deepshare.net
深度之眼

《Deep Learning》
深度前馈网络

导师：Johnson

deepshare.net
深度之眼

深度前馈网络

Deep Feedforward Network

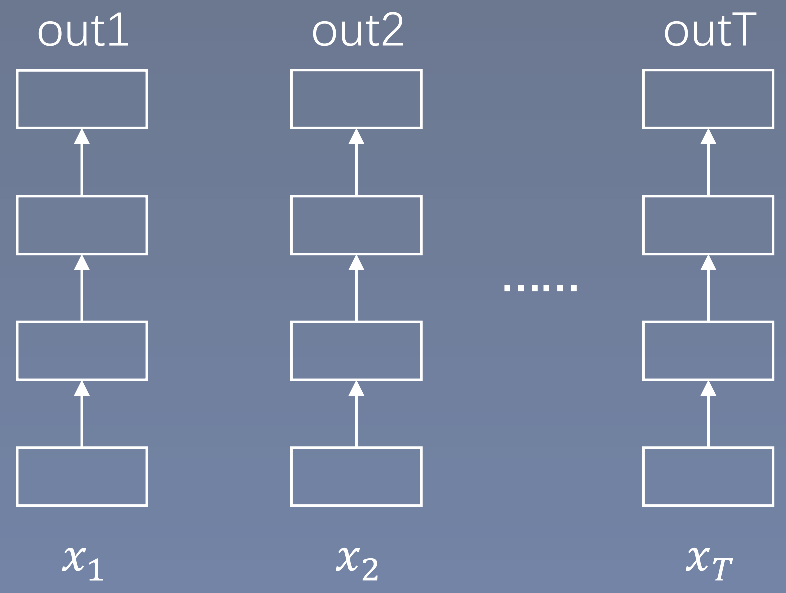
主要内容

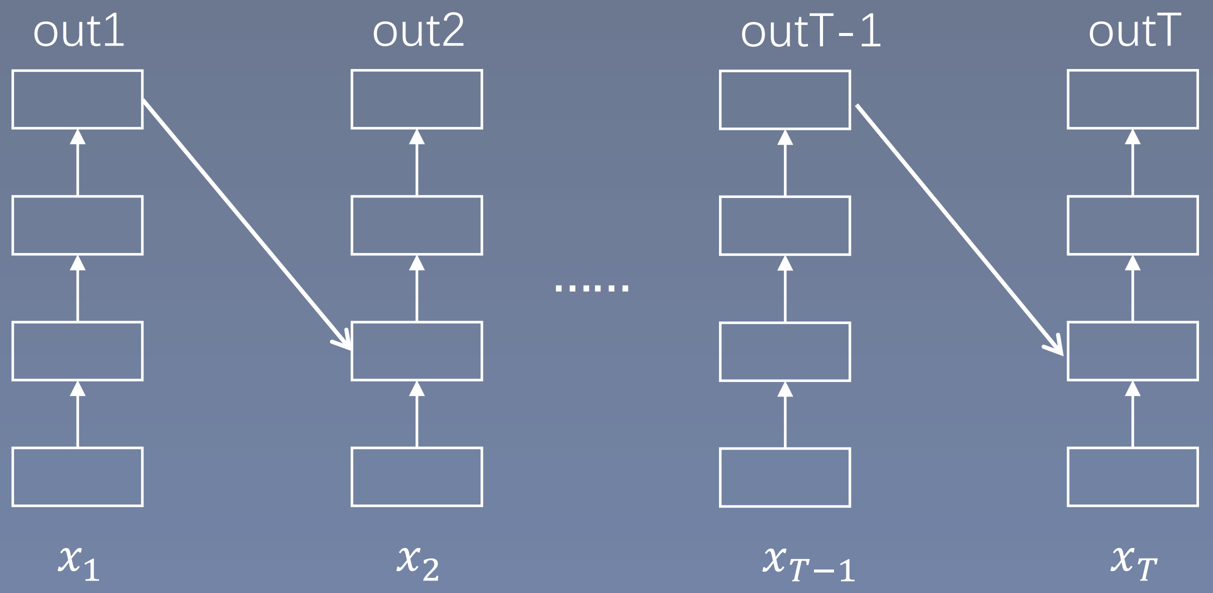
第六章深度前馈网络

深度前馈网络（deep feedforward network），也叫作**前馈神经网络**（feedforward neural network）或者**多层感知机**（multilayer perceptron, MLP），是典型的深度学习模型。前馈网络的目标是近似某个函数 f^*。例如，对于分类器，$\boldsymbol{y} = f^*(\boldsymbol{x})$ 将输入 $\boldsymbol{x}$ 映射到一个类别 y。前馈网络定义了一个映射 $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$，并且学习参数 $\boldsymbol{\theta}$ 的值，使它能够得到最佳的函数近似。

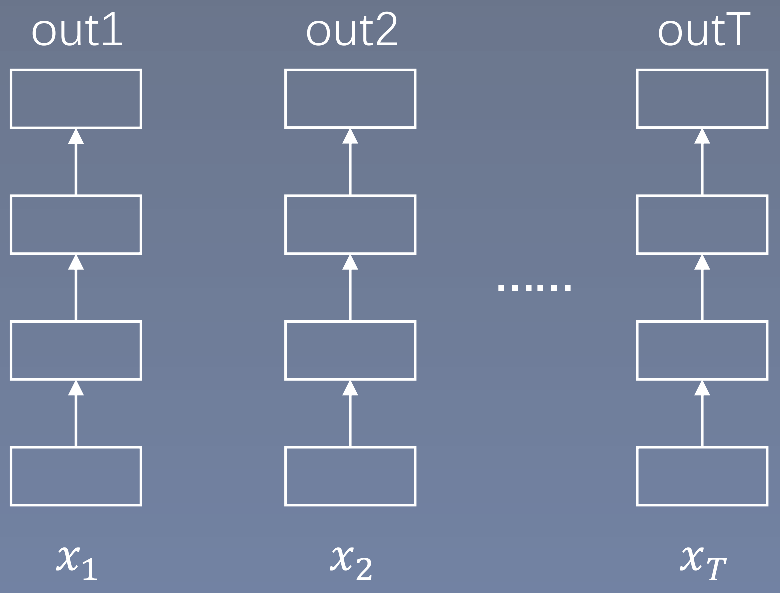
这种模型被称为**前向**（feedforward）的，是因为信息流过 $\boldsymbol{x}$ 的函数，流经用于定义 f 的中间计算过程，最终到达输出 $\boldsymbol{y}$。在模型的输出和模型本身之间没有**反馈**（feedback）连接。当前馈神经网络被扩展成包含反馈连接时，它们被称为**循环神经网络**（recurrent neural network），在第十章介绍。

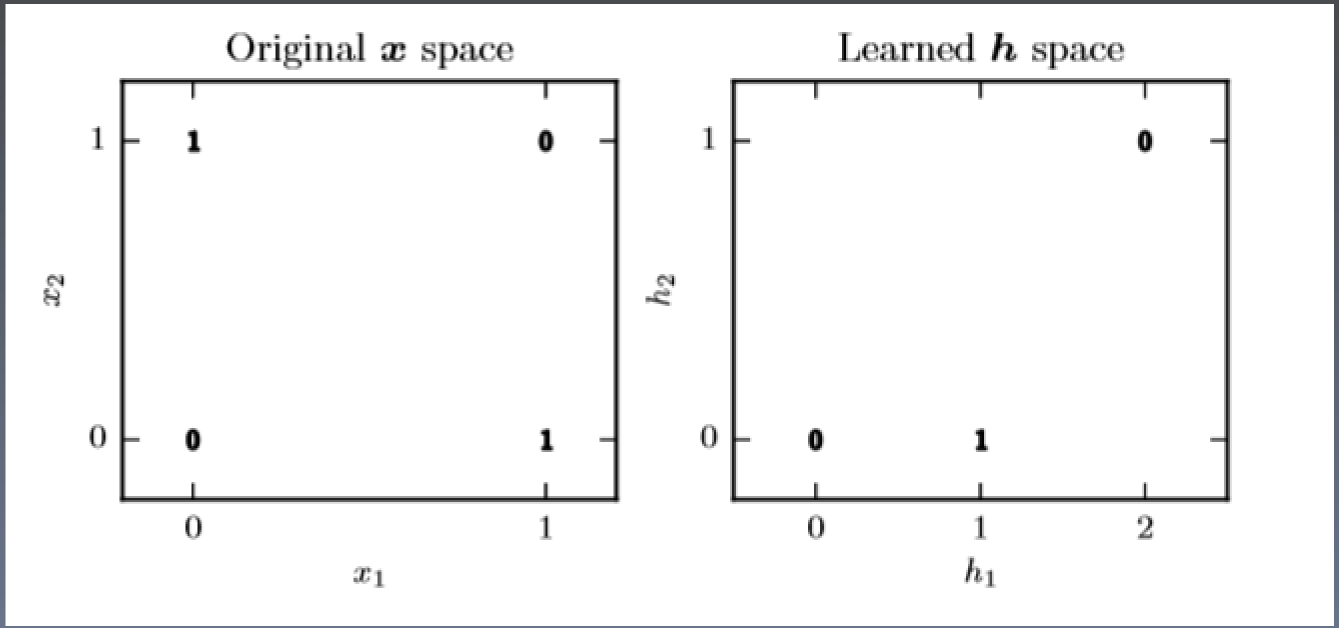
设x1,x2,x3,……xT 为语音序列



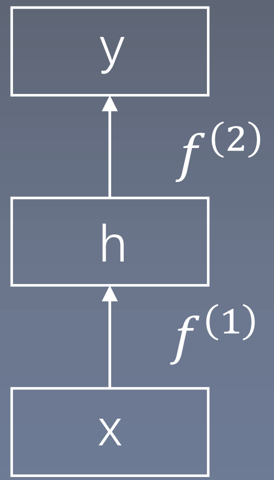


前馈神经网络被称作**网络**（network）是因为它们通常用许多不同函数复合在一起来表示。该模型与一个有向无环图相关联，而图描述了函数是如何复合在一起的。例如，我们有三个函数 $f^{(1)}, f^{(2)}$ 和 $f^{(3)}$ 连接在一个链上以形成 $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$。这些链式结构是神经网络中最常用的结构。在这种情况下，$f^{(1)}$ 被称为网络的**第一层**（first layer），$f^{(2)}$ 被称为**第二层**（second layer），以此类推。链的全长称为模型的**深度**（depth）。正是因为这个术语才出现了“深度学习”这个名字。前馈网络的最后一层被称为**输出层**（output layer）。





deepshare.net
深度之眼



$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{x}^\top \boldsymbol{w} + b.$$

简单的单层线性函数无法解决异或问题

解决办法：增加深度，即加入隐层单元h

$$\boldsymbol{h} = f^{(1)}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c})$$

$y = f^{(2)}(\boldsymbol{h}; \boldsymbol{w}, b)$，完整的模型是 $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = f^{(2)}(f^{(1)}(\boldsymbol{x}))$。

$f^{(1)}$ 应该是哪种函数？线性模型到目前为止都表现不错，让 $f^{(1)}$ 也是线性的似乎很有诱惑力。可惜的是，如果 $f^{(1)}$ 是线性的，那么前馈网络作为一个整体对于输入仍然是线性的。暂时忽略截距项，假设 $f^{(1)}(\boldsymbol{x}) = \boldsymbol{W}^\top \boldsymbol{x}$ 并且 $f^{(2)}(\boldsymbol{h}) = \boldsymbol{h}^\top \boldsymbol{w}$，那么 $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{W}^\top \boldsymbol{x}$。我们可以将这个函数重新表示成 $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{w}'$ 其中 $\boldsymbol{w}' = \boldsymbol{W}\boldsymbol{w}$。

$h = f^{(1)}(x; W, c)$ 　 $y = f^{(2)}(h; w, b)$，完整的模型是 $f(x; W, c, w, b) = f^{(2)}(f^{(1)}(x))$

使用非线性激活函数relu 　 $g(z) = \max\{0, z\}$

$$f(x; W, c, w, b) = w^{\top} \max\{0, W^{\top} x + c\} + b.$$

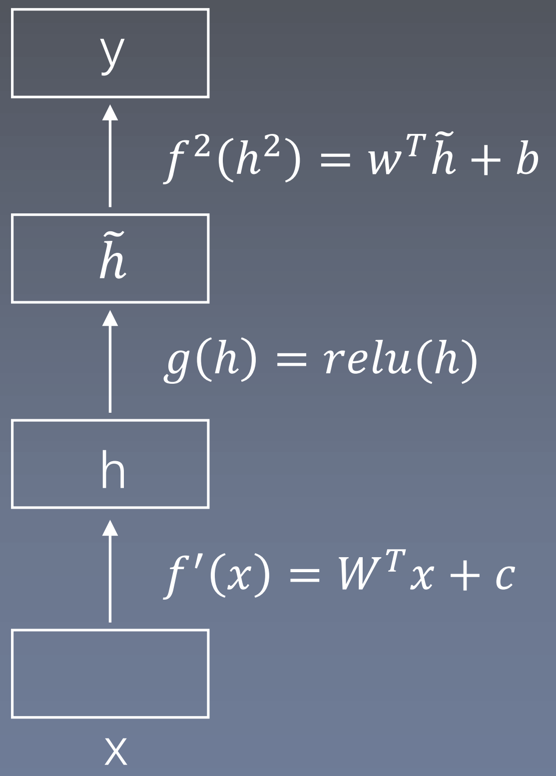
我们现在可以给出 XOR 问题的一个解。令

$$W = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix},$$

$$c = \begin{bmatrix}0\\ -1\end{bmatrix},$$

$$w = \begin{bmatrix}1\\ -2\end{bmatrix},$$

以及 $b = 0$。

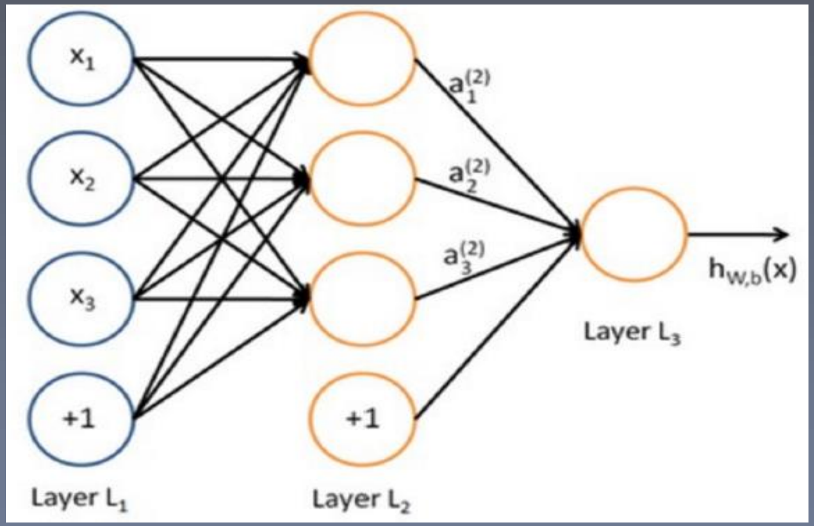


$$X = \begin{bmatrix}0 & 0\\ 0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix}. \quad XW = \begin{bmatrix}0 & 0\\ 1 & 1\\ 1 & 1\\ 2 & 2\end{bmatrix}. \quad +c \quad \begin{bmatrix}0 & -1\\ 1 & 0\\ 1 & 0\\ 2 & 1\end{bmatrix}. \quad \begin{bmatrix}0 & 0\\ 1 & 0\\ 1 & 0\\ 2 & 1\end{bmatrix}. \; * \; w \quad \begin{bmatrix}0\\ 1\\ 1\\ 0\end{bmatrix}.$$

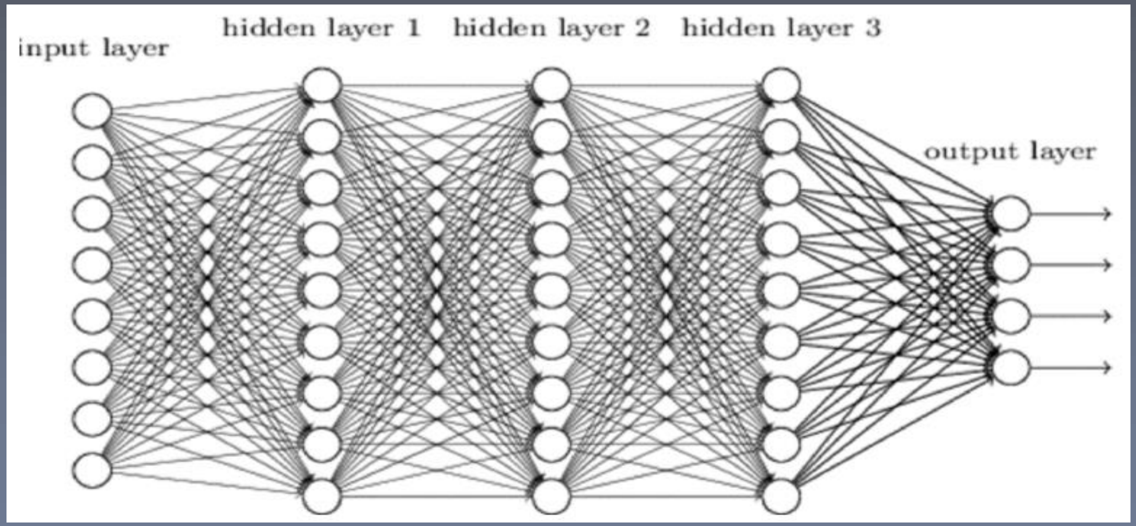
神经网络结构
Neural Network Structure

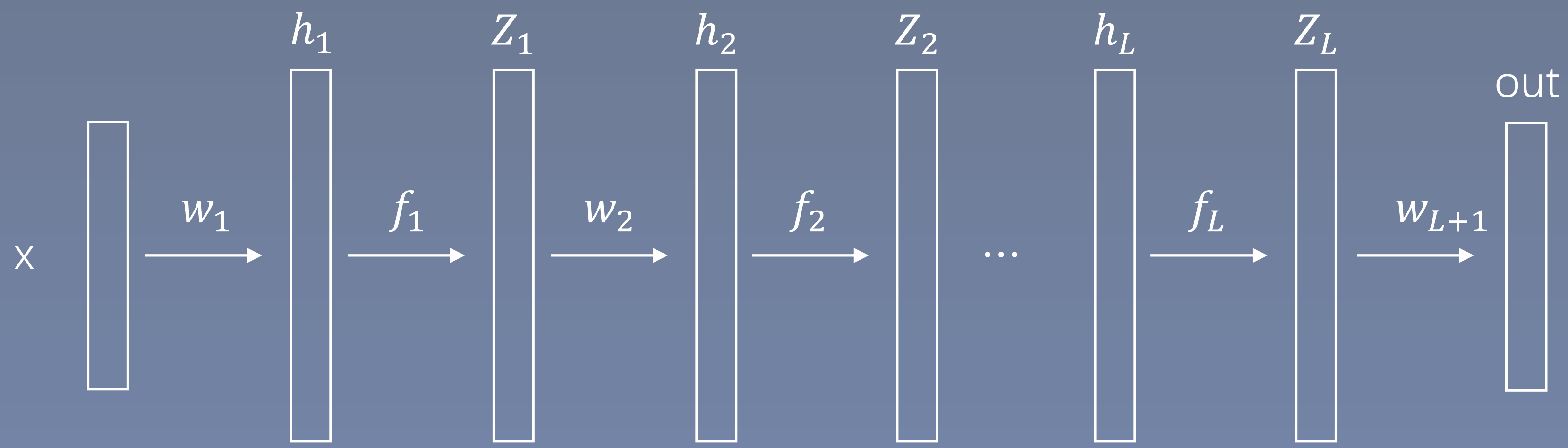
deepshare.net
深度之眼

添加少量隐层 → 浅层神经网络（SNN）



添加多中间层 → 深度神经网络（DNN）





其中Wi为矩阵，fi为非线性激活函数，如relu,sigmoid,tanh等等

神经网络表达力与过拟合

Neural Network Expressiveness and Overfitting

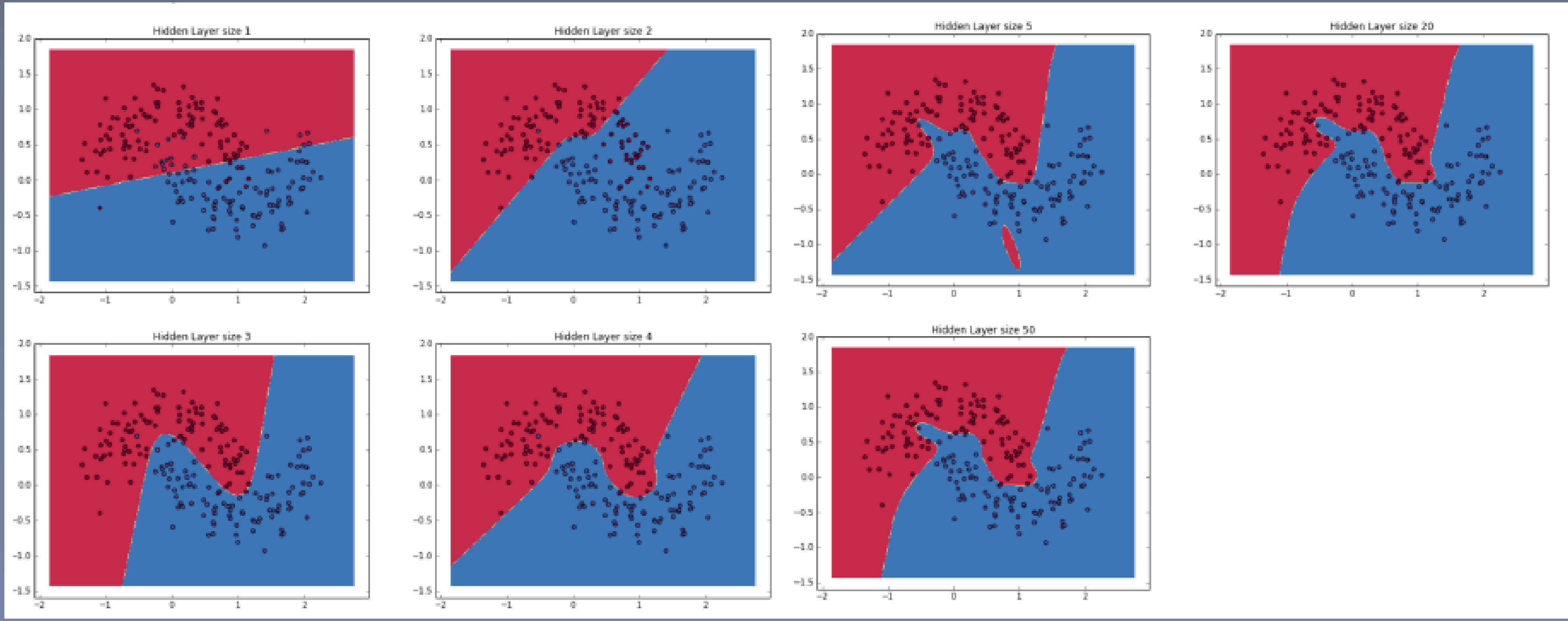
deepshare.net
深度之眼

- 理论上说单隐层神经网络可以逼近任何连续函数(只要隐层的神经元个数足够多)。
- 虽然从数学上看表达能力一致,但是多隐藏层的神经网络比单隐藏层的神经网络工程效果好很多。
- 对于一些分类数据(比如CTR预估里),3层神经网络效果优于2层神经网络,但是如果把层数再不断增加(4,5,6层),对最后结果的帮助就没有那么大的跳变了。
- 图像数据比较特殊,是一种深层(多层次)的结构化数据,深层次的卷积神经网络,能够更充分和准确地把这些层级信息表达出来。

神经网络表达力与过拟合

Neural Network Expressiveness and Overfitting

- 提升隐层层数或者隐层神经元个数,神经网络“容量”会变大,空间表达力会变强。
- 过多的隐层和神经元节点,会带来过拟合问题不要试图通过降低神经网络参数量来减缓过拟合,用正则化战者 dropout

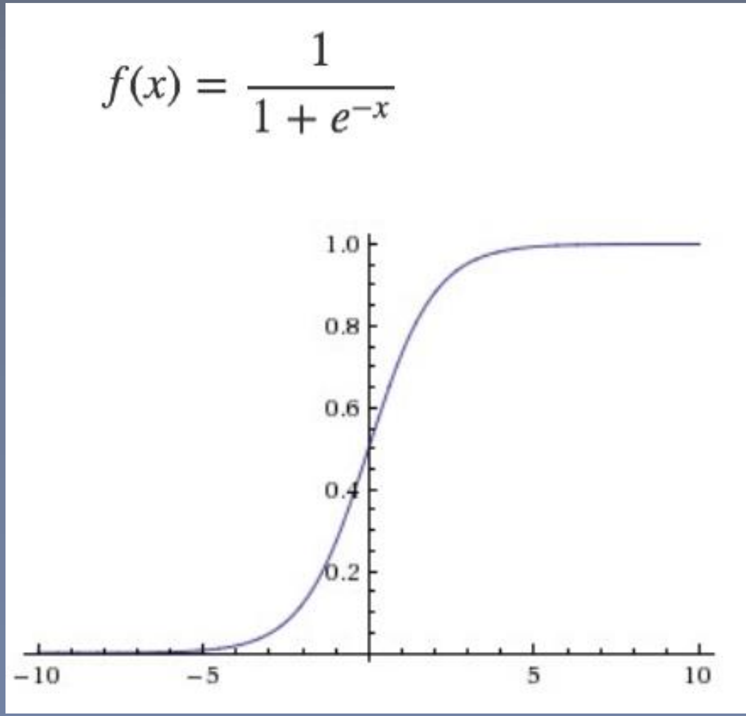


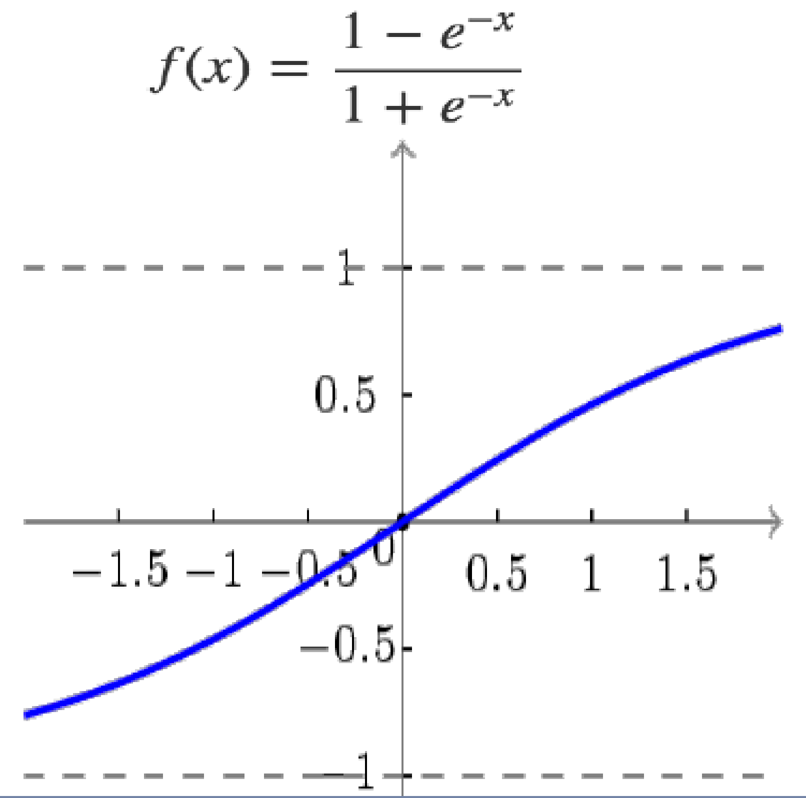
传递函数（激活函数）

Activation function

deepshare.net
深度之眼

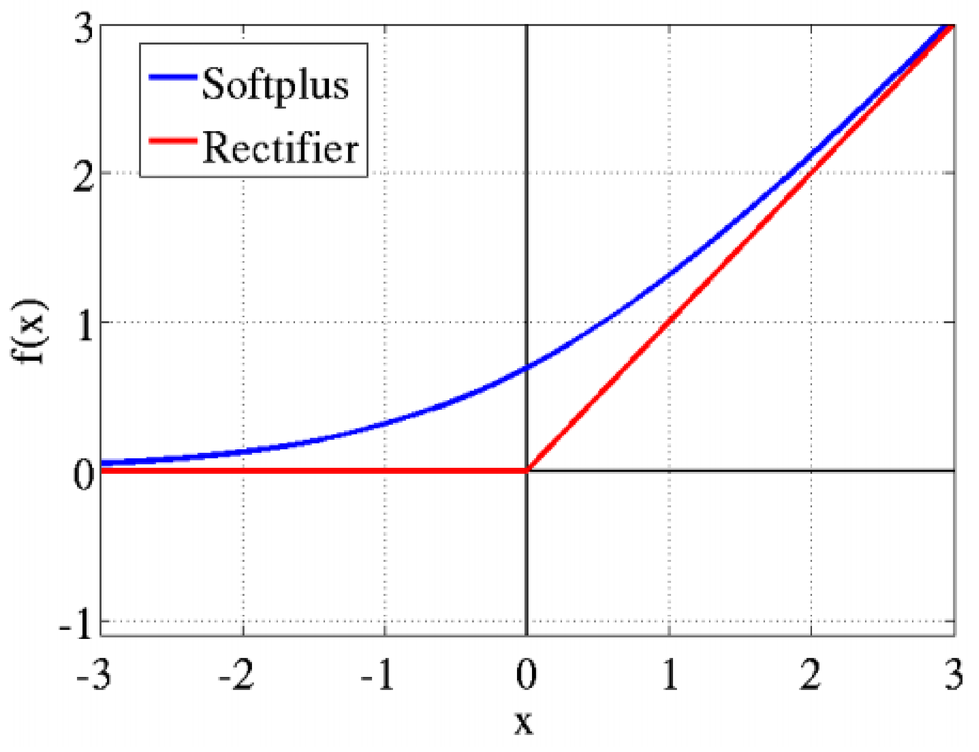
Sigmoid函数 **VS** tanh（双曲正切）函数

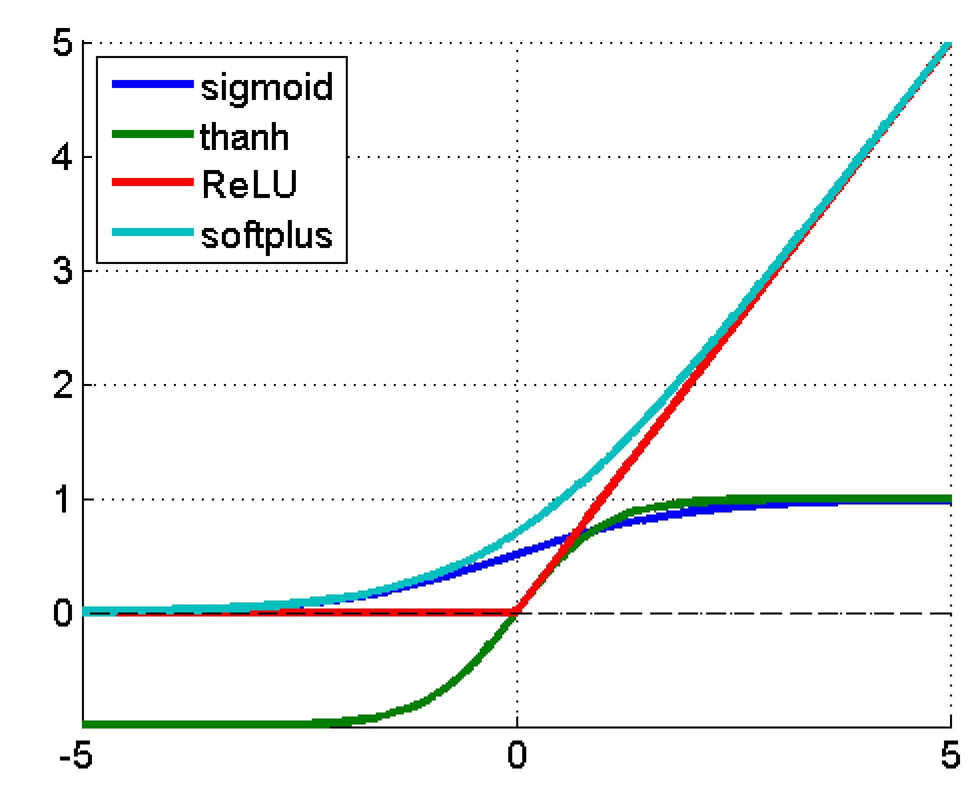




传递函数（激活函数）： Relu函数

Activation function





传递函数（激活函数）

Activation function

deepshare.net
深度之眼

为什么通常Relu比sigmoid和tanh强，有什么不同？

- 主要是因为它们梯度特性不同。sigmoid和tanh的梯度在饱和区域非常平缓，接近于0，很容易造成梯度消失（vanishing gradient）的问题，减缓收敛速度。梯度消失在网络层数多的时候尤其明显，是加深网络结构的主要障碍之一。
- Relu的梯度大多数情况下是常数，有助于解决深层网络的收敛问题。Relu的另一个优势是在生物上的合理性，它是单边的，相比sigmoid和tanh，更符合生物神经元的特征。
- 而提出sigmoid和tanh，主要是因为它们全程可导。还有表达区间问题，sigmoid和tanh区间是0到1，或着-1到1，在表达上，尤其是输出层的表达上有优势。
- ReLU更容易学习优化。因为其分段线性性质，导致其前传，后传，求导都是分段线性。而传统的sigmoid函数，由于两端饱和，在传播过程中容易丢弃信息：

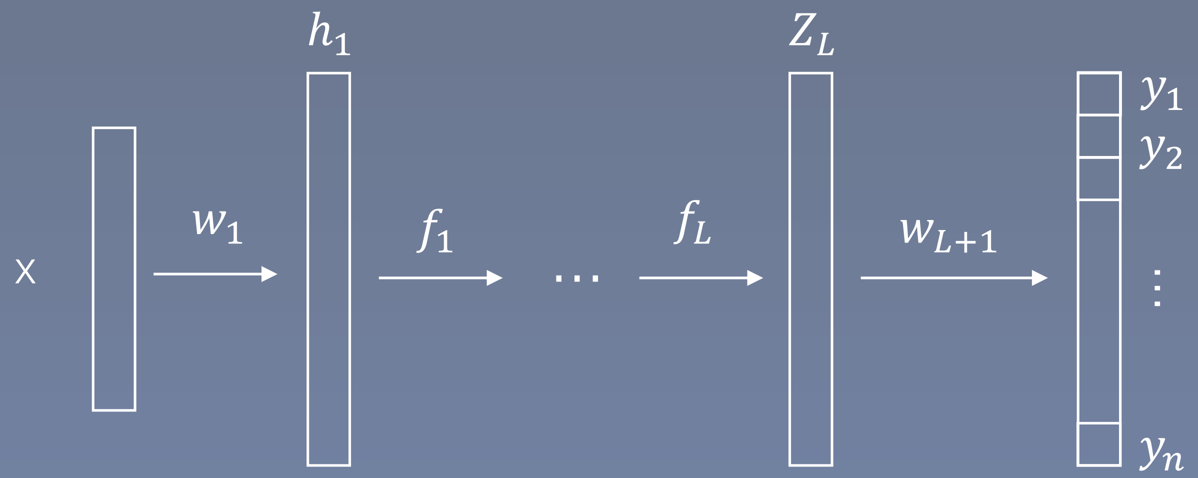
损失函数

loss function

两类问题

1.回归问题(MSE)

2.分类问题(cross entropy)

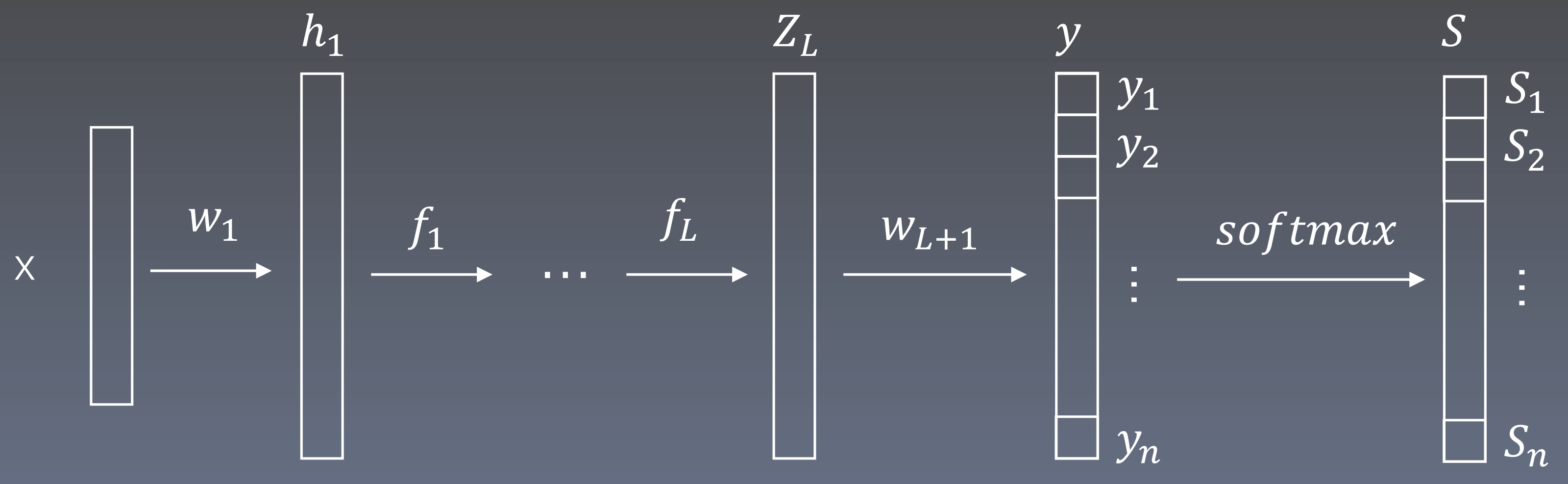


$$J = \frac{1}{2N}\sum_{i=1}^{N}\left\|y^i - \tilde{y}^i\right\|^2$$

$$\frac{\partial J}{\partial y^i} = \frac{1}{2N}\sum_{i=1}^{N}\left(y^i - \tilde{y}^i\right) * 2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \tilde{y}^i\right)$$

deepshare.net
深度之眼



$$S_k = \frac{e^{y_k}}{\sum_{i=1}^{n} e^{y_i}}$$

KL距离（相对熵）

KL距离，是Kullback-Leibler差异（Kullback-Leibler Divergence）的简称，也叫做相对熵（Relative Entropy）。它衡量的是相同事件空间里的两个概率分布的差异情况。其物理意义是：在相同事件空间里，概率分布P(x)对应的每个事件，若用概率分布 Q(x)编码时，平均每个基本事件（符号）编码长度增加了多少比特。我们用D（P||Q）表示KL距离，计算公式如下：

$$D(P\|Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}$$

当两个概率分布完全相同时，即P(X)=Q(X)，其相对熵为0 。我们知道，概率分布P(X)的信息熵为：

对于分类问题来讲，一个样本对应的网络的输出S(s1,s2,...,sn)是一个概率分布，而这个样本的标注 $\tilde{S}$ 一般为(0,0,...1,0,0...0),也可以看做一个概率分布(硬分布)

cross entropy可以看成是$\tilde{s}$与s之间的KL距离.

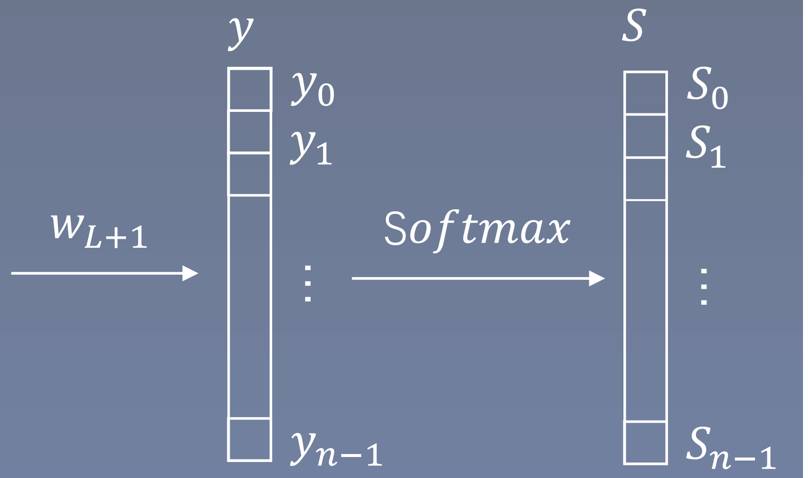
$$D(\tilde{s}||s) = \sum \tilde{s} \log \frac{\tilde{s}}{s}$$

$$D(\tilde{s}||s) = \sum \tilde{s}\log\frac{\tilde{s}}{s}$$

假设$\tilde{S} = (0,0,\cdots,1,0,\cdots,0)$ ，其中1为$\tilde{s}$的第k个元素（索引从0开始）

令$S = (s_0, s_1, \cdots, s_k, \cdots, s_{n-1})$

$$D(\tilde{s}||s) = 1\log\frac{1}{s_k} = -\log s_k \ (CE\text{损失函数})$$



$$J = -\log s_k \text{(可看做目标类别概率最大)}$$

$$= -\log\frac{e^{y_k}}{\sum_{i=0}^{n-1} e^{y_i}}$$

$$\frac{\partial J}{\partial y_m} = \frac{\partial J}{\partial y_m}(\log\sum_{i=0}^{n-1} e^{y_i} - y_k)$$

$$= \frac{e^{y_m}}{\sum_{i=0}^{n-1} e^{y_i}} - \delta(m=k)$$

$$= s_m - \delta(m=k)$$

写成向量形式：$\dfrac{\partial J}{\partial y} = S - \tilde{S}$

$S = (s_0, s_1, \cdots, s_k, \cdots, s_{n-1})$

$\tilde{S} = (0,0,\cdots,1,0,\cdots,0)$

为了简单，这里使用了单个样本推导

批量梯度下降

Batch Gradient Descent

优点：

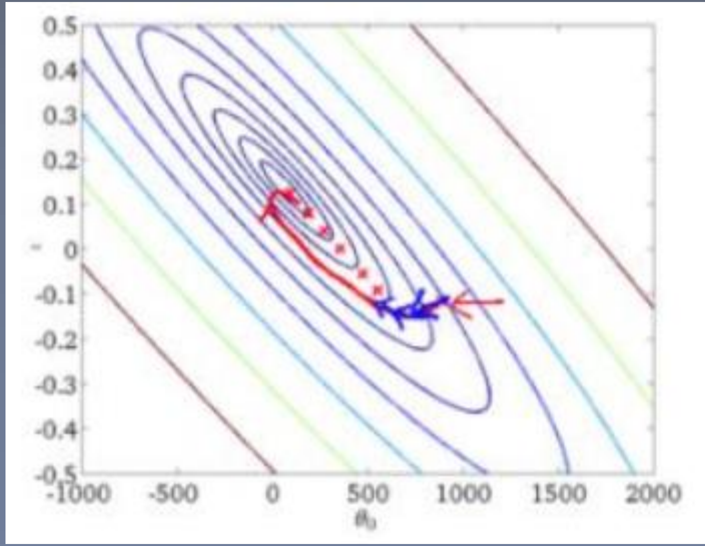
（1）一次迭代是对所有样本进行计算，此时利用矩阵进行操作，实现了并行。

（2）由全数据集确定的方向能够更好地代表样本总体，从而更准确地朝向极值所在的方向。当目标函数为凸函数时，BGD一定能够得到全局最优。

缺点：

（1）当样本数目 m 很大时，每迭代一步都需要对所有样本计算，训练过程会很慢。

从迭代的次数上来看，BGD迭代的次数相对较少。其迭代的收敛曲线示意图可以表示如下：

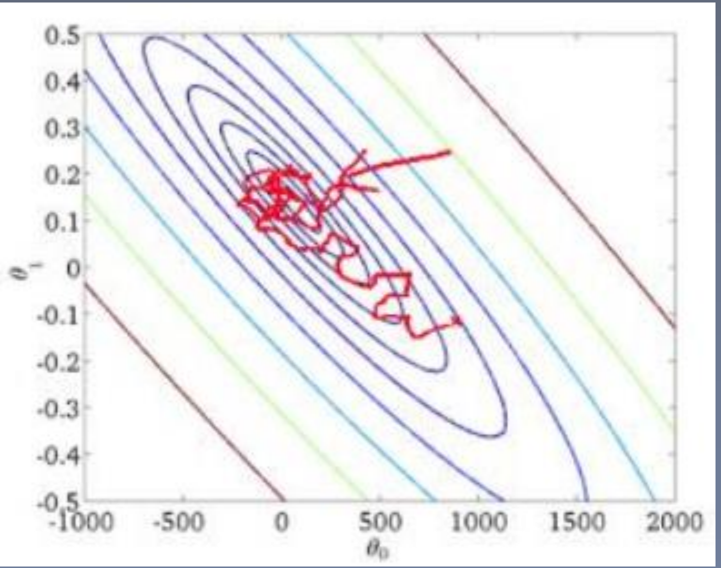


随机梯度下降

Stochastic Gradient Descent

优点：

（1）由于不是在全部训练数据上的损失函数，而是在每轮迭代中，随机优化某一条训练数据上的损失函数，这样每一轮参数的更新速度大大加快。



缺点：

（1）准确度下降。由于即使在目标函数为强凸函数的情况下，SGD仍旧无法做到线性收敛。

（2）可能会收敛到局部最优，由于单个样本并不能代表全体样本的趋势。

（3）不易于并行实现。

小批量梯度下降

Mini-Batch Gradient Descent

优点：

（1）通过矩阵运算，每次在一个batch上优化神经网络参数并不会比单个数据慢太多。

（2）每次使用一个batch可以大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近梯度下降的效果。

（3）可实现并行化。

缺点：

（1）batch_size的不当选择可能会带来一些问题。

小批量梯度下降

Mini-Batch Gradient Descent

deepshare.net
深度之眼

batcha_size的选择带来的影响：

（1）在合理地范围内，增大batch_size的好处：

a. 内存利用率提高了，大矩阵乘法的并行化效率提高。

b. 跑完一次 epoch（全数据集）所需的迭代次数减少，对于相同数据量的处理速度进一步加快。

c. 在一定范围内，一般来说 Batch_Size 越大，其确定的下降方向越准，引起训练震荡越小。

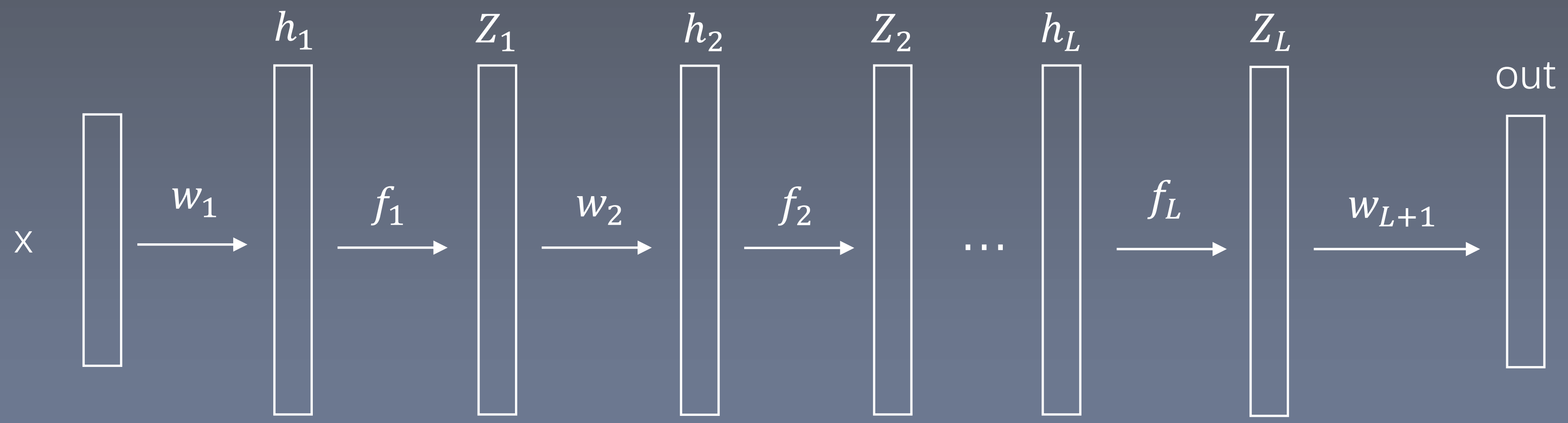
（2）盲目增大batch_size的坏处：

a. 内存利用率提高了，但是内存容量可能撑不住了。

b. 跑完一次 epoch（全数据集）所需的迭代次数减少，要想达到相同的精度，其所花费的时间大大增加了，从而对参数的修正也就显得更加缓慢。

c. Batch_Size 增大到一定程度，其确定的下降方向已经基本不再变化。

前向传播



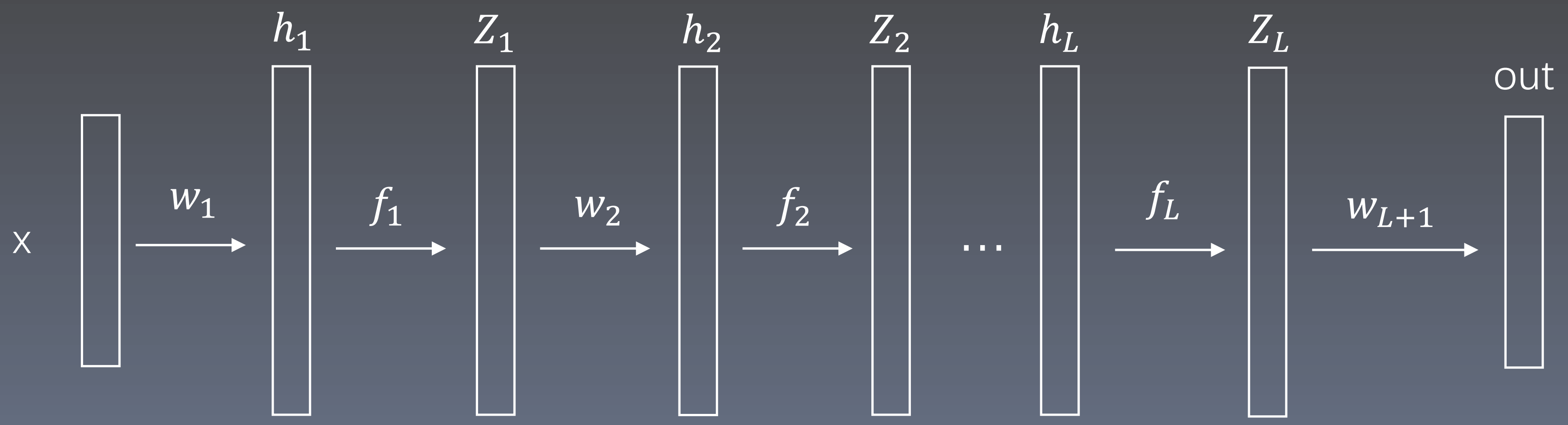
假设X为$N * M$的矩阵（其中N为样本个数，M为特征维数）

h_1与Z_1的维数为$m_1 \rightarrow W_1$为$m * m_1$的矩阵, $b_1 \in \mathbb{R}^{m_1}$,

h_2与Z_2的维数为$m_2 \rightarrow W_2$为$m_1 * m_2$的矩阵, $b_2 \in \mathbb{R}^{m_2}$,

$\vdots$

h_L与Z_L的维数为$m_L \rightarrow W_L$为$m_{L-1} * m_L$的矩阵, $b_L \in \mathbb{R}^{m_L}$.



前向算法:

$h_1 = xw_1 + \tilde{b}_1, Z_1 = f_1(h_1)$, $\tilde{b}_1$为${b_1}^T$后沿着行方向扩展N行

$h_2 = Z_1 w_2 + \tilde{b}_2, Z_2 = f_2(h_2)$

$\vdots$

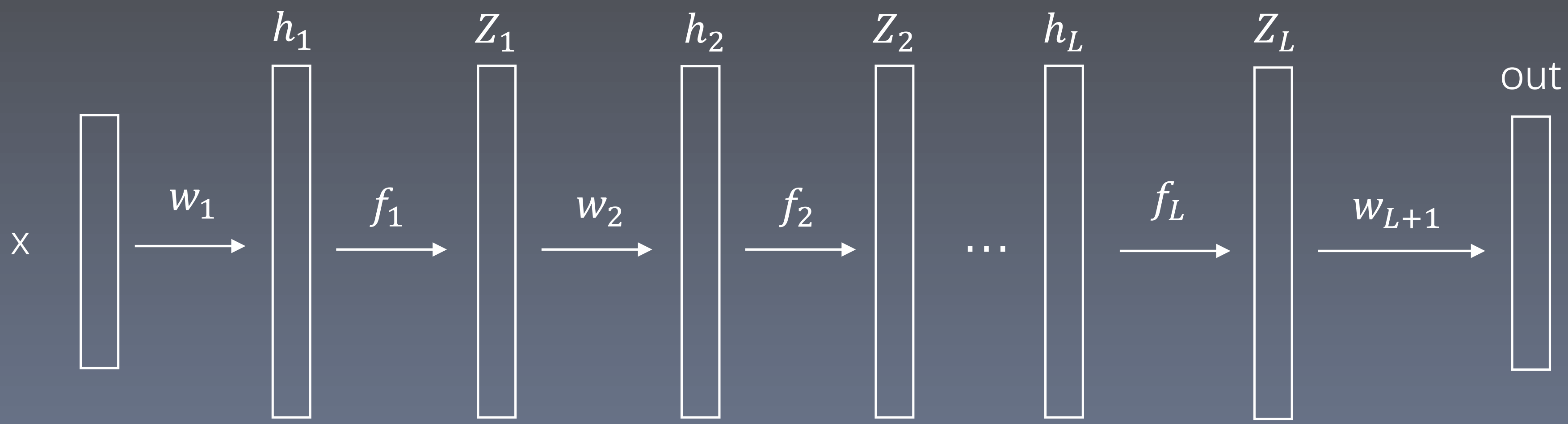
$h_L = Z_{L-1} w_L + \tilde{b}_L, Z_L = f_L(h_L)$

$out = Z_L w_{L+1} + \tilde{b}_{L+1}$

假设输出为n维,
则out矩阵为N*n

$\frac{\partial J}{\partial out}$ 根据mse或者是ce准则来求出

反向传播



$$out = Z_L w_{L+1} + \tilde{b}_{L+1}$$

$$Z_L = \begin{pmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \end{pmatrix}_{2\times 3}, W_{L+1} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}_{3\times 2}, \tilde{b}_{L+1} = \begin{pmatrix} b_1 & b_2 \\ b_1 & b_2 \end{pmatrix}_{2\times 2}, out = \begin{pmatrix} o_{11} & o_{12} \\ o_{21} & o_{22} \end{pmatrix}$$

$$Z_L \cdot W_{L+1} = \begin{pmatrix} z_{11}w_{11} + z_{12}w_{21} + z_{13}w_{31} & , & z_{11}w_{12} + z_{12}w_{22} + z_{13}w_{32} \\ z_{21}w_{11} + z_{22}w_{21} + z_{23}w_{31} & , & z_{21}w_{12} + z_{22}w_{22} + z_{23}w_{32} \end{pmatrix}$$

$$o_{11} = z_{11}w_{11} + z_{12}w_{21} + z_{13}w_{31} + b_1,$$

$$o_{12} = z_{11}w_{12} + z_{12}w_{22} + z_{13}w_{32} + b_2,$$

$$o_{21} = z_{21}w_{11} + z_{22}w_{21} + z_{23}w_{31} + b_1,$$

$$o_{22} = z_{21}w_{12} + z_{22}w_{22} + z_{23}w_{32} + b_2.$$

$$o_{11}=z_{11}w_{11}+z_{12}w_{21}+z_{13}w_{31}+b_1,$$
$$o_{12}=z_{11}w_{12}+z_{12}w_{22}+z_{13}w_{32}+b_2,$$
$$o_{21}=z_{21}w_{11}+z_{22}w_{21}+z_{23}w_{31}+b_1,$$
$$o_{22}=z_{21}w_{12}+z_{22}w_{22}+z_{23}w_{32}+b_2.$$

$$out=Z_LW_{L+1}+\tilde{b}_{L+1}$$

deepshare.net

深度之眼

$$\frac{\partial J}{\partial w_{11}}=\frac{\partial J}{\partial o_{11}}z_{11}+\frac{\partial J}{\partial o_{21}}z_{21},\frac{\partial J}{\partial w_{12}}=\frac{\partial J}{\partial o_{12}}z_{11}+\frac{\partial J}{\partial o_{22}}z_{21}$$
$$\frac{\partial J}{\partial w_{21}}=\frac{\partial J}{\partial o_{11}}z_{12}+\frac{\partial J}{\partial o_{21}}z_{22},\frac{\partial J}{\partial w_{22}}=\frac{\partial J}{\partial o_{12}}z_{12}+\frac{\partial J}{\partial o_{22}}z_{22}$$
$$\frac{\partial J}{\partial w_{31}}=\frac{\partial J}{\partial o_{11}}z_{13}+\frac{\partial J}{\partial o_{21}}z_{23},\frac{\partial J}{\partial w_{32}}=\frac{\partial J}{\partial o_{12}}z_{13}+\frac{\partial J}{\partial o_{22}}z_{23}$$
$$\begin{pmatrix}\frac{\partial J}{\partial w_{11}} & \frac{\partial J}{\partial w_{12}}\\ \frac{\partial J}{\partial w_{21}} & \frac{\partial J}{\partial w_{22}}\\ \frac{\partial J}{\partial w_{31}} & \frac{\partial J}{\partial w_{32}}\end{pmatrix}=\begin{pmatrix}z_{11} & z_{21}\\ z_{12} & z_{22}\\ z_{13} & z_{23}\end{pmatrix}\begin{pmatrix}\frac{\partial J}{\partial o_{11}} & \frac{\partial J}{\partial o_{12}}\\ \frac{\partial J}{\partial o_{21}} & \frac{\partial J}{\partial o_{22}}\end{pmatrix}$$
$$\frac{\partial J}{\partial W_{L+1}}=Z_L{}^T\frac{\partial J}{\partial out}$$

$$\begin{cases}\frac{\partial J}{\partial b_1}=\frac{\partial J}{\partial o_{11}}+\frac{\partial J}{\partial o_{21}}\\ \frac{\partial J}{\partial b_2}=\frac{\partial J}{\partial o_{12}}+\frac{\partial J}{\partial o_{22}}\end{cases},\Rightarrow\left(\frac{\partial J}{\partial b}\right)^T=\begin{pmatrix}\frac{\partial J}{\partial b_1} & \frac{\partial J}{\partial b_2}\end{pmatrix}=\begin{pmatrix}\frac{\partial J}{\partial o_{11}}+\frac{\partial J}{\partial o_{21}} & \frac{\partial J}{\partial o_{12}}+\frac{\partial J}{\partial o_{22}}\end{pmatrix}=\text{将}\frac{\partial J}{\partial out}\text{的每一行加起来}$$

$$o_{11}=z_{11}w_{11}+z_{12}w_{21}+z_{13}w_{31}+b_1,$$
$$o_{12}=z_{11}w_{12}+z_{12}w_{22}+z_{13}w_{32}+b_2,$$
$$o_{21}=z_{21}w_{11}+z_{22}w_{21}+z_{23}w_{31}+b_1,$$
$$o_{22}=z_{21}w_{12}+z_{22}w_{22}+z_{23}w_{32}+b_2.$$

$$out=Z_L W_{L+1}+\tilde{b}_{L+1}$$

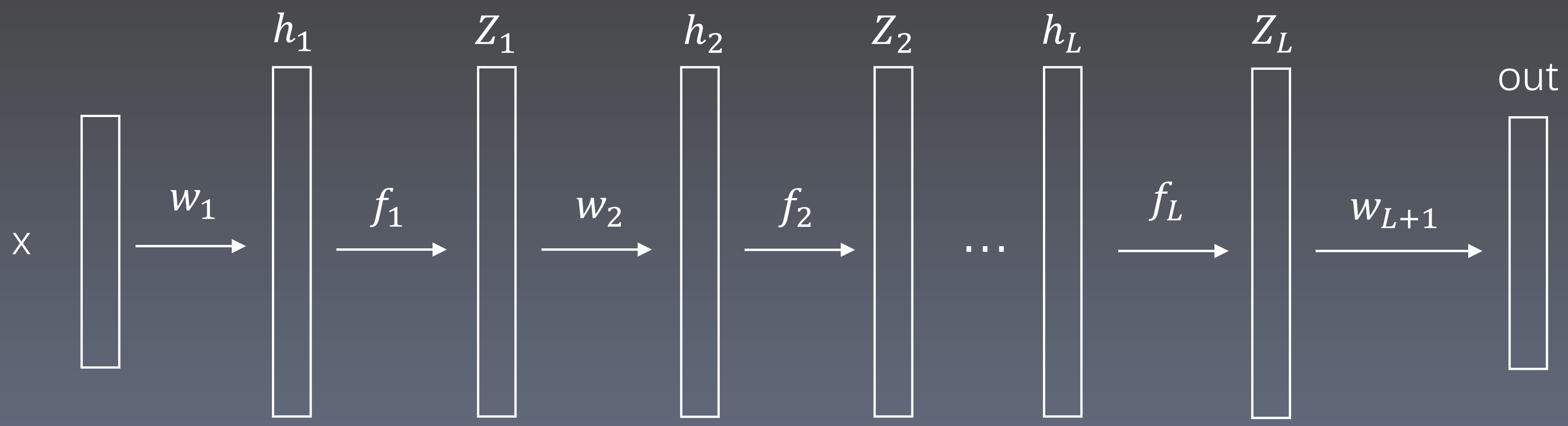
deepshare.net
深度之眼

$$\frac{\partial J}{\partial z_{11}}=\frac{\partial J}{\partial o_{11}}w_{11}+\frac{\partial J}{\partial o_{12}}w_{12};\frac{\partial J}{\partial z_{12}}=\frac{\partial J}{\partial o_{11}}w_{21}+\frac{\partial J}{\partial o_{12}}w_{22};\frac{\partial J}{\partial z_{13}}=\frac{\partial J}{\partial o_{11}}w_{31}+\frac{\partial J}{\partial o_{12}}w_{32}$$

$$\frac{\partial J}{\partial z_{21}}=\frac{\partial J}{\partial o_{21}}w_{11}+\frac{\partial J}{\partial o_{22}}w_{12};\frac{\partial J}{\partial z_{22}}=\frac{\partial J}{\partial o_{21}}w_{21}+\frac{\partial J}{\partial o_{12}}w_{22};\frac{\partial J}{\partial z_{23}}=\frac{\partial J}{\partial o_{21}}w_{31}+\frac{\partial J}{\partial o_{22}}w_{32}$$

$$\begin{pmatrix}\frac{\partial J}{\partial z_{11}} & \frac{\partial J}{\partial z_{12}} & \frac{\partial J}{\partial z_{13}}\\ \frac{\partial J}{\partial z_{21}} & \frac{\partial J}{\partial z_{22}} & \frac{\partial J}{\partial z_{23}}\end{pmatrix}=\begin{pmatrix}\frac{\partial J}{\partial o_{11}} & \frac{\partial J}{\partial o_{12}}\\ \frac{\partial J}{\partial o_{21}} & \frac{\partial J}{\partial o_{22}}\end{pmatrix}\begin{pmatrix}w_{11} & w_{21} & w_{31}\\ w_{12} & w_{22} & w_{32}\end{pmatrix}$$

$$\frac{\partial J}{\partial Z_L}=\frac{\partial J}{\partial out}W_{L+1}^T$$



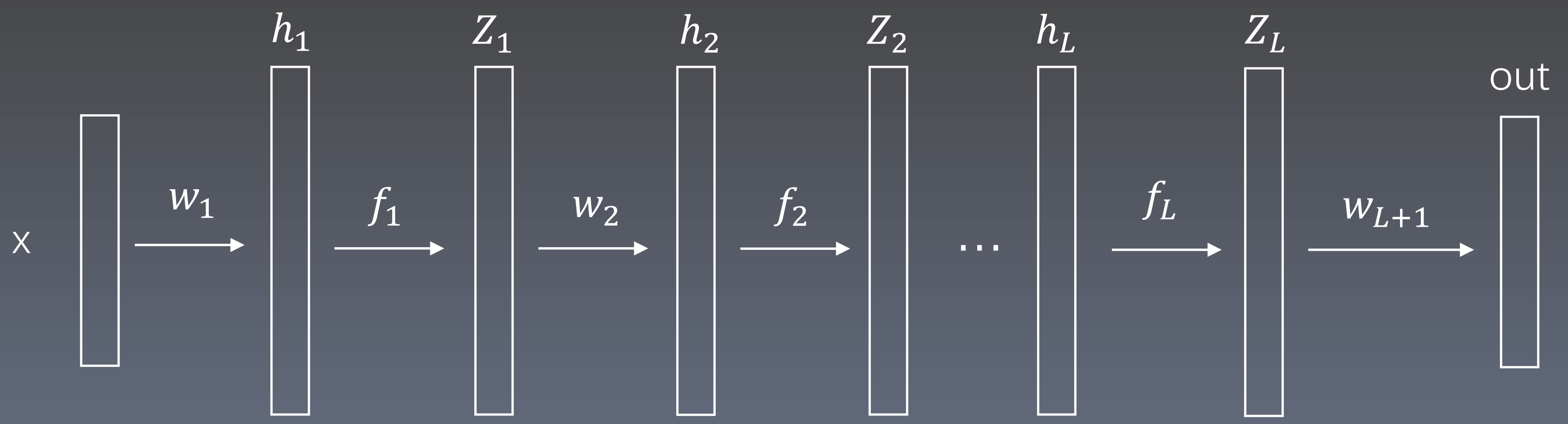
$$Z_L = f_L(h_L)$$

①f_L为sigmoid

$$Z_L = \frac{1}{He^{-h_L}}$$

$$\frac{\partial J}{\partial h_L} = \frac{\partial J}{\partial Z_L}\frac{dz_L}{dh_L} = \frac{\partial J}{\partial Z_L}\frac{e^{-hL}}{(1+e^{-h_L})^2} = \frac{\partial J}{\partial Z_L}\frac{1}{1+e^{-h_L}}\frac{e^{-h_L}}{1+e^{-h_L}}$$

$$= \frac{\partial J}{\partial Z_L} Z_L(1 - Z_L)$$

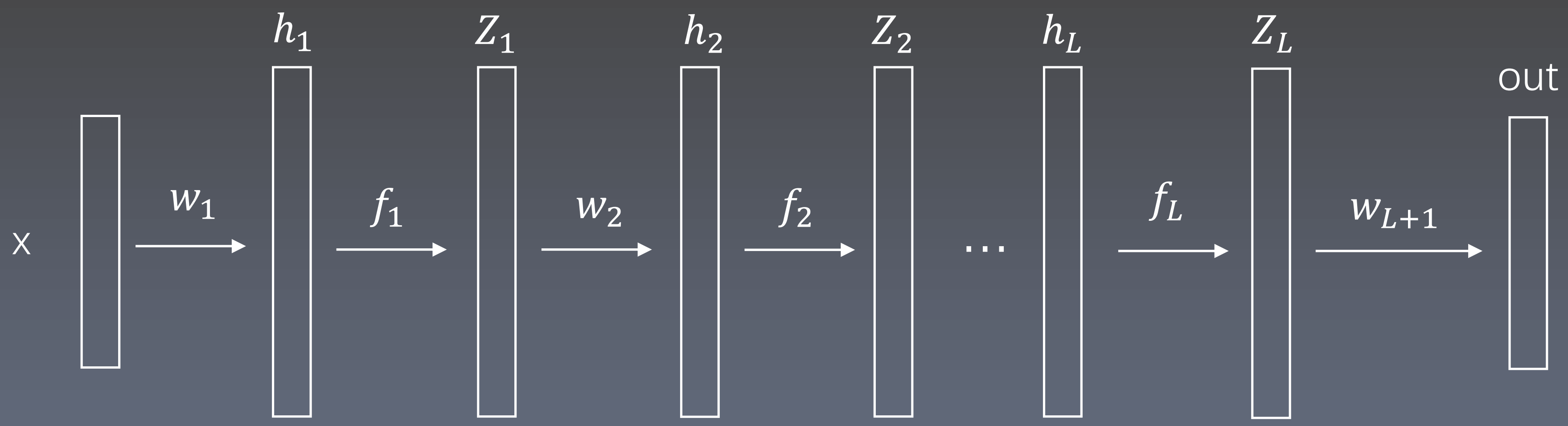


②f_L为tanh

$$Z_L = \frac{e^{h_L} - e^{-h_L}}{e^{h_L} + e^{-h_L}}$$

$$\frac{\partial J}{\partial h_L} = \frac{\partial J}{\partial Z_L}\frac{dZ_L}{dh_L} = \frac{\partial J}{\partial Z_L}\frac{4}{(e^{h_L} + e^{-h_L})^2} = \frac{\partial J}{\partial Z_L}[1 - (\frac{e^{h_L} - e^{-h_L}}{e^{h_L} + e^{-h_L}})^2]$$

$$= \frac{\partial J}{\partial z_L}[1 - {z_L}^2]$$

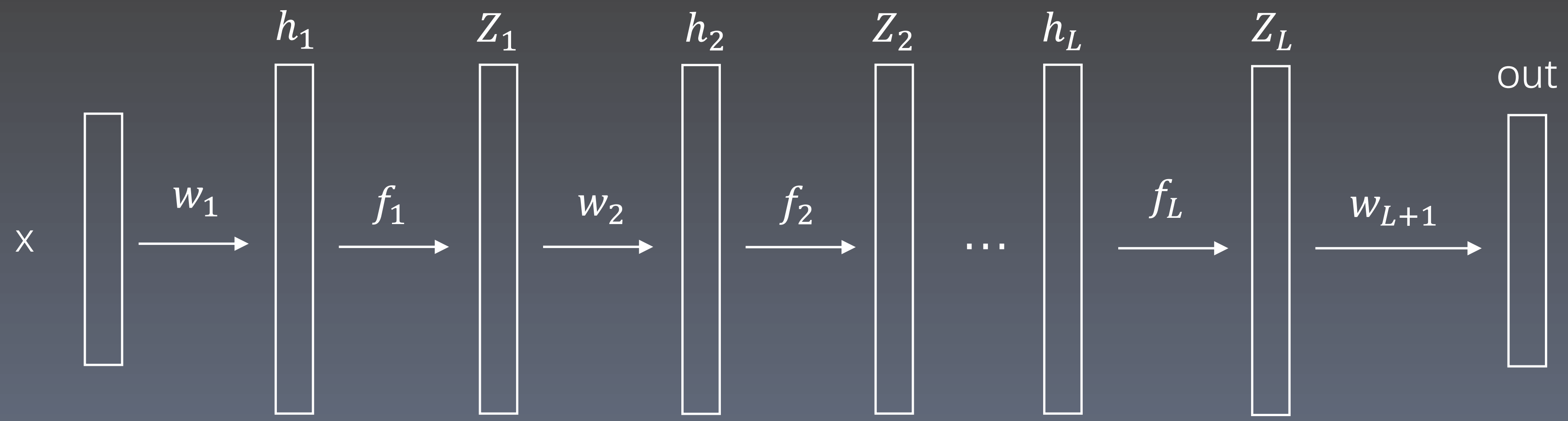


deepshare.net
深度之眼

③f_L为relu

$$Z_L = relu(h_L) = \begin{cases} 0, h_L \leq 0 \\ h_L, h_L > 0 \end{cases}$$

$$\frac{\partial J}{\partial h_L} = \frac{\partial J}{\partial Z_L}\frac{dZ_L}{dh_L} = \begin{cases} 0, h_L \leq 0 \\ \dfrac{\partial J}{\partial Z_L}, h_L > 0 \end{cases}$$



不同算法：

$$
\frac{\partial J}{\partial out} \rightarrow \boxed{\left\{ \begin{aligned}
&\frac{\partial J}{\partial w_{L+1}} = z_L^T \frac{\partial J}{\partial out} \\
&\frac{\partial J}{\partial z_L} = \frac{\partial J}{\partial out} w_{L+1}^T \\
&\left(\frac{\partial J}{\partial b_{L+1}}\right)^T = SumRow\left(\frac{\partial J}{\partial out}\right) \\
&w_{L+1}^{t+1} = w_{L+1}^{t} - \eta \frac{\partial J}{\partial w_{L+1}} \\
&b_{L+1}^{t+1} = b_{L+1}^{t} - \eta \frac{\partial J}{\partial b_{L+1}}
\end{aligned} \right.} \rightarrow \boxed{\frac{\partial J}{\partial h_L}} \rightarrow \boxed{\left\{ \begin{aligned}
&\frac{\partial J}{\partial w_L} = z_L^T \frac{\partial J}{\partial h_L} \\
&\frac{\partial J}{\partial z_{L\text{-}1}} = \frac{\partial J}{\partial h_L} w_L^T \\
&\quad\vdots \\
&\quad\vdots
\end{aligned} \right.} \cdots
$$

deepshare.net

深度之眼

联系我们:

电话: 18001992849

邮箱: service@deepshare.net

Q Q: 2677693114

公众号

客服微信