机器学习

Machine learning

第十章 神经网络与深度学习（2）

Neural Network and Deep Learning

授课人：周晓飞
zhouxiaofei@iie.ac.cn
2020-12-19

第十章 神经网络与深度学习

10.1 概述

10.2 多层感知机

10.3 卷积网络

10.4 Recurrent 网络

10.5 深度学习

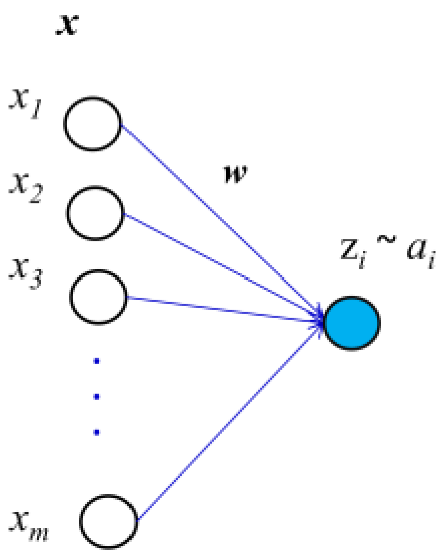
多层感知机

网络结构

神经元的运算

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$a = f(z)$$



多层感知机

网络结构

多层感知机

含有数据输入层、1 个以上隐藏层、1 个输出层；

各层神经元全连接，同一层神经元之间无连接。

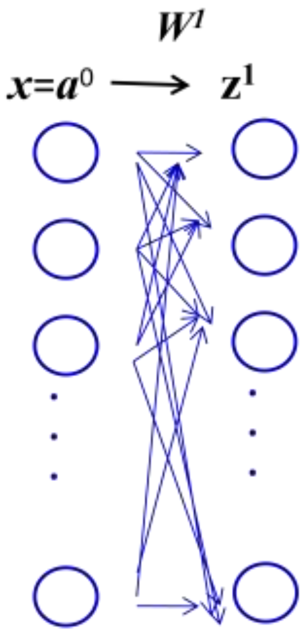
多层感知机

网络结构

多层感知机

含有数据输入层、1 个以上隐藏层、1 个输出层；

各层神经元全连接，同一层神经元之间无连接。



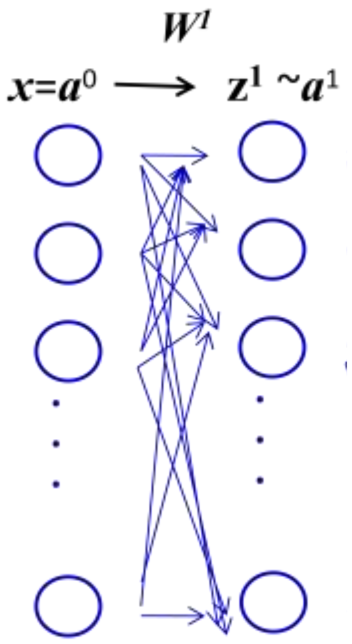
多层感知机

网络结构

多层感知机

含有数据输入层、1 个以上隐藏层、1 个输出层；

各层神经元全连接，同一层神经元之间无连接。



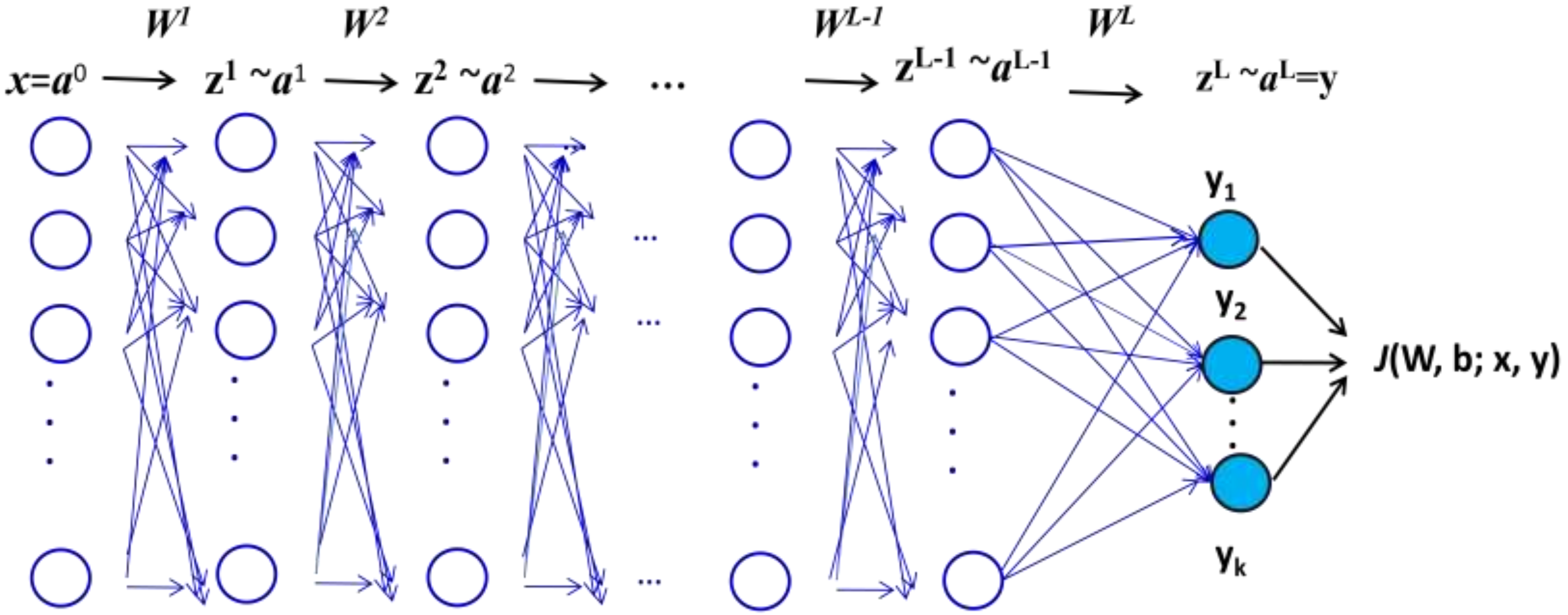
多层感知机

网络结构

多层感知机

含有数据输入层、1 个以上隐藏层、1 个输出层；

各层神经元全连接，同一层神经元之间无连接。



网络结构

多层感知机的运算

$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)})$$

Definitions:

- L Number of Layers;
- n^l Number of neurons in l-th layer;
- $f_l(\cdot)$ Activation function in l-th layer;
- $W^{(l)} \in \mathbb{R}^{n^l \times n^{l-1}}$ weight matrix between $l-1$-th layer and l-th layer;
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n^l}$ bias vector between $l-1$-th layer and l-th layer;
- $\mathbf{z}^{(l)} \in \mathbb{R}^{n^l}$ state vector of neurons in l-th layer;
- $\mathbf{a}^{(l)} \in \mathbb{R}^{n^l}$ activation vector of neurons in l-th layer.

激活函数（包括硬门限阈值函数），是导致网络运算非线性的直接原因。

多层感知机

问题描述

学习问题

- 数据

$$\mathcal{T} = \{ \mathbf{x}^{(i)}, y^{(i)} \}_{n=1}^{N}$$

- 输出的瞬时（第 *i* 次）损失

$$J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})$$

- 训练样本上的（N 个）平均损失

$$\sum_{i=1}^{N} J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})$$

学习目标：调整神经元连接权重值，使得平均误差能量最小。

两种方法：批量学习和在线学习。

多层感知机

问题描述

目标：最小化损失函数

$$J(W,\mathbf{b}) = \sum_{i=1}^{N} J(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)})$$

$$\begin{aligned} W^{(l)} &= W^{(l)} - \alpha\frac{\partial J(W,\mathbf{b})}{\partial W^{(l)}}, \\ &= W^{(l)} - \alpha\sum_{i=1}^{N}\left(\frac{\partial J(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)})}{\partial W^{(l)}}\right), \\ \mathbf{b}^{(l)} &= \mathbf{b}^{(l)} - \alpha\frac{\partial J(W,\mathbf{b})}{\partial \mathbf{b}^{(l)}}, \\ &= \mathbf{b}^{(l)} - \alpha\sum_{i=1}^{N}\left(\frac{\partial J(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)})}{\partial \mathbf{b}^{(l)}}\right), \end{aligned}$$

重要的问题：求取参数梯度

多层感知机

问题描述

目标：最小化损失函数

$$J(W,\mathbf{b}) = \sum_{i=1}^{N} J(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)}) + \boxed{\frac{1}{2}\lambda\|W\|_F^2},$$

可以引入正则项

$$W^{(l)} = W^{(l)} - \alpha\frac{\partial J(W,\mathbf{b})}{\partial W^{(l)}},$$

$$= W^{(l)} - \alpha\sum_{i=1}^{N}\left(\frac{\partial J(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)})}{\partial W^{(l)}}\right),$$

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha\frac{\partial J(W,\mathbf{b})}{\partial \mathbf{b}^{(l)}},$$

$$= \mathbf{b}^{(l)} - \alpha\sum_{i=1}^{N}\left(\frac{\partial J(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)})}{\partial \mathbf{b}^{(l)}}\right),$$

重要的问题：求取参数梯度

问题描述

批量学习（Batch Learning）

- N 个样本（一个 batch）
- 随机采样 batch 训练样本集
- Batch-by-Batch 调整权值

优点 ：

梯度向量形式固定，有利于并行处理；

缺点：

需要内存资源大。

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial J(W, \mathbf{b})}{\partial W^{(l)}},$$
$$= W^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial W^{(l)}}\right),$$
$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \frac{\partial J(W, \mathbf{b})}{\partial \mathbf{b}^{(l)}},$$
$$= \mathbf{b}^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}}\right),$$

多层感知机

问题描述

在线学习（Online Learning）

- sample-by-sample 调整权值

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial W^{(l)}}$$

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}}$$

优点 ：

容易执行、存储量小、有效解决大规模和困难模式的分类。

缺点：

学习过程随机、不稳定。

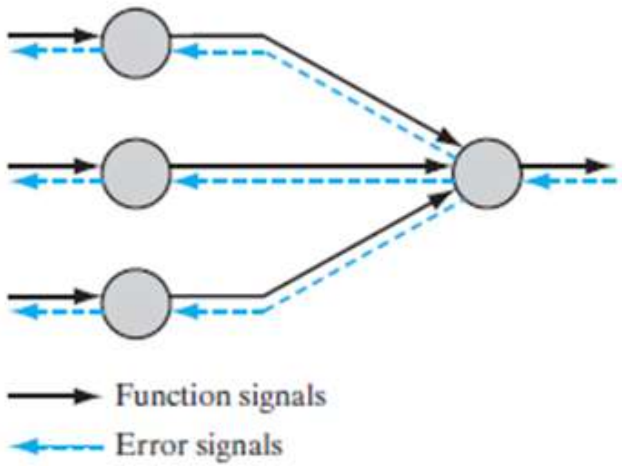
多层感知机

BP 基本思想

两个方向的信号流、两个方向的函数运算

函数信号：计算输出函数信号

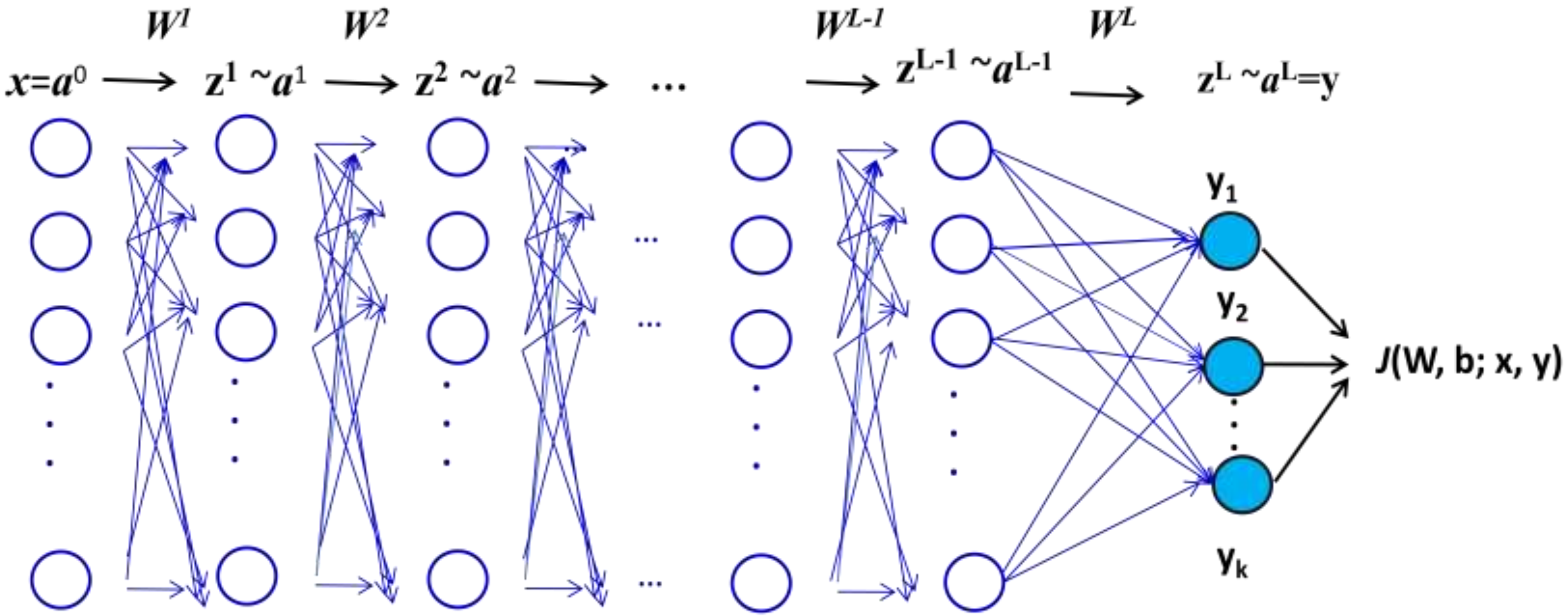
误差信号：计算梯度向量



多层感知机

BP 基本思想

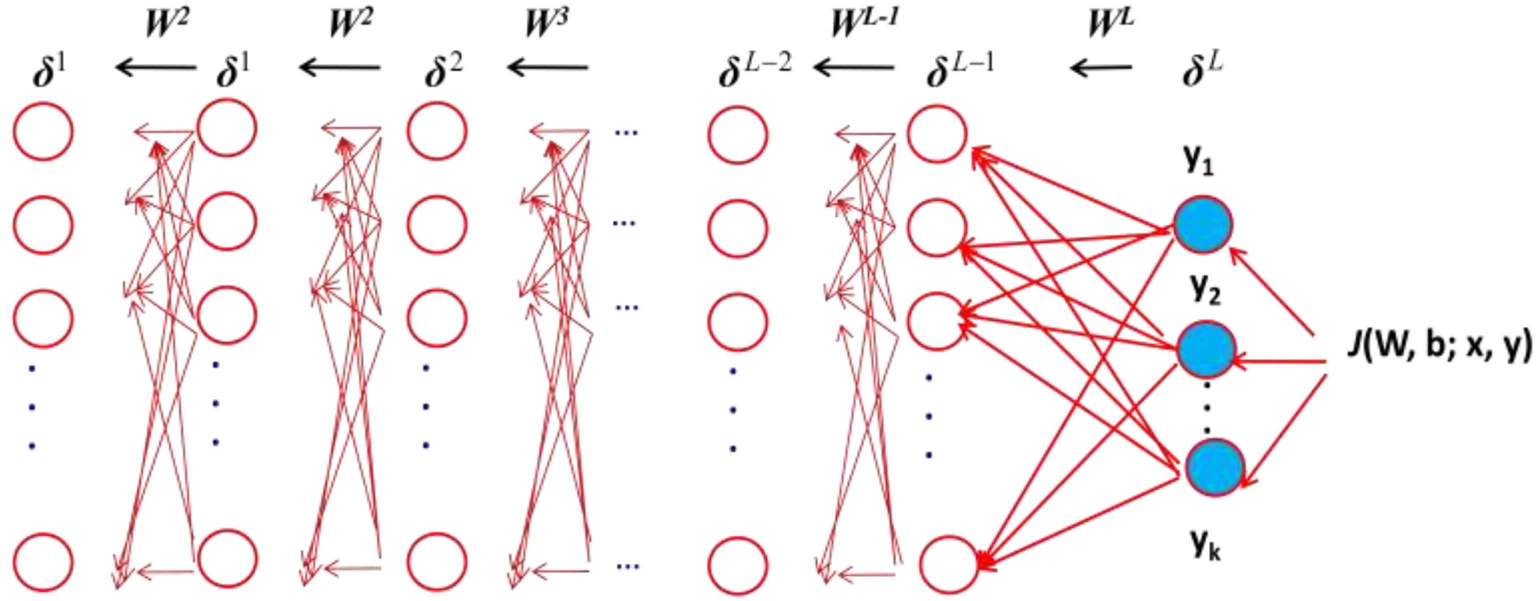
数据前馈运算



多层感知机

BP 基本思想

梯度反馈运算



多层感知机

BP 基本思想

变量关系

$z^1 = g_1(x, W^1)$,

$z^2 = g_2(z^1, W^2)$,

… …

$z^{l-1} = g_{l-1}(z^{l-2}, W^{l-1})$,

$z^{l} = g_{l}(z^{l-1}, W^{l})$,

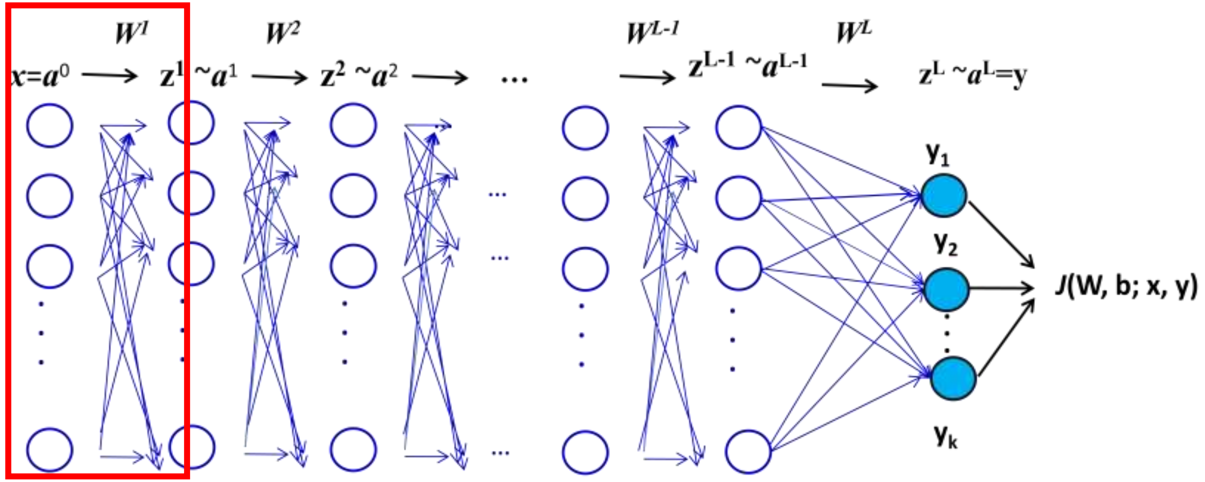
$z^{l+1} = g_{l+1}(z^{l}, W^{l+1})$,

… …

$z^{L} = g_{L}(z^{L-1}, W^{L})$,

$y = f(z^{L})$,

$J(W, y)$

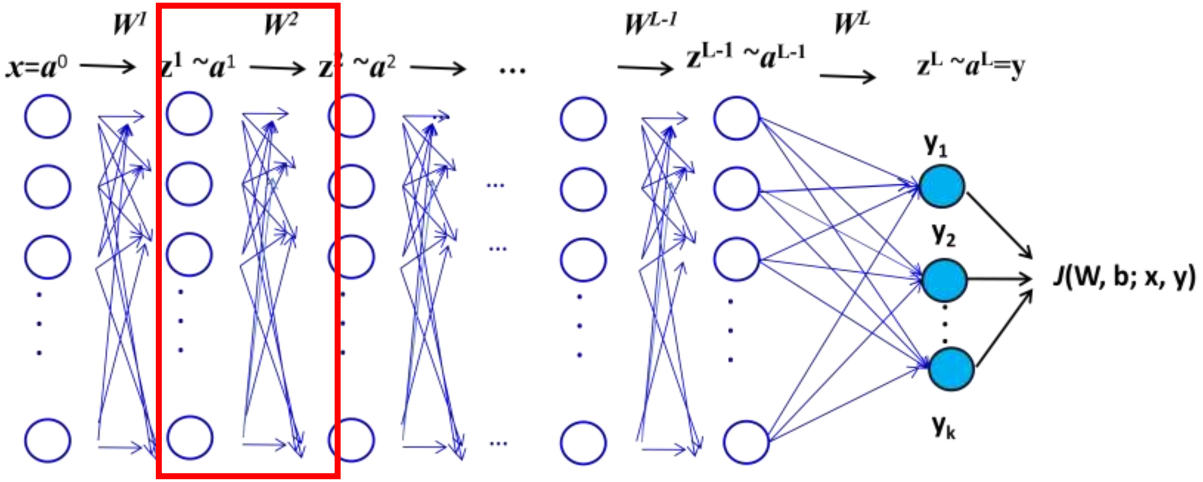


多层感知机

BP 基本思想

变量关系

$\mathbf{z}^1 = g_1(x, \mathbf{W}^1)$,
$\mathbf{z}^2 = g_2(\mathbf{z}^1, \mathbf{W}^2)$,
… …
$\mathbf{z}^{l-1} = g_{l-1}(\mathbf{z}^{l-2}, \mathbf{W}^{l-1})$,
$\mathbf{z}^{l} = g_{l}(\mathbf{z}^{l-1}, \mathbf{W}^{l})$,
$\mathbf{z}^{l+1} = g_{l+1}(\mathbf{z}^{l}, \mathbf{W}^{l+1})$,
… …
$\mathbf{z}^{L} = g_{L}(\mathbf{z}^{L-1}, \mathbf{W}^{L})$,
$\mathbf{y} = f(\mathbf{z}^{L})$,
$J(\mathbf{W}, \mathbf{y})$

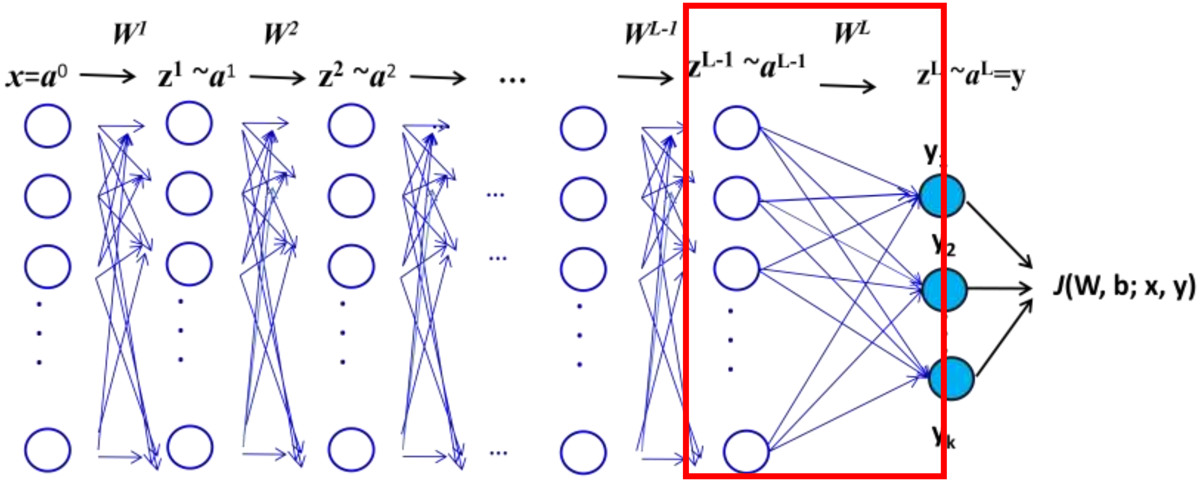


多层感知机

BP 基本思想

变量关系

$\mathbf{z}^1 = g_1(x, \mathbf{W}^1)$,
$\mathbf{z}^2 = g_2(\mathbf{z}^1, \mathbf{W}^2)$,
... ...
$\mathbf{z}^{l-1} = g_{l-1}(\mathbf{z}^{l-2}, \mathbf{W}^{l-1})$,
$\mathbf{z}^{l} = g_{l}(\mathbf{z}^{l-1}, \mathbf{W}^{l})$,
$\mathbf{z}^{l+1} = g_{l+1}(\mathbf{z}^{l}, \mathbf{W}^{l+1})$,
... ...
$\mathbf{z}^{L} = g_L(\mathbf{z}^{L-1}, \mathbf{W}^L)$,
$\mathbf{y} = f_L(\mathbf{z}^L)$,
$J(\mathbf{W}, \mathbf{y})$

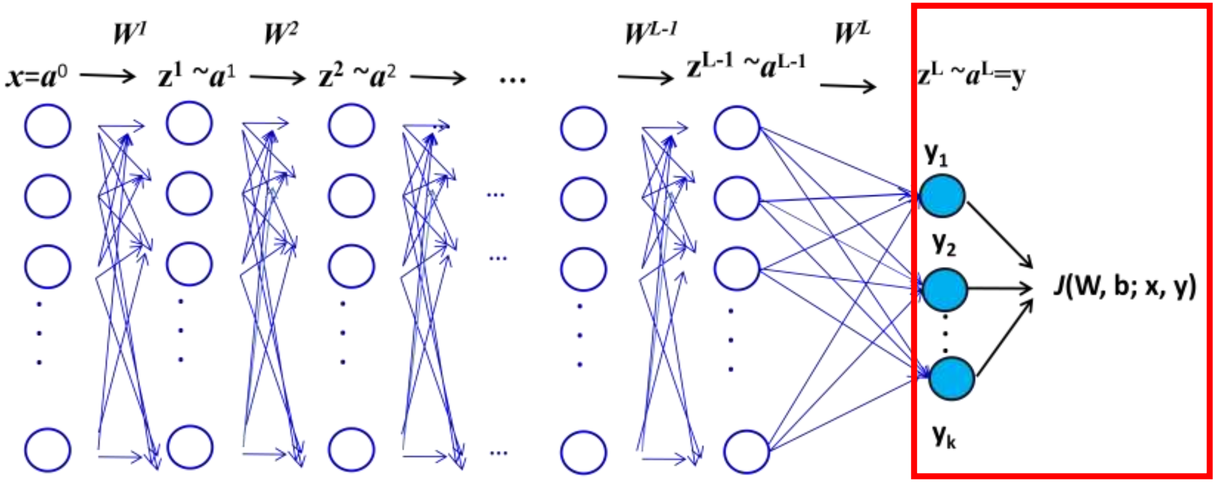


多层感知机

BP 基本思想

变量关系

$\mathbf{z}^1 = g_1(x,\mathbf{W}^1)$,
$\mathbf{z}^2 = g_2(\mathbf{z}^1,\mathbf{W}^2)$,
… …
$\mathbf{z}^{l-1} = g_{l-1}(\mathbf{z}^{l-2},\mathbf{W}^{l-1})$,
$\mathbf{z}^{l} = g_{l}(\mathbf{z}^{l-1},\mathbf{W}^{l})$,
$\mathbf{z}^{l+1} = g_{l+1}(\mathbf{z}^{l},\mathbf{W}^{l+1})$,
… …
$\mathbf{z}^{L} = g_{L}(\mathbf{z}^{L-1},\mathbf{W}^{L})$,
$\mathbf{y} = f_L(\mathbf{z}^L)$,
$J(\mathbf{W},\mathbf{y})$



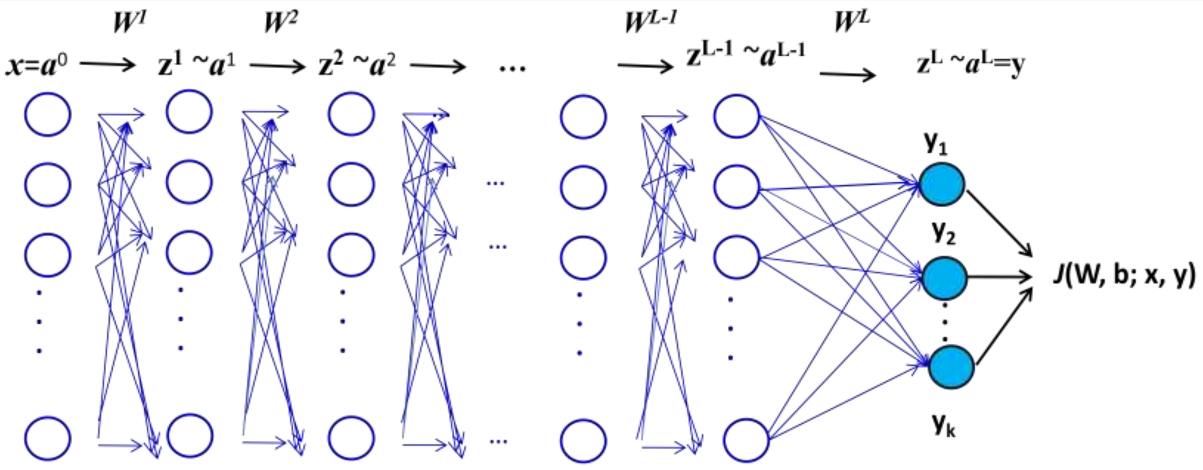
多层感知机

BP 基本思想

变量关系

$\mathbf{z}^1 = g_1(\boldsymbol{x},\mathbf{W}^1)$,
$\mathbf{z}^2 = g_2(\mathbf{z}^1,\mathbf{W}^2)$,
… …
$\mathbf{z}^{l-1} = g_{l-1}(\mathbf{z}^{l-2},\mathbf{W}^{l-1})$,
$\mathbf{z}^{l} = g_{l}(\mathbf{z}^{l-1},\mathbf{W}^{l})$,
$\mathbf{z}^{l+1} = g_{l+1}(\mathbf{z}^{l},\mathbf{W}^{l+1})$,
… …
$\mathbf{z}^{L} = g_L(\mathbf{z}^{L-1},\mathbf{W}^{L})$,
$\mathbf{y} = f(\mathbf{z}^L)$,
$J(\mathbf{W},\mathbf{y})$

$J(\mathbf{W},\mathbf{y})$与 x 的变量依赖：$J(\mathbf{W},\mathbf{y})=J\left(f\left(g_L\left(\ldots g_2(g_1(\boldsymbol{x},\mathbf{W}^1),\mathbf{W}^2),\ldots \mathbf{W}^L\right)\right)\right)$



多层感知机

BP 基本思想

变量关系

$z^1 = g_1(x, W^1)$,

$z^2 = g_2(z^1, W^2)$,

… …

$z^{l-1} = g_{l-1}(z^{l-2}, W^{l-1})$,

$z^{l} = g_{l}(z^{l-1}, W^{l})$,

$z^{l+1} = g_{l+1}(z^{l}, W^{l+1})$,

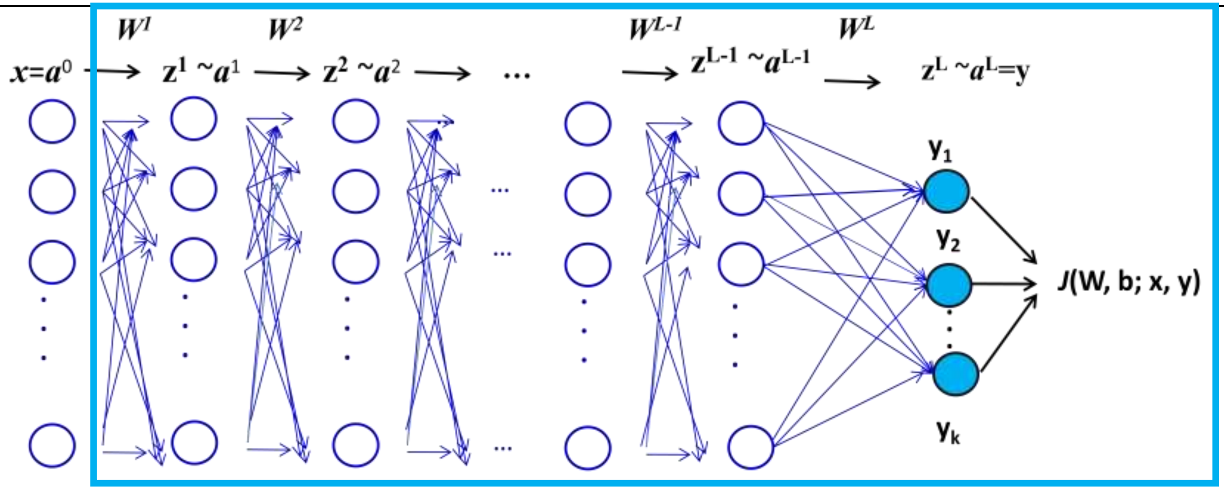
… …

$z^L = g_L(z^{L-1}, W^L)$,

$y = f(z^L)$,

$J(W, y)$

J(W,y)与 x 的变量依赖：$J(W,y) = J(f(g_L(\ldots g_2(g_1(x, W^1), W^2), \ldots W^L)))$



多层感知机

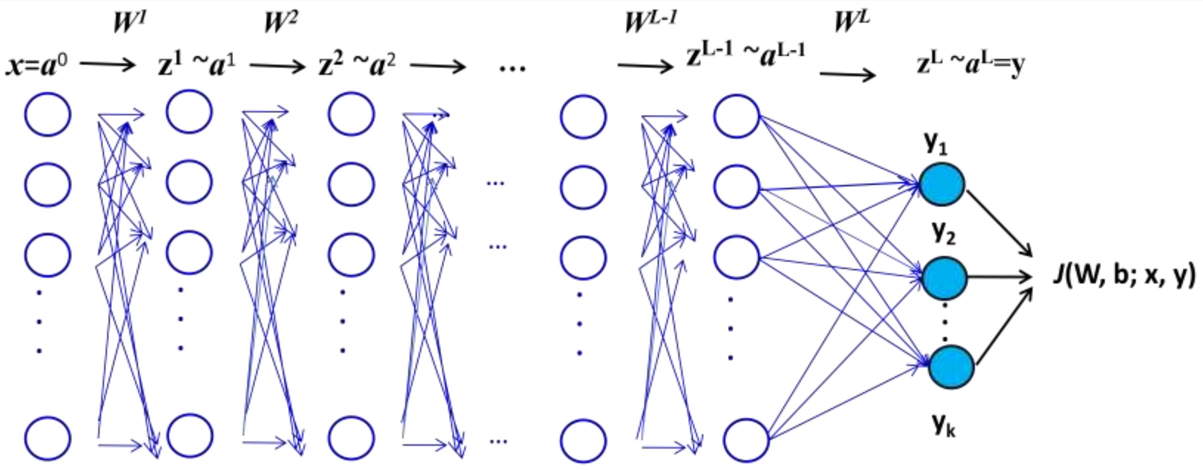
BP 基本思想

变量关系

$\mathbf{z}^1 = g_1(x,\mathbf{W}^1)$,
$\mathbf{z}^2 = g_2(\mathbf{z}^1,\mathbf{W}^2)$,
… …

$\mathbf{z}^l = g_l(\mathbf{z}^{l-1},\mathbf{W}^l)$,
$\mathbf{z}^{l+1} = g_{l+1}(\mathbf{z}^l,\mathbf{W}^{l+1})$,
… …
$\mathbf{z}^L = g_L(\mathbf{z}^{L-1},\mathbf{W}^L)$,
$\mathbf{y} = f(\mathbf{z}^L)$,
$J(\mathbf{W},\mathbf{y})$

$J(\mathbf{W},\mathbf{y})$与 x 的变量依赖：$J(\mathbf{W},\mathbf{y})=J\,(f\,(g_L\,(\ldots g_2(g_1(x,\mathbf{W}^1),\mathbf{W}^2),\ldots\mathbf{W}^L))$
$J(\mathbf{W},\mathbf{y})$与 $\mathbf{z}^1$ 的变量依赖：$J(\mathbf{W},\mathbf{y})=J\,(f\,(g_L\,(\ldots g_2(\mathbf{z}^1,\mathbf{W}^2),\ldots\mathbf{W}^L))$



多层感知机

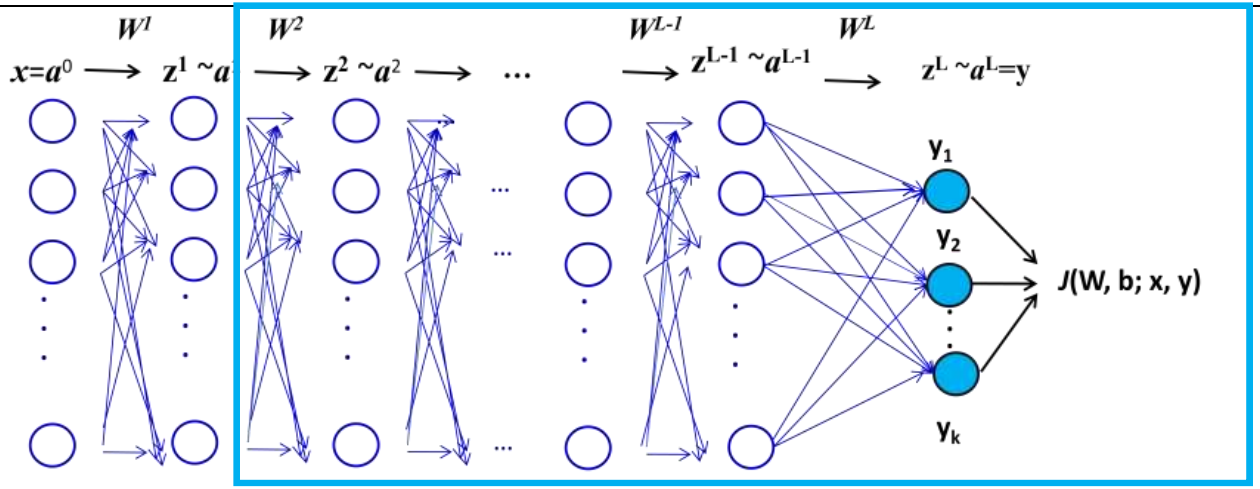
BP 基本思想

变量关系

$z^1 = g_1(x, W^1)$,
$z^2 = g_2(z^1, W^2)$,
… …

$z^l = g_l(z^{l-1}, W^l)$,
$z^{l+1} = g_{l+1}(z^l, W^{l+1})$,
… …
$z^L = g_L(z^{L-1}, W^L)$,
$y = f(z^L)$,
$J(W, y)$

J(W,y)与 x 的变量依赖：$J(W,y)=J(f(g_L(\ldots g_2(g_1(x,W^1),W^2),\ldots W^L))$
J(W,y)与 z^1 的变量依赖：$J(W,y)=J(f(g_L(\ldots g_2(z^1,W^2),\ldots W^L))$



多层感知机

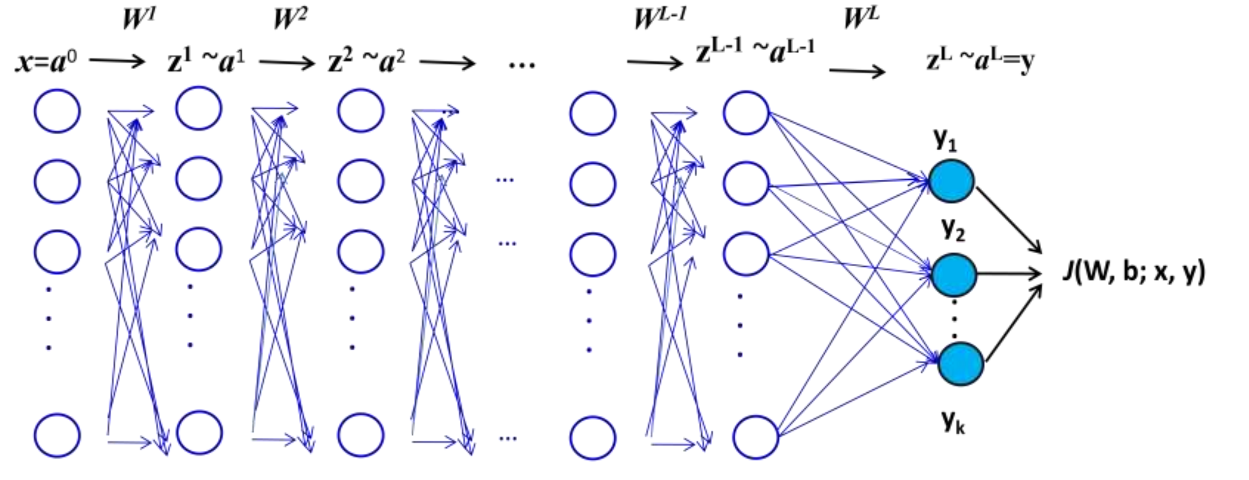
BP 基本思想

变量关系

$z^1 = g_1(x, W^1)$,
$z^2 = g_2(z^1, W^2)$,
… …

$z^{l+1} = g_{l+1}(z^l, W^{l+1})$,
$z^{l+2} = g_{l+2}(z^{l+1}, W^{l+2})$,
… …
$z^L = g_L(z^{L-1}, W^L)$,
$y = f(z^L)$,
$J(W,y)$

J(W,y)与 x 的变量依赖：$J(W,y)=J(f(g_L(\ldots g_2(g_1(x,W^1),W^2),\ldots W^L))$
J(W,y)与 z^1 的变量依赖：$J(W,y)=J(f(g_L(\ldots g_2(z^1,W^2),\ldots W^L))$
J(W,y)与 z^2 的变量依赖：$J(W,y)=J(f(g_L(\ldots g_3(z^2,W^3),\ldots W^L))$
… …
J(W,y)与 z^l 的变量依赖：$J(W,y)=J(f(g_L(\ldots g_{l+1}(z^l,W^{l+1})\ W^{l+2},\ldots W^L))$



多层感知机

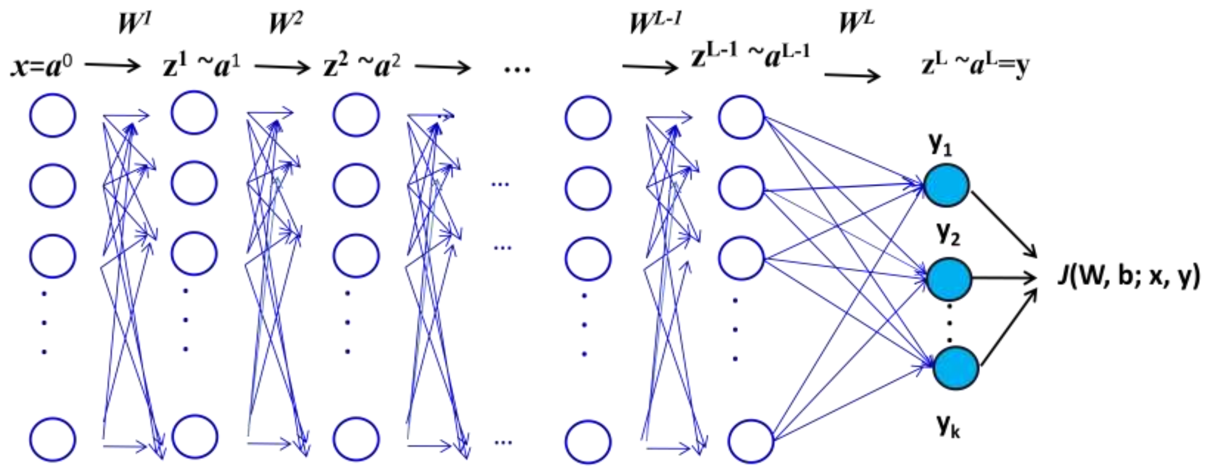
BP 基本思想

变量关系

$\mathbf{z}^1 = g_1(x,\mathbf{W}^1)$,
$\mathbf{z}^2 = g_2(\mathbf{z}^1,\mathbf{W}^2)$,
… …

$\mathbf{z}^{l+1} = g_{l+1}(\mathbf{z}^l,\mathbf{W}^{l+1})$,
$\mathbf{z}^{l+2} = g_{l+2}(\mathbf{z}^{l+1},\mathbf{W}^{l+2})$,
… …
$\mathbf{z}^L = g_L(\mathbf{z}^{L-1},\mathbf{W}^L)$,
$\mathbf{y} = f(\mathbf{z}^L)$,
$J(\mathbf{W},\mathbf{y})$

J(W,y)与 x 的变量依赖：$J(\mathbf{W},\mathbf{y})=J(f(g_L(\ldots g_2(g_1(x,\mathbf{W}^1),\mathbf{W}^2),\ldots\mathbf{W}^L))$
J(W,y)与 $\mathbf{z}^1$ 的变量依赖：$J(\mathbf{W},\mathbf{y})=J(f(g_L(\ldots g_2(\mathbf{z}^1,\mathbf{W}^2),\ldots\mathbf{W}^L))$
J(W,y)与 $\mathbf{z}^2$ 的变量依赖：$J(\mathbf{W},\mathbf{y})=J(f(g_L(\ldots g_3(\mathbf{z}^2,\mathbf{W}^3),\ldots\mathbf{W}^L))$
… …
J(W,y)与 $\mathbf{z}^l$ 的变量依赖：$J(\mathbf{W},\mathbf{y})=J(f(g_L(\ldots g_{l+1}(\mathbf{z}^l,\mathbf{W}^{l+1})\ \mathbf{W}^{l+2},\ldots\mathbf{W}^L))$
J(W,y)与 $\mathbf{z}^{l+1}$ 的变量依赖：$J(\mathbf{W},\mathbf{y})=J(f(g_L(\ldots g_{l+2}(\mathbf{z}^{l+1},\mathbf{W}^{l+2}),\ldots\mathbf{W}^L))$



多层感知机

BP 基本思想

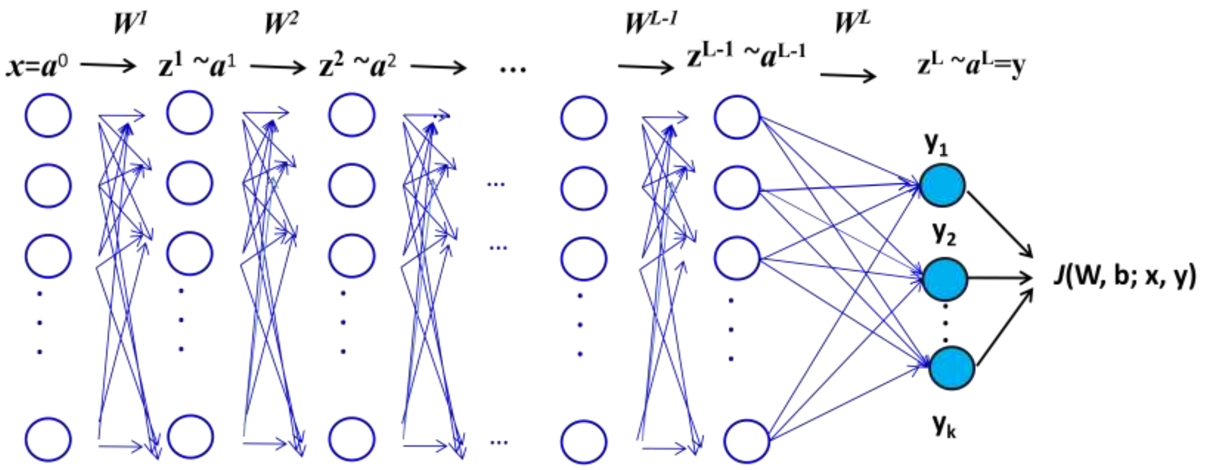
变量关系

$z^1 = g_1(x, W^1)$,
$z^2 = g_2(z^1, W^2)$,
… …

$z^{l+1} = g_{l+1}(z^l, W^{l+1})$, （3）
$z^{l+2} = g_{l+2}(z^{l+1}, W^{l+2})$,
… …
$z^L = g_L(z^{L-1}, W^L)$,
$y = f(z^L)$,
$J(W,y)$

$J(W,y)$与 z^l 的变量依赖（1）
可以分解为（2）（3）

$J(W,y)$与 x 的变量依赖： $J(W,y)=J(f(g_L(\ldots g_2(g_1(x,W^1),W^2),\ldots W^L))$
$J(W,y)$与 z^1 的变量依赖： $J(W,y)=J(f(g_L(\ldots g_2(z^1,W^2),\ldots W^L))$
$J(W,y)$与 z^2 的变量依赖： $J(W,y)=J(f(g_L(\ldots g_3(z^2,W^3),\ldots W^L))$
… …
$J(W,y)$与 z^l 的变量依赖： （1） $J(W,y)=J(f(g_L(\ldots g_{l+1}(z^l,W^{l+1})\, W^{l+2},\ldots W^L))$
$J(W,y)$与 z^{l+1} 的变量依赖：（2） $J(W,y)=J(f(g_L(\ldots g_{l+2}(z^{l+1},W^{l+2}),\ldots W^L))$



多层感知机

BP 基本思想

局部梯度的迭代

$$\boldsymbol{x}=\boldsymbol{a}^{(0)}\rightarrow \mathbf{z}^{(1)}\sim \boldsymbol{a}^{(1)}\rightarrow \mathbf{z}^{(2)}\sim \boldsymbol{a}^{(2)}\rightarrow \cdots \rightarrow \mathbf{z}^{(l)}\sim \boldsymbol{a}^{(l)}\rightarrow \mathbf{z}^{(l+1)}\sim \boldsymbol{a}^{(l+1)}\rightarrow \cdots \mathbf{z}^{(L)}\sim \boldsymbol{a}^{(L)}=\boldsymbol{y}\rightarrow J(\boldsymbol{W},\boldsymbol{b};\boldsymbol{x},\boldsymbol{y})$$

$$\boldsymbol{x}=\boldsymbol{a}^{(0)}\rightarrow \mathbf{z}^{(1)}\sim \boldsymbol{a}^{(1)}\rightarrow \mathbf{z}^{(2)}\sim \boldsymbol{a}^{(2)}\rightarrow \cdots \rightarrow \mathbf{z}^{(l)}\sim \boldsymbol{a}^{(l)}\rightarrow \mathbf{z}^{(l+1)}\sim \boldsymbol{a}^{(l+1)}\rightarrow \cdots \mathbf{z}^{(L)}\sim \boldsymbol{a}^{(L)}=\boldsymbol{y}\rightarrow J(\boldsymbol{W},\boldsymbol{b};\boldsymbol{x},\boldsymbol{y})$$

$$\boldsymbol{x}=\boldsymbol{a}^{(0)}\rightarrow \mathbf{z}^{(1)}\sim \boldsymbol{a}^{(1)}\rightarrow \mathbf{z}^{(2)}\sim \boldsymbol{a}^{(2)}\rightarrow \cdots \rightarrow \mathbf{z}^{(l)}\sim \boldsymbol{a}^{(l)}\rightarrow \mathbf{z}^{(l+1)}\sim \boldsymbol{a}^{(l+1)}\rightarrow \cdots \mathbf{z}^{(L)}\sim \boldsymbol{a}^{(L)}=\boldsymbol{y}\rightarrow J(\boldsymbol{W},\boldsymbol{b};\boldsymbol{x},\boldsymbol{y})$$

多层感知机

BP 基本思想

局部梯度的迭代

$$x=a^{(0)}\to \mathbf{z}^{(1)}\sim a^{(1)}\to \mathbf{z}^{(2)}\sim a^{(2)}\to\cdots\to \mathbf{z}^{(l)}\sim a^{(l)}\to \mathbf{z}^{(l+1)}\sim a^{(l+1)}\to\cdots \mathbf{z}^{(L)}\sim a^{(L)}=y\to J(W,b;x,y)$$

$$x=a^{(0)}\to \mathbf{z}^{(1)}\sim a^{(1)}\to \mathbf{z}^{(2)}\sim a^{(2)}\to\cdots\to \mathbf{z}^{(l)}\sim a^{(l)}\to \mathbf{z}^{(l+1)}\sim a^{(l+1)}\to\cdots \mathbf{z}^{(L)}\sim a^{(L)}=y\to J(W,b;x,y)$$

$$x=a^{(0)}\to \mathbf{z}^{(1)}\sim a^{(1)}\to \mathbf{z}^{(2)}\sim a^{(2)}\to\cdots\to \mathbf{z}^{(l)}\sim a^{(l)}\to \mathbf{z}^{(l+1)}\sim a^{(l+1)}\to\cdots \mathbf{z}^{(L)}\sim a^{(L)}=y\to J(W,b;x,y)$$

$$\boldsymbol{\delta}^{(l)}=\frac{\partial J(\boldsymbol{W},\boldsymbol{b};\boldsymbol{x},\boldsymbol{y})}{\partial \mathbf{z}^{(l)}}$$

$$=\frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{z}^{(l)}}\cdot\frac{\partial J(\boldsymbol{W},\boldsymbol{b};\boldsymbol{x},\boldsymbol{y})}{\partial \mathbf{z}^{(l+1)}}$$

$$=\frac{\partial \boldsymbol{a}^{(l)}}{\partial \mathbf{z}^{(l)}}\cdot\frac{\partial \mathbf{z}^{(l+1)}}{\partial \boldsymbol{a}^{(l)}}\cdot\frac{\partial J(\boldsymbol{W},\boldsymbol{b};\boldsymbol{x},\boldsymbol{y})}{\partial \mathbf{z}^{(l+1)}}$$

$$=\frac{\partial \boldsymbol{a}^{(l)}}{\partial \mathbf{z}^{(l)}}\cdot\frac{\partial \mathbf{z}^{(l+1)}}{\partial \boldsymbol{a}^{(l)}}\cdot\boldsymbol{\delta}^{(l+1)}$$

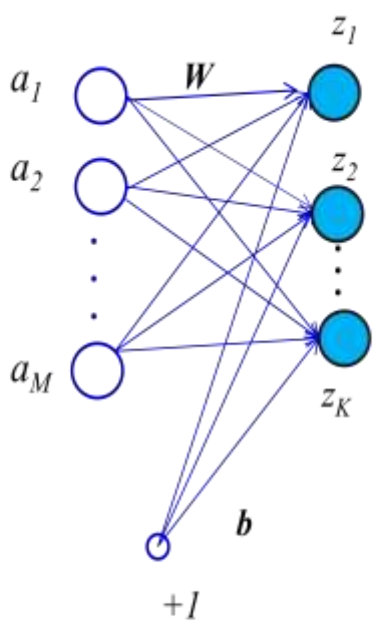
$$\delta^{(l)}\triangleq\frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l)}}$$

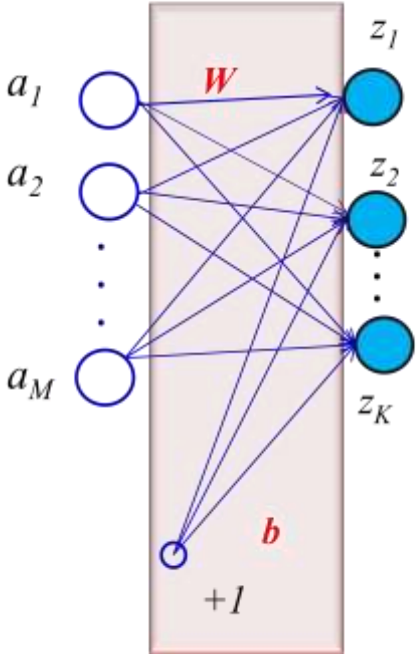
$$=\frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}}\cdot\frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}}\cdot\frac{\partial J(W,\mathbf{b};\mathbf{x},y)}{\partial \mathbf{z}^{(l+1)}}$$

附：BP 算法矩阵形式

“多元--向量” 的求导

$$\boldsymbol{z}=\boldsymbol{W}\boldsymbol{a}+\boldsymbol{b}, \quad \boldsymbol{W}\in R^{M\times K},\ \boldsymbol{b}\in R^{K},\ \boldsymbol{a}\in R^{M},\ \boldsymbol{z}\in R^{K}$$





$$\Rightarrow \boxed{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{a}}=\left[\quad ? \quad\right]_{K\times M}=\boldsymbol{W}^{T}}$$

多层感知机

附：BP 算法矩阵形式

激活函数”的求导

特别的向量函数： $\boldsymbol{a}=\boldsymbol{f}(\boldsymbol{z})$， 其中 $f_1(.)=f_2(.)=...=f_K(.)=f(.)$，并且 $a_i=f(z_i)$。（$\boldsymbol{a}\in R^M$，$\boldsymbol{z}\in R^K$ ）

变量依赖

$$a_1 \xrightarrow{f} f(z_1)$$
$$a_2 \xrightarrow{f} f(z_2)$$
$$\vdots$$
$$a_K \xrightarrow{f} f(z_K)$$

$$\Rightarrow \quad \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z}} = \begin{bmatrix} & ? & \end{bmatrix}_{K\times K}$$

梯度关系 $$\frac{\partial a_j}{\partial z_i} = \begin{cases} \frac{\partial a_i}{\partial z_i}, & i=j \\ 0, & i\neq j \end{cases}$$

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z}} = \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial a_K}{\partial z_K} \end{bmatrix}_{K\times K} = \begin{bmatrix} f'(z_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f'(z_K) \end{bmatrix}_{K\times K} = diag(f'(\boldsymbol{z}))$$

BP 基本思想

BP 算法矩阵迭代公式

$$
\begin{aligned}
\delta^{(l)} &\triangleq \frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l)}} \\
&= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l+1)}} \\
&= \mathrm{diag}(f_l'(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^\top \cdot \delta^{(l+1)} \\
&= f_l'(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^\top \delta^{(l+1)}),
\end{aligned}
$$

多层感知机

BP 基本思想

如果能够求取 z^l 的局部梯度，就可求 W^l 和 b^l 的梯度

变量关系

$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

因此参数的更新公式（矩阵形式）：

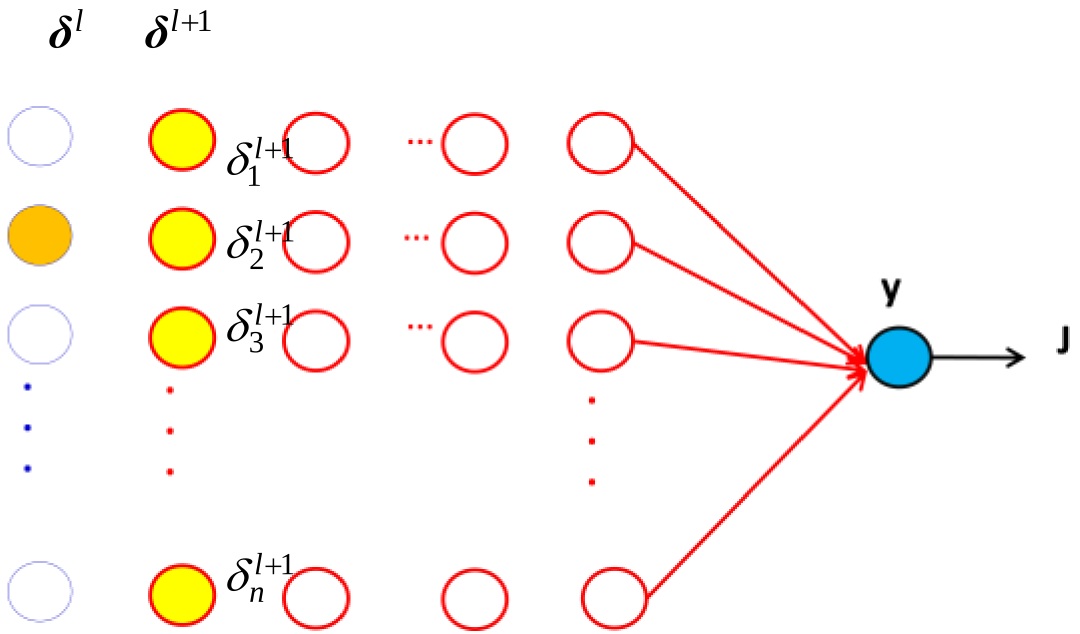
$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^{\top}.$$

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}.$$

多层感知机

BP 算法

问题： 假设已知 l+1 层局部梯度，求 l 层局部梯度



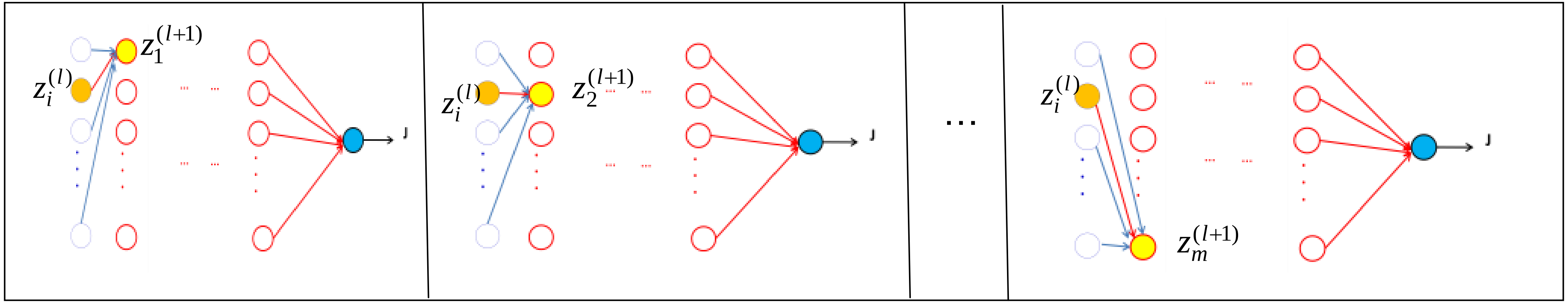
多层感知机

BP 算法

变量依赖关系：

- J 与 $\mathbf{z}^{l+1}$ 的变量依赖： $J = J(f(g_L(\ldots g_{l+2}(\mathbf{z}^{l+1}, \mathbf{W}^{l+2}), \ldots \mathbf{W}^L))$
- $\mathbf{z}^{l+1}$ 与 $\mathbf{z}^{l}$ 的变量依赖： $\mathbf{z}^{l+1} = g_{l+1}(\mathbf{z}^{l}, \mathbf{W}^{l+1})$

即 $z_j^{(l+1)} = \sum_i a_i^{(l)} w_{ij}^{(l+1)} = \sum_i f_i(z_i^{(l)}) w_{ij}^{(l+1)}$，如图



多层感知机

BP 算法

变量依赖关系：

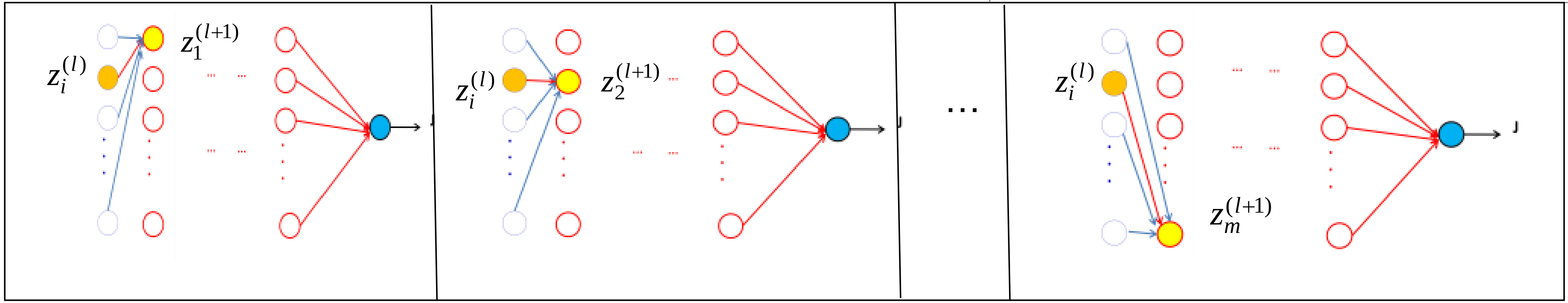
- J 与 $\mathbf{z}^{l+1}$ 的变量依赖： J=J (f (g_L (...g_{l+2}($\mathbf{z}^{l+1}$, $\mathbf{W}^{l+2}$),...$\mathbf{W}^L$)) ,
- $\mathbf{z}^{l+1}$ 与 $\mathbf{z}^{l}$ 的变量依赖： $\mathbf{z}^{l+1}= g_{l+1}(\mathbf{z}^{l}, \mathbf{W}^{l+1})$,

  即 $z_j^{(l+1)} = \sum_i a_i^{(l)} w_{ij}^{(l+1)} = \sum_i f_i(z_i^{(l)}) w_{ij}^{(l+1)}$， 如图

梯度传递关系：

⇨ l+1 层局部梯度是已知：$\delta_j^{l+1} = \dfrac{\partial L}{\partial z_j^{l+1}}$

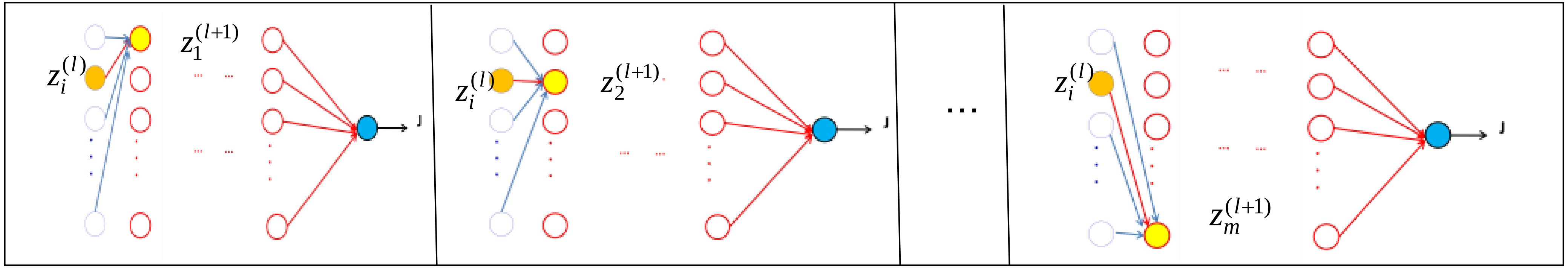
⇨ $$\frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}} = \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} \cdot \frac{\partial z_j^{(l+1)}}{\partial a_i^{(l)}} = f_i'(z_i^{(l)}) w_{ij}^{(l+1)}$$



BP 算法

局部梯度迭代公式：

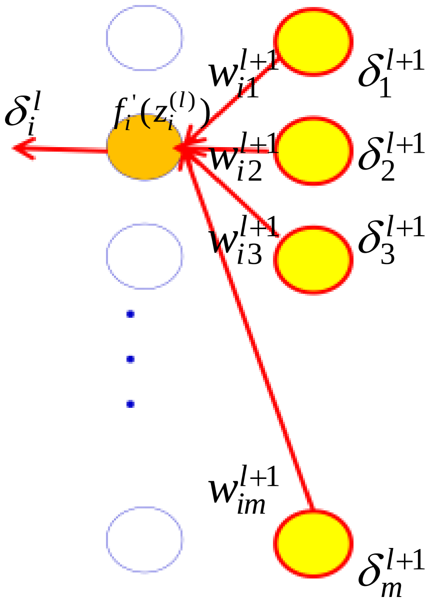
$$\delta_i^{(l)} = \frac{\partial L}{\partial z_i^l} = \boxed{\sum_{j=1}^{m} \frac{\partial z_j^{l+1}}{\partial z_i^l} \frac{\partial L}{\partial z_j^{l+1}} = \sum_{j=1}^{m} \frac{\partial z_j^{l+1}}{\partial z_i^l} \delta_j^{(l+1)}} = \sum_{j=1}^{m} f_i^{'}(z_i^{(l)}) w_{ij}^{(l+1)} \delta_j^{(l+1)} = f_i^{'}(z_i^{(l)}) \sum_{j=1}^{m} w_{ij}^{(l+1)} \delta_j^{(l+1)}$$



多层感知机

BP 算法

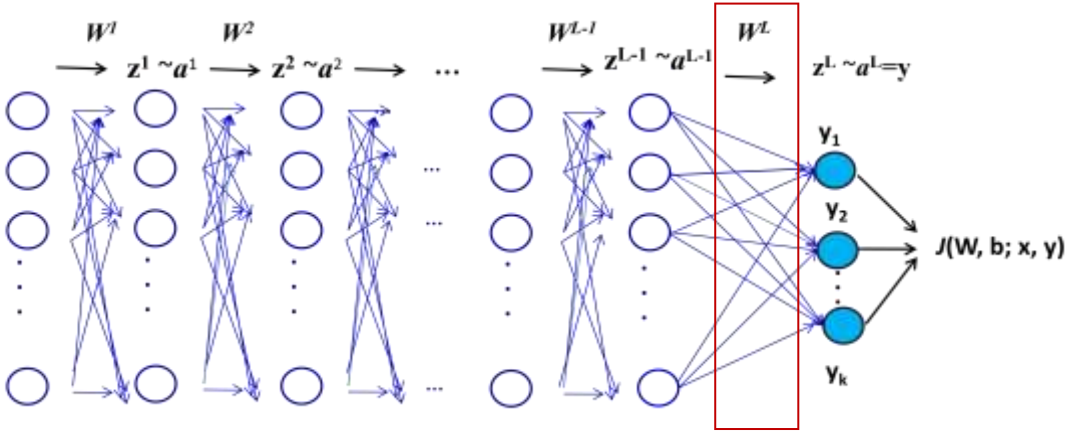
局部梯度沿着网络，反向计算

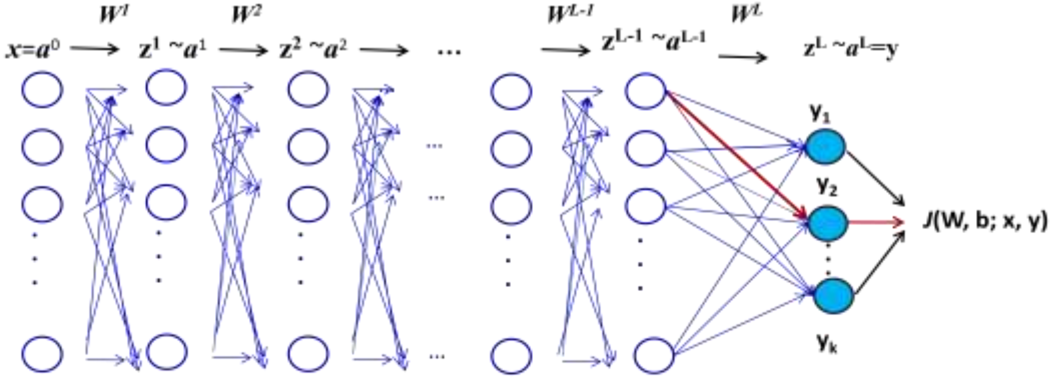


$$\delta_i^{(l)} = f_i'(z_i^{(l)}) \sum_{j=1}^{m} w_{ij}^{(l+1)} \delta_j^{(l+1)}$$

BP 算法

输出层参数的学习





（1） 局部梯度

$$\delta_k{}^L = \frac{\partial J}{\partial z_k^L} = \frac{\partial a_k^L}{\partial z_k^L}\frac{\partial J}{\partial a_k}$$

$$= f_k^{'}(z_k^L)\frac{\partial J}{\partial a_k}$$

（2） 参数梯度

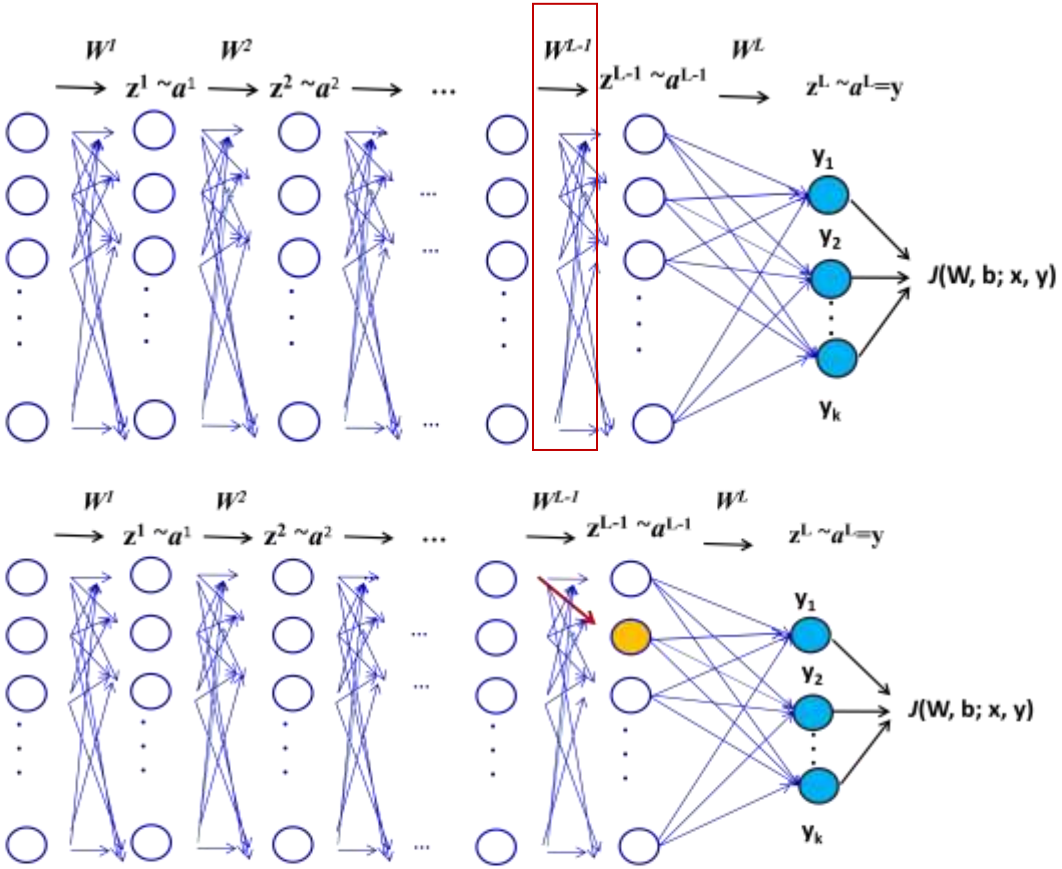
$$\frac{\partial J}{\partial w_{ik}^L} = \frac{\partial z_k^L}{\partial w_{ik}^L}\frac{\partial J}{\partial z_k^L} = \frac{\partial z_k^L}{\partial w_{ik}^L}\delta_k{}^L = a_i^{L-1}\delta_k{}^L$$

$$\frac{\partial J}{\partial b_k^L} = \frac{\partial z_k^L}{\partial b_k^L}\frac{\partial J}{\partial z_k^L} = \delta_k{}^L$$

多层感知机

BP 算法

隐层参数的学习



（3） 局部梯度反向传递

$$\delta_i^{(l)} = f_i^{'}(z_i^{(l)})\sum_{j=1}^{m} w_{ij}^{(l+1)}\delta_j^{(l+1)}$$

（4） 参数梯度

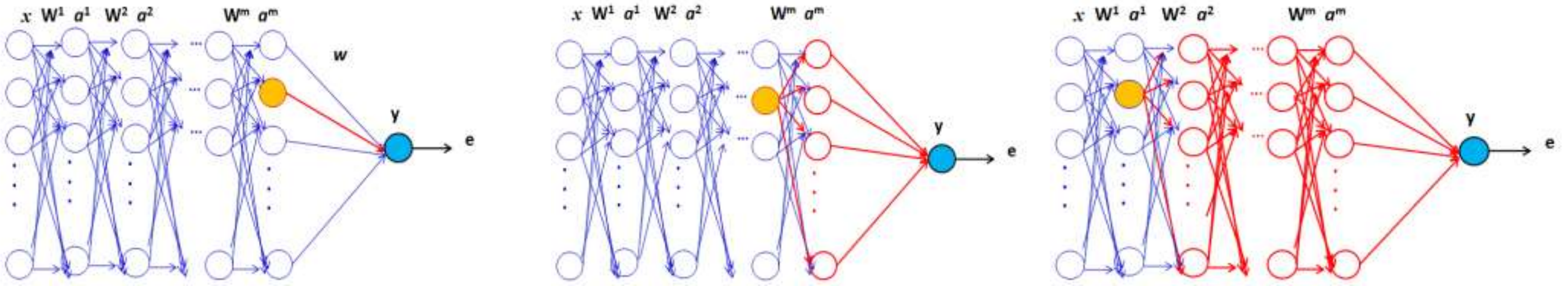
$$\frac{\partial J}{\partial w_{ik}^{l}} = \frac{\partial z_k^L}{\partial w_{ik}^{l}}\frac{\partial J}{\partial z_k^{l}} = a_i^{l-1}\delta_k^l$$

$$\frac{\partial J}{\partial b_k^{l}} = \frac{\partial z_k^L}{\partial b_k^{l}}\frac{\partial J}{\partial z_k^{l}} = \delta_k^l$$

多层感知机

BP 算法

梯度迭代



可见，反向越深的隐层与目标函数之间的变量依赖关系越复杂。

BP 算法通过梯度迭代的策略，解决了这一问题。

BP 算法小结

流程概要

(1) 数据初始化

(2) Epoch 采样

(3) 前向计算

(4) 反向梯度计算

输出层： $\delta_k{}^L = f_k^{'}(z_k^L)\frac{\partial J}{\partial y_k}$ ； 隐藏层： $\delta_i^{(l)} = f_i^{'}(z_i^{(l)})\sum_{j=1}^{m} w_{ij}^{(l+1)}\delta_j^{(l+1)}$

(5) 求参数梯度： $\frac{\partial J}{\partial w_{ik}^l} = a_i^{l-1}\delta_k^l$, $\frac{\partial J}{\partial b_k^l} = \frac{\partial z_k^L}{\partial b_k^l}\frac{\partial J}{\partial z_k^l} = \delta_k^l$

(6) 迭代(2)-(5).

多层感知机

激活函数

- Logistic Function

$$y_j(n) = \varphi_j(v_j(n)) = \frac{1}{1 + \exp(-av_j(n))}, \qquad a > 0$$

$$\varphi_j'(v_j(n)) = \frac{a\exp(-av_j(n))}{[1 + \exp(-av_j(n))]^2}$$

$$\varphi_j'(v_j(n)) = ay_j(n)[1 - y_j(n)]$$

多层感知机

激活函数

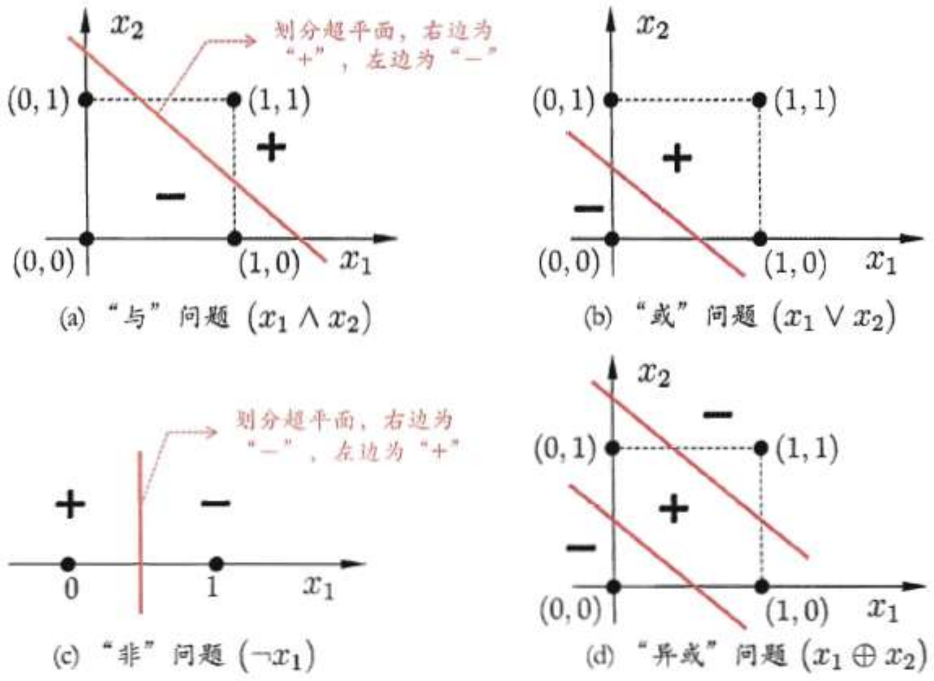
- Hyperbolic tangent function

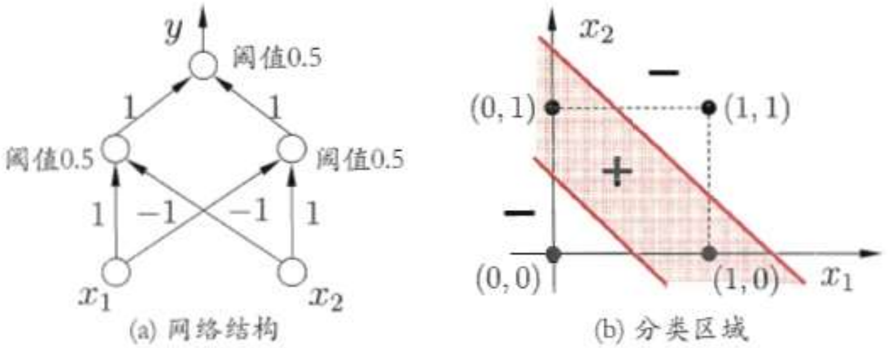
$$y_j(n) = \varphi_j(v_j(n)) = a \tanh(bv_j(n))$$

$$\begin{aligned}\varphi_j'(v_j(n)) &= ab \operatorname{sech}^2(bv_j(n)) \\ &= ab(1 - \tanh^2(bv_j(n))) \\ &= \frac{b}{a}[a - y_j(n)][a + y_j(n)]\end{aligned}$$

异或问题

$$0 \oplus 0 = 0 \quad 1 \oplus 1 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 0 = 1$$





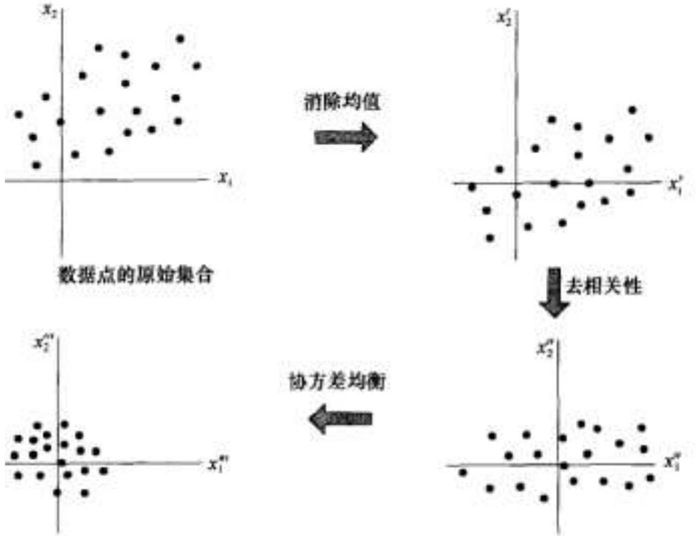
多层感知机

改善性能的试探法

标准化输入

每一个输入变量都需要预处理，使得它关于整个训练集求平均的均值接近 0，或者与标准偏差相比是比较小的（LeCun，1993）。

依次执行三个标准化步骤的结果：消除均值、去相关性以及协方差均衡。



4.11 二维输入空间的消除均值、去相关性以及协方差均衡运算的图示

多层感知机

函数逼近

通用逼近定理

令 $\varphi(\cdot)$是一个非常数的、有界的和单调增的连续函数。令 I_{m_0} 表示 m_0 维单位超立方体 $[0,1]^{m_0}$。I_{m_0} 上连续函数空间用 $C(I_{m_0})$表示。那么，给定任何函数 $f \ni C(I_{m_0})$和 $\varepsilon>0$，存在这样的一个整数 m_1 和实常数 α_i，b_i 和 w_{ij}，其中 $i=1,\cdots,m_1, j=1,\cdots,m_0$，使我们可以定义

$$F(x_1,\cdots,x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i\right) \qquad (4.88)$$

作为 $f(\cdot)$函数的一个近似实现；也就是说，

$$|F(x_1,\cdots,x_{m_0}) - f(x_1,\cdots,x_{m_0})| < \varepsilon$$

对存在于输入空间中的所有 $x_1,x_2,\cdots,x_{m_0}$ 均成立。

第十一章 神经网络与深度学习

14.1 概述

14.2 多层感知机

14.3 卷积网络

14.4 Recurrent 网络

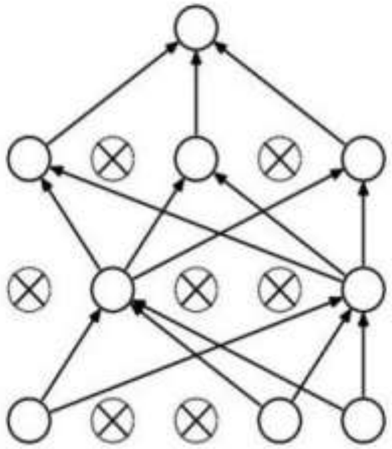
14.5 深度学习

卷积网络

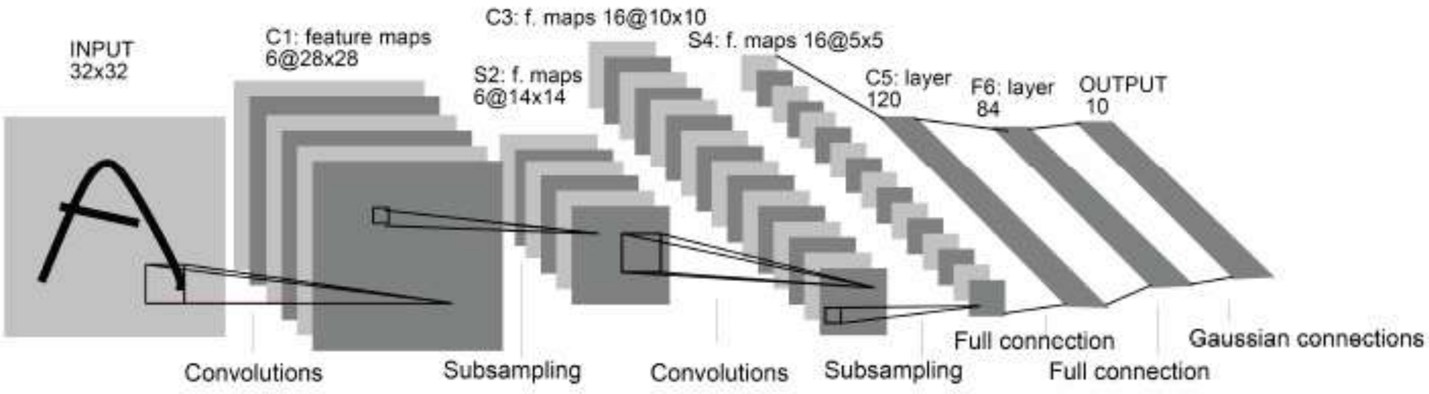
网络结构

多层感知机如何约简网络？

- Dropout



- CNN (Shared weight)



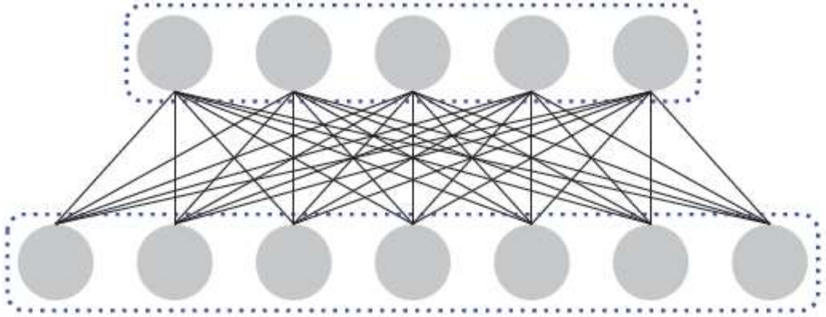
LeNet-5 （LeCun,1989）

卷积网络

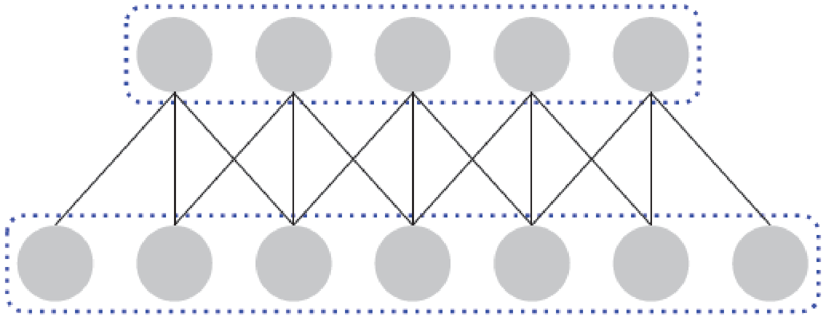
卷积层

卷积层具有局部连接和权重共享特点。

一维情况为例



全连接

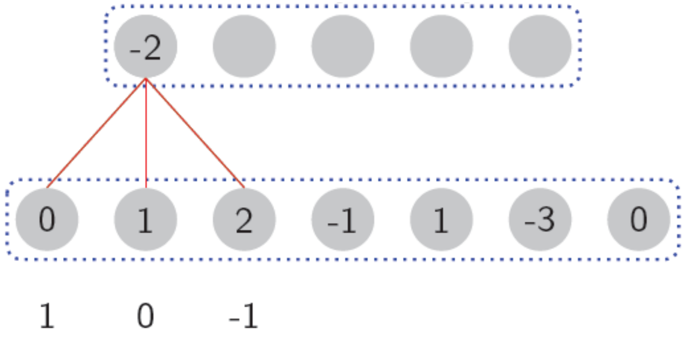


卷积层

卷积网络

卷积层

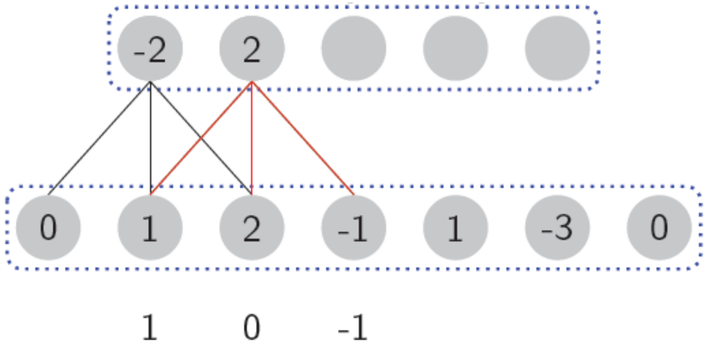
一维卷积



卷积网络

卷积层

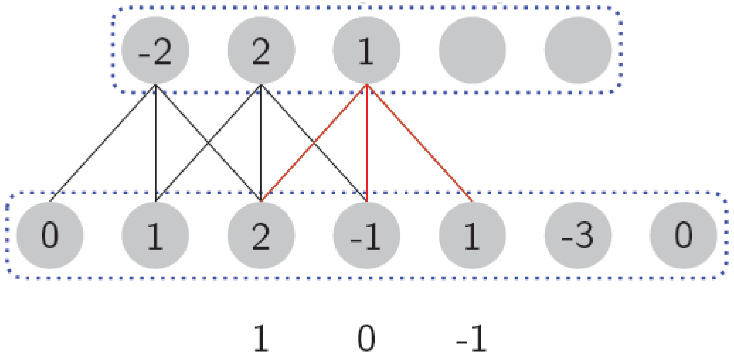
一维卷积



卷积网络

卷积层

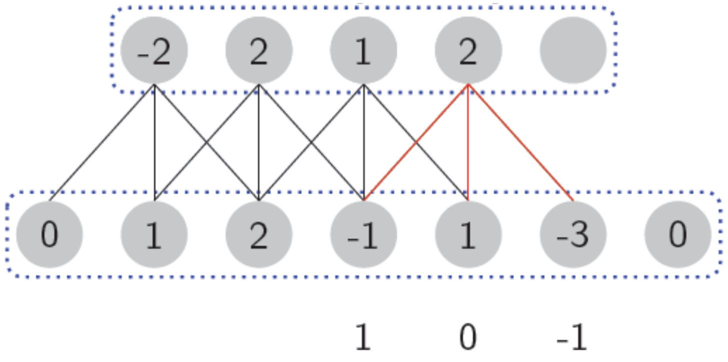
一维卷积



卷积网络

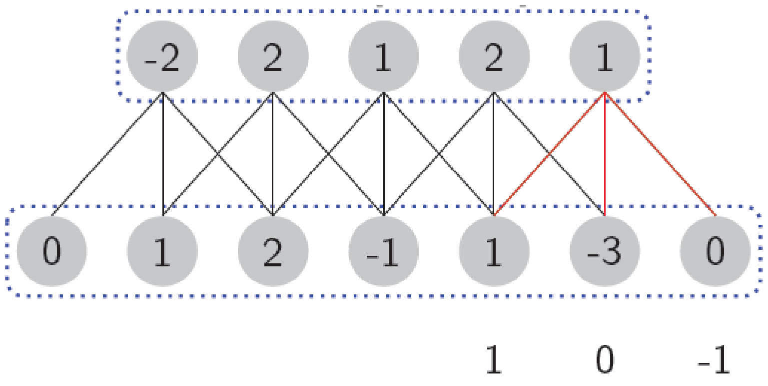
卷积层

一维卷积



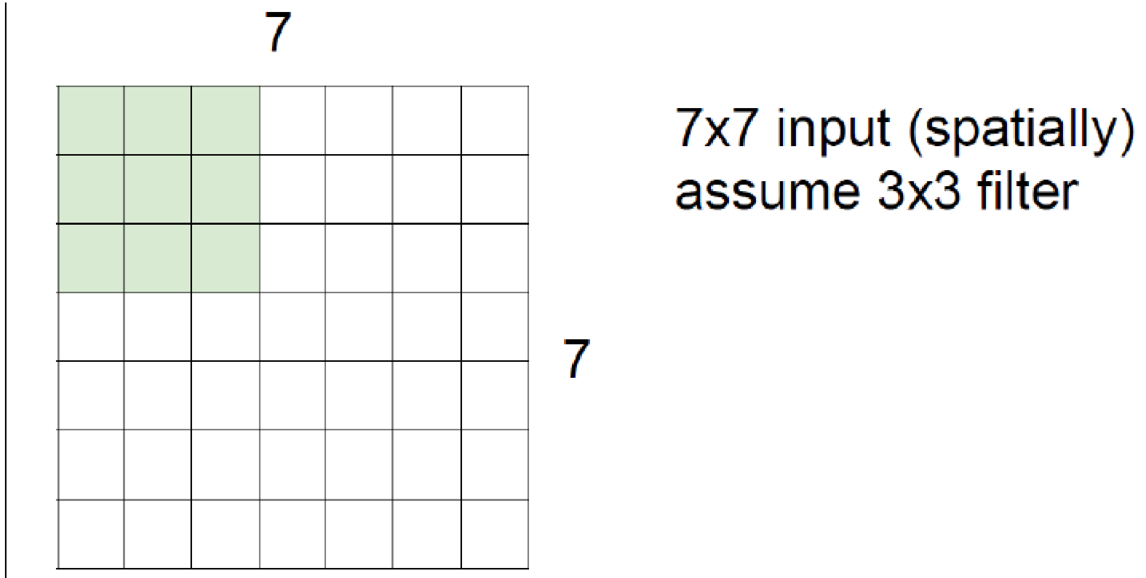
卷积层

一维卷积



卷积层

二维卷积

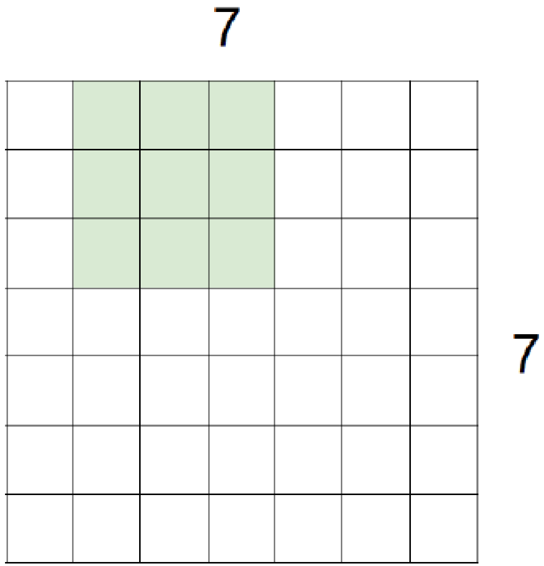


[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

卷积网络

卷积层

二维卷积

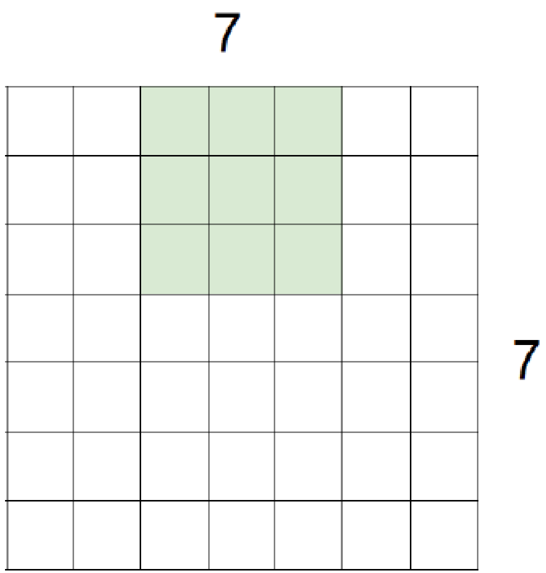


7x7 input (spatially)
assume 3x3 filter

[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

卷积层

二维卷积



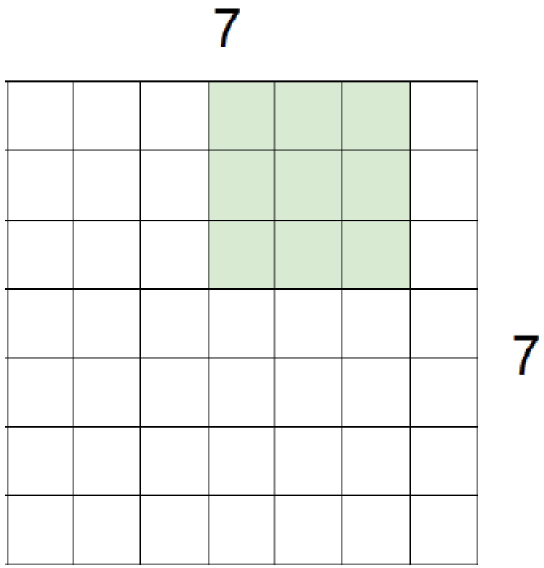
7x7 input (spatially)
assume 3x3 filter

[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

卷积网络

卷积层

二维卷积



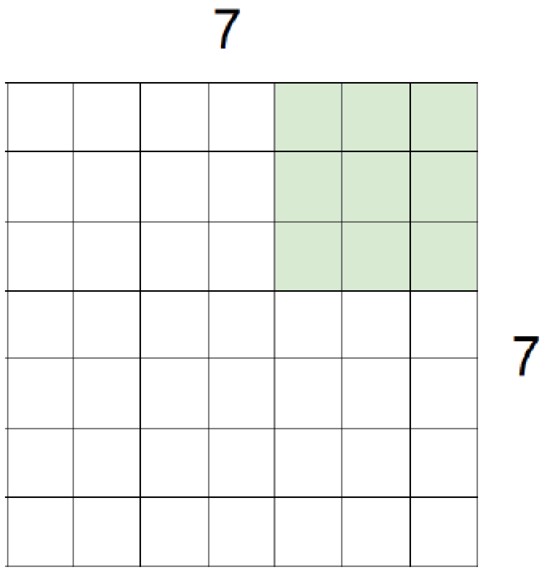
7x7 input (spatially)
assume 3x3 filter

[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

卷积网络

卷积层

二维卷积



7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Output size:

(N - F) / stride + 1

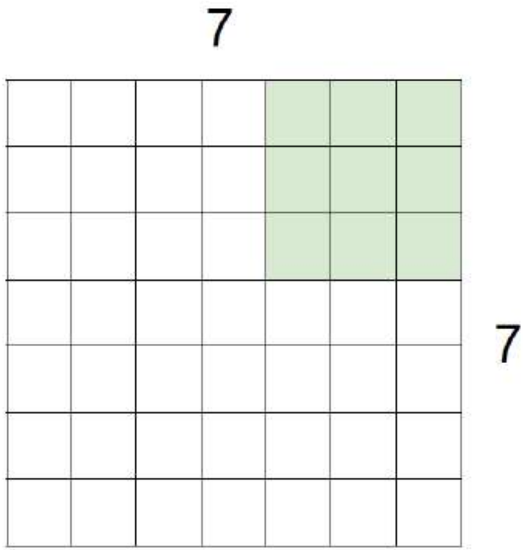
[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

卷积网络

卷积层

卷积层的输出尺度

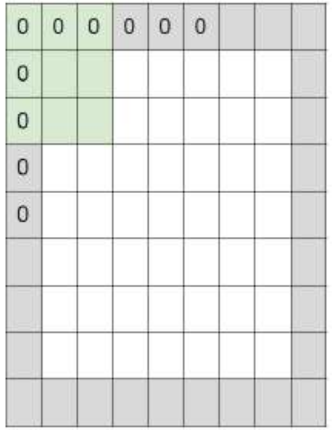
（参数：Filter、Stride、Pad）



7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Output size:
(N - F) / stride + 1



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Output size:
(N+2P - F) / stride + 1

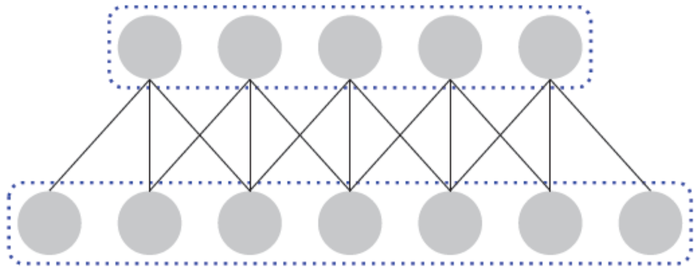
[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

卷积网络

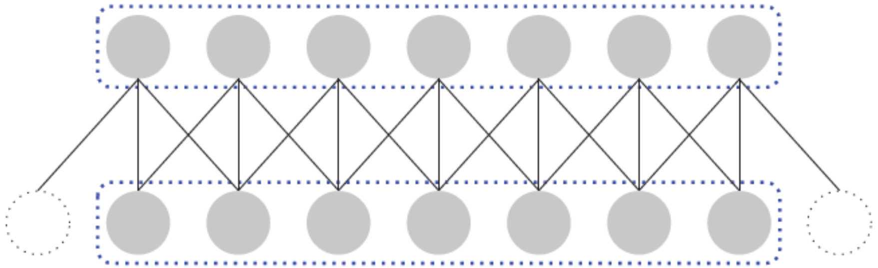
卷积层

卷积层的输出尺度

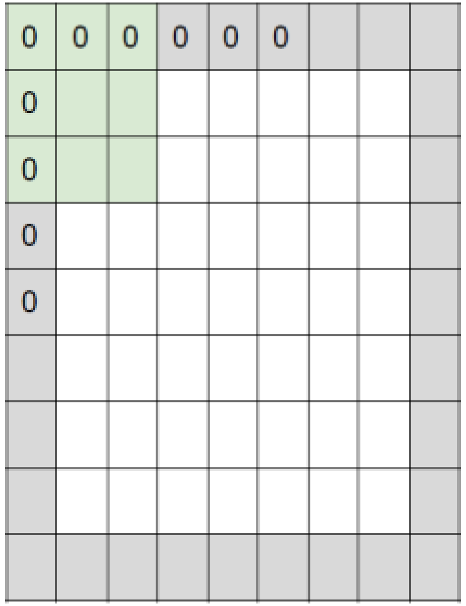
- **Pad： 填充设置**



(a) 窄卷积 Padding=0



(b) 等长卷积 Padding=1

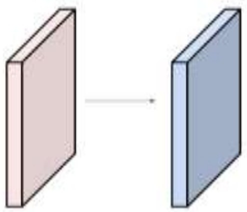


卷积层

卷积层的输出尺度

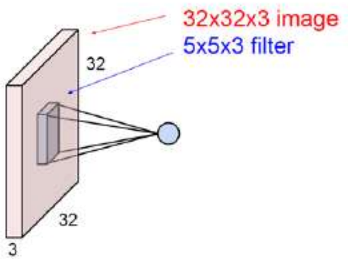
Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2



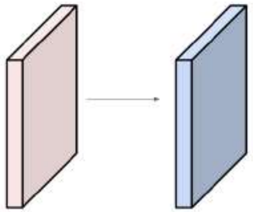
Output volume size: ?

[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.



Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2



Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

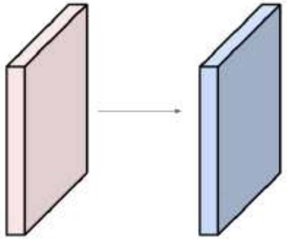
卷积层

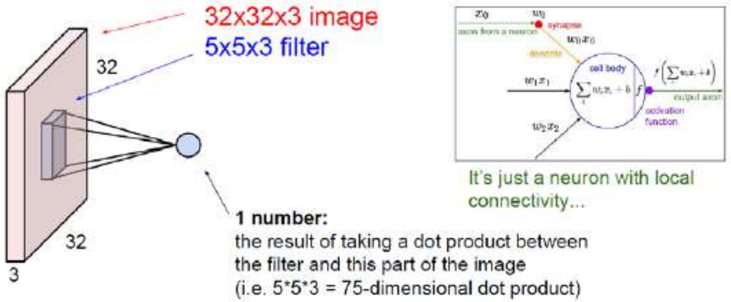
卷积层的参数个数

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

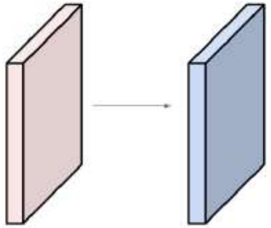
Number of parameters in this layer?





Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

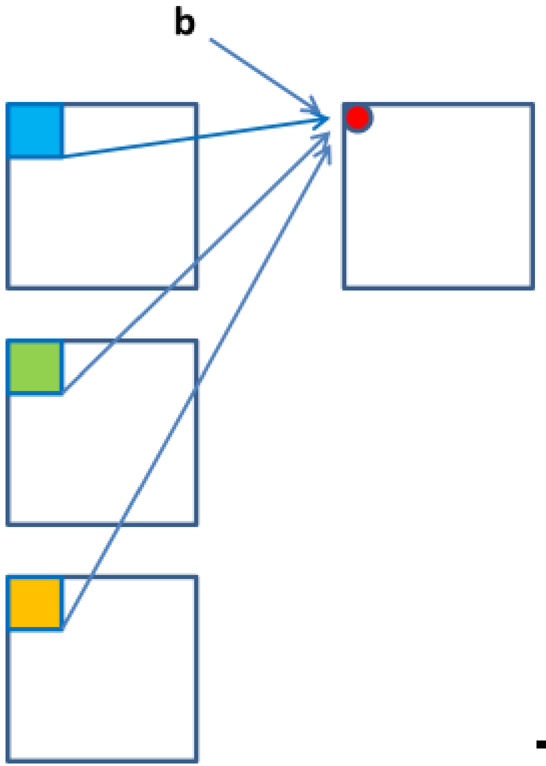


Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

卷积网络

卷积层

1 个 filter 的卷积过程

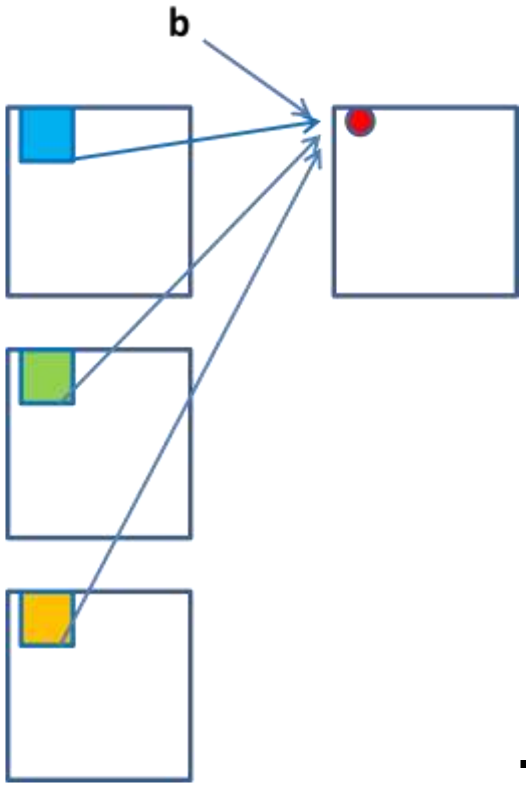


一个 filter 参数： F×F×3+1

卷积网络

卷积层

1 个 filter 的卷积过程

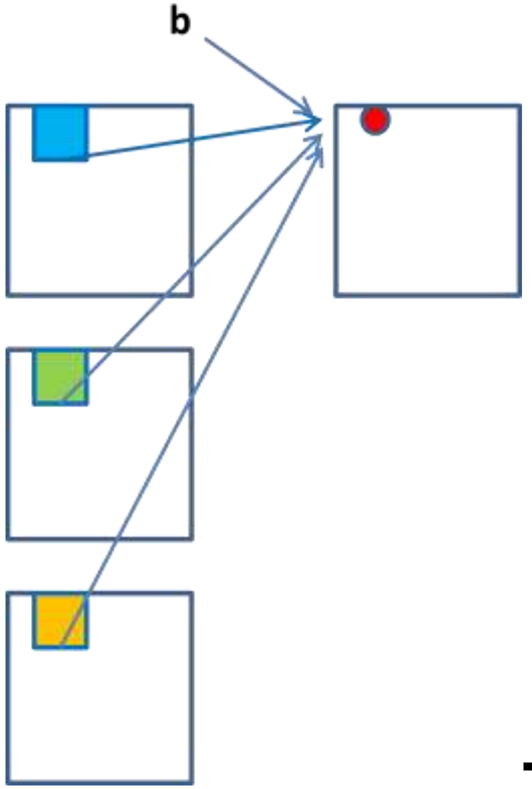


一个 filter 参数： F×F×3+1

卷积网络

卷积层

1 个 filter 的卷积过程



一个 filter 参数： F×F×3+1

卷积层

操作流程

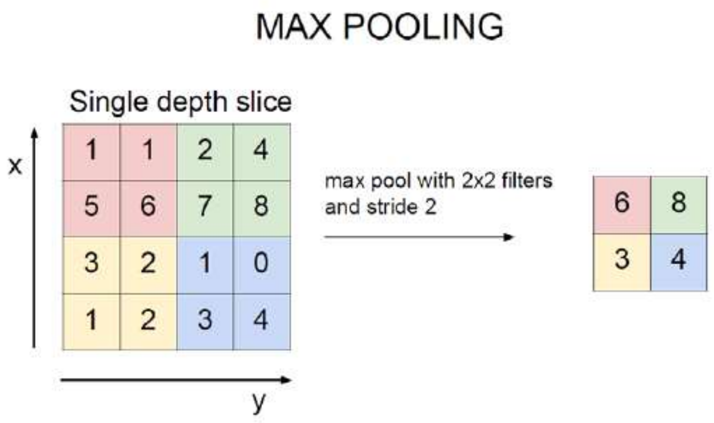
Summary. To summarize, the Conv Layer:

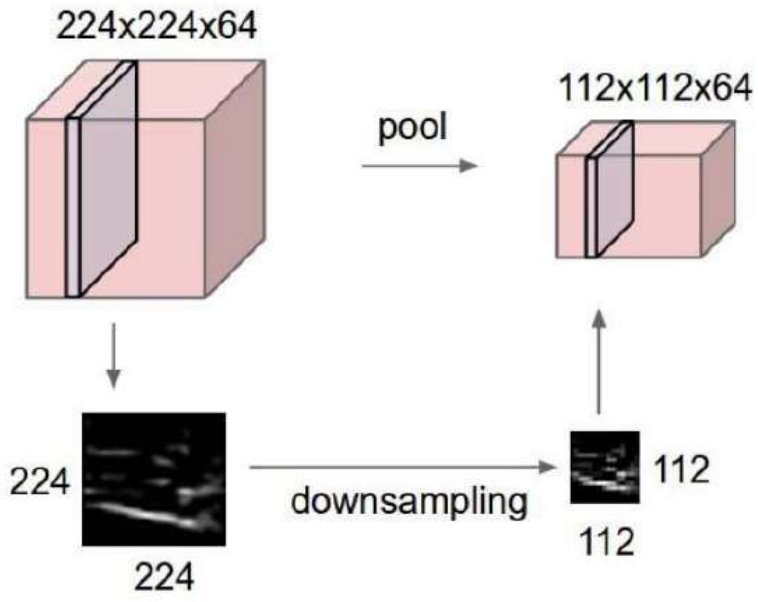
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

子采样层

每个通道，通过下采样，缩减尺度。

二维情况为例：





[1] Stanford Class-CS231n (2016): Convolutional Neural Networks for Visual Recognition.

子采样层

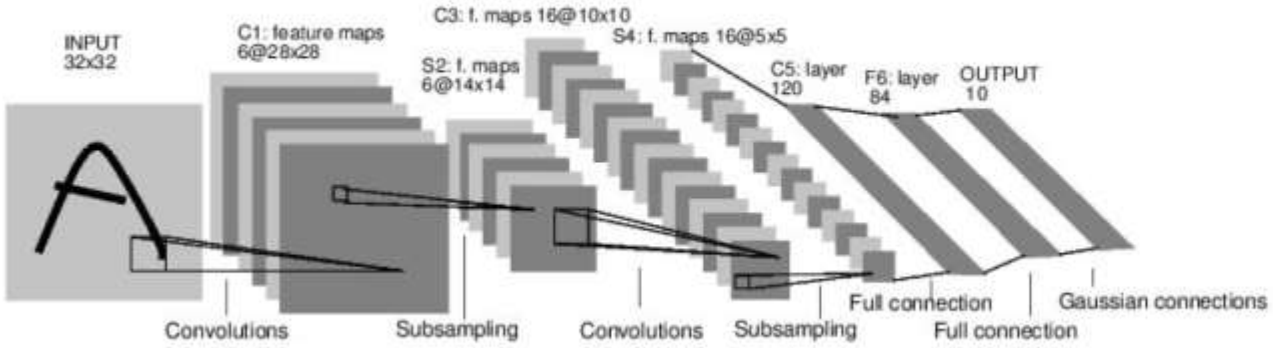
操作流程

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

典型实例

Case Study: LeNet-5

[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

S2-C3 层采用了连接表，

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

本讲参考文献

1. Stanford Class-CS231n: Convolutional Neural Networks for Visual Recognition。
2. Simon Haykin，Neural Network and Learning Machine. 3rd
3. Simon Haykin，申富饶等译，神经网络与学习机器，第三版。
4. 邱锡鹏，《深度学习与自然语言处理》Slides@CCF ADL 20160529。