

深度学习讨论班

第二节

Multi-layer perceptron (多层感知机)

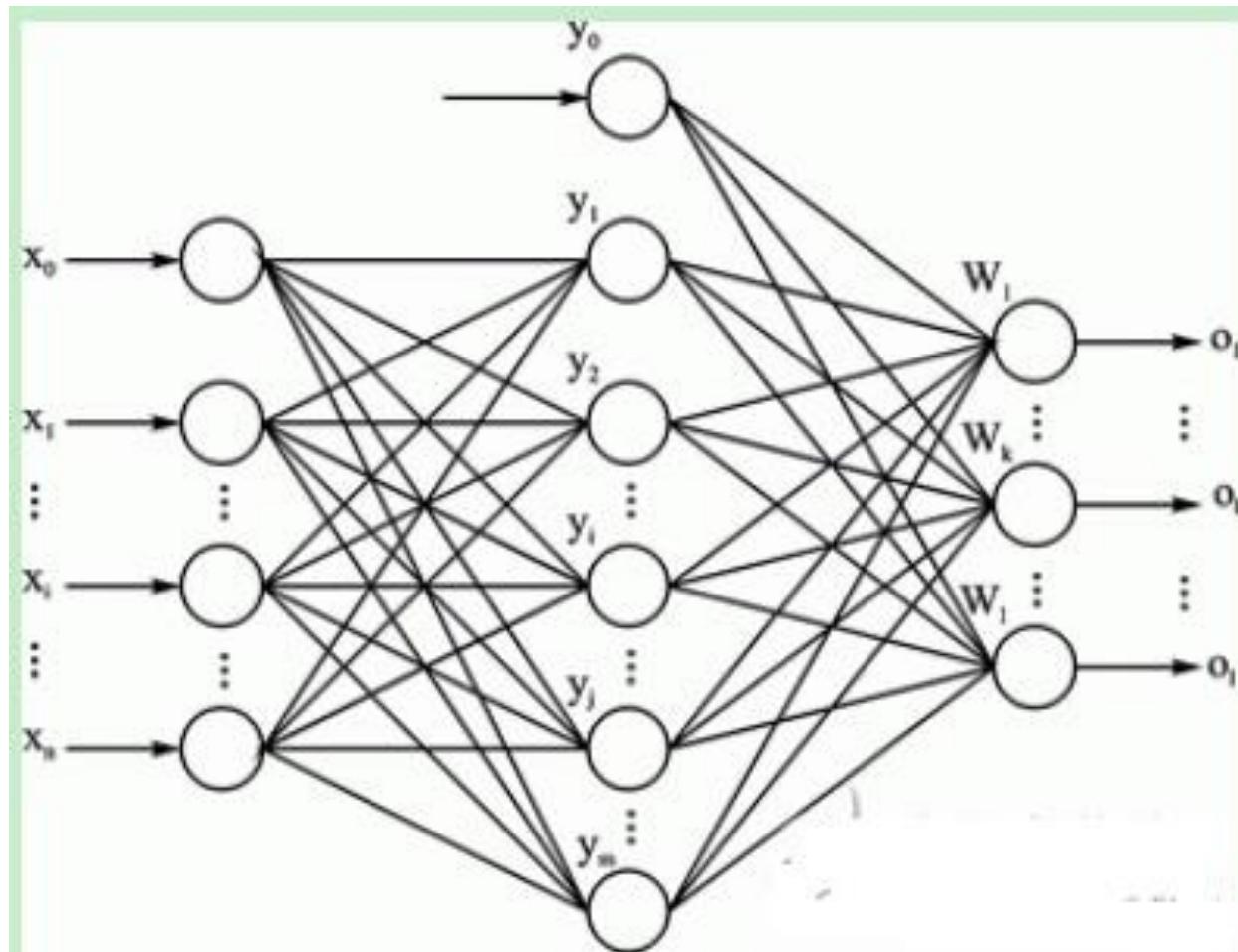
黄雷

2016-12-6

上一讲主要内容

- Basic Concept
 - Machine learning
 - Neural network
 - Deep network
- History of neural network
 - Perceptron
 - BackPropagation
 - Deep learning
- Application

多层感知机（前向神经网络）



outline

- Linear classifier (简单线性分类器)
 - One neuron (一个神经元)
 - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
 - Model representation (模型表示)
 - Loss function: the goal for learning
 - Training
 - Gradient based optimization
 - backpropagation

One example(一个贯穿全文的例子)

- Classification tasks
 - Binary classification(二分类): is cat?
 - Multiple classification (多分类) : is cat, dog, others?
- Assumption
 - The feature vectors of the images are provided, 3-dimensinal vectors



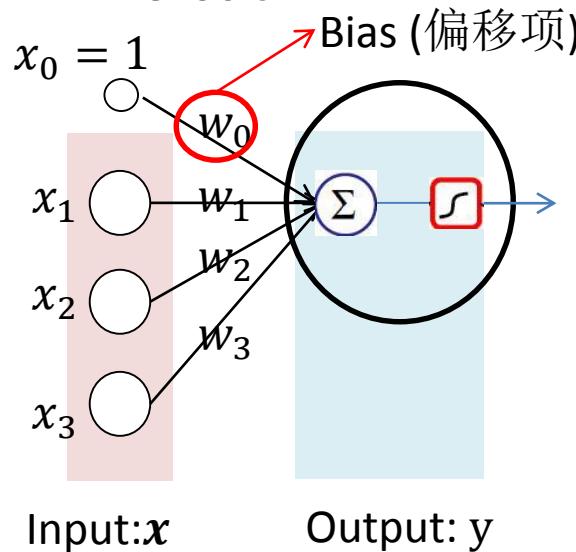
$$(x_1, x_2, x_3)^T$$

outline

- Linear classifier (简单线性分类器)
 - One neuron (一个神经元)
 - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
 - Model representation (模型表示)
 - Loss function: the goal for learning
 - Training
 - Gradient based optimization
 - backpropagation

Linear Classifier (线性分类器)

- One neuron
 - Binary classification (二分类问题)
is cat?

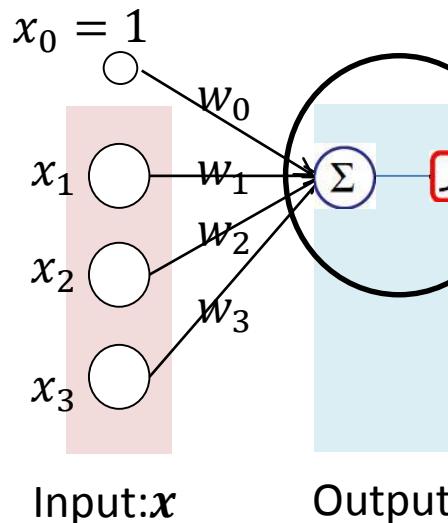


$$a = \sum_{i=0}^3 w_i x_i = \mathbf{w} \cdot \mathbf{x}$$
$$y = \sigma(a) = \frac{1}{1 + e^{-a}}$$

$$(x_1, x_2, x_3)^T$$

Linear Classifier

- One example



$$\text{Model: } \mathbf{w} = (w_0, w_1, w_2, w_3)^T \\ = (2, 0, 0, 4)^T$$



$$(x_1, x_2, x_3)^T = (2, 2, 3)^T$$

$$a = 2 * 1 + 0 * 2 + 0 * 2 + 4 * 3 = 14$$

$$y = \varphi(a) = \frac{1}{1 + e^{-14}} > 0.5$$



$$(x_1, x_2, x_3)^T = (1, 2, -3)^T$$

$$a = 2 * 1 + 0 * 1 + 0 * 2 + 4 * (-3) = -10$$

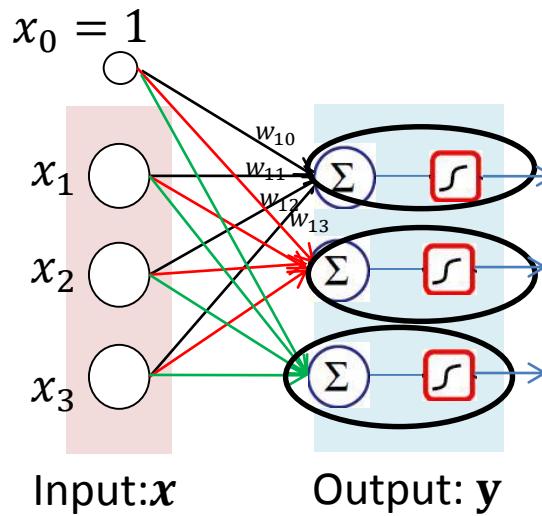
$$y = \varphi(a) = \frac{1}{1 + e^{10}} < 0.5$$

outline

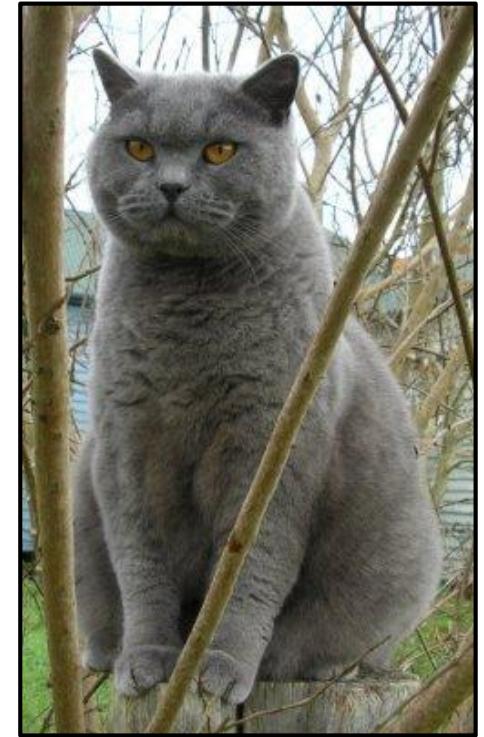
- Linear classifier (简单线性分类器)
 - One neuron (一个神经元)
 - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
 - Model representation (模型表示)
 - Loss function: the goal for learning
 - Training
 - Gradient based optimization
 - backpropagation

Linear Classifier (线性分类器)

- Multiple neurons
 - Multiple classification: is cat? dog? others?



w_{ij} : 第 i 个输出神经元, 连接第 j 个输入单元



(x_1, x_2, x_3)

$$a_i = \sum_{j=0}^3 w_{ij} x_i = \mathbf{w}_i \cdot \mathbf{x}, \quad i = (1, 2, 3)$$

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

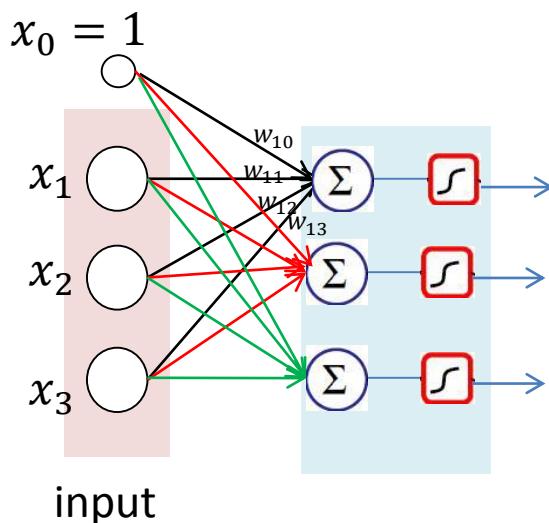


$$\mathbf{a} = \mathbf{W} \cdot \mathbf{x}$$

$$\mathbf{y} = \sigma(\mathbf{a})$$

Linear Classifier (线性分类器)

- One example



Model: $W =$

2	0	0	4
0	0	-2	-3
1	-1	0	0



$$(x_1, x_2, x_3)^T = (2, 2, 3)^T$$

$$\begin{aligned} \mathbf{a} &= \begin{array}{|c|c|c|c|} \hline 2 & 0 & 0 & 4 \\ \hline 0 & 0 & -2 & -3 \\ \hline 1 & -1 & 0 & 0 \\ \hline \end{array} * (1, 2, 2, 3)^T \\ &= (14, -13, -1) \end{aligned}$$

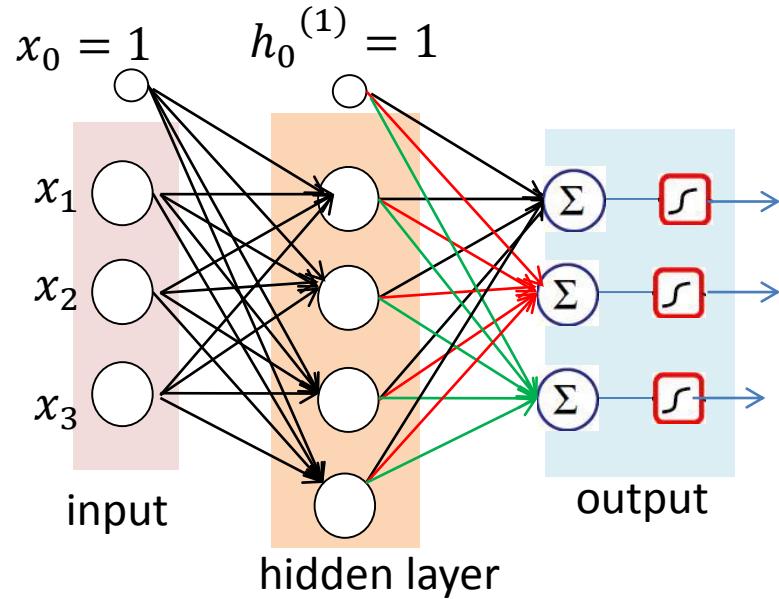
$$\mathbf{y} = \left(\frac{1}{1+e^{-14}}, \frac{1}{1+e^{13}}, \frac{1}{1+e^1} \right)^T$$

outline

- Linear classifier (简单线性分类器)
 - One neuron (一个神经元)
 - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
 - Model representation (模型表示)
 - Loss function: the goal for learning
 - Training
 - Gradient based optimization
 - backpropagation

Multi-layer perceptron (多层感知机)

- Multi-layer perceptron or feed-forward neural network



x_i : 第*i*个输入节点

$h_i^{(k)}$: 第*k*层隐藏层的第*i*个节点

$w_{ij}^{(k)}$: 第*k*层隐藏层, 第*i*个输出神经元,
连接第*j*个输入神经元

y_i : 第*i*个输出节点

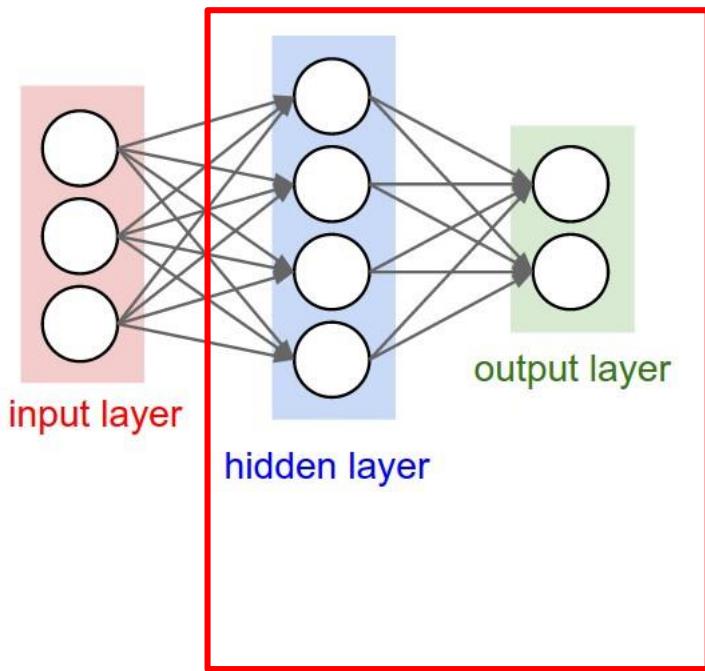
Pre-activation $\rightarrow \mathbf{a}^{(1)} = \mathbf{W}^{(1)} \cdot \mathbf{x}$
activation $\rightarrow \mathbf{h}^{(1)} = \sigma(\mathbf{a}^{(1)})$

$$\mathbf{a}^{(2)} = \mathbf{W}^{(2)} \cdot \mathbf{h}^{(1)}$$
$$\mathbf{y} = \sigma(\mathbf{a}^{(2)})$$

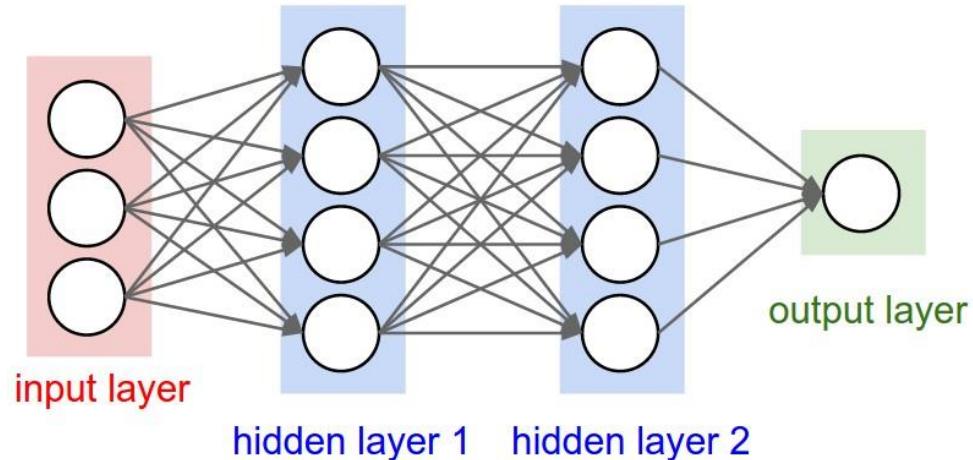


$$\mathbf{a}^{(i)} = \mathbf{W}^{(i)} \cdot \mathbf{h}^{(i-1)}$$
$$\mathbf{h}^{(i)} = \sigma(\mathbf{a})$$
$$(\mathbf{h}^{(0)} = \mathbf{x}, \mathbf{h}^{(L)} = \mathbf{y})$$

Neural Networks: Architectures

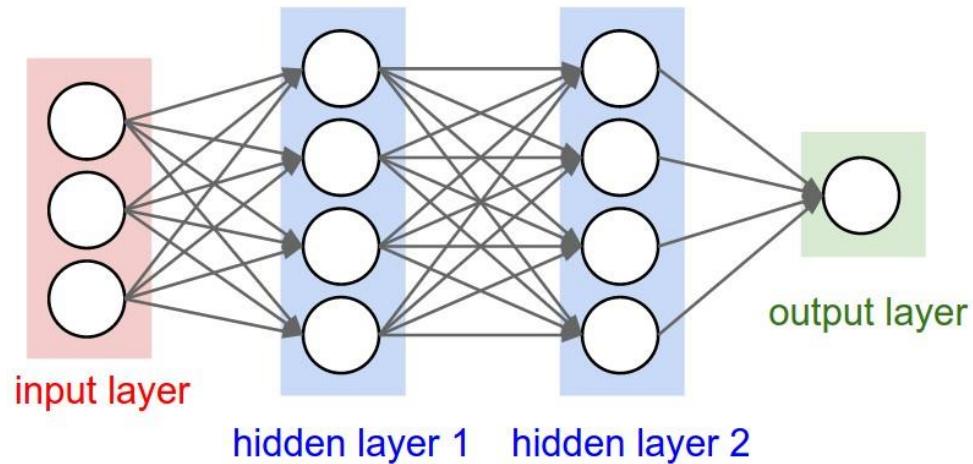
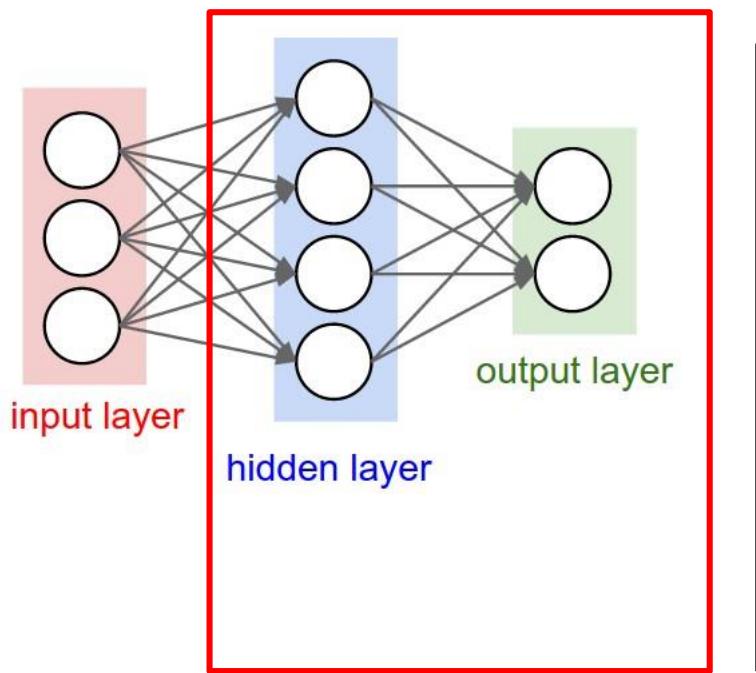


“2-layer Neural Net”, or
“1-hidden-layer Neural Net”



“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

Neural Networks: Architectures



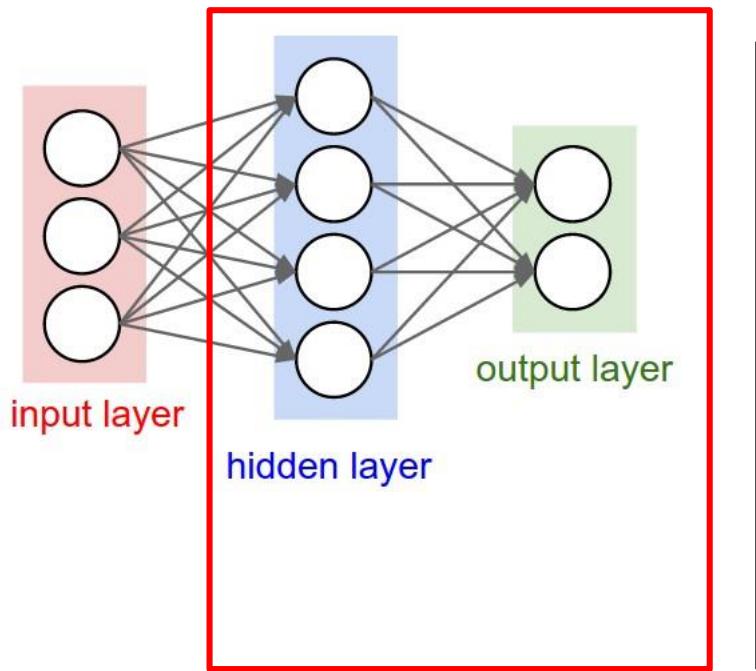
Number of Neurons: ?

Number of Weights: ?

Number of Parameters: ?

Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

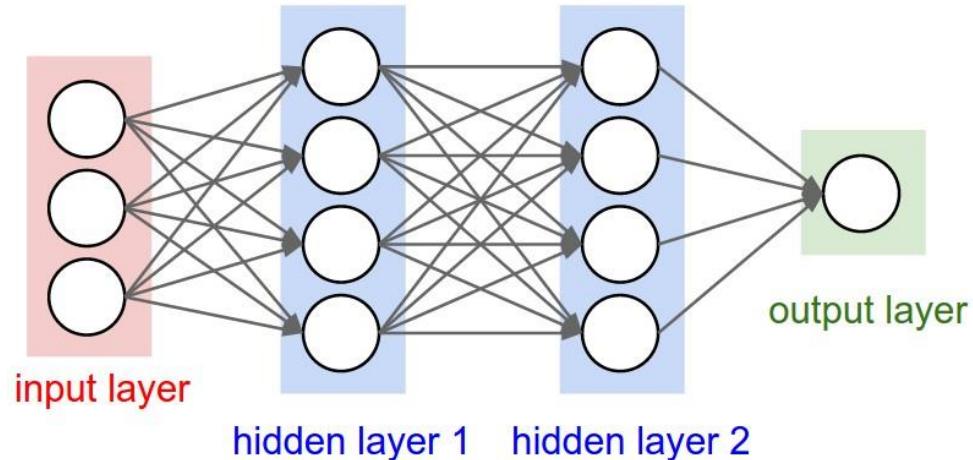
Neural Networks: Architectures



Number of Neurons: $4+2 = 6$

Number of Weights: $[4 \times 3 + 2 \times 4] = 20$

Number of Parameters: $20 + 6 = 26$ (biases!)

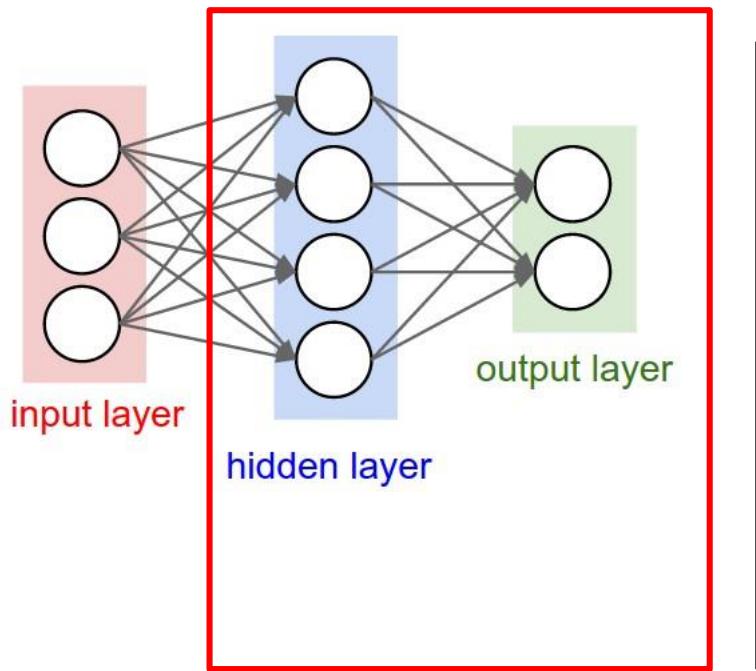


Number of Neurons: ?

Number of Weights: ?

Number of Parameters: ?

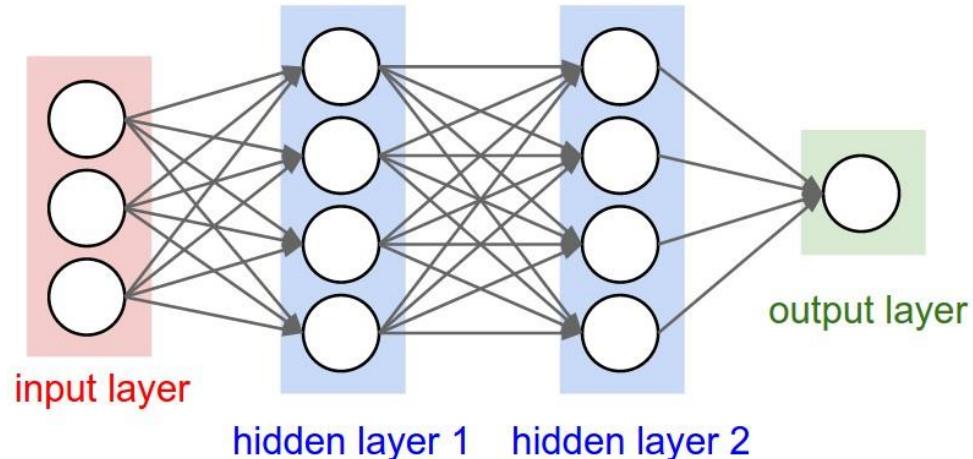
Neural Networks: Architectures



Number of Neurons: $4+2 = 6$

Number of Weights: $[4 \times 3 + 2 \times 4] = 20$

Number of Parameters: $20 + 6 = 26$ (biases!)

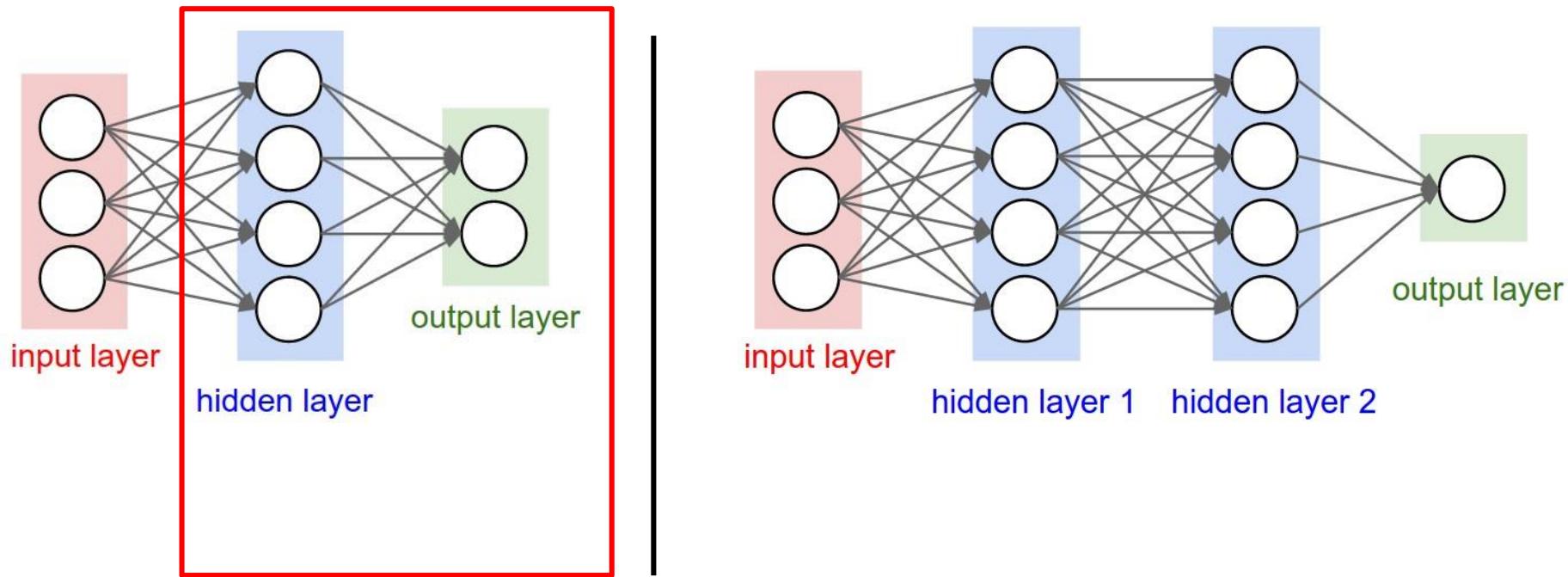


Number of Neurons: $4 + 4 + 1 = 9$

Number of Weights: $[4 \times 3 + 4 \times 4 + 1 \times 4] = 32$

Number of Parameters: $32 + 9 = 41$

Neural Networks: Architectures



Modern CNNs: ~10 million neurons

Human visual cortex: ~5 billion neurons

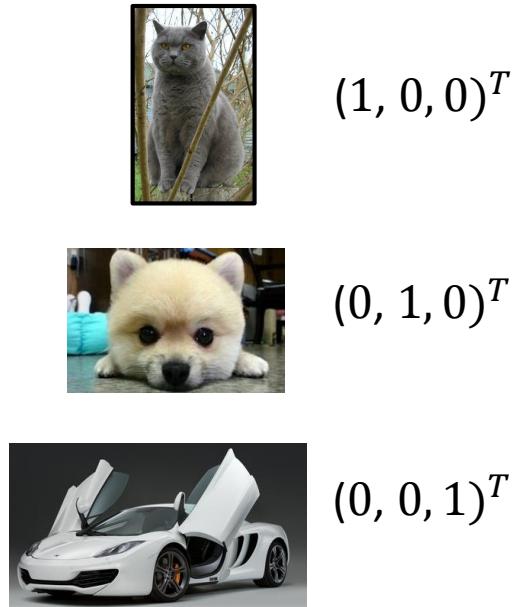
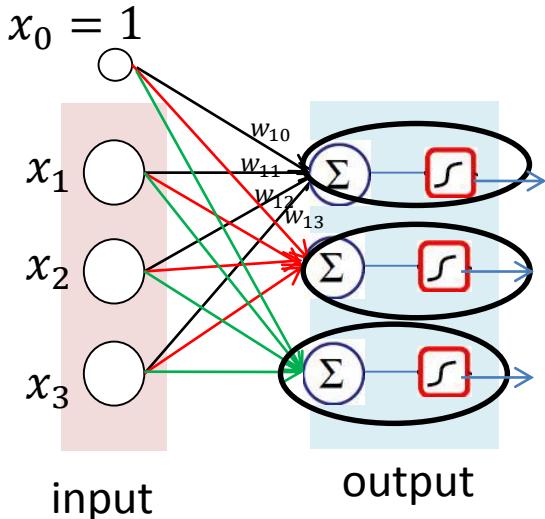
Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

outline

- Linear classifier (简单线性分类器)
 - One neuron (一个神经元)
 - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
 - Model representation (模型表示)
 - Loss function: the goal for learning
 - Training
 - Gradient based optimization
 - backpropagation

Target of learning: Loss function

- Loss function



$$\begin{aligned} L &= (\mathbf{y} - \hat{\mathbf{y}})^2 \\ &= (1 - y_1)^2 + y_2^2 + y_3^2 \end{aligned}$$

$$\begin{aligned} \mathbf{a} &= \mathbf{W} \cdot \mathbf{x} \\ \mathbf{y} &= \sigma(\mathbf{a}) \end{aligned}$$

E.g. Loss: Mean Squared Error (均方误差): $L=(\mathbf{y} - \hat{\mathbf{y}})^2$
objective function: $\min L=(\mathbf{y} - \hat{\mathbf{y}})^2$

outline

- Linear classifier (简单线性分类器)
 - One neuron (一个神经元)
 - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
 - Model representation (模型表示)
 - Loss function: the goal for learning
 - **Training**
 - Gradient based optimization
 - backpropagation

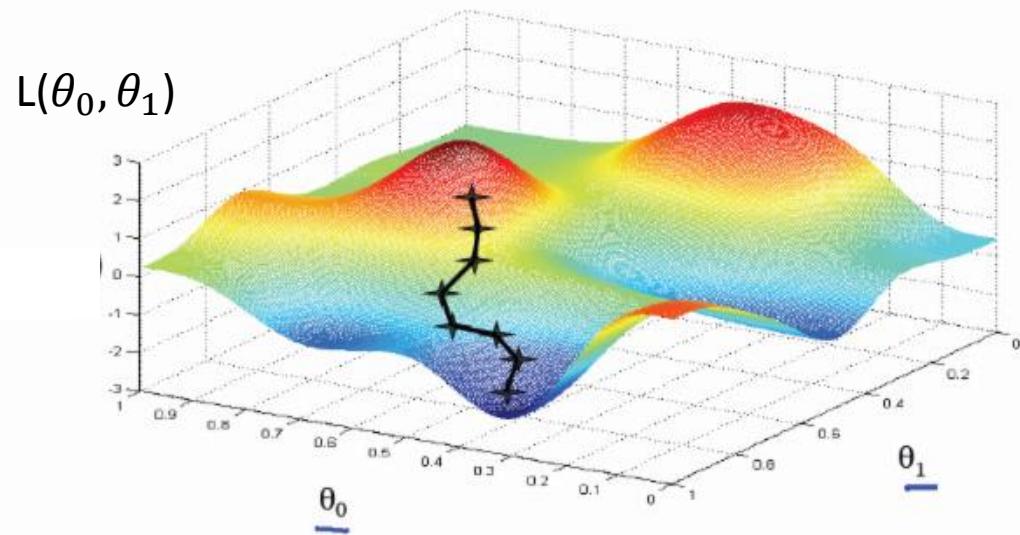
How to learn: adjust parameters

- Gradient descent (梯度下降法)

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \frac{d L}{d \mathbf{W}^{(t)}},$$

学习率

下降方向为负的梯度方向



梯度方向: $\left(\frac{dL(\theta_0, \theta_1)}{\theta_0}, \frac{dL(\theta_0, \theta_1)}{\theta_1}\right)$

Calculate gradient: back-Propagation

- 0.derivative

- In 1-dimension, the derivative of a scalar function :

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

- In multiple dimension, the **gradient** is the vector of (partial derivatives).

$$\frac{\text{df}(\theta)}{\theta} = \left(\frac{df(\theta_0, \theta_1)}{d\theta_0}, \frac{df(\theta_0, \theta_1)}{d\theta_1} \right), \quad \theta = (\theta_0, \theta_1)$$

最小均方误差损失 Loss: $L = (\mathbf{y} - \hat{\mathbf{y}})^2$

backPropagation

- 1. Basic operation in neural network

addition: $f(x,y) = x + y$ $\frac{df}{x} = 1$ $\frac{df}{y} = 1$

multiplication : $f(x,y) = xy$ $\frac{df}{x} = y$ $\frac{df}{y} = x$

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

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}} \right) \left(\frac{1}{1+e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$

backPropagation

- 2. Chain rule

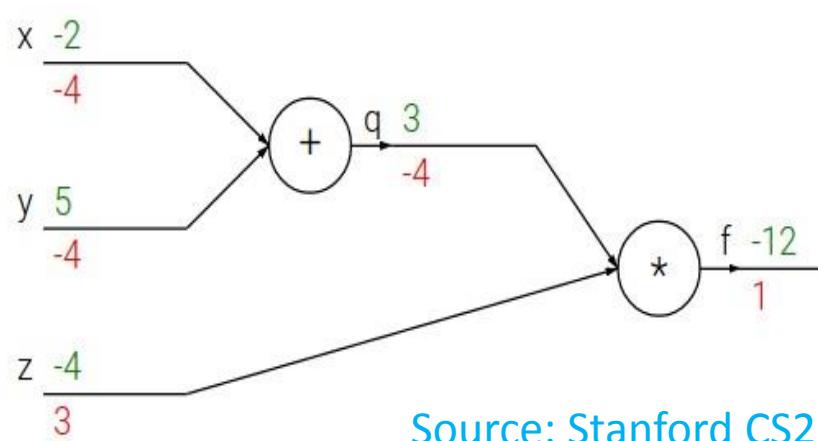
➤ Compound expressions(复合表达式): $f(x, y, z) = (x + y)z$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

➤ Chain rule(链规则):

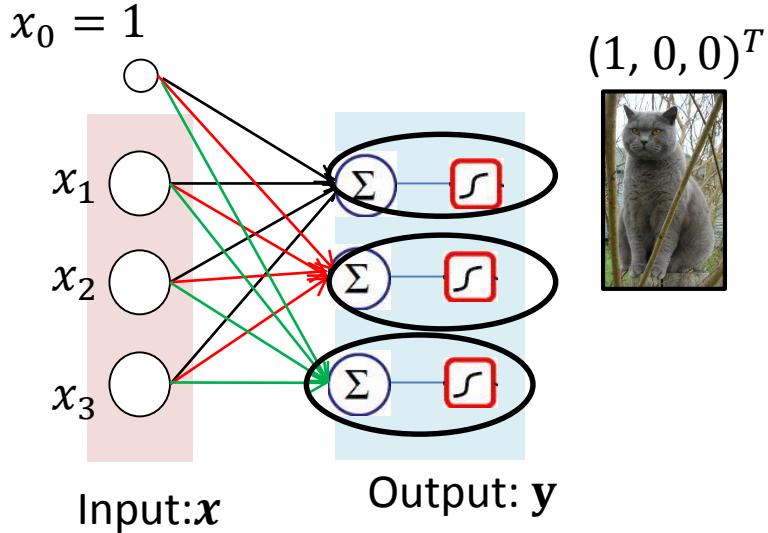
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$



Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

backPropagation

- Linear Classifier



➤ 1. 根据输入，计算输出值：

$$a_i = \sum_{j=0}^3 w_{ij} x_i = \mathbf{w}_i \cdot \mathbf{x}, \quad i = (1, 2, 3)$$

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

MSE Loss: $L = (\mathbf{y} - \hat{\mathbf{y}})^2$
 $= (1 - y_1)^2 + y_2^2 + y_3^2$

➤ 2. 根据链规则，计算梯度 $\frac{dL}{dw}$:

$$\frac{dL}{y_1} = 2(y_1 - 1)$$

$$\frac{dL}{y_i} = 2y_i, (i=2,3)$$

$$\frac{dL}{a_i} = \frac{dL}{y_i} \frac{dy_i}{a_i} = \frac{dL}{y_i} \sigma(a_i)(1 - \sigma(a_i))$$

$$\frac{dL}{w_{ij}} = \frac{dL}{a_i} \frac{da_i}{w_{ij}} = \frac{dL}{a_i} x_{ij}$$

$$= \frac{dL}{y_i} \sigma(a_i)(1 - \sigma(a_i))$$

\downarrow

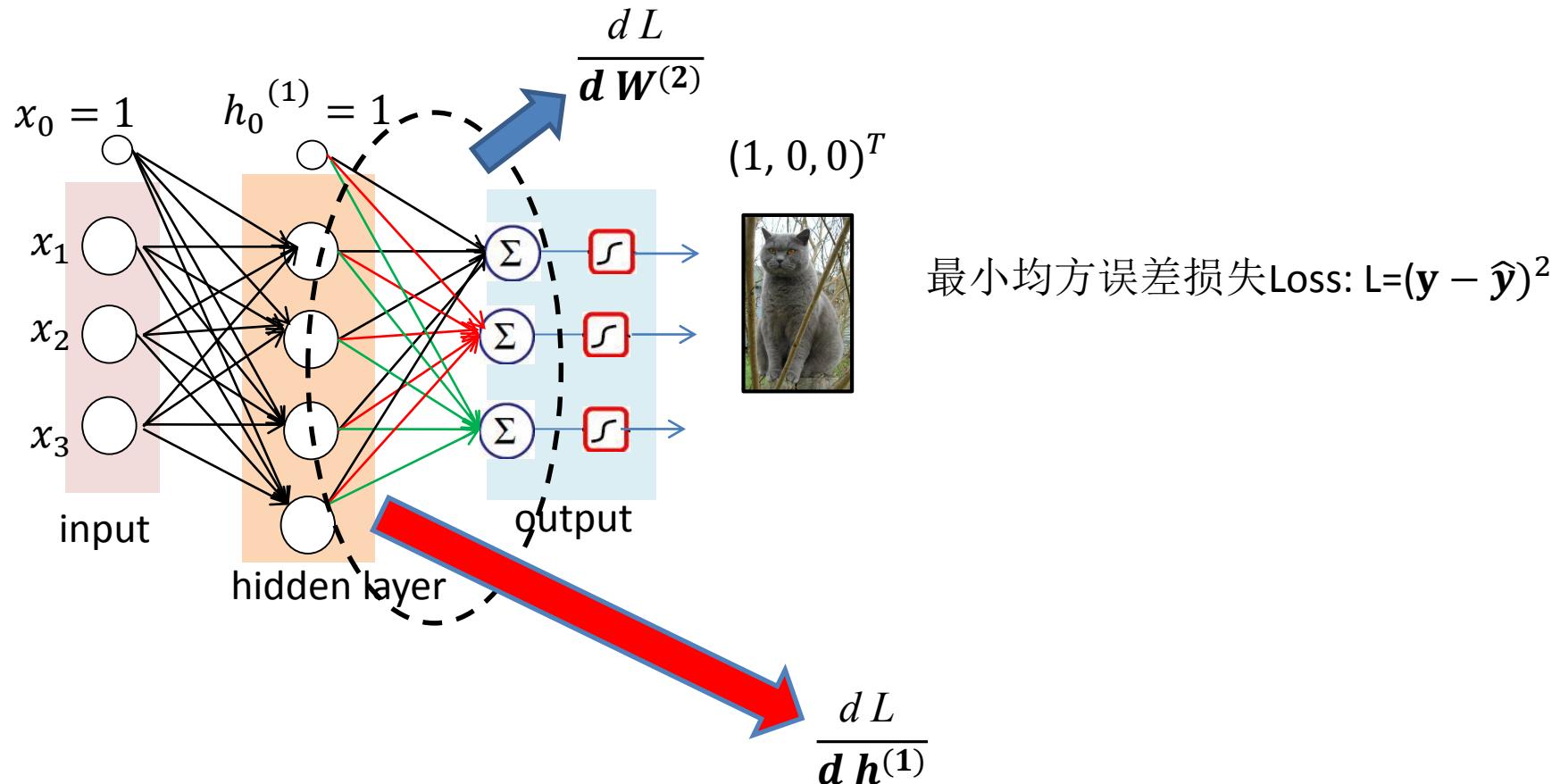
$$\frac{dL}{\mathbf{y}} = 2(\mathbf{y} - \hat{\mathbf{y}})$$

$$\frac{dL}{\mathbf{a}} = 2[(\mathbf{y} - \hat{\mathbf{y}}) \cdot \sigma(\mathbf{a}) \cdot (1 - \sigma(\mathbf{a}))]^T$$

$$\frac{dL}{\mathbf{w}} = 2[(\mathbf{y} - \hat{\mathbf{y}}) \cdot \sigma(\mathbf{a}) \cdot (1 - \sigma(\mathbf{a}))] \mathbf{x}$$

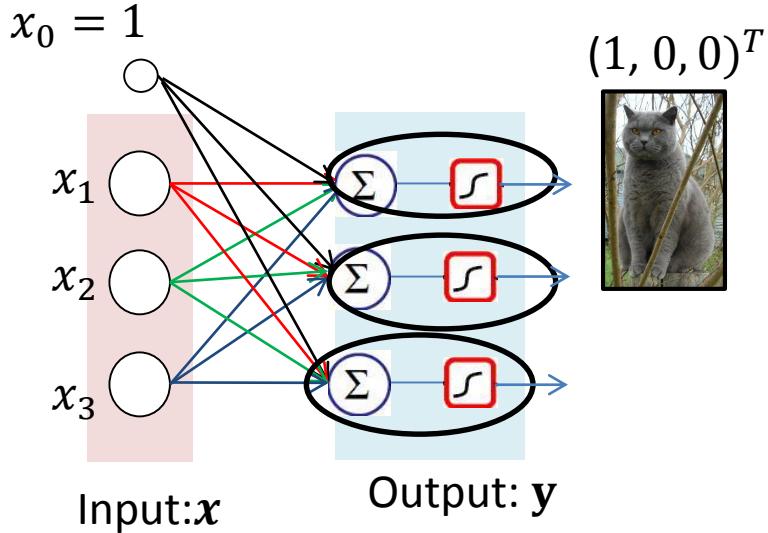
backPropagation

- Multi-Layer perceptron



backPropagation

- Linear Classifier



➤ 2. 根据链规则，计算梯度 $\frac{dL}{dw}$:

$$\frac{dL}{y_1} = 2(1 - y_1)$$

$$\frac{dL}{y_i} = 2y_i, (i=2,3)$$

$$\frac{dL}{a_i} = \frac{dL}{y_i} \frac{dy_i}{a_i} = \frac{dL}{y_i} \sigma(a_i)(1 - \sigma(a_i))$$

➤ 1. 根据输入，计算输出值：

$$a_i = \sum_{j=0}^3 w_{ij} x_i = \mathbf{w}_i \cdot \mathbf{x}, \quad i = (1, 2, 3)$$

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

$$\text{MSE Loss: } L = (\mathbf{y} - \hat{\mathbf{y}})^2 \\ = (1 - y_1)^2 + y_2^2 + y_3^2$$

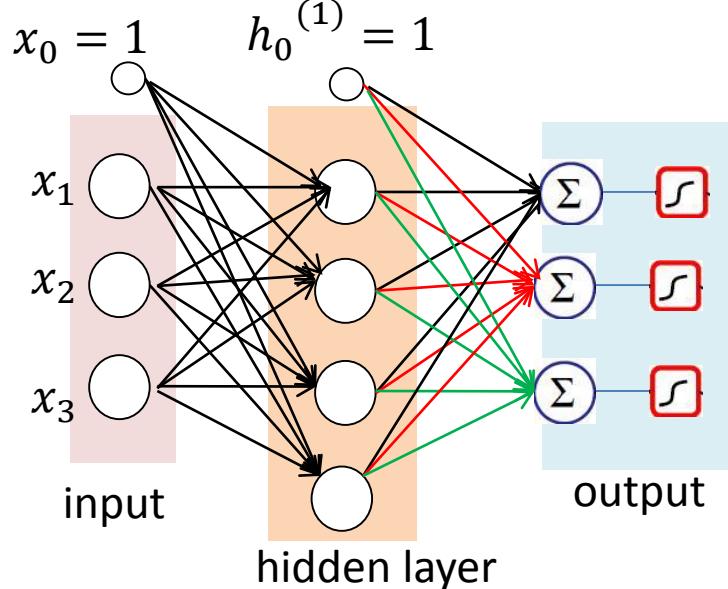
$$\frac{dL}{x_i} = \frac{dL}{a_i} \frac{da_i}{x_i} = \frac{dL}{a_i} \sum_{j=0}^3 w_{ij}$$



$$\frac{dL}{x} = \frac{dL}{a} W$$

backPropagation

- Multi-layer perceptron



➤ 1 根据输入，计算输出值：

$$\mathbf{a}^{(1)} = \mathbf{W}^{(1)} \cdot \mathbf{x}$$

$$\mathbf{h}^{(1)} = \sigma(\mathbf{a}^{(1)})$$

$$\mathbf{a}^{(2)} = \mathbf{W}^{(2)} \cdot \mathbf{h}^{(1)}$$

$$\mathbf{y} = \sigma(\mathbf{a}^{(2)})$$

➤ MSE Loss: $L = (\mathbf{y} - \hat{\mathbf{y}})^2$

➤ 2 根据链规则，计算梯度 $\frac{dL}{dx}$:

$$(1, 0, 0)^T$$



BackPropagation

$$\frac{dL}{\mathbf{y}} = 2(\mathbf{y} - \hat{\mathbf{y}})$$

$$\frac{dL}{da^{(2)}} = \frac{dL}{dy} \cdot \sigma(a^{(2)}) \cdot (1 - \sigma(a^{(2)}))$$

$$\frac{dL}{dh^{(1)}} = \frac{dL}{da^{(2)}} W^{(2)}$$

$$\frac{dL}{a^{(1)}} = \frac{dL}{dh^{(1)}} \cdot \sigma(a^{(1)}) \cdot (1 - \sigma(a^{(1)}))$$

$$\frac{dL}{dx} = \frac{dL}{da^{(1)}} W^{(1)}$$

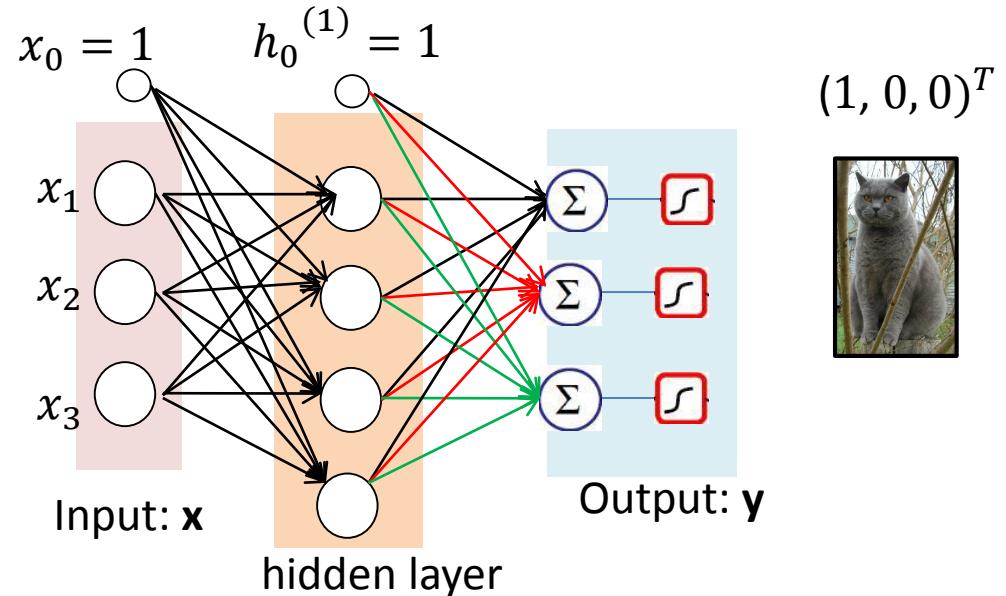
➤ 3. 根据链规则，计算梯度 $\frac{dL}{dw}$:

$$\frac{dL}{dW^{(2)}} = \frac{dL}{da^{(2)}} h^{(1)}$$

$$\frac{dL}{dW^{(1)}} = \frac{dL}{da^{(1)}} \cancel{x}$$

backPropagation

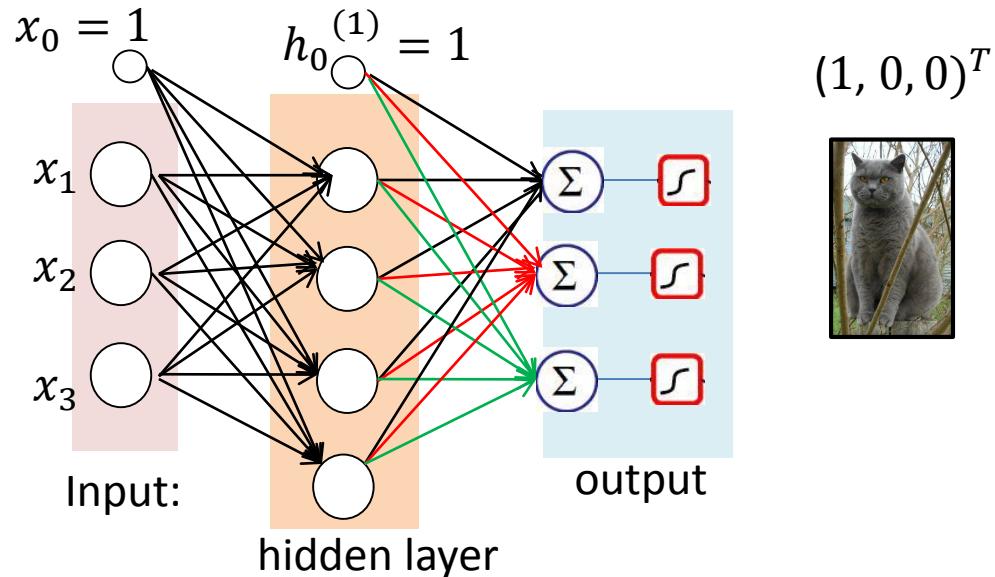
- Conclusion
 - Calculate gradient
 - Similar as forward:
 - Input: $\frac{d L}{dy}$
 - Output: $\frac{d L}{dx}$



Multi-layer perceptron

- Training Algorithm

- 0. 初始化权重 $\mathbf{W}^{(0)}$
- 1. 前向过程:
 - 1.1 根据输入, 计算输出值 \mathbf{y}
 - 1.2. 计算损失函数值 $L(\mathbf{y}, \hat{\mathbf{y}})$ 。
- 2. 后向传播
 - 计算 $\frac{d L}{\mathbf{y}}$
 - 后向传播直到计算 $\frac{d L}{\mathbf{x}}$
- 3. 计算梯度 $\frac{d L}{d \mathbf{W}}$
- 4. 更新梯度
$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \frac{d L}{d \mathbf{W}^{(t)}}$$



Engineering in practice

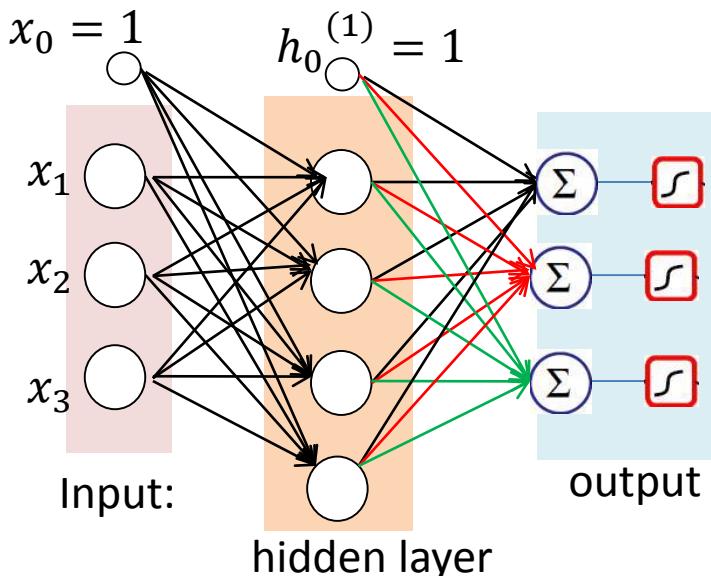
➤ Torch 平台

- **Model construction**

```
function create_model()
    model = nn.Sequential()
    model:add(nn.Linear(3, 4))
    model:add(nn.Sigmoid())
    model:add(nn.Linear(4, 3))
    model:add(nn.Sigmoid())
    criterion = nn.MSECriterion()
    return model, criterion
end
```

- **Training per iteration:**

```
-- forward
outputs = model:forward(X)
loss = criterion:forward(outputs, Y)
-- backward
dloss_doutput = criterion:backward(outputs, Y)
model:backward(X, dloss_doutput)
```

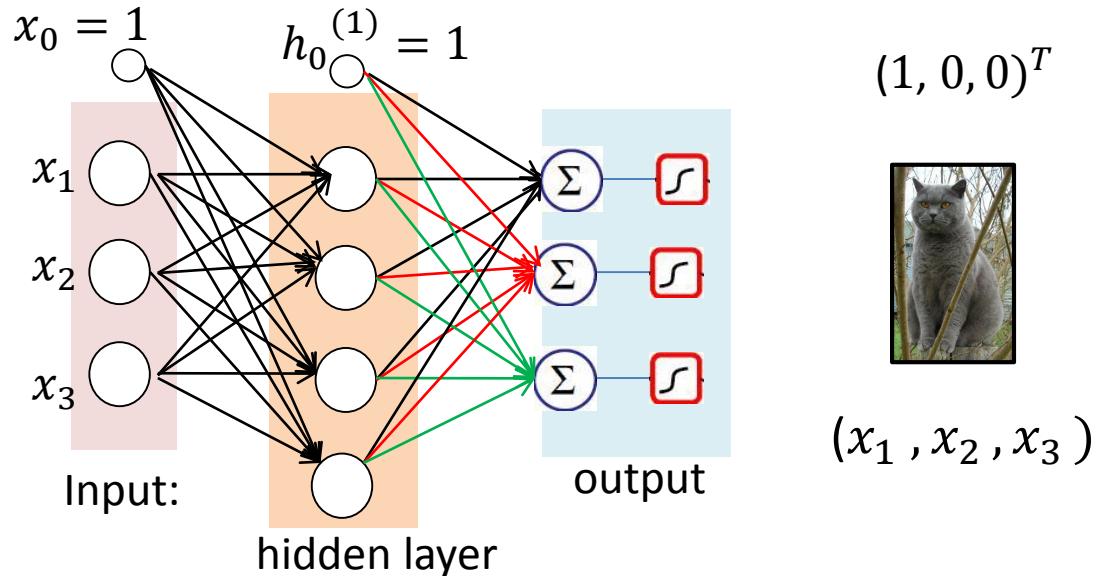


Some analyses

- Feature extraction
 - Pixel-wise input
 - High dimension
 - Correlation between features



Convolutional Neural Network(CNN), 卷积神经网络



(x_1, x_2, x_3)



08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	71	31	45
49	49	99	40	17	81	18	57	60	87	17	40	98	49	69	14	56	62	00	
81	49	31	73	55	79	14	29	93	72	40	67	00	30	09	49	13	36	65	
92	70	95	23	04	60	11	42	63	05	56	01	32	56	71	37	02	36	91	
22	31	16	71	51	67	13	59	41	92	36	54	22	40	40	28	66	33	13	80
24	47	31	05	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
15	35	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
47	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	03	14	88	34	89	63	72
41	36	23	09	75	00	76	44	20	45	35	14	00	62	33	97	34	31	33	95
78	17	53	28	22	78	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	94	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	08	48	35	71	89	07	08	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	21	73	99	13	88	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	14	26	28	79	33	27	98	66
44	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69	
64	42	26	73	05	55	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	05	88	69	82	67	59	85	74	04	36	16
40	79	35	29	78	31	90	01	71	32	49	74	48	04	14	24	23	57	05	54
01	70	54	73	83	51	54	69	36	92	33	48	61	43	52	01	79	14	45	

What the computer sees