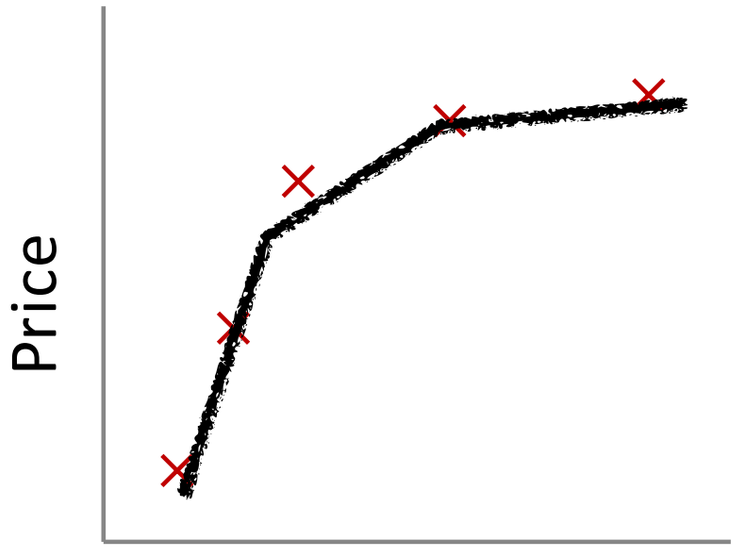
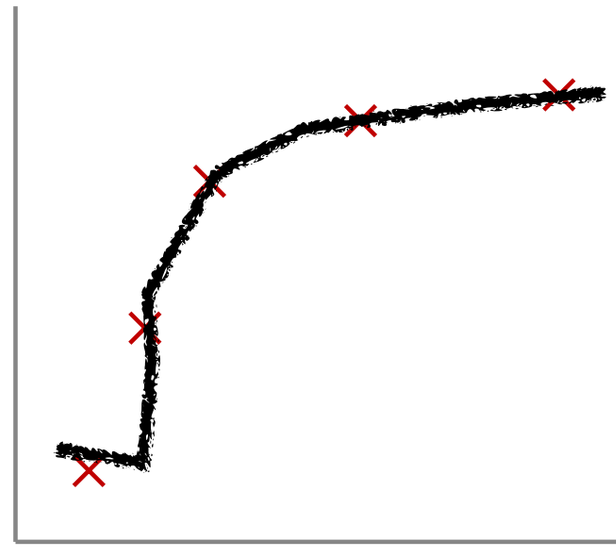
Review

Regularization



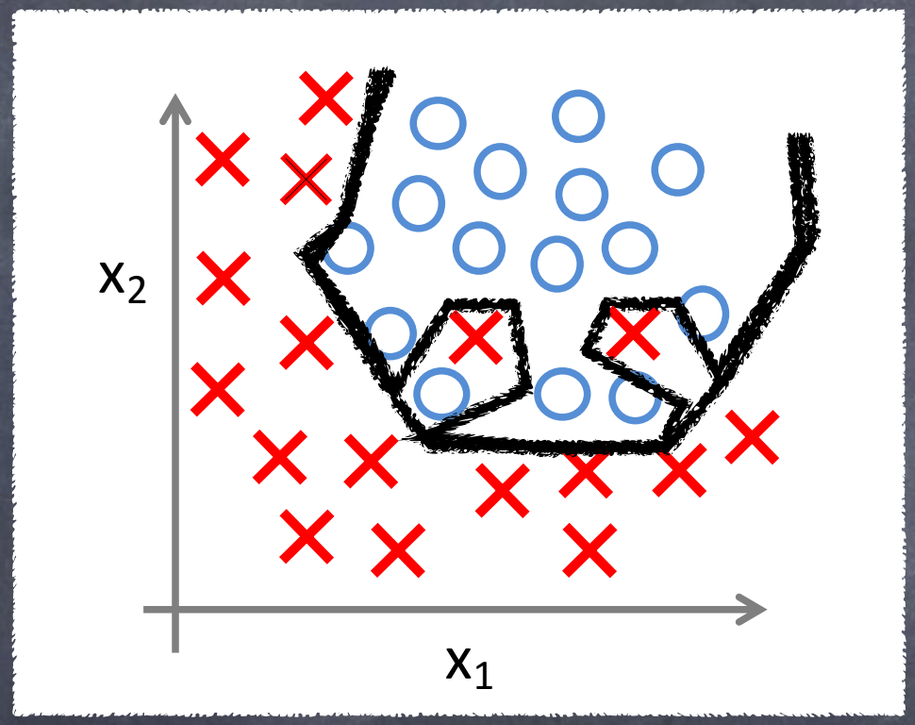


$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Suppose we penalize and make these two small

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000 * \theta_3^2 + 1000 * \theta_4^2$$


x_2
x_1

Regularization

Small values for parameters

- "Simpler" hypothesis
- Less prone to overfitting

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

$$\min_\theta J(\theta)$$

Regularization parameter

Regularized Gradient descent

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \quad \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

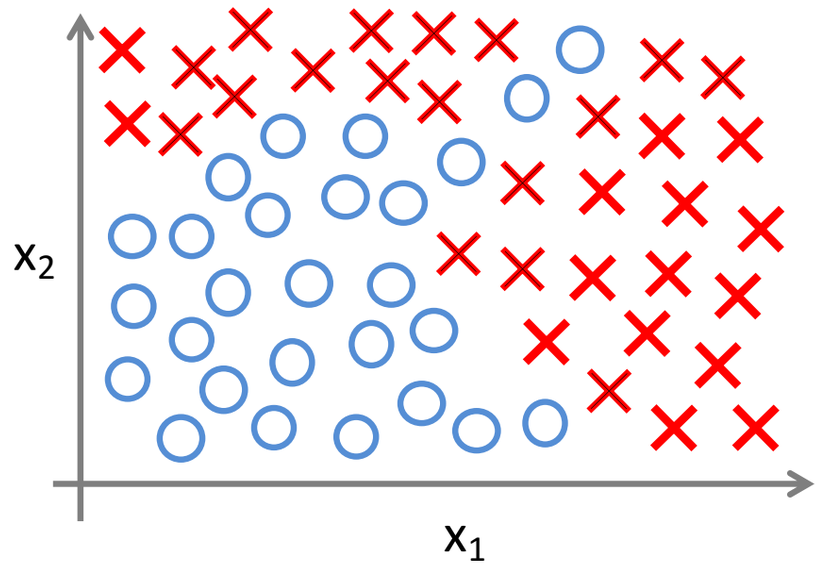
$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$\}$

Regularized Normal Equation

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

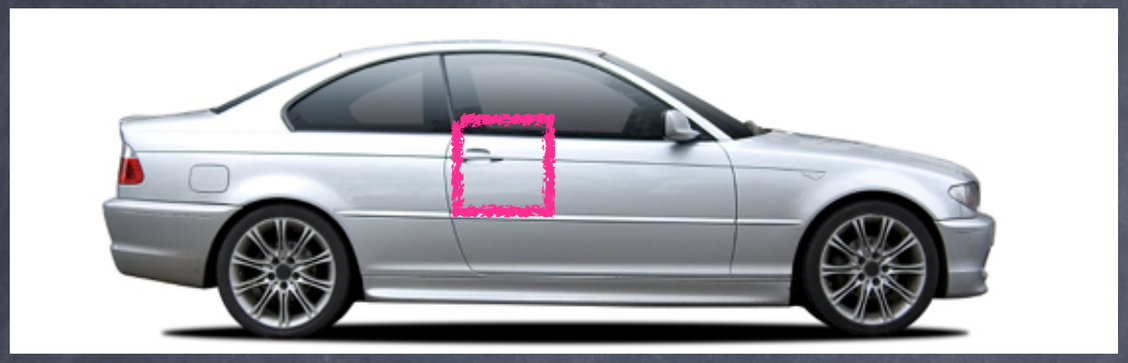


Logistic Regression

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

$x_1 =$ size
$x_2 =$ # bedrooms
$x_3 =$ # floors
$x_4 =$ age
$\dots$
x_{100}

A lot of features!



But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Cars





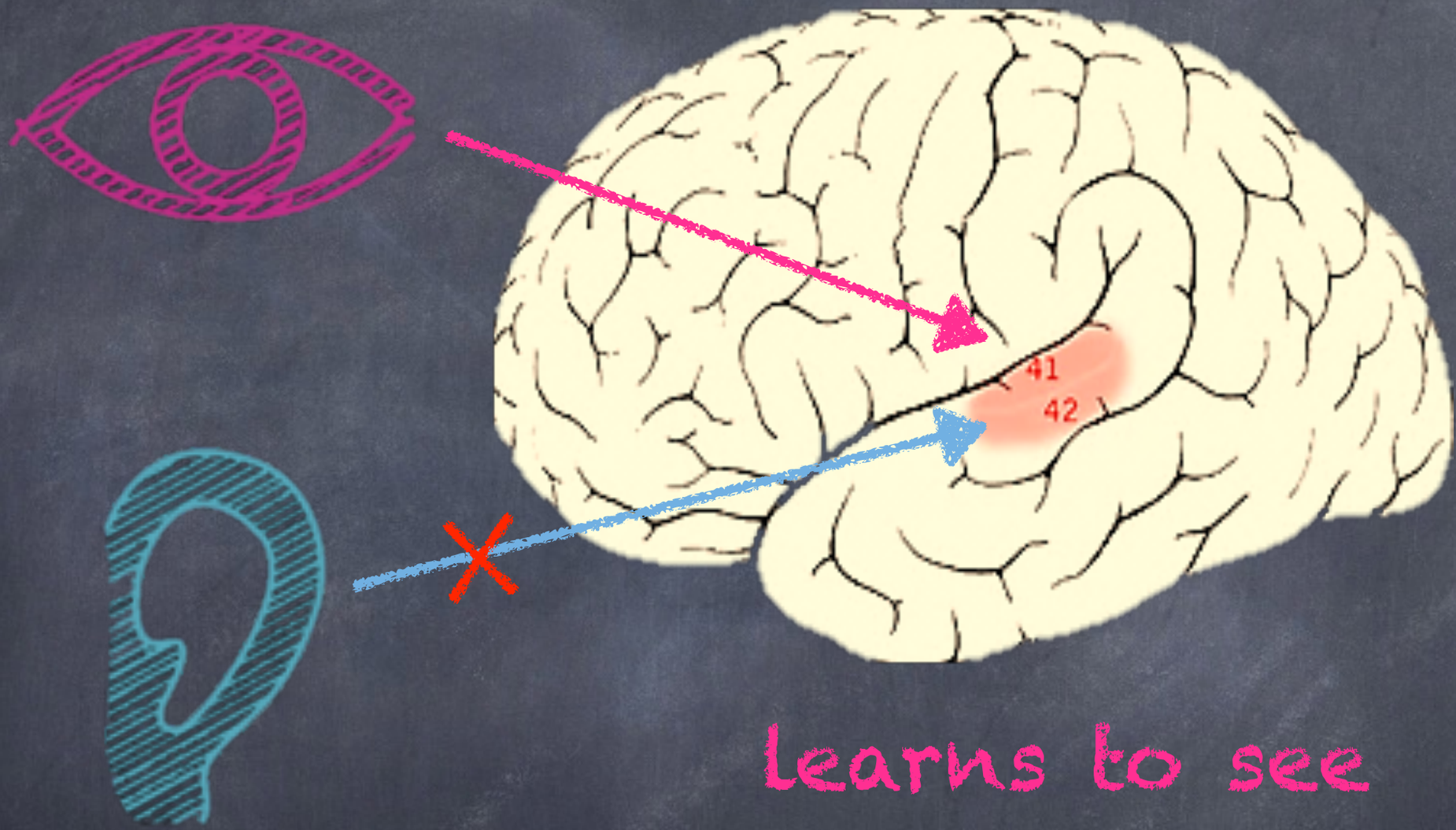

Not a car






Is it a car?

Neural Network



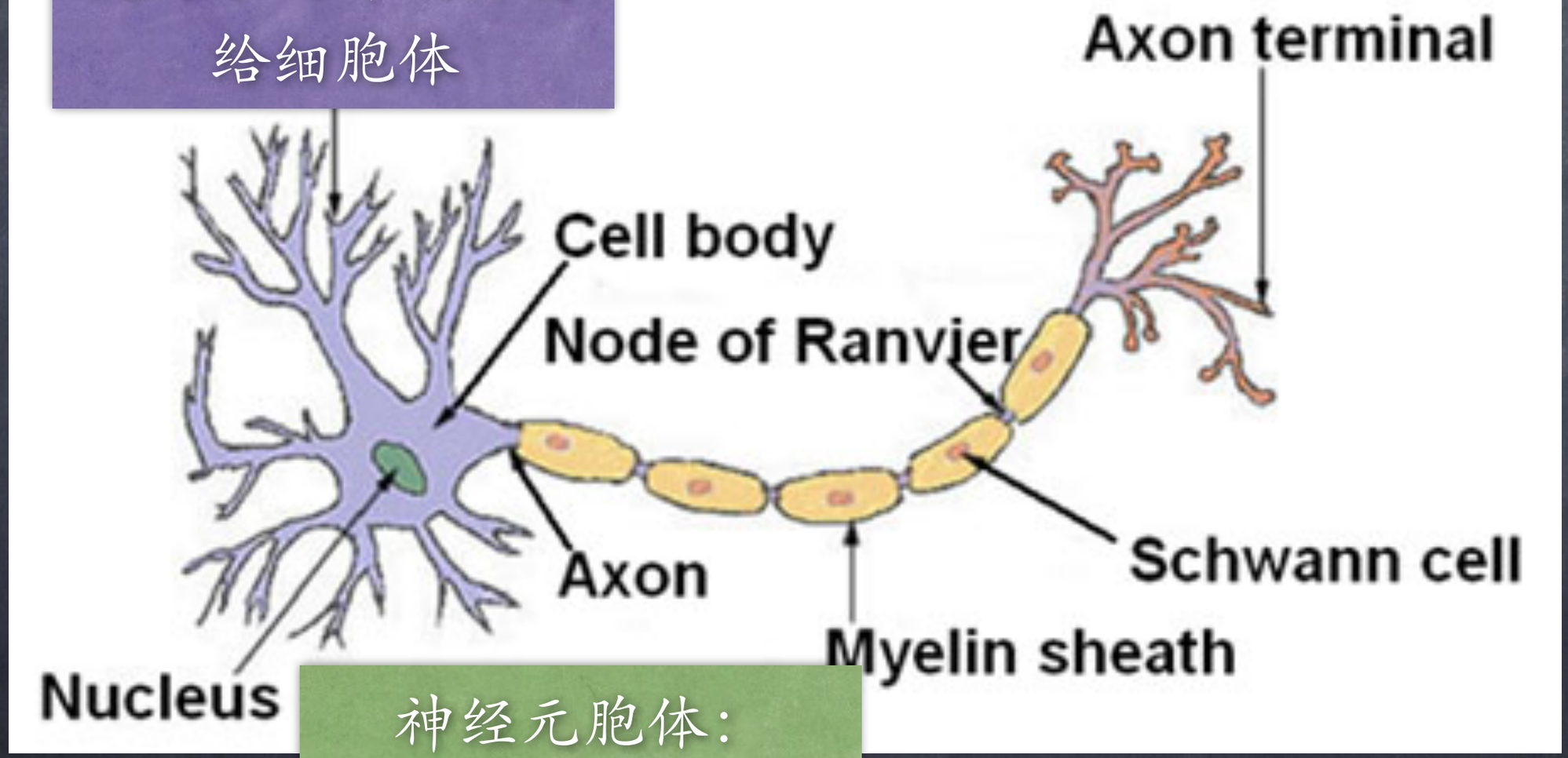
learns to see

One neural architecture learns to solve different problems!

Neuron in the brain

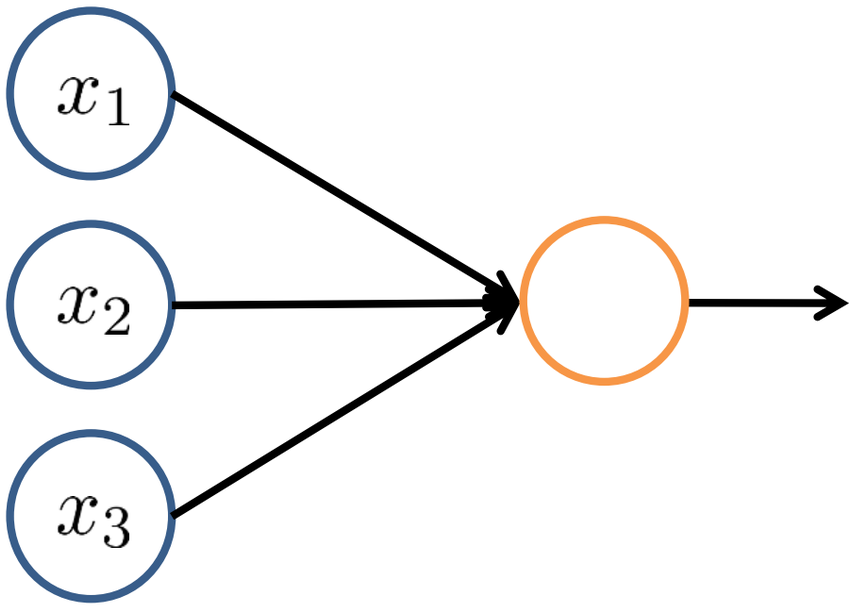
树突：
接受其他神经元轴突传来的冲动并传给细胞体

轴突末梢：
接受外来刺激，再由细胞体传出



神经元胞体：
联络和整合输入信息并传出信息

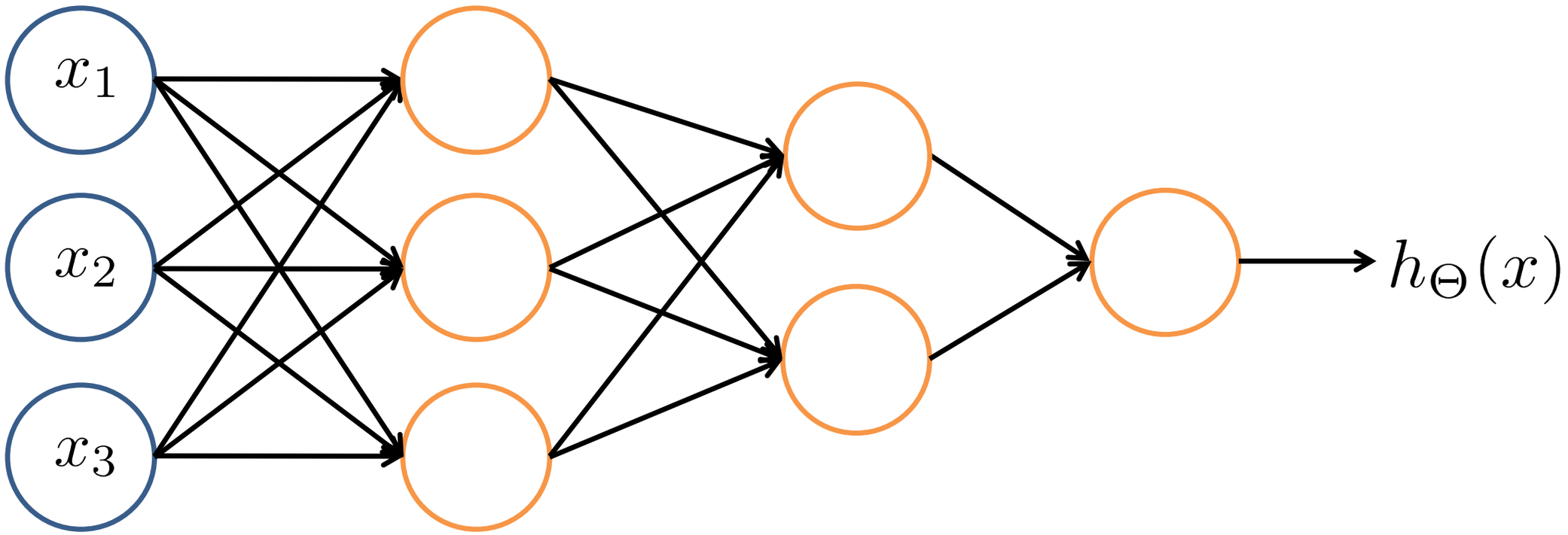
Neuron model: Logistic unit



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

Sigmoid (logistic) activation function.

Neural network architectures



Layer 1

input layer

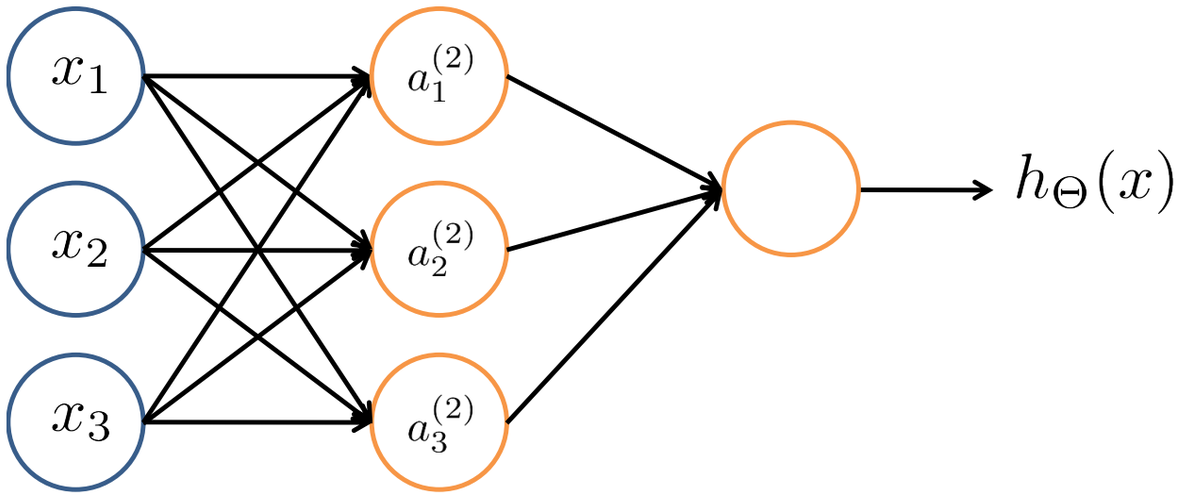
Layer 2, 3

hidden layer

Layer 4

output layer

Neural Network



$\Theta^{(j)} =$

matrix of weights controlling function mapping from layer j to layer j + 1

$a_i^{(j)}$ = "activation" of unit i in layer j

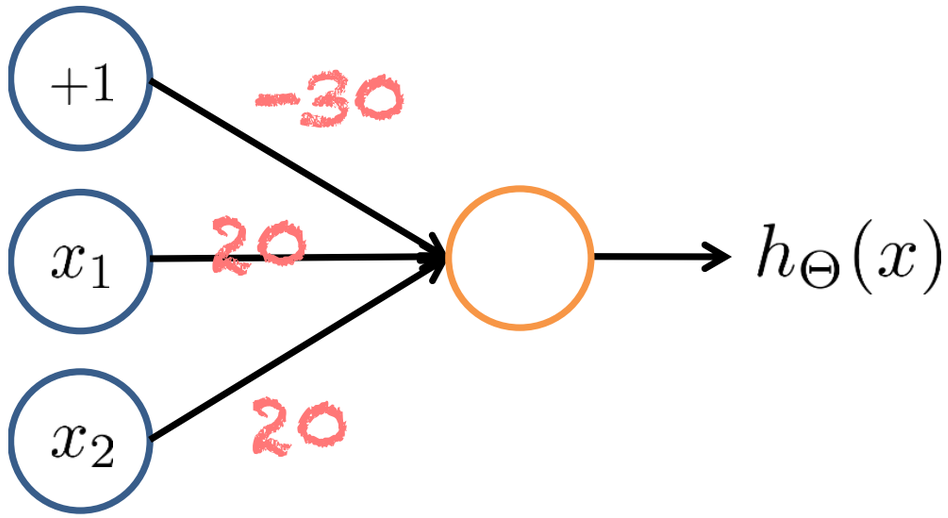
激活/激励函数

$s_{j+1} \times (s_j + 1)$

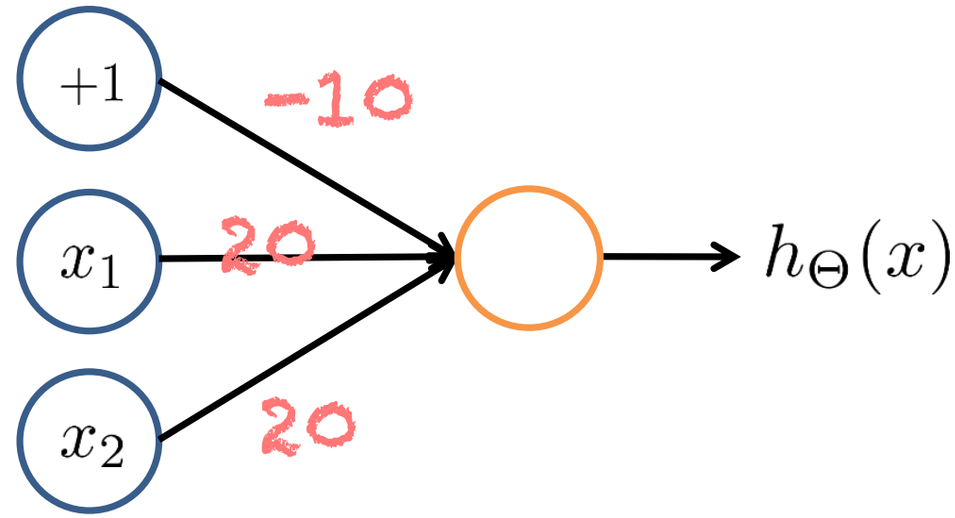
$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

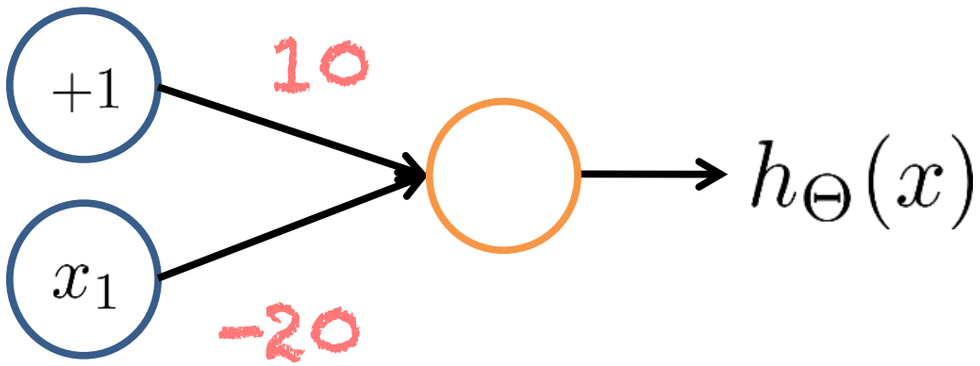
Simple Example: AND



Simple Example: OR

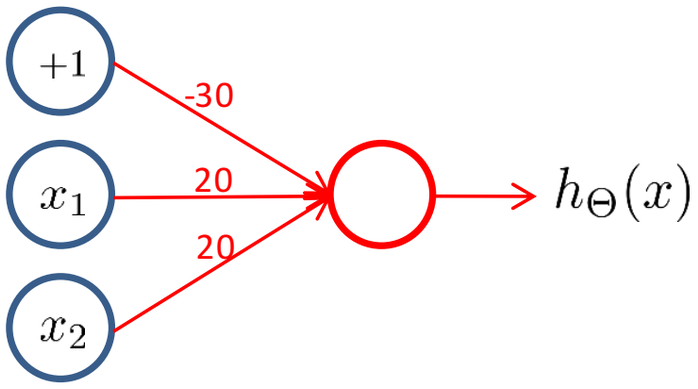


Simple Example: NOT

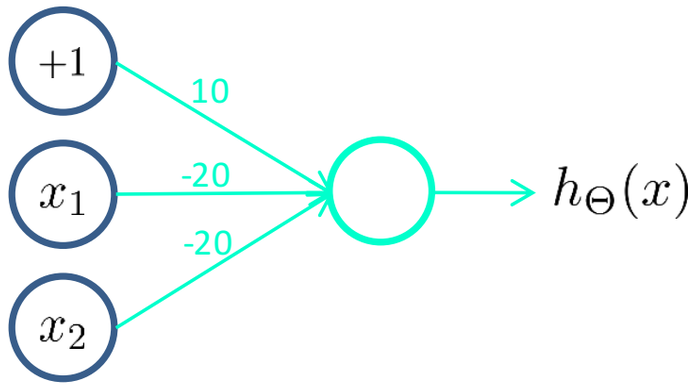


输入A	输入B	输出F
0	0	1
0	1	0
1	0	0
1	1	1

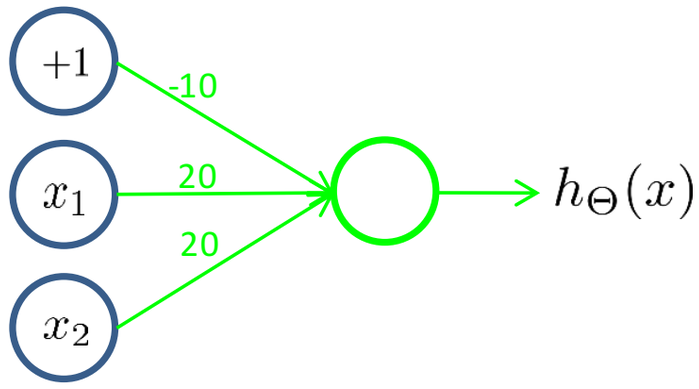
Putting it together: x_1 XNOR x_2



x_1 AND x_2

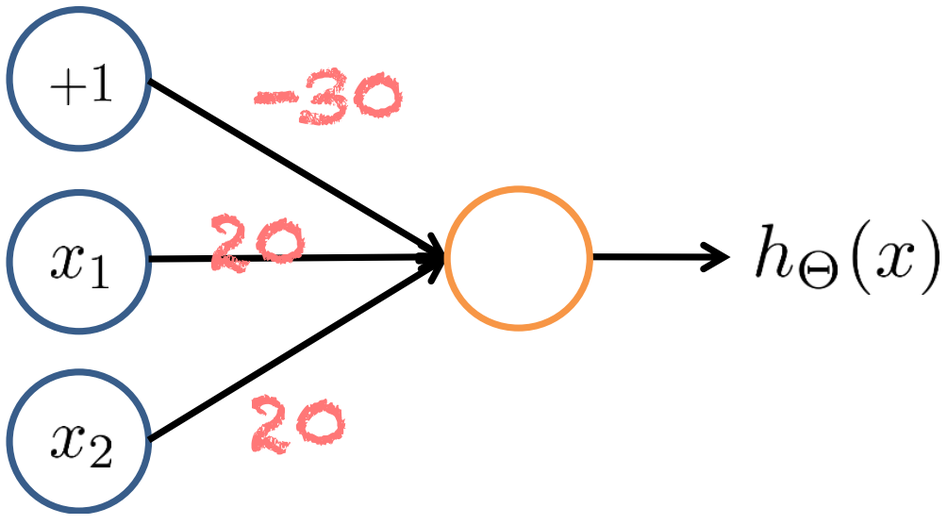


(NOT x_1) AND (NOT x_2)



x_1 OR x_2

Simple Example: AND



How to fit the parameters?

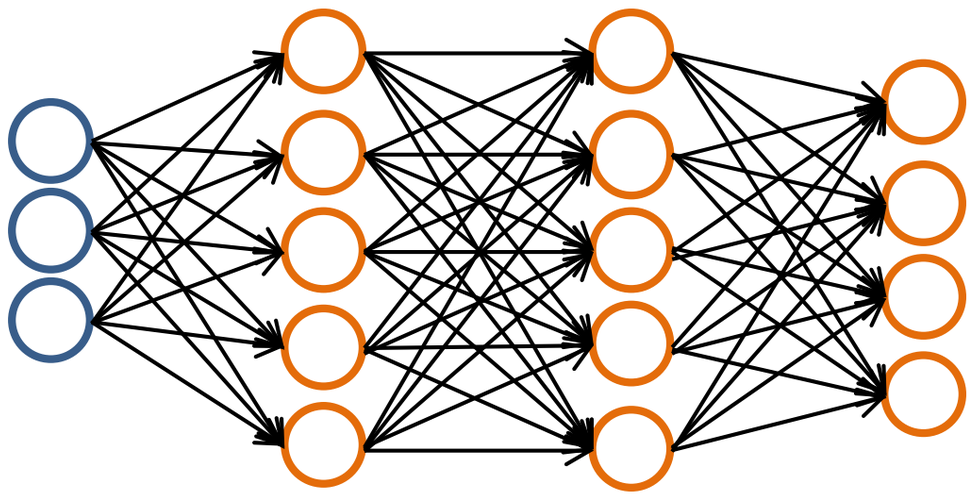
Multiple output units: one-vs-all

Pedestrian

Car

Motorcycle

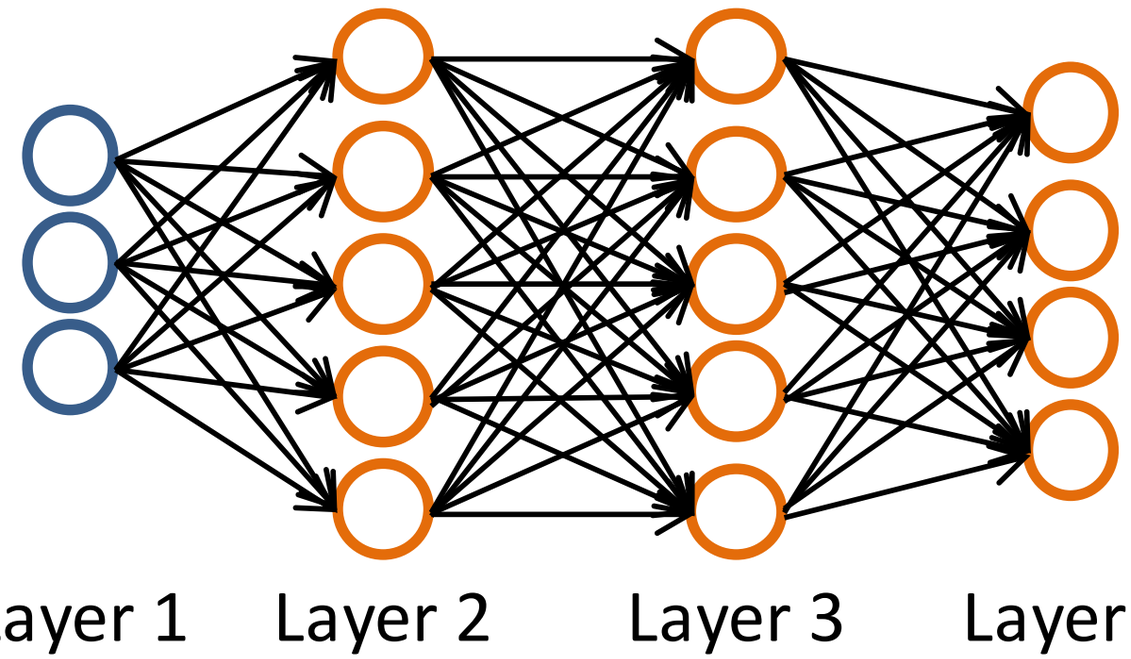
Truck



$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle



$L =$ number of layers

$s_l =$ number of units in layer L, excluding bias unit

Training dataset

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

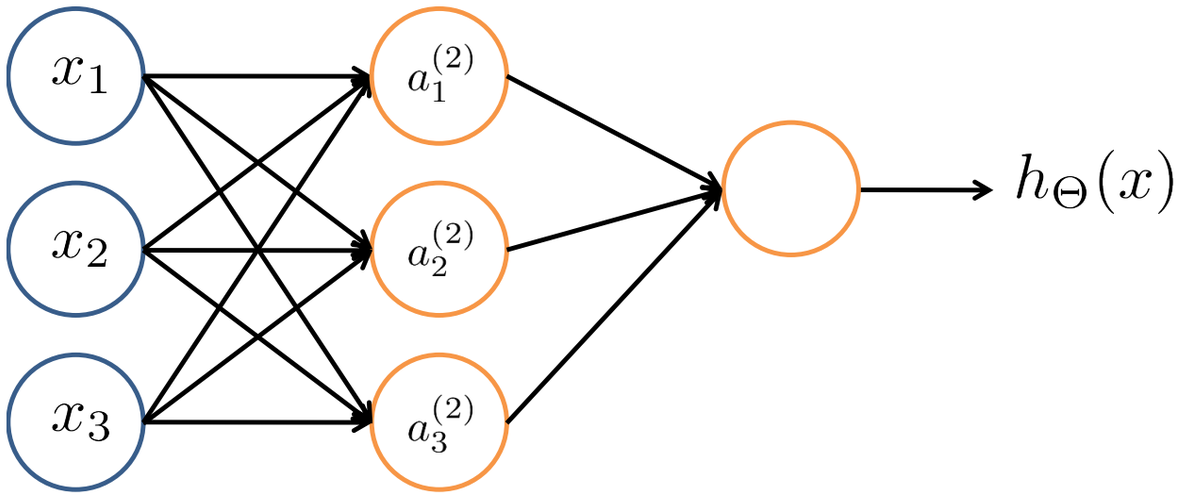
Binary Classification

$y = 0$ or 1

Multi-class Classification

$y \in \mathbb{R}^K$

Neural Network



$\Theta^{(j)} =$

matrix of weights controlling function mapping from layer j to layer j + 1

$a_i^{(j)}$ = "activation" of unit i in layer j

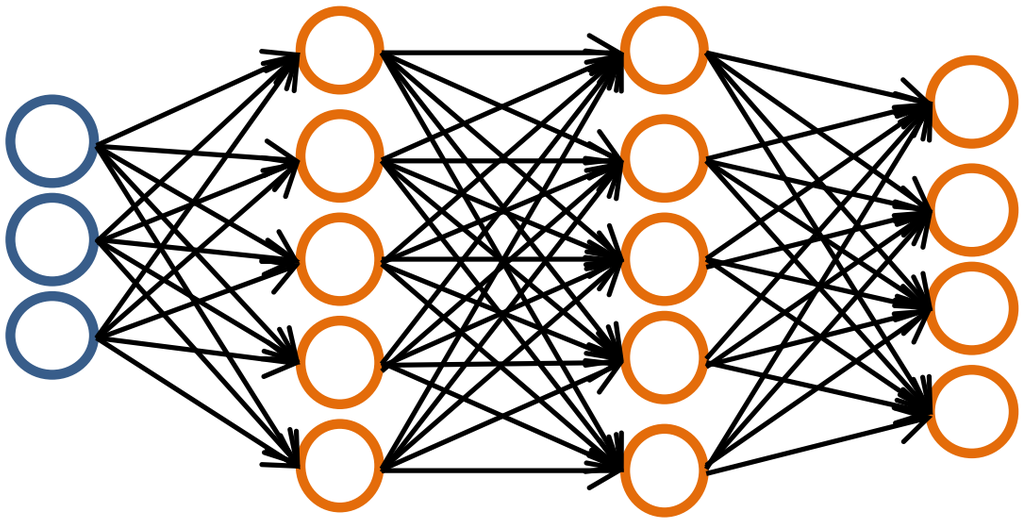
激活/激励函数

$s_{j+1} \times (s_j + 1)$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

Forward Propagation



Given one training example (x, y)

Forward Propagation:

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)}) \ (\text{add } a_0^{(2)})$$

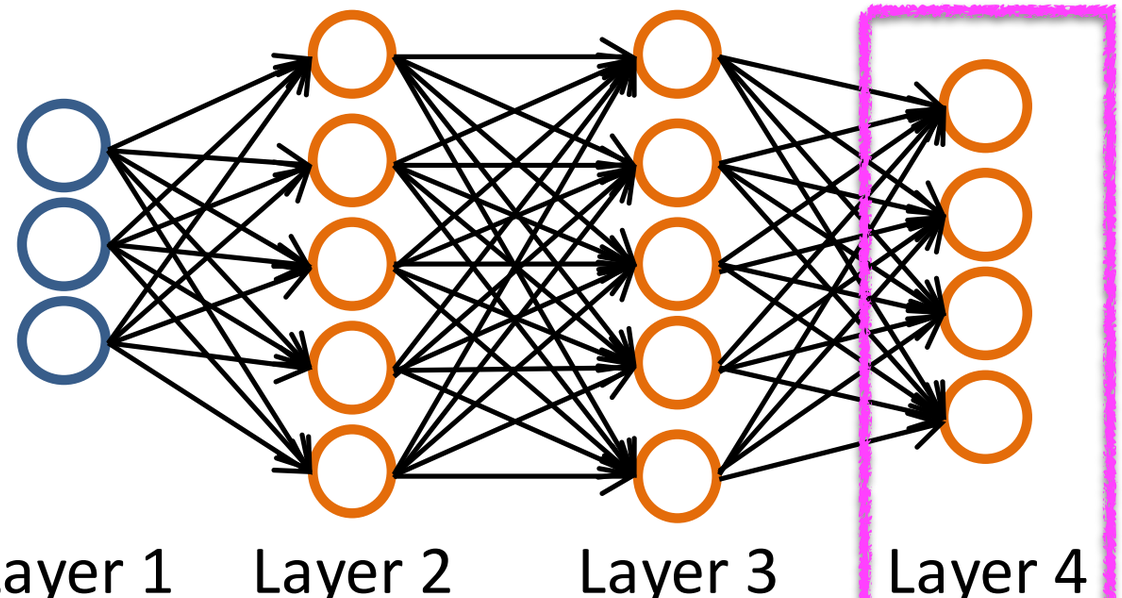
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$a^{(3)} = g(z^{(3)}) \ (\text{add } a_0^{(3)})$$

$$z^{(4)} = \Theta^{(3)} a^{(3)}$$

$$a^{(4)} = h_\Theta(x) = g(z^{(4)})$$

Backpropagation



Forward Propagation:

$a^{(1)} = x$

$z^{(2)} = \Theta^{(1)} a^{(1)}$

$a^{(2)} = g(z^{(2)})$ (add $a_0^{(2)}$)

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

$a^{(3)} = g(z^{(3)})$ (add $a_0^{(3)}$)

$z^{(4)} = \Theta^{(3)} a^{(3)}$

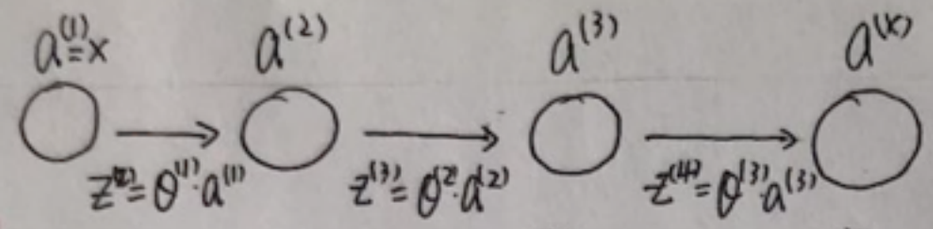
$a^{(4)} = h_\Theta(x) = g(z^{(4)})$

$\delta_j^{(l)}$ = "error" of unit *j* in layer *l*

$\delta_j^{(4)} = a_j^{(4)} - y_j$ for each output unit (L=4)

$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(4)})$

$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(3)})$



y 真实值

$a^{(2)}=g(z^{(2)})$ $a^{(3)}=g(z^{(3)})$ $a^{(k)}=g(z^{(k)})$

损失函数 $J(\hat{y},y)=\frac{1}{2}(\hat{y}-y)^2=\frac{1}{2}(a^{(k)}-y)^2$

(1)

$$\frac{\partial J}{\partial a^{(k)}}=a^{(k)}-y \rightarrow \delta^{(4)}$$

$$\frac{\partial J}{\partial \theta^{(3)}}=\frac{\partial J}{\partial a^{(k)}}\cdot\frac{\partial a^{(k)}}{\partial z^{(k)}}\cdot\frac{\partial z^{(k)}}{\partial \theta^{(3)}}=\delta^{(k)}\cdot g'(z^{(k)})\cdot a^{(3)}$$

(2)

$$\frac{\partial J}{\partial a^{(3)}}=\frac{\partial J}{\partial a^{(k)}}\cdot\frac{\partial a^{(k)}}{\partial a^{(3)}}$$

$$=\frac{\partial J}{\partial a^{(k)}}\cdot\frac{\partial a^{(k)}}{\partial z^{(k)}}\frac{\partial z^{(k)}}{\partial a^{(3)}}$$

$$=\delta^{(k)}\cdot g'(z^{(k)})\cdot\theta^{(3)}$$

$$=(\theta^{(3)})^T\cdot\delta^{(k)}\cdot g'(z^{(k)})\rightarrow\delta^{(3)}$$

$$\frac{\partial J}{\partial \theta^{(2)}}=\frac{\partial J}{\partial a^{(3)}}\cdot\frac{\partial a^{(3)}}{\partial z^{(3)}}\cdot\frac{\partial z^{(3)}}{\partial \theta^{(2)}}=\delta^{(3)}\cdot g'(z^{(3)})\cdot a$$

(3):

$$\delta^{(2)}=(\theta^{(2)})^T\cdot\delta^{(3)}\cdot g'(z^{(3)}) \qquad \frac{\partial J}{\partial \theta^{(1)}}=\delta^{(2)}\cdot g'(z^{(2)})\cdot a^{(1)}$$

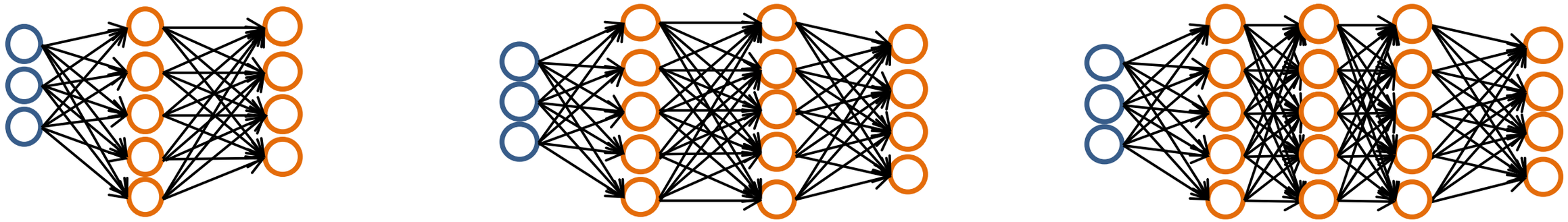
$$\theta^{(3)}:=\theta^{(3)}-\eta\cdot\frac{\partial J}{\partial \theta^{(3)}}$$

$$\theta^{(2)}:=\theta^{(2)}-\eta\frac{\partial J}{\partial \theta^{(2)}}$$

$$\theta^{(1)}:=\theta^{(1)}-\eta\frac{\partial J}{\partial \theta^{(1)}}$$

Training a Neural Network

1. pick a network architecture



Reasonable defaults:

- 1 hidden layer;
- or if >1 hidden layer, have same number of hidden units in each layer (usually the more the better)