning algorithms to perform complete function.

the "one learning algorithm hypothesis"

the same brain tissues can process any senses necessary; areas of the brain can repurpose themselves.

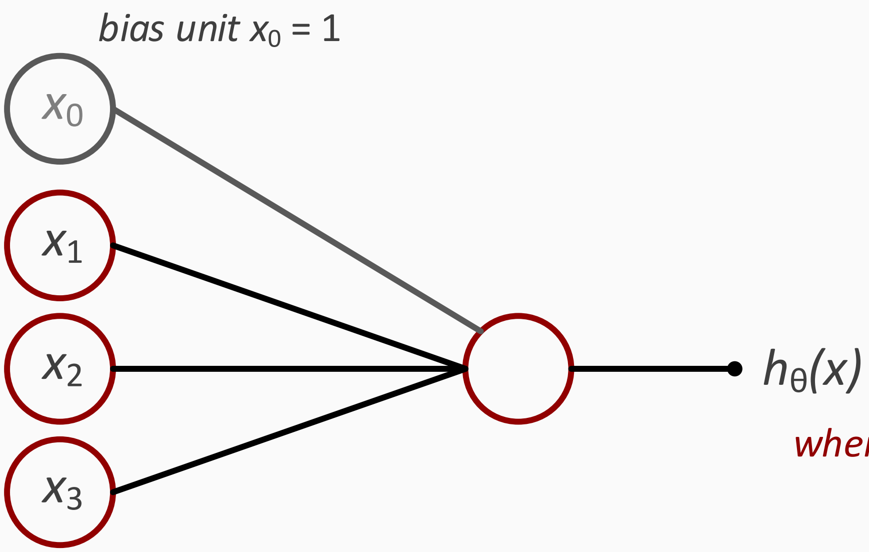


neural networks

model representations i

In artificial neural network, neurons in the brain are modeled as simple logistic unit:

single neuron model: logistic unit



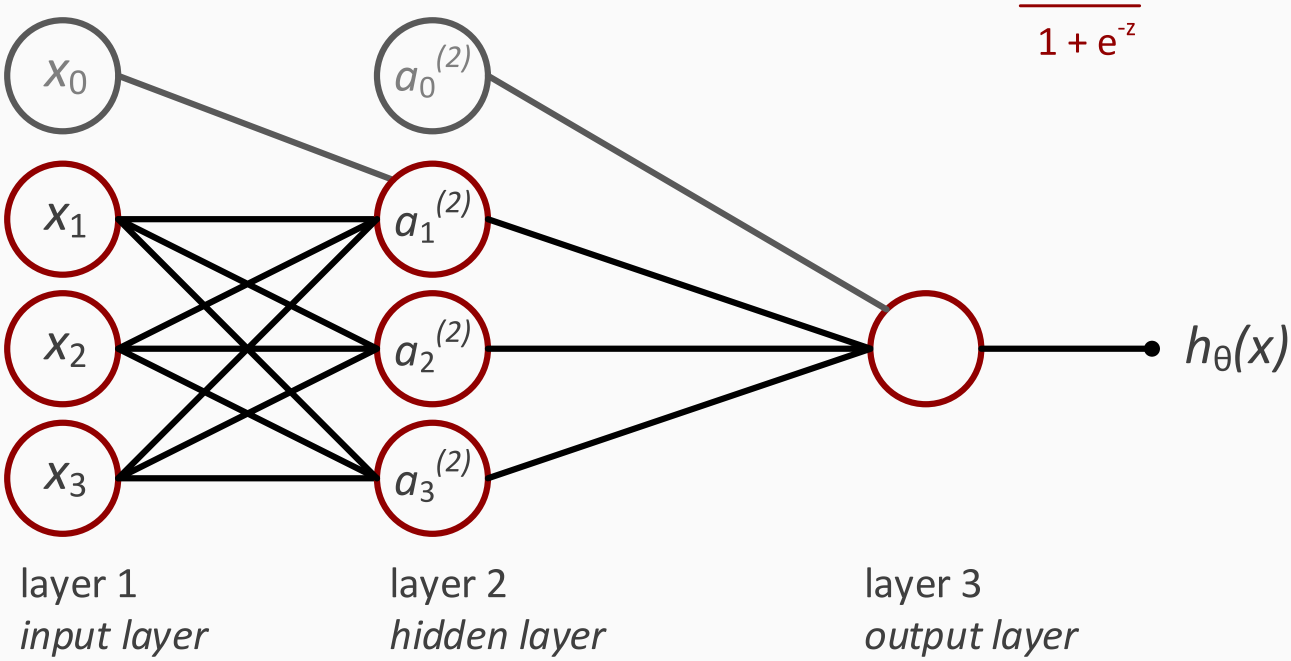
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

where: $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$

Sigmoid (logistic) activation function

neural network

where: $g(z) = \dfrac{1}{1 + e^{-z}}$



computational method for neural network

$a_i^{(j)}$ = the **activation** of unit $\boldsymbol{i}$ in layer $\boldsymbol{j}$

$\theta^{(j)}$ = the matrix of weights controlling function mapping from layer $\boldsymbol{j}$ to layer $\boldsymbol{j}$ + 1

$$a_1^{(2)} = g\left(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\right)$$

$$a_2^{(2)} = g\left(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\right)$$

$$a_3^{(2)} = g\left(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\right)$$

$$h_\theta(x) = g\left(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}\right)$$

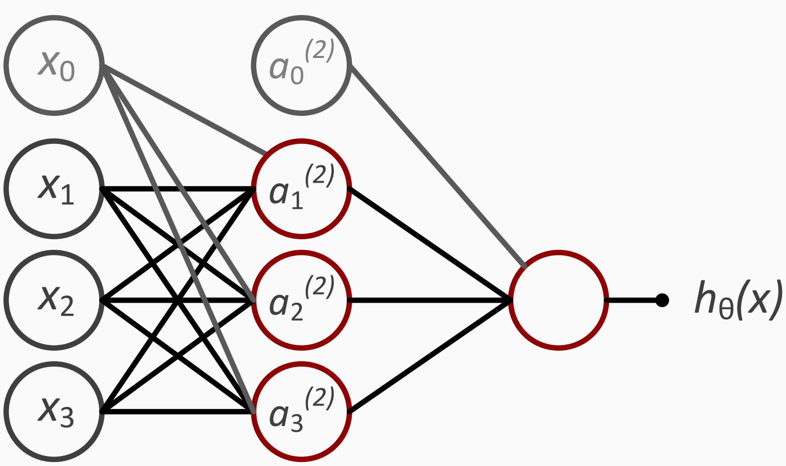
If network has $\boldsymbol{s_j}$ units in layer $\boldsymbol{j}$, $\boldsymbol{s_{j+1}}$ units in layer $\boldsymbol{j+1}$, then $\boldsymbol{\theta^{(j)}}$ will be a matrix with a dimension of $\boldsymbol{s_{j+1} \times (s_j + 1)}$

model representations ii

The superscripts of the variables represent the values associated to the neural network layer.

Forward propagation: Vectorized implementation

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$



$$z^{(2)} = g\left(\theta^{(1)}a^{(1)}\right)$$

$$a^{(2)} = g\left(z^{(2)}\right)$$

To compensate for the bias unit:

Add $a_0^{(2)} = 1$

$$z^{(3)} = \theta^{(2)}a^{(2)}$$

$$h_\theta(x) = a^{(3)} = g\left(z^{(3)}\right)$$

$$a_1^{(2)} = g\left(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\right)$$

$$a_2^{(2)} = g\left(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\right)$$

$$a_3^{(2)} = g\left(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\right)$$

$$h_\theta(x) = g\left(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}\right)$$

$$a_1^{(2)} = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(z_2^{(2)}\right)$$

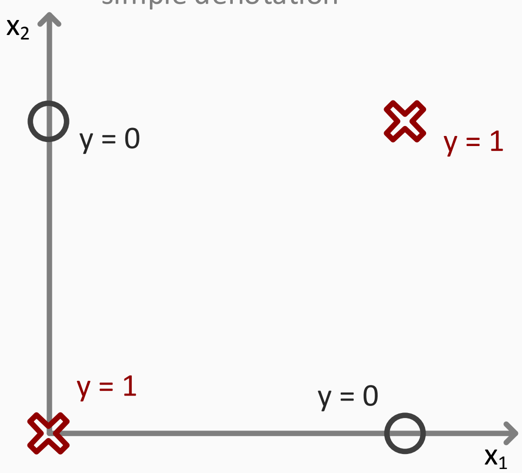
application

examples and intuitions i

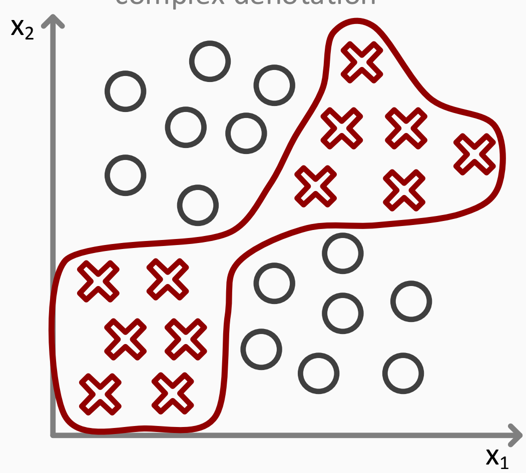
nonlinear classification: XOR/XNOR

x_1, x_2 are binary (0 or 1)

simple denotation



complex denotation

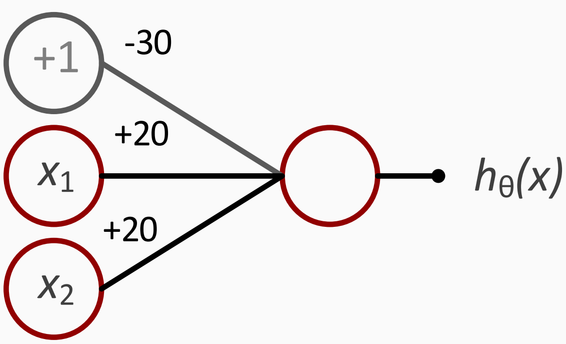


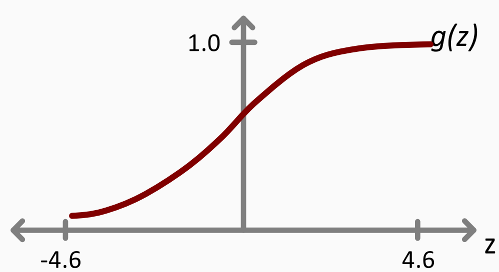
$y = x_1 \text{ XOR } x_2$ *(true only if exactly 1 of x_1 **or** x_2 = 1)*

$$\left.\begin{array}{l} x_1 \text{ XNOR } x_2 \\ \text{NOT } (x_1 \text{ XOR } x_2) \end{array}\right\} \text{same}$$

*(true if x_1 **nor** x_2 = 1)*

example: AND





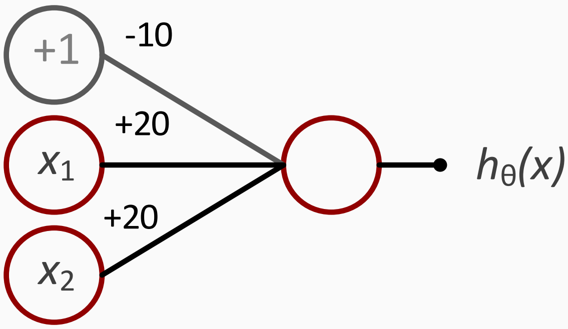
$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$

x_1	x_2	$h_\theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

The sigmoid function graphed above denotes *g(z)*

The example neural network computation with the **AND** logical above and the corresponding calculations in the table prove the logical **AND** function. The function is true when $h_\theta(x) \approx x_1 \text{ AND } x_2$. This can be seen in the results above when the computations of $h_\theta(x) = g(z)$ output a true value when both x_1 and x_2 return the values of true.

example: OR



$$h_\theta(x) = g(-10 + 20x_1 + 20x_2)$$

x_1	x_2	$h_\theta(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	$g(10) \approx 1$
1	1	$g(10) \approx 1$

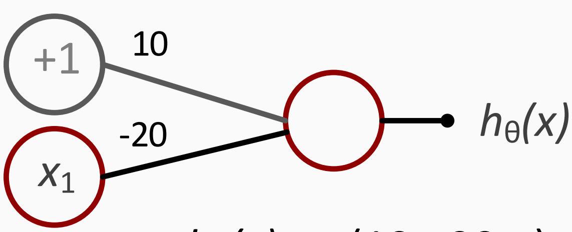
The example neural network computation with the **OR** logical above and the corresponding calculations in the table prove the logical **OR** function. The function is true when $h_\theta(x) \approx x_1 \text{ OR } x_2$. This can be seen in the results above when the computations of $h_\theta(x) = g(z)$ output a true value when either x_1 or x_2 return the values of true.

examples and intuitions ii

Previous examples explored the ways to compute nonlinear hypothesis with neural nets works to compute the **AND** and **OR** function. Additionally, **NOT** can be computed equally:

x_1 AND x_2 $\qquad\qquad$ x_1 OR x_2

negation: NOT x_1



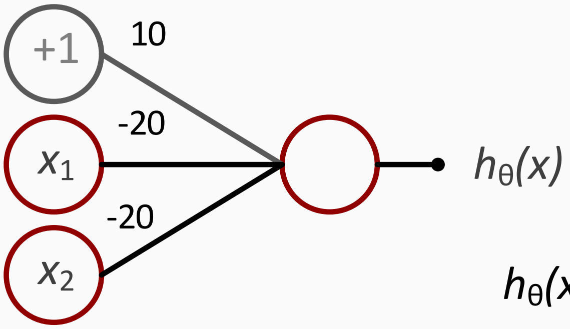
x_1	$h_\theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

$$h_\theta(x) = g(10 - 20x_1)$$

The general idea of negation (NOT) is when a large negative weights (-20 in this example) are attached to the variables (x_1 in this example); the result is negation as illustrated. Additionally, in computing the function of (NOT x_1) AND (NOT x_2), negative weights will be attached to both x_1 *and* x_2.

(NOT x_1) AND (NOT x_2) = 1 ; if and only if $x_1 = x_2 = 0$

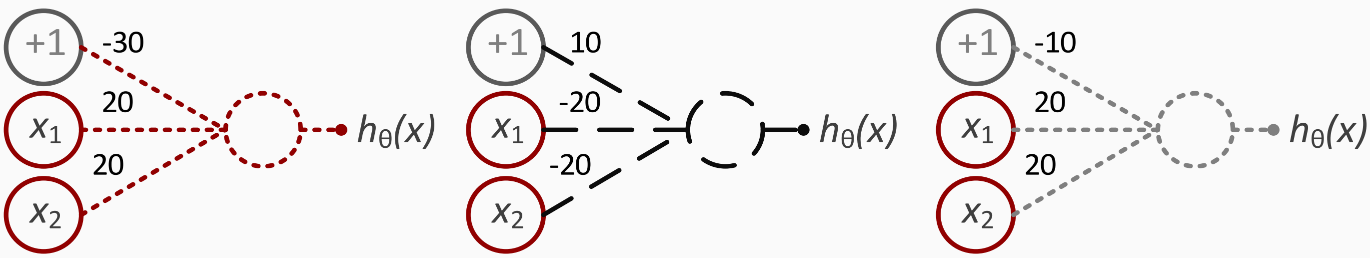
example: (NOT x_1) AND (NOT x_2)



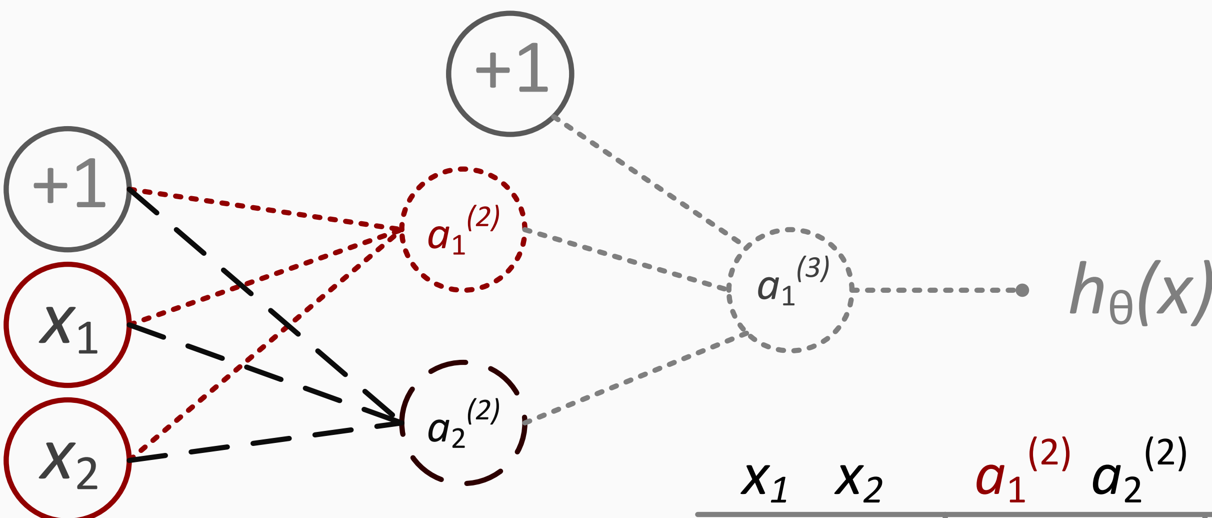
x_1	x_2	$h_\theta(x)$
0	0	$g(10) \approx 1$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(-30) \approx 0$

$$h_\theta(x) = g(10 - 20x_1 - 20x_2)$$

combining the networks: x_1 XNOR x_2



x_1 AND x_2 (NOT x_1) AND (NOT x_2) x_1 OR x_2



$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$

$$h_\theta(x) = g(10 - 20x_1 - 20x_2)$$

$$h_\theta(x) = g(-10 + 20x_1 + 20x_2)$$

x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

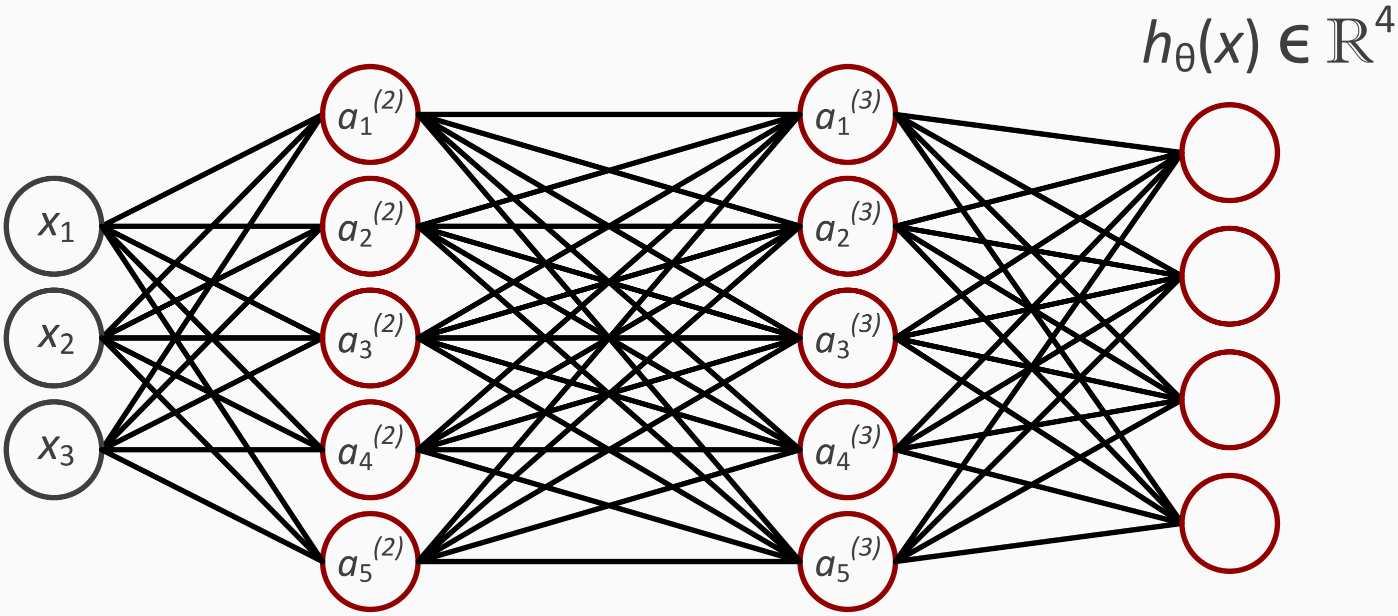
Thus, $h_\theta x = 1$ when either both x_1 and x_2 are 0; or when x_1 and x_2 are both 1.

In other words, $h_\theta x$ outputs 1 at the latter two locations and 0 otherwise.

multiclass classification

Multiclass classification is essentially an extension of the one-vs-all method as illustrated. The output is now a vector of 4 numbers.

multiple output units: one-vs-all



Want $h_\theta x \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad h_\theta x \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad h_\theta x \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad h_\theta x \approx \begin{bmatrix}0\\0\\0\\1\end{bmatrix},$ etc.

In the training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}),\ldots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ will be represented as one of the following:

$Y^{(i)}$ = one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$