深度学习讨论班

第二节
Multi-layer perceptron
(多层感知机)

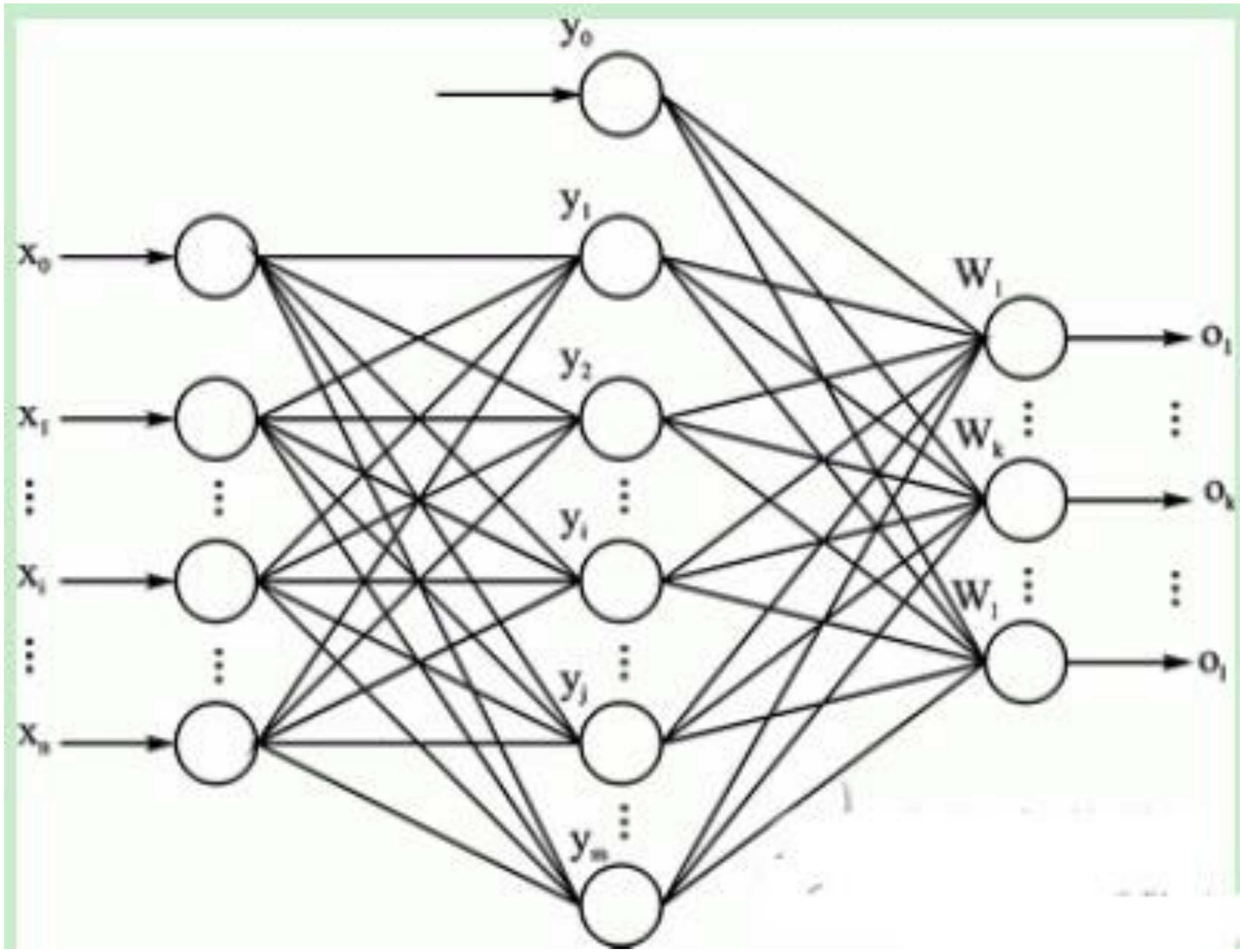
黄雷

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

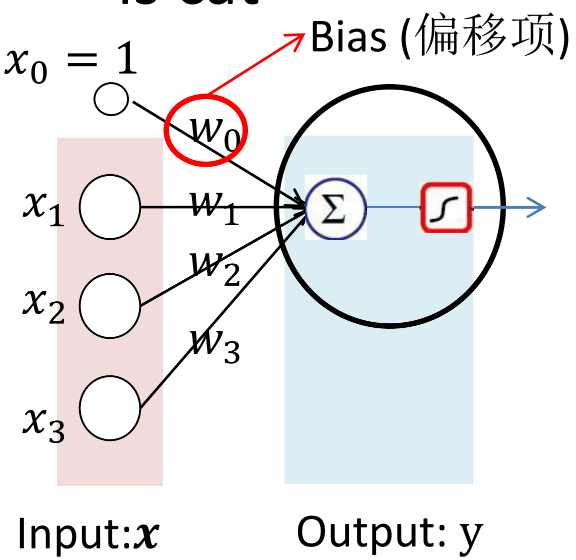
$$(x_1, x_2, x_3)^T$$

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?



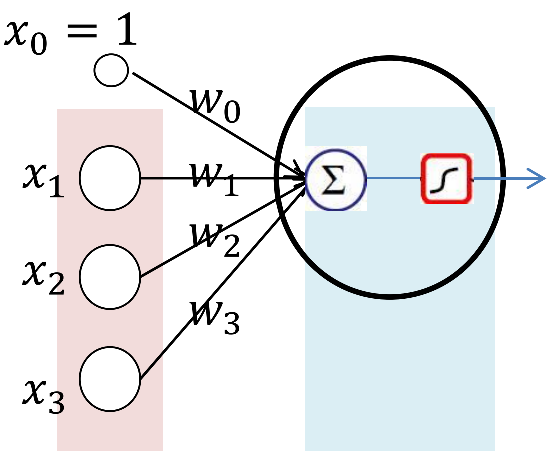
$(x_1, x_2, x_3)^T$

$$a = \sum_{i=0}^{3} w_i x_i = \mathbf{w} \cdot \boldsymbol{x}$$

$$y = \sigma(a) = \frac{1}{1 + e^{-a}}$$

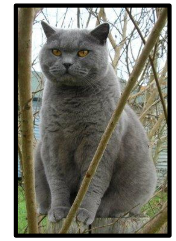
Linear Classifier

- One example



Input: $\boldsymbol{x}$ Output: y

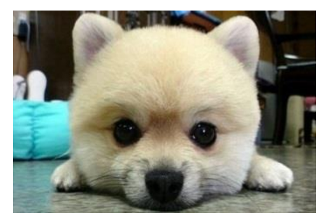
Model: $\mathbf{w}$= $(w_0, w_1, w_2, w_3)^T$
=$(2, 0,0, 4)^T$



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

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*

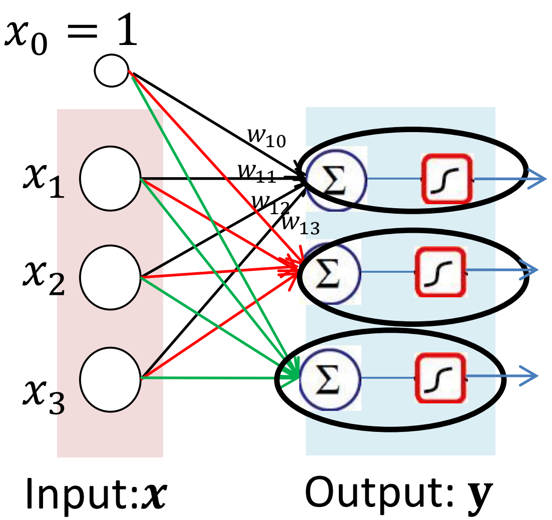
$$y = \varphi(a) = \frac{1}{1 + e^{10}} < 0.5$$

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？



(x_1, x_2, x_3)

w_{ij}: 第 i 个输出神经元，连接第j 个输入单元

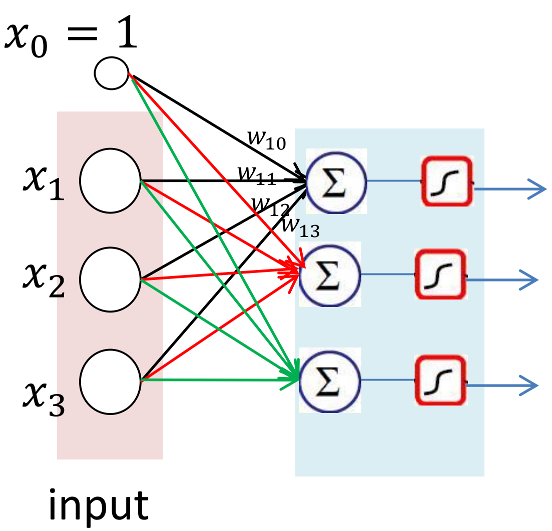
$$a_i = \sum_{j=0}^{3} w_{ij} x_i = \boldsymbol{w}_{i.} \cdot \boldsymbol{x}, \qquad i = (1,2,3)$$

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

$$\boldsymbol{a} = \boldsymbol{W} \cdot \boldsymbol{x}$$
$$\mathbf{y} = \boldsymbol{\sigma}(\boldsymbol{a})$$

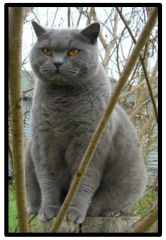
Linear Classifier (线性分类器)

- One example



Model: $\boldsymbol{W}$=

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



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

$\boldsymbol{a} =$

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

*(1,2,2,3)^T

$= (14, -13, -1)$

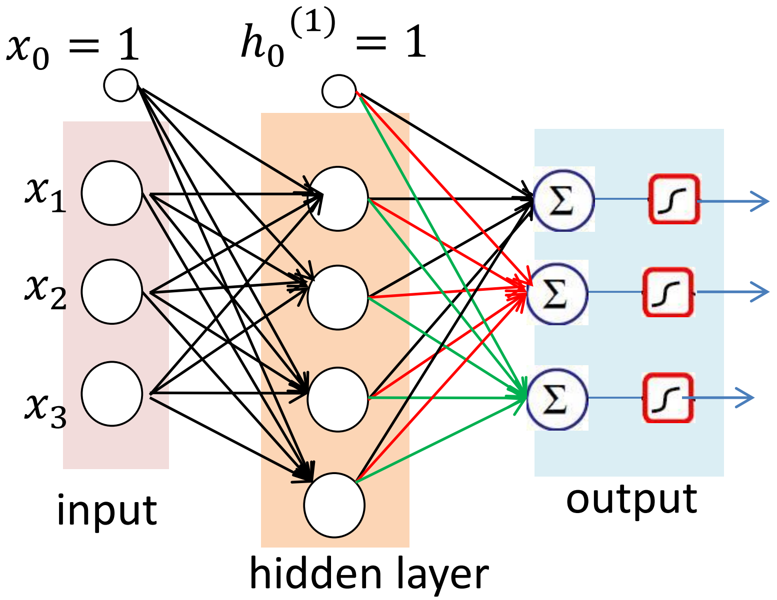
$$\mathbf{y} = \left(\frac{1}{1+e^{-14}}, \frac{1}{1+e^{13}}, \frac{1}{1+e^{1}}\right)^T$$

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

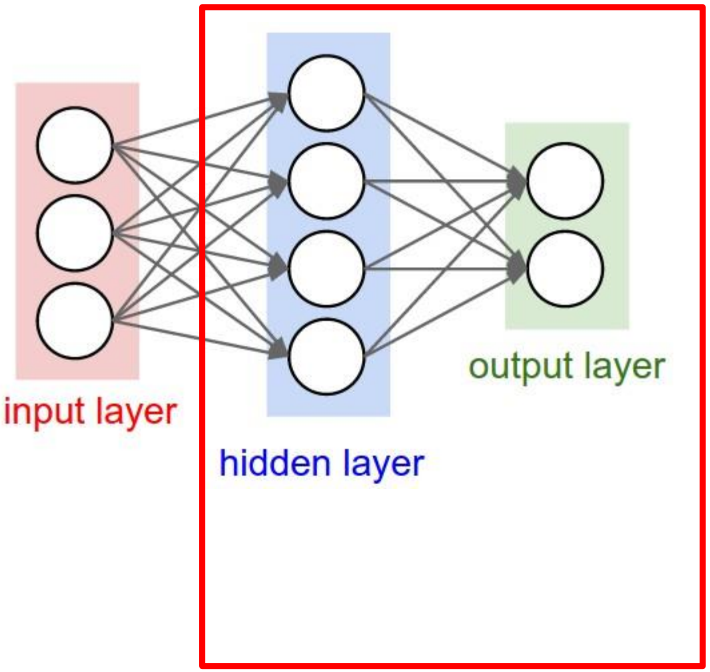
activation → $\boldsymbol{h}^{(1)} = \sigma(\boldsymbol{a}^{(1)})$

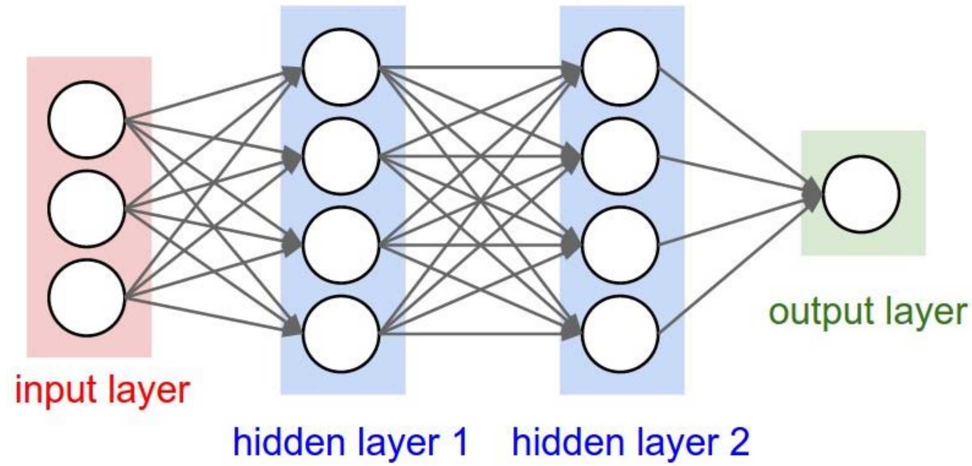
$$\boldsymbol{a}^{(2)} = \boldsymbol{W}^{(2)} \cdot \boldsymbol{h}^{(1)}$$
$$\boldsymbol{y} = \sigma(\boldsymbol{a}^{(2)})$$

⇒

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

Neural Networks: Architectures



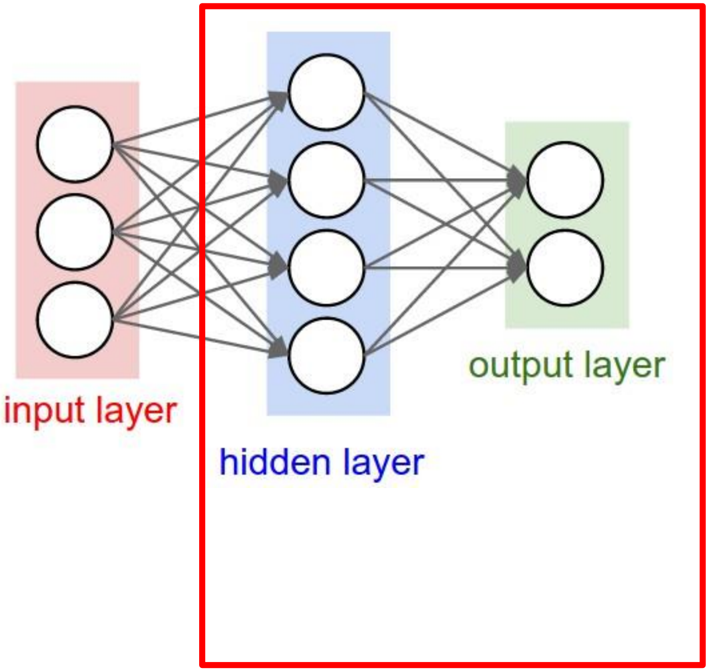


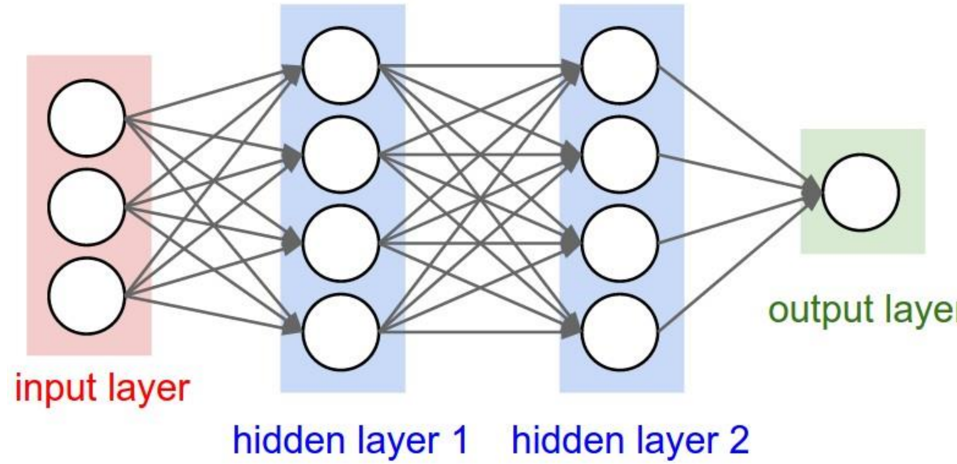
“2-layer Neural Net”, or
“1-hidden-layer Neural Net”

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

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

Neural Networks: Architectures

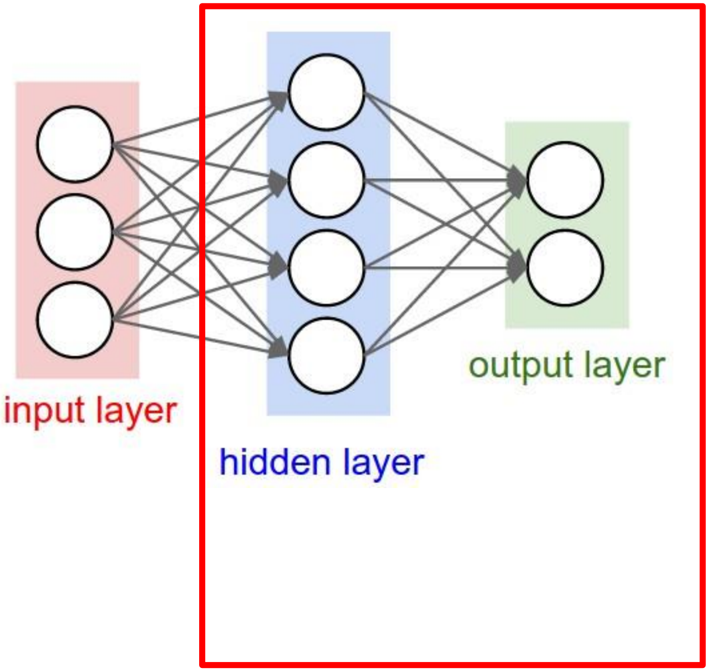


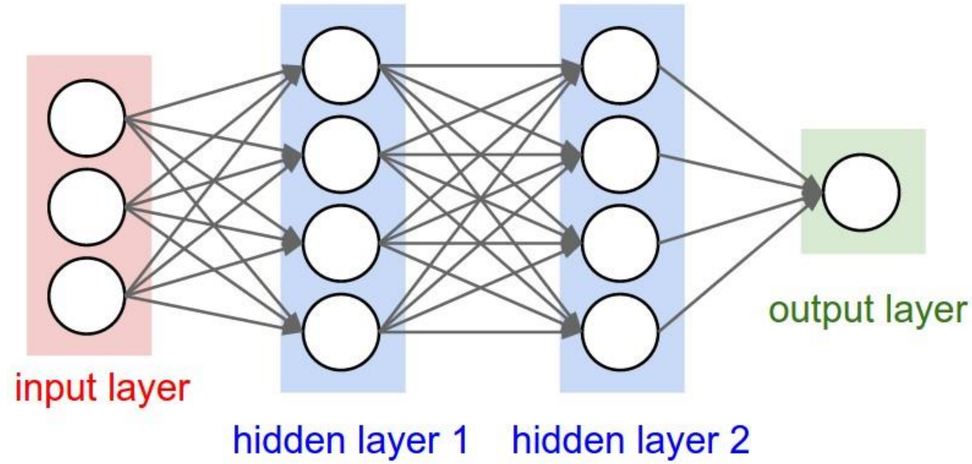


Number of Neurons: ?
Number of Weights: ?
Number of Parameters: ?

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

Neural Networks: Architectures



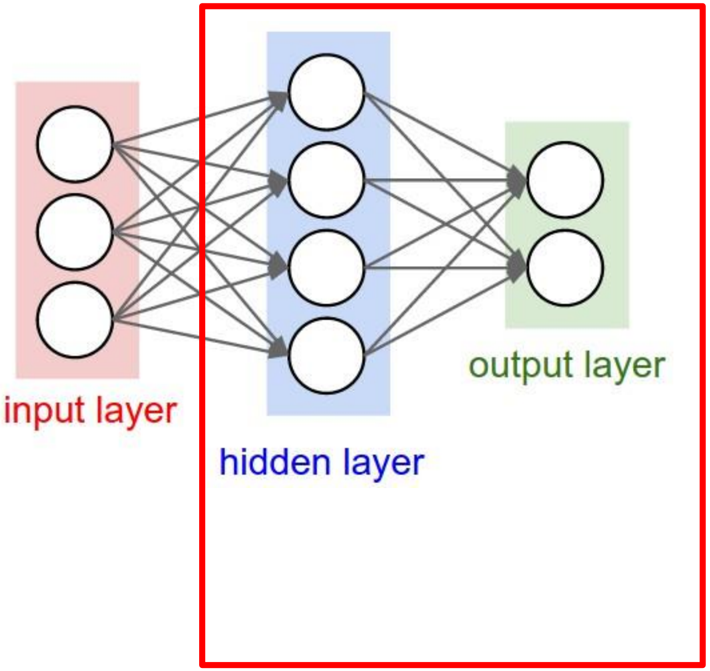


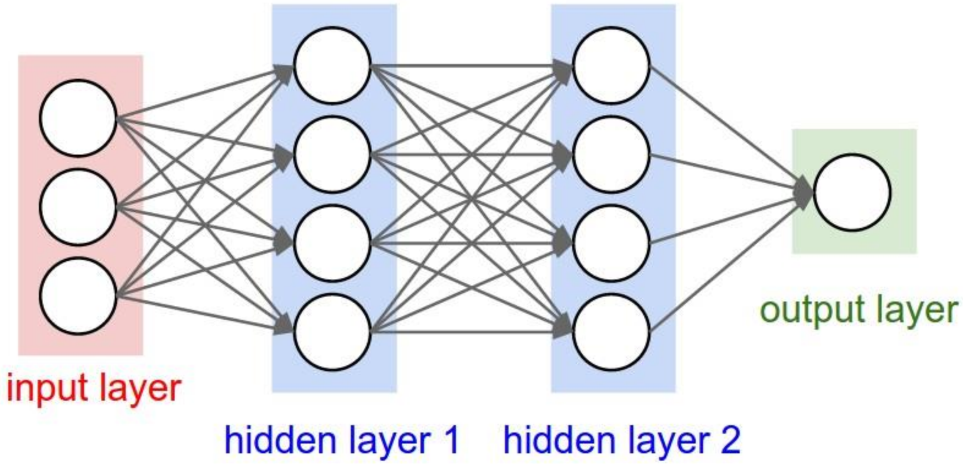
Number of Neurons: 4+2 = 6
Number of Weights: [4x3 + 2x4] = 20
Number of Parameters: 20 + 6 = 26 (biases!)

Number of Neurons: ?
Number of Weights: ?
Number of Parameters: ?

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

Neural Networks: Architectures



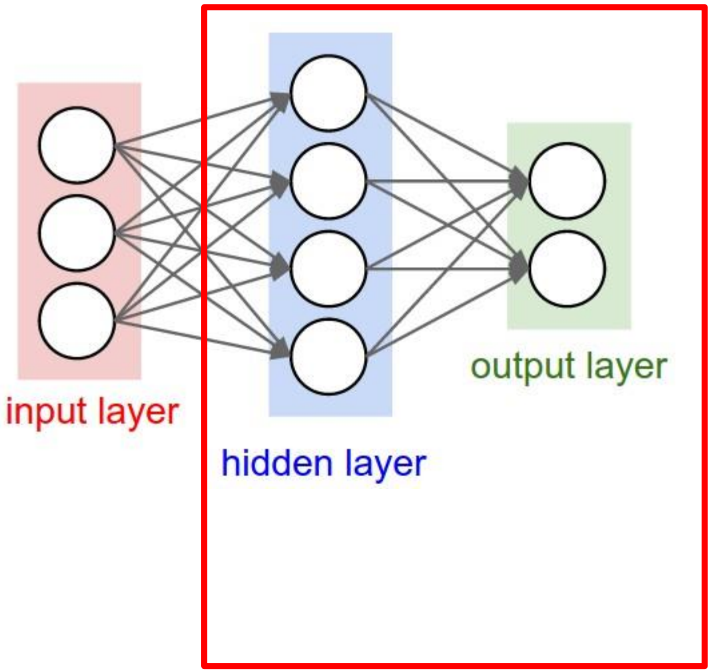


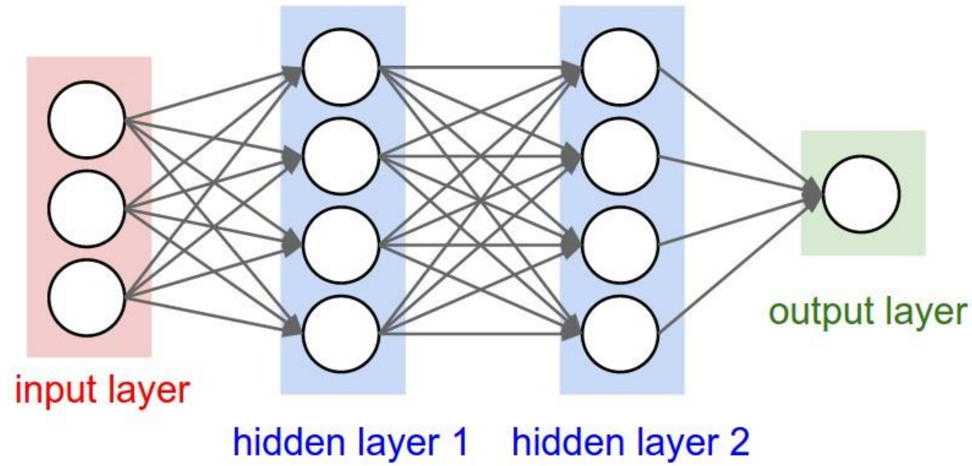
Number of Neurons: 4+2 = 6
Number of Weights: [4x3 + 2x4] = 20
Number of Parameters: 20 + 6 = 26 (biases!)

Number of Neurons: 4 + 4 + 1 = 9
Number of Weights: [4x3+4x4+1x4]=32
Number of Parameters: 32+9 = 41

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

Neural Networks: Architectures





Modern CNNs: ~10 million neurons
Human visual cortex: ~5 billion neurons

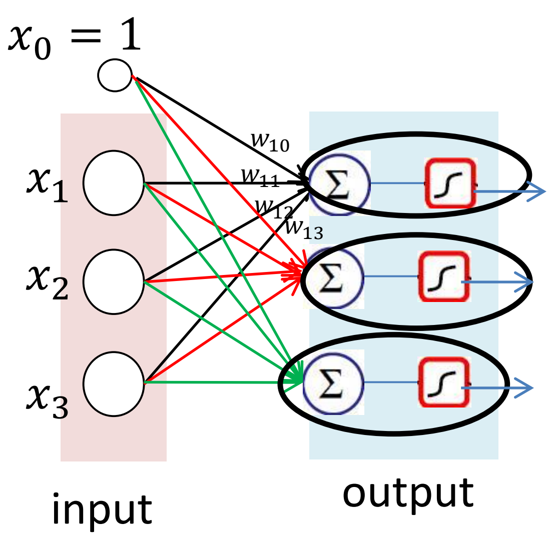
Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

outline

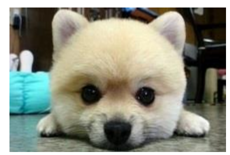
- Linear classifier (简单线性分类器)
  - One neuron (一个神经元)
  - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
  - Model representation (模型表示)
  - Loss function: the goal for learning
  - Training
    - Gradient based optimization
    - backpropagation

Target of learning: Loss function

- Loss function



$(1, 0, 0)^T$ ⇨ L=$(\mathbf{y} - \hat{\mathbf{y}})^2$
=$(1 - y_1)^2 + y_2{}^2 + y_3{}^2$



$(0, 1, 0)^T$

$(0, 0, 1)^T$

$$\boldsymbol{a} = \boldsymbol{W} \cdot \boldsymbol{x}$$
$$\mathbf{y} = \sigma(\boldsymbol{a})$$

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

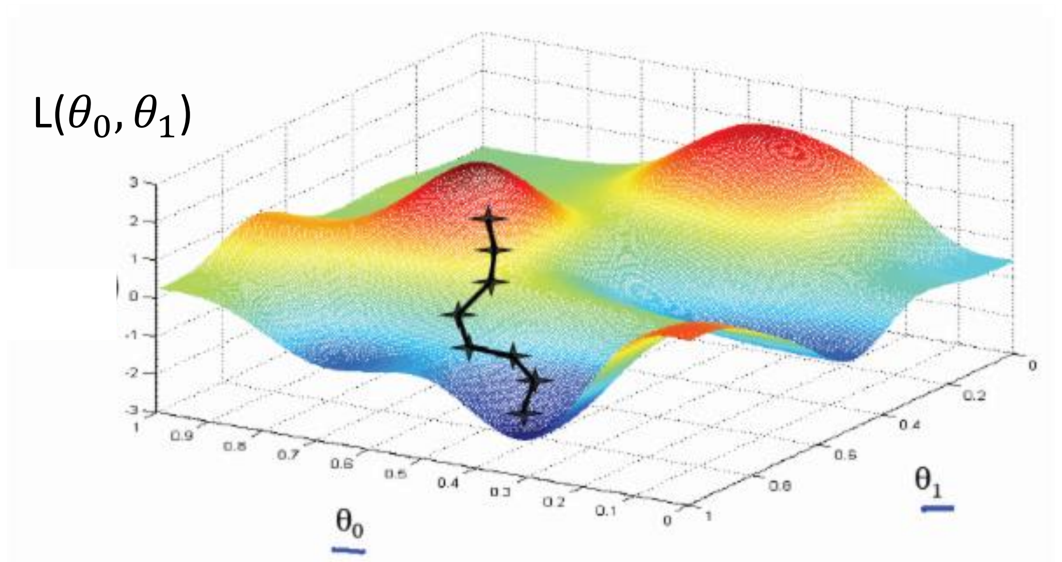
How to learn: adjust parameters

- Gradient descent (梯度下降法)

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

学习率

下降方向为负的梯度方向



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

Calculate gradient: back-Propagation

- 0.derivative

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

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

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

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

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

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： $\sigma(x) = \frac{1}{1+e^{-x}}$

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

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

backPropagation

- 2.Chain rule

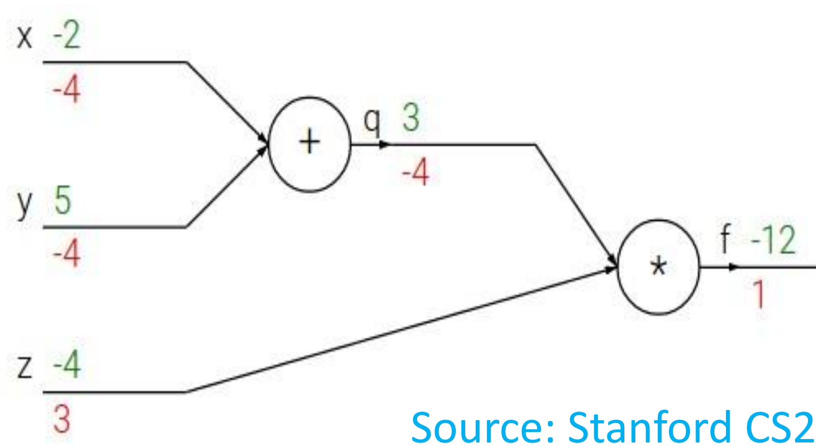
  - Compound expressions(复合表达式): $f(x,y,z) = (x+y)z$

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

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

  - Chain rule(链规则):

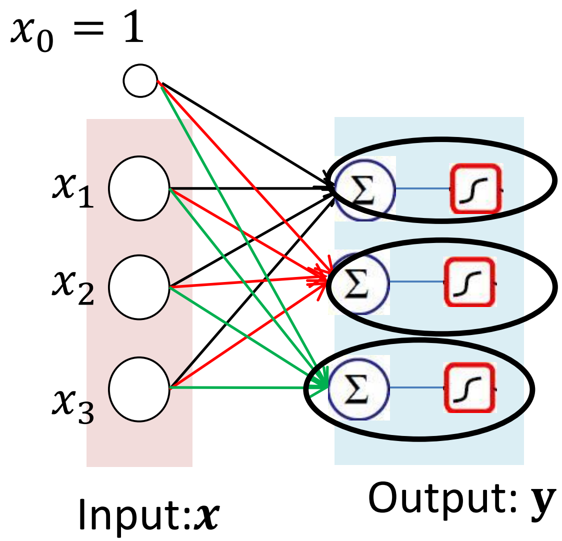
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$



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

backPropagation

- Linear Classifier



$(1, 0, 0)^T$

➢ 1.根据输入，计算输出值：

$$a_i = \sum_{j=0}^{3} w_{ij} x_i = \boldsymbol{w}_{i.} \cdot \boldsymbol{x}, \qquad i = (1,2,3)$$

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

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 $\frac{dL}{d\boldsymbol{W}}$ ：

$$\frac{\mathrm{d\,L}}{y_1} = 2(y_1 - 1)$$

$$\frac{\mathrm{d\,L}}{y_i} = 2y_i, \text{(i=2,3)}$$

$$\frac{\mathrm{d\,L}}{a_i} = \frac{\mathrm{d\,L}}{y_i}\frac{\mathrm{d}y_i}{a_i} = \frac{\mathrm{d\,L}}{y_i}\sigma(a_i)(1\text{-}\sigma(a_i))$$

$$\frac{\mathrm{d\,L}}{w_{ij}} = \frac{\mathrm{d\,L}}{a_i}\frac{\mathrm{d}a_i}{w_{ij}} = \frac{\mathrm{d\,L}}{a_i}x_{ij}$$

$$= \frac{\mathrm{d\,L}}{y_i}\sigma(a_i)(1\text{-}\sigma(a_i))$$

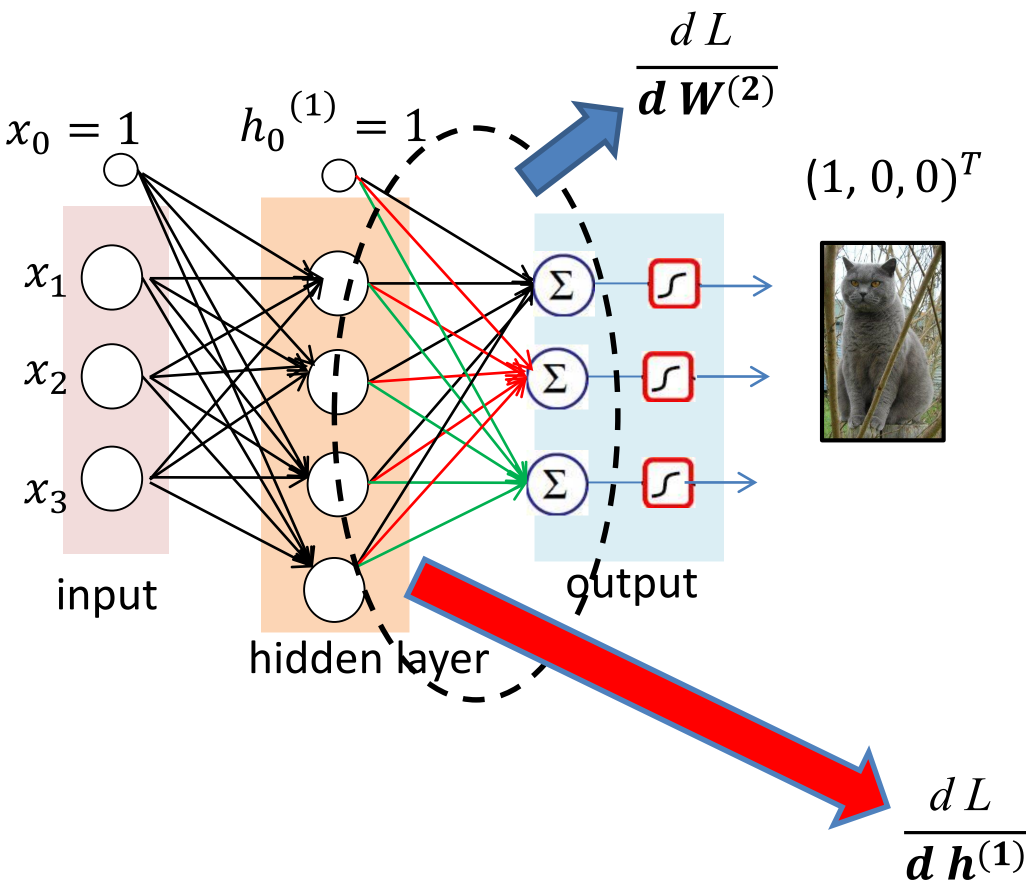
$$\frac{\mathrm{d\,L}}{\mathbf{y}} = 2(\mathbf{y} - \hat{\mathbf{y}})$$

$$\frac{\mathrm{d\,L}}{\boldsymbol{a}} = 2[(\mathbf{y} - \hat{\mathbf{y}}) \cdot \sigma(\boldsymbol{a}) \cdot (1 - \sigma(\boldsymbol{a}))]^T$$

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

backPropagation

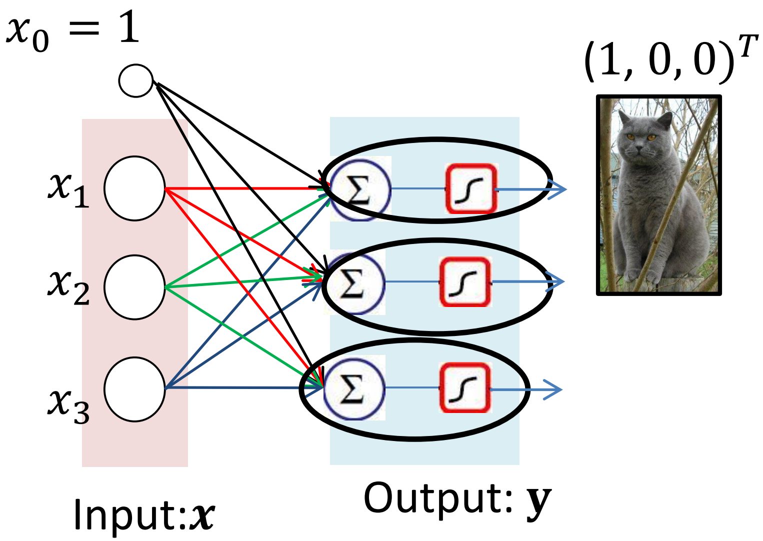
- Multi-Layer perceptron



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

backPropagation

- Linear Classifer



➢ 1.根据输入，计算输出值：

$$a_i = \sum_{j=0}^{3} w_{ij} x_i = \boldsymbol{w}_{i.} \cdot \boldsymbol{x}, \qquad i = (1,2,3)$$

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

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 $\frac{d\,L}{d\,\boldsymbol{W}}$ ：

$$\frac{\mathrm{d\,L}}{y_1} = 2(1 - y_1)$$

$$\frac{\mathrm{d\,L}}{y_i} = 2y_i, \text{(i=2,3)}$$

$$\frac{\mathrm{d\,L}}{a_i} = \frac{\mathrm{d\,L}}{y_i}\frac{\mathrm{d}y_i}{a_i} = \frac{\mathrm{d\,L}}{y_i}\sigma(a_i)(1\text{-}\sigma(a_i))$$

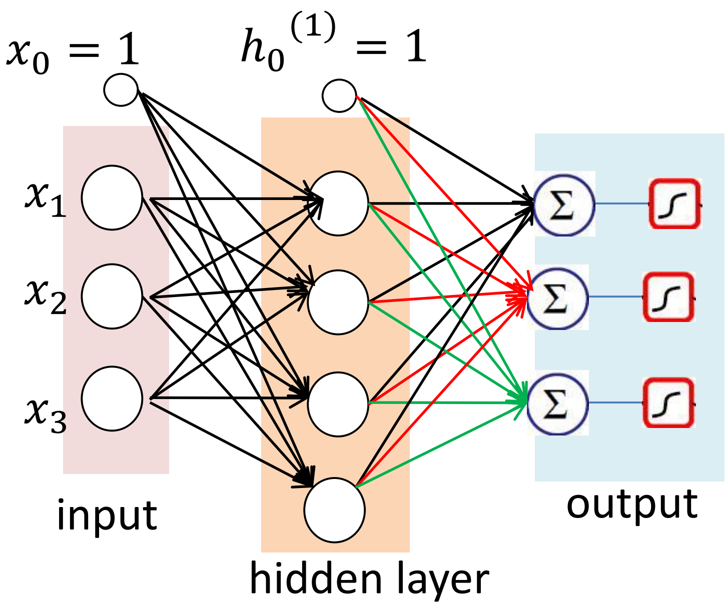
$$\frac{\mathrm{d\,L}}{x_i} = \frac{\mathrm{d\,L}}{a_i}\frac{\mathrm{d}a_i}{x_i} = \frac{\mathrm{d\,L}}{a_i}\sum_{j=0}^{3} w_{ij}$$

⇩

$$\frac{\mathrm{d\,L}}{\boldsymbol{x}} = \frac{\mathrm{d\,L}}{\boldsymbol{a}}\boldsymbol{W}$$

backPropagation

- Multi-layer perceptron



$(1, 0, 0)^T$

➤ 1根据输入，计算输出值：

$$\boldsymbol{a}^{(1)} = \boldsymbol{W}^{(1)} \cdot \boldsymbol{x}$$
$$\boldsymbol{h}^{(1)} = \sigma\left(\boldsymbol{a}^{(1)}\right)$$
$$\boldsymbol{a}^{(2)} = \boldsymbol{W}^{(2)} \cdot \boldsymbol{h}^{(1)}$$
$$\boldsymbol{y} = \sigma\left(\boldsymbol{a}^{(2)}\right)$$

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

➤ 2根据链规则，计算梯度$\frac{d\,L}{d\,\boldsymbol{x}}$：

$$\frac{\mathrm{d\,L}}{\mathbf{y}} = 2(\mathbf{y} - \hat{\mathbf{y}})$$
$$\frac{\mathrm{d\,L}}{d\boldsymbol{a}^{(2)}} = \frac{\mathrm{d\,L}}{\mathbf{dy}} \cdot \sigma\left(\boldsymbol{a}^{(2)}\right) \cdot \left(1 - \sigma\left(\boldsymbol{a}^{(2)}\right)\right)$$
$$\frac{d\,L}{d\,\boldsymbol{h}^{(1)}} = \frac{\mathrm{d\,L}}{d\boldsymbol{a}^{(2)}} \boldsymbol{W}^{(2)}$$
$$\frac{\mathrm{d\,L}}{\boldsymbol{a}^{(1)}} = \frac{\mathrm{d\,L}}{d\,\boldsymbol{h}^{(1)}} \cdot \sigma\left(\boldsymbol{a}^{(1)}\right) \cdot \left(1 - \sigma\left(\boldsymbol{a}^{(1)}\right)\right)$$
$$\frac{d\,L}{d\boldsymbol{x}} = \frac{\mathrm{d\,L}}{d\boldsymbol{a}^{(1)}} \boldsymbol{W}^{(1)}$$

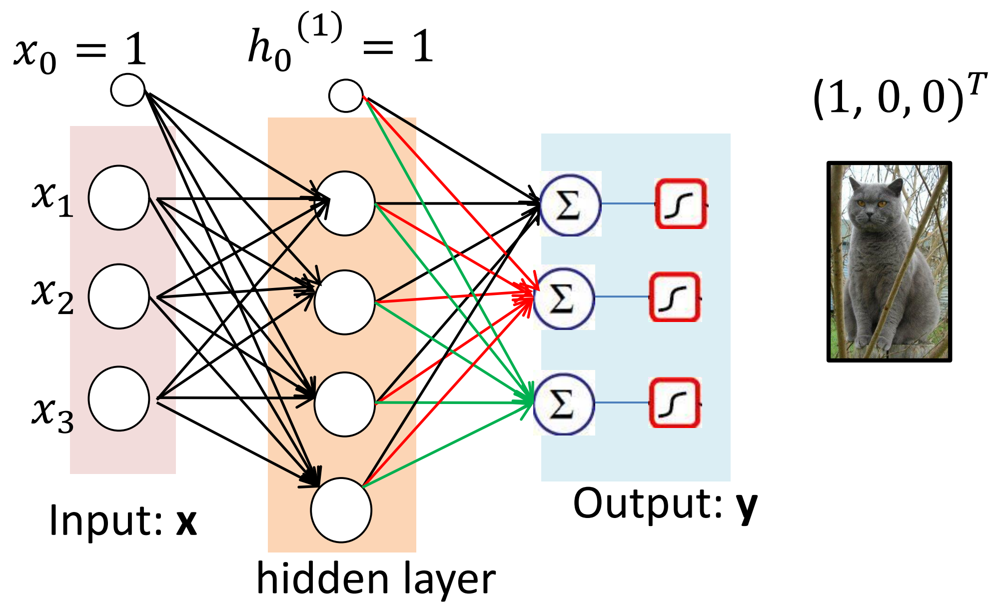
BackPropagation

➤ 3.根据链规则，计算梯度 $\frac{d\,L}{d\,\boldsymbol{W}}$ ：

$$\frac{d\,L}{d\,\boldsymbol{W}^{(2)}} = \frac{\mathrm{d\,L}}{d\boldsymbol{a}^{(2)}} \boldsymbol{h}^{(1)}$$
$$\frac{d\,L}{d\,\boldsymbol{W}^{(1)}} = \frac{\mathrm{d\,L}}{d\boldsymbol{a}^{(1)}} \boldsymbol{x}$$

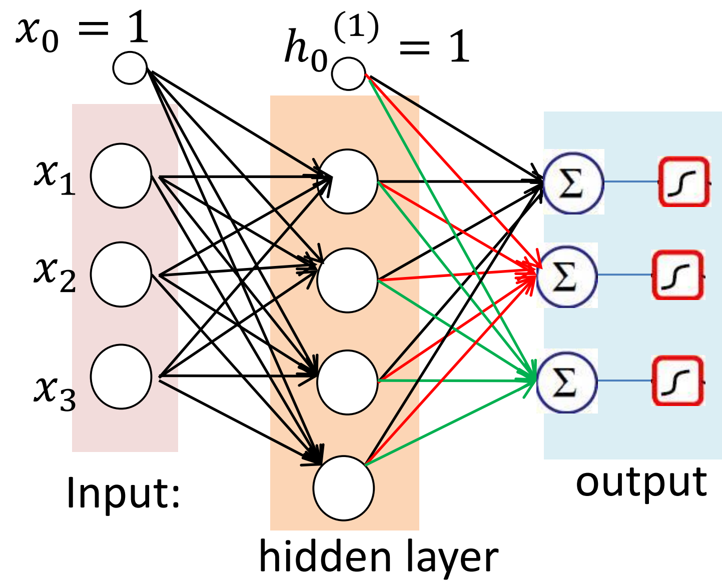
backPropagation

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

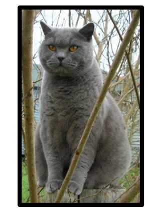


Multi-layer perceptron

- Training Algorithm



$(1, 0, 0)^T$



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

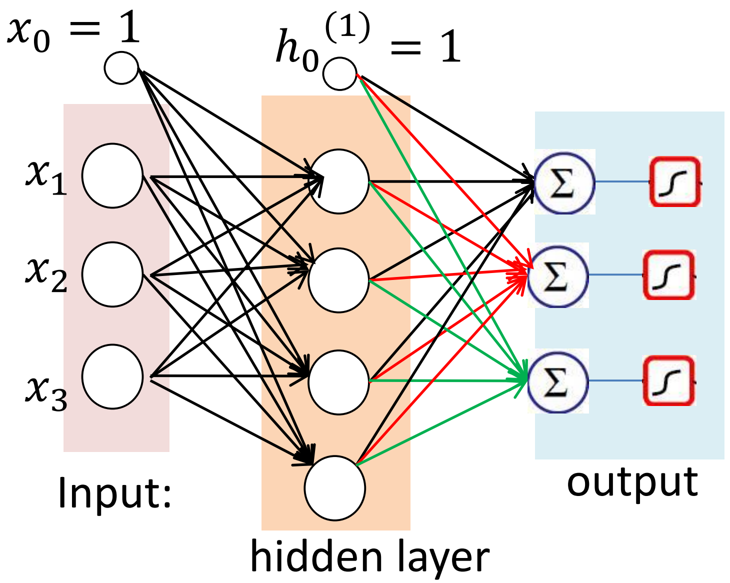
$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta \frac{d\,\mathrm{L}}{\boldsymbol{d}\,\boldsymbol{W}^{(t)}}$$

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
```



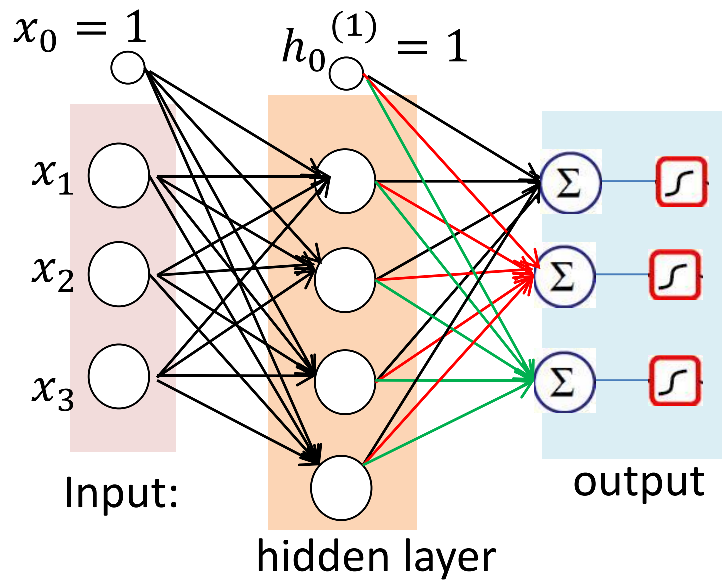
- **Training per iteration:**

```
-- 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



$(1, 0, 0)^T$

(x_1, x_2, x_3)