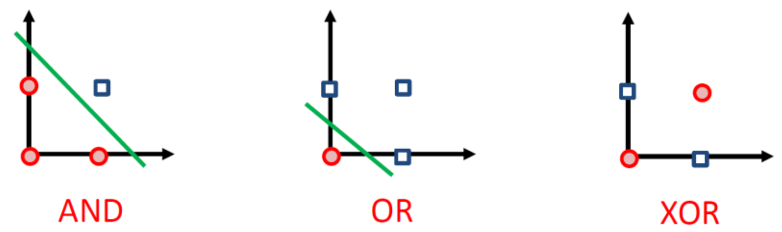
感知器与前馈神经网

主要内容

- 问题的引出
- 多层感知机
  - 神经元
  - 激活函数
- 反向传播算法

问题的引出

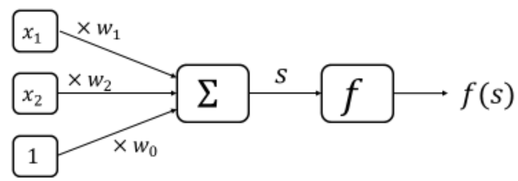
- 线性函数能力不足

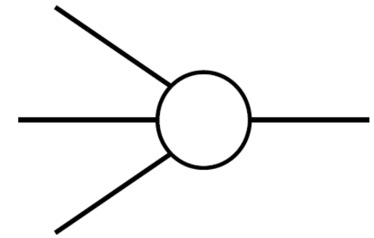


- 将多个线性函数级联是否可行?
  - 思考矩阵乘法!
- 回忆我们处理非线性问题的手段
  - 提升特征维度（显示的特征变换)
  - 利用核函数（隐式的特征变换)
- 线性变换+非线性变换

神经元（neuron）

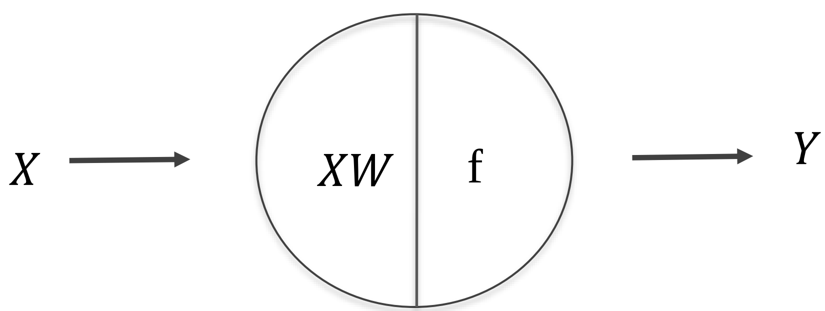
- 感知器模型
  - 对比线性分类器
  - 对比SVM
  - 对比逻辑回归





- x_1, x_2 – inputs
- w_1, w_2 – synaptic weights
- w_0 – bias weight
- f – activation function

-



神经元（向量到标量）

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- 神经元是如下定义的非线性函数

$$y = f(x_1, x_2, \cdots, x_n) = a\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

或者

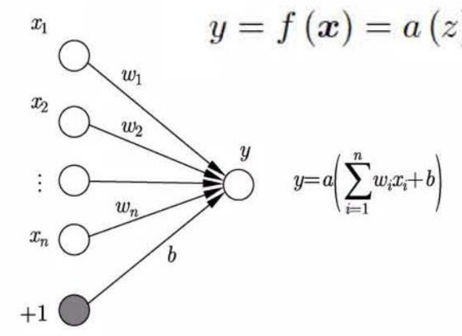
$$y = f\left(x_1, x_2, \cdots, x_n\right) = a\left(z\right), z = \sum_{i=1}^{n} w_i x_i + b$$

z 是净输入，w和b是权重和偏置

a是激活函数，例如Sigmoid函数

$$y = f\left(\boldsymbol{x}\right) = a\left(z\right), \quad z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

$$a\left(z\right) = \frac{1}{1 + \mathrm{e}^{-z}}$$



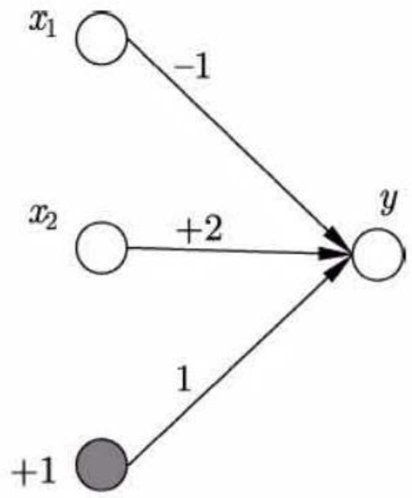
神经元

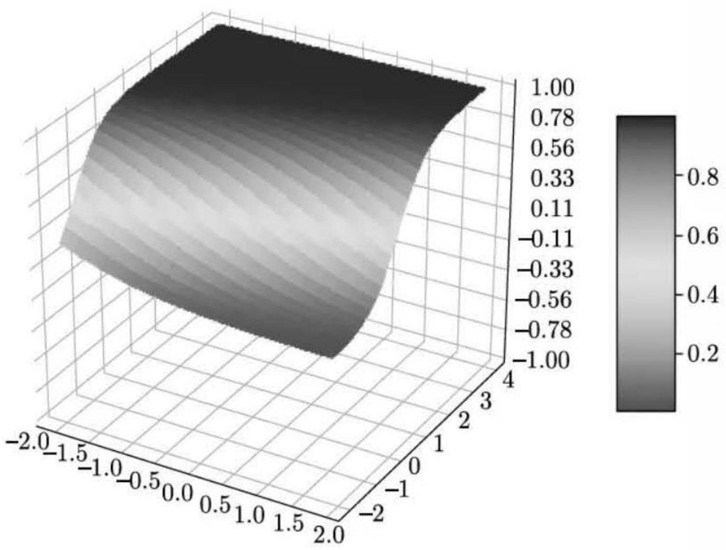
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

$$y = f(\boldsymbol{x}) = a(z), \quad z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

- 例子（利用Sigmoid函数激活）





常用的激活函数

- 阶梯函数

$$f(s)=\begin{cases}1, & if\ s\geq 0\\ 0, & if\ s<0\end{cases}$$

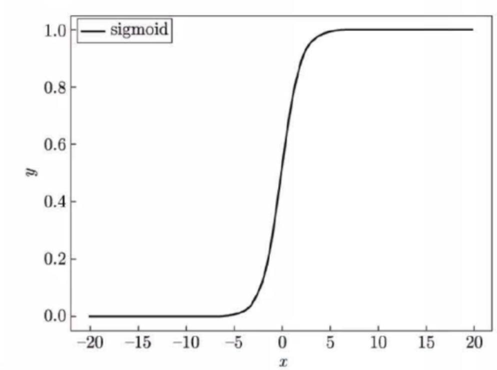
- 符号函数

$$f(s)=\begin{cases}1, & if\ s\geq 0\\ -1, & if\ s<0\end{cases}$$

- Sigmoid函数

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

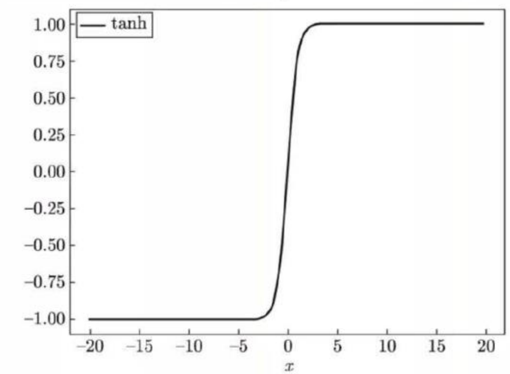
$$a'(z)=a(z)(1-a(z))$$



- 双曲正切函数

$$a(z)=\tanh(z)=\frac{\mathrm{e}^{z}-\mathrm{e}^{-z}}{\mathrm{e}^{z}+\mathrm{e}^{-z}}$$

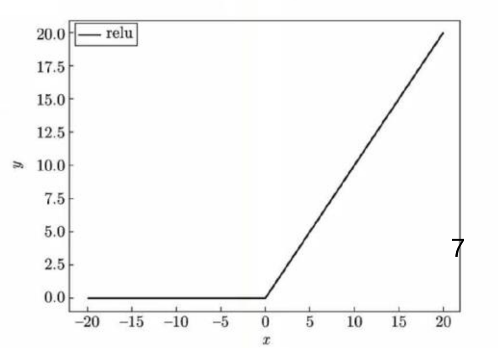
$$a'(z)=1-a(z)^2$$



- 整流线性函数（relu）

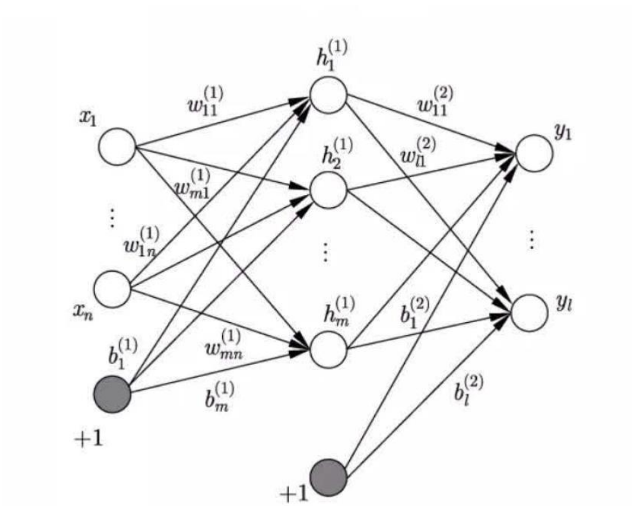
$$a(z)=\mathrm{relu}(z)=\max(0,z)$$

$$a'(z)=\begin{cases}1, & z>0\\ 0, & 其他\end{cases}$$



前馈神经网络

- 由多层神经元组成
- 层间互连、层内不连接
- 前一层神经元的输出是后一层的输入
- 表示输入信号（向量）到输出（向量）非线性变换
  - 矩阵表示



$$\boldsymbol{h}^{(1)} = f^{(1)}(\boldsymbol{x}) = a(\boldsymbol{z}^{(1)}) = a\left(\boldsymbol{W}^{(1)^{\mathrm{T}}}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right)$$

$$\boldsymbol{y} = f^{(2)}\left(\boldsymbol{h}^{(1)}\right) = g(\boldsymbol{z}^{(2)}) = g\left(\boldsymbol{W}^{(2)^{\mathrm{T}}}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right)$$

多层前馈神经网

- 多层前馈网络的矩阵表示

$$
\begin{cases}
\boldsymbol{h}^{(1)} = f^{(1)}(\boldsymbol{x}) = a(\boldsymbol{z}^{(1)}) = a\left(\boldsymbol{W}^{(1)\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) \\
\boldsymbol{h}^{(2)} = f^{(2)}\left(\boldsymbol{h}^{(1)}\right) = a(\boldsymbol{z}^{(2)}) = a\left(\boldsymbol{W}^{(2)\mathrm{T}}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right) \\
\vdots \\
\boldsymbol{h}^{(s-1)} = f^{(s-1)}\left(\boldsymbol{h}^{(s-2)}\right) = a(\boldsymbol{z}^{(s-1)}) = a\left(\boldsymbol{W}^{(s-1)\mathrm{T}}\boldsymbol{h}^{(s-2)} + \boldsymbol{b}^{(s-1)}\right) \\
\boldsymbol{y} = \boldsymbol{h}^{(s)} = f^{(s)}\left(\boldsymbol{h}^{(s-1)}\right) = g(\boldsymbol{z}^{(s)}) = g\left(\boldsymbol{W}^{(s)\mathrm{T}}\boldsymbol{h}^{(s-1)} + \boldsymbol{b}^{(s)}\right)
\end{cases}
$$

- 线性变换+非线性变换
  - 非线性变换是关键
  - 多次线性等同一次线性变换

神经网络常承担的任务

- 回归-输出实数
  - g是恒等函数

$$y = g(z) = z, \quad z = \boldsymbol{w}^{(s)\mathrm{T}} \boldsymbol{h}^{(s-1)} + b^{(s)}$$

- 二类分类-输出类别的后验概率值
  - g是S型函数

$$P(y=1|\boldsymbol{x}) = g(z) = \frac{1}{1+\mathrm{e}^{-z}}, \quad z = \boldsymbol{w}^{(s)\mathrm{T}} \boldsymbol{h}^{(s-1)} + b^{(s)}$$

- 多类分类-输出各类别的后验概率值
  - g是softmax函数

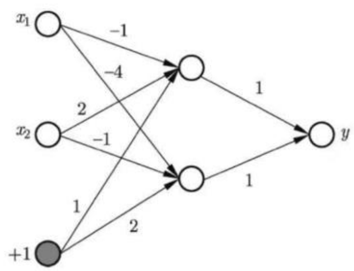
$$P(y_k=1|\boldsymbol{x}) = g(z_k) = \frac{\mathrm{e}^{z_k}}{\sum_{i=1}^{l} \mathrm{e}^{z_i}}, \quad z_k = \boldsymbol{w}_k^{(s)\mathrm{T}} \boldsymbol{h}^{(s-1)} + b_k^{(s)}, \quad k = 1, 2, \cdots, l$$

- 多标签分类-多个二类分类
  - 多个g，g是S型函数

$$P(y_k=1|\boldsymbol{x}) = g(z_k) = \frac{1}{1+\mathrm{e}^{-z_k}}, \quad z_k = \boldsymbol{w}_k^{(s)\mathrm{T}} \boldsymbol{h}^{(s-1)} + b_k^{(s)}, \quad k = 1, 2, \cdots, l$$

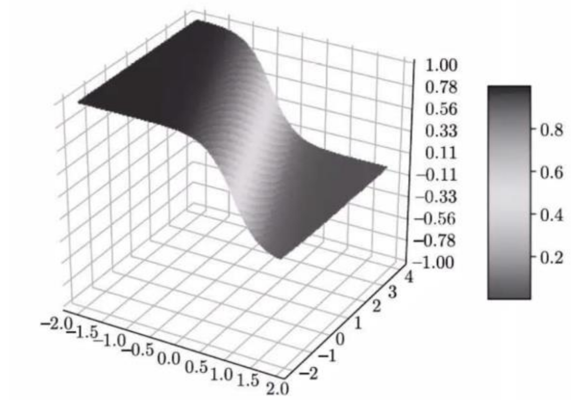
神经网络例子

- 两层神经网络，S型激活函数



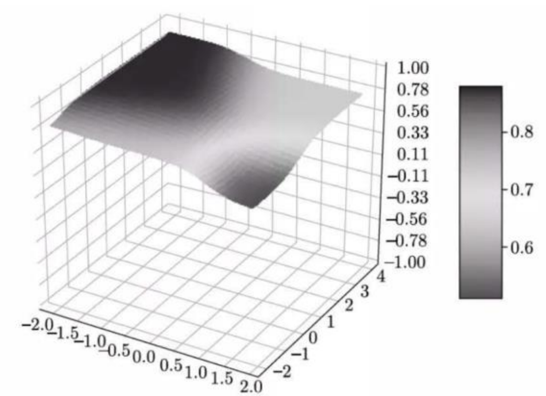
$$h_1^{(1)} = \frac{1}{1+\mathrm{e}^{x_1-2x_2-1}}$$

$$h_2^{(1)} = \frac{1}{1+\mathrm{e}^{4x_1+x_2-2}}$$

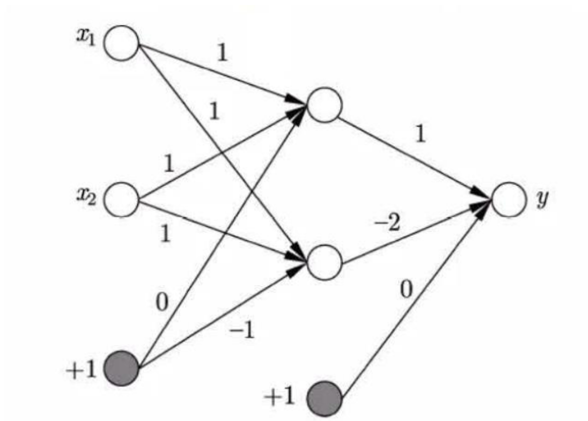


- 第二层输出两类

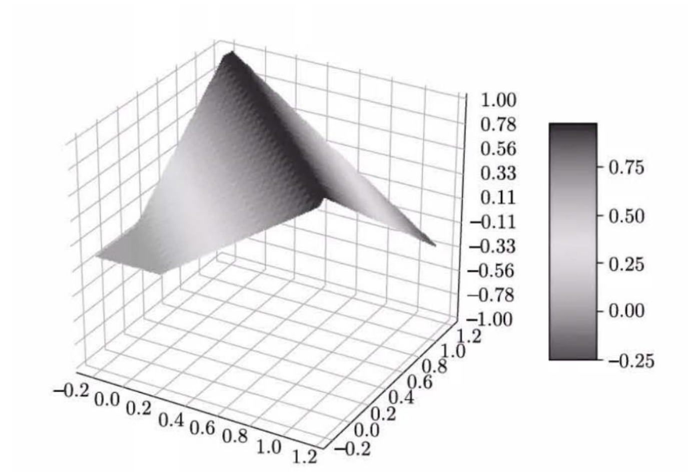
$$f(x_1,x_2) = \sigma\left(\frac{1}{1+\mathrm{e}^{x_1-2x_2-1}} + \frac{1}{1+\mathrm{e}^{4x_1+x_2-2}}\right)$$



XOR问题网络

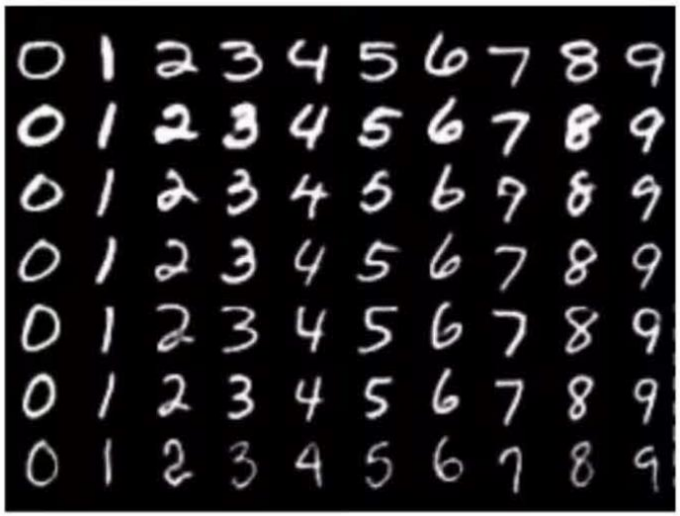


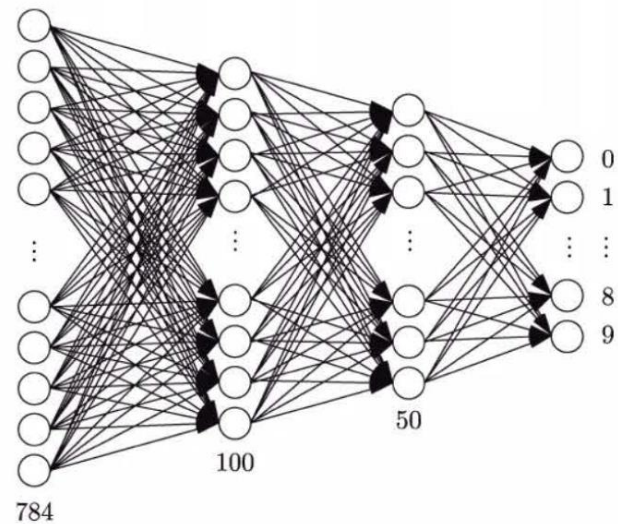
x_1
x_2
+1
+1
1
1
1
1
0
−1
1
−2
0
y



1.00
0.78
0.56
0.33
0.11
−0.11
−0.33
−0.56
−0.78
−1.00
0.75
0.50
0.25
0.00
−0.25
−0.2 0.0 0.2 0.4 0.6 0.8 1.0 1.2
−0.2 0.0 0.2 0.4 0.6 0.8 1.0 1.2

手写数字识别网络





前馈神经网络的表示能力

- 线性模型可以用神经网络表示
  - 多类分类（逻辑回归）

$$P(y_k=1|\boldsymbol{x})=f(\boldsymbol{x})=\frac{e^{(\boldsymbol{w}_k^{(1)\mathrm{T}}\boldsymbol{x}+b_k^{(1)})}}{\sum_{i=1}^{l}e^{(\boldsymbol{w}_i^{(1)\mathrm{T}}\boldsymbol{x}+b_i^{(1)})}},\quad k=1,2,\cdots,l$$

  - 两类分类（很多种网络皆可表示）

$$f(\boldsymbol{x})=\tanh(\boldsymbol{w}^{(1)^{\mathrm{T}}}\boldsymbol{x}+b^{(1)})$$

  - 线性和非线性SVM

$$y=\begin{cases}+1, & \operatorname{sign}(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})+b)\geqslant 0\\ -1, & \text{其他}\end{cases}$$

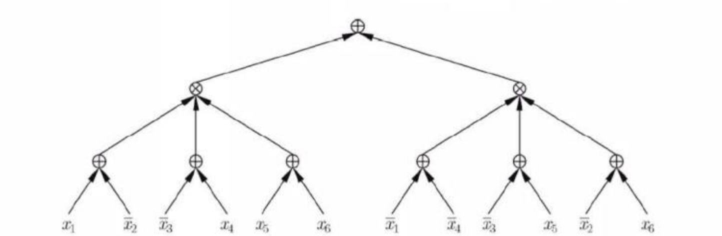
  - PCA

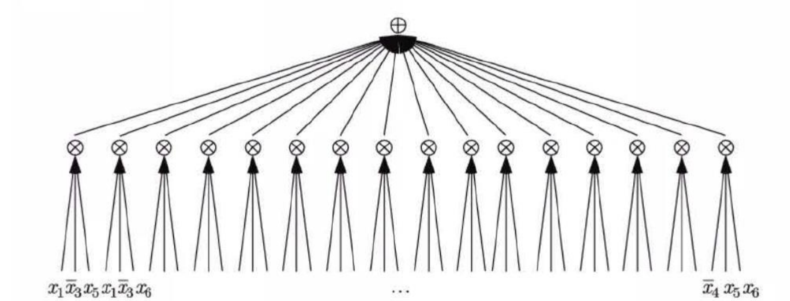
函数等价性及网络深度与广度

- 前馈神经网络y=f(x;θ)有大量等价函数
  - 即参数θ不同，但函数f对相同的输入x，产生相同的y
  - 考虑某个隐层有m个神经元，激活函数是双曲正切函数
    - 存在多少这样等价的参数向量?

- 深度与广度

$$(x_1+\bar{x}_2)(\bar{x}_3+x_4)(x_5+x_6)=x_1\bar{x}_3x_5+x_1\bar{x}_3x_6+x_1x_4x_5+x_1x_4x_6+\bar{x}_2\bar{x}_3x_5+\bar{x}_2\bar{x}_3x_6+\bar{x}_2x_4x_5+\bar{x}_2x_4x_6$$





  - 深度网络往往具有更低的复杂度
  - 更深层次的联系还有待进一步研究

前馈神经网络的参数学习

- 训练数据

$$\mathcal{T}=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_N,y_N)\}$$

- 学习神经网络模型y=f(x;ɵ)中的参数
  - 网络架构确定：层数、每层神经元数、激活函数等

- 学习问题-》优化问题

$$\hat{\boldsymbol{\theta}}=\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\sum_{i=1}^{N}L(f(\boldsymbol{x}_i;\boldsymbol{\theta}),y_i)+\lambda\cdot\Omega(f)\right]$$

- 不考虑正则、损失函数通常与最大似然等价

$$\hat{\boldsymbol{\theta}}=\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\sum_{i=1}^{N}\log P_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}_i)\right]$$

几种任务的对应损失函数

- 回归问题

$$P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) \sim N(f(\boldsymbol{x};\boldsymbol{\theta}), \sigma^2)$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i;\boldsymbol{\theta}))^2 + N \log \sigma + \frac{N}{2} \log 2\pi \right]$$

若方差固定，则等价

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2} (y_i - f(\boldsymbol{x}_i;\boldsymbol{\theta}))^2$$

- 二类分类问题

$$p = P_{\boldsymbol{\theta}}(y=1|\boldsymbol{x}) = f(\boldsymbol{x};\boldsymbol{\theta})$$

交叉熵

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ - \sum_{i=1}^{N} [y_i \log f(\boldsymbol{x};\boldsymbol{\theta}) + (1-y_i) \log(1 - f(\boldsymbol{x};\boldsymbol{\theta}))] \right\}$$

几种任务的对应损失函数

- 多类分类

$$p = P_{\theta}(y_k = 1|x) = f(x;\theta)$$

交叉熵

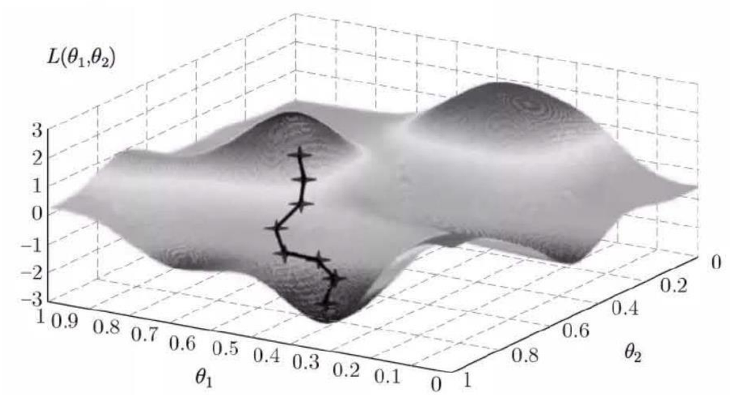
$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\left\{-\sum_{i=1}^{N}\left[\sum_{k=1}^{l} y_{ik}\log f(x;\theta)\right]\right\}$$

前馈神经网络的学习算法

- 非凸优化问题

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \frac{1}{N}\sum_{i=1}^{N} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$$

  - 从函数的等价性上看，最优解不唯一



梯度下降

- 目标函数

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} L_i(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$$

- 损失是各样本损失和
- 若损失函数是参数的连续函数（最好可导）
  - 梯度下降

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

算法 23.1（梯度下降法）

输入：网络架构 $f(\boldsymbol{x};\boldsymbol{\theta})$，训练数据集 T。

输出：神经网络参数向量 $\hat{\boldsymbol{\theta}}$。

超参数：学习率 η。

1. 随机初始化参数向量 $\boldsymbol{\theta}$；
2. Do while($\boldsymbol{\theta}$ 不收敛) {

   针对所有样本，按照以下公式，更新参数向量 $\boldsymbol{\theta}$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

   }
3. 返回学习到的参数向量 $\hat{\boldsymbol{\theta}}$。

随机梯度下降

- 损失函数的梯度实际上是基于训练样本的采样结果

$$\theta \leftarrow \theta - \eta \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i(\theta)}{\partial \theta}$$

  - 效率低（所有训练）
  - 容易陷入局部极优解
- 随机梯度算法
  - 随机打乱样本顺序
  - 将样本分成m组，每一组n个样本

$$\theta \leftarrow \theta - \eta \frac{1}{n}\sum_{j=1}^{n}\frac{\partial L_j(\theta)}{\partial \theta}$$

算法 23.2（随机梯度下降法）

输入：网络架构 $f(\boldsymbol{x};\boldsymbol{\theta})$，训练数据集 $\mathcal{T}$。

输出：神经网络参数向量 $\hat{\boldsymbol{\theta}}$。

超参数：学习率 η，小批量样本容量 n。

1. 随机打乱样本顺序，将样本分成 m 个组，每一组有 n 个样本；
2. 随机初始化参数向量 $\boldsymbol{\theta}$；
3. Do while($\boldsymbol{\theta}$ 不收敛) {

   For ($i = 1, 2, \cdots, m$) {

   针对第 i 个组的 n 个样本，按照以下公式，更新参数向量 $\boldsymbol{\theta}$

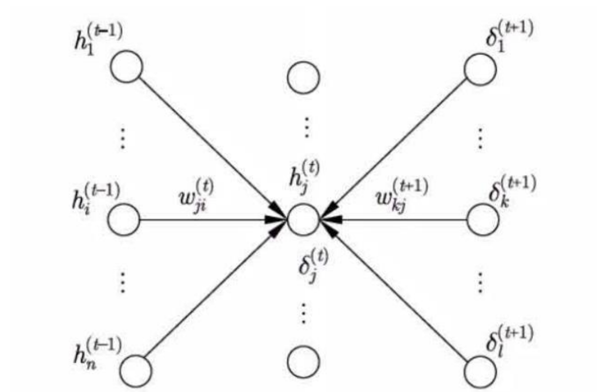
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{1}{n}\sum_{j=1}^{n}\frac{\partial L_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

   }

   }
4. 返回学习到的参数向量 $\hat{\boldsymbol{\theta}}$。

反向传播算法

- 参数学习问题的关键在于梯度的计算
  - 符合函数-》链式法则
  - 层次结构-》递归算法



- 考虑s层神经网络，第t层神经元

$$h_j^{(t)} = a\left(z_j^{(t)}\right), \quad j = 1, 2, \cdots, m$$

$$z_j^{(t)} = \sum_{i=1}^{n} w_{ji}^{(t)} h_i^{(t-1)} + b_j^{(t)}$$

- 第t+1层

$$h_k^{(t+1)} = a\left(z_k^{(t+1)}\right), \quad k = 1, 2, \cdots, l$$

$$z_k^{(t+1)} = \sum_{j=1}^{m} w_{kj}^{(t+1)} h_j^{(t)} + b_k^{(t+1)}$$

反向传播算法

- 计算损失函数对所有参数的梯度
  - 根据链式法则，可知

$$\frac{\partial L}{\partial w_{ji}^{(t)}} = \frac{\partial L}{\partial z_j^{(t)}} \frac{\partial z_j^{(t)}}{\partial w_{ji}^{(t)}}$$

$$\frac{\partial L}{\partial b_j^{(t)}} = \frac{\partial L}{\partial z_j^{(t)}} \frac{\partial z_j^{(t)}}{\partial b_j^{(t)}}$$

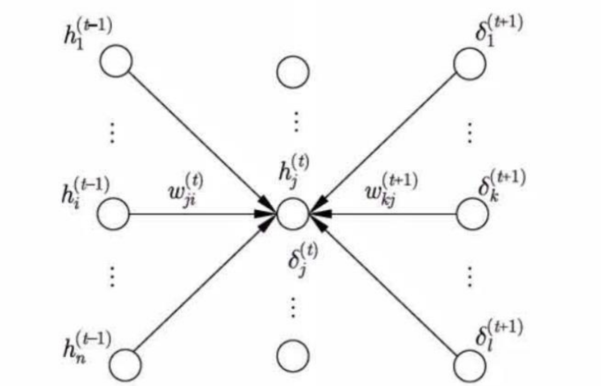
  - 考虑损失函数对第t层净输入的梯度

$$\delta_j^{(t)} = \frac{\partial L}{\partial z_j^{(t)}}, \quad j = 1, 2, \cdots, m$$

称为第t层"误差"，因此有

$$\frac{\partial L}{\partial w_{ji}^{(t)}} = \delta_j^{(t)} h_i^{(t-1)}$$

$$\frac{\partial L}{\partial b_j^{(t)}} = \delta_j^{(t)}$$



反向传播算法

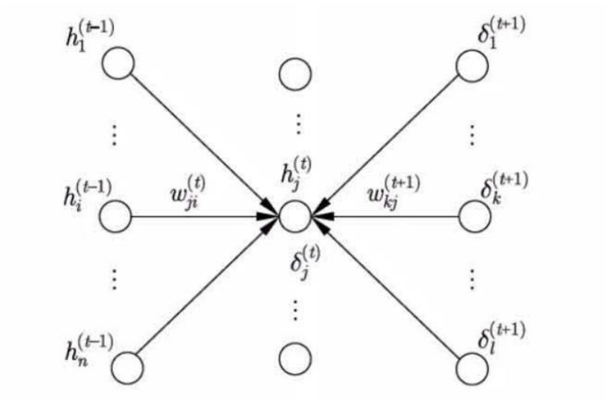
- 考察相邻两层间的“误差”关系

$$\delta_j^{(t)} = \frac{\partial L}{\partial z_j^{(t)}} = \sum_{k=1}^{l} \frac{\partial L}{\partial z_k^{(t+1)}} \frac{\partial z_k^{(t+1)}}{\partial z_j^{(t)}}, \quad j = 1, 2, \cdots, m$$

- 有

$$\delta_j^{(t)} = \frac{\mathrm{d}a}{\mathrm{d}z_j^{(t)}} \sum_{k=1}^{l} w_{kj}^{(t+1)} \delta_k^{(t+1)}$$

  - 导出了层间“误差”间的递归关系

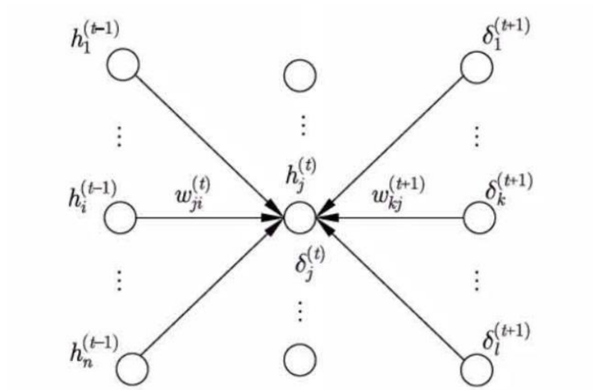


反向传播算法

- 递归的求解关键在“出口”，对于神经网络的出口是“输出层”
  - 第s层（输出层）的误差为

$$\delta_k^{(s)} = \frac{\partial L}{\partial z_k^{(s)}}, \quad k = 1, 2, \cdots, l$$

  - 根据链式法则有



$$\delta_k^{(s)} = \frac{\mathrm{d}g}{\mathrm{d}z_k^{(s)}} \frac{\partial L}{\partial h_k^{(s)}}, \quad k = 1, 2, \cdots, l$$

- 针对常见的两种任务：回归和聚类
  - 回归：损失函数

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2} \left(y_i - f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right)^2$$

  - 激活函数为恒等函数

$$\boxed{\delta^{(s)} = h^{(s)} - y}$$

反向传播算法

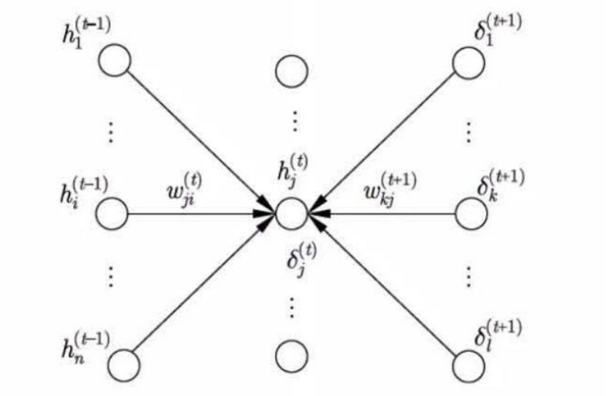
- 针对常见的两种任务：回归和聚类
  - 聚类：多类分类问题

    损失函数：交叉熵

    $$-\sum_{k=1}^{l} y_k \log h_k^{(s)}$$

    激活函数为

    $$g(z) = \frac{e^z}{\sum_{z'} e^{z'}}$$



$$\boxed{\delta_k^{(s)} = h_k^{(s)} - y_k, \quad k = 1, 2, \cdots, l}$$

- 无论“回归”还是“分类”任务，损失函数对最后一层的神经元的净输入的导数是都是神经元输入与目标（label）之间的差值

反向传播算法

算法 23.3 （前馈神经网络的反向传播算法）

输入：神经网络 $f(\boldsymbol{x};\boldsymbol{\theta})$，参数向量 $\boldsymbol{\theta}$，一个样本 $(\boldsymbol{x},\boldsymbol{y})$。

输出：更新的参数向量 $\boldsymbol{\theta}$。

超参数：学习率 η。

{

1. 正向传播，得到各层输出 $\boldsymbol{h}^{(1)},\boldsymbol{h}^{(2)},\cdots,\boldsymbol{h}^{(s)}$

$$\boldsymbol{h}^{(0)} = \boldsymbol{x}$$

For $t = 1,2,\cdots,s$, do{

$$\boldsymbol{z}^{(t)} = \boldsymbol{W}^{(t)}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}^{(t)}$$

$$\boldsymbol{h}^{(t)} = a(\boldsymbol{z}^{(t)})$$

}

$$f(\boldsymbol{x}) = \boldsymbol{h}^{(s)}$$

2. 反向传播，得到各层误差 $\boldsymbol{\delta}^{(s)},\cdots,\boldsymbol{\delta}^{(2)},\boldsymbol{\delta}^{(1)}$，同时计算各层的梯度，更新各层的参数。

计算输出层的误差

$$\boldsymbol{\delta}^{(s)} = \boldsymbol{h}^{(s)} - \boldsymbol{y}$$

For $t = s,\cdots,2,1$, do{

反向传播算法

计算第 t 层的梯度

$$\nabla_{\boldsymbol{W}^{(t)}} L = \boldsymbol{\delta}^{(t)} \cdot \boldsymbol{h}^{(t-1)^{\mathrm{T}}}$$

$$\nabla_{\boldsymbol{b}^{(t)}} L = \boldsymbol{\delta}^{(t)}$$

根据梯度下降公式更新第 t 层的参数

$$\boldsymbol{W}^{(t)} \leftarrow \boldsymbol{W}^{(t)} - \eta \nabla_{\boldsymbol{W}^{(t)}} L$$

$$\boldsymbol{b}^{(t)} \leftarrow \boldsymbol{b}^{(t)} - \eta \nabla_{\boldsymbol{b}^{(t)}} L$$

If $(t > 1)$ {

将第 t 层的误差传到第 $t-1$ 层

$$\boldsymbol{\delta}^{(t-1)} = \frac{\partial a}{\partial \boldsymbol{z}^{(t-1)}} \odot \left(\boldsymbol{W}^{(t)^{\mathrm{T}}} \cdot \boldsymbol{\delta}^{(t)} \right)$$

}

}

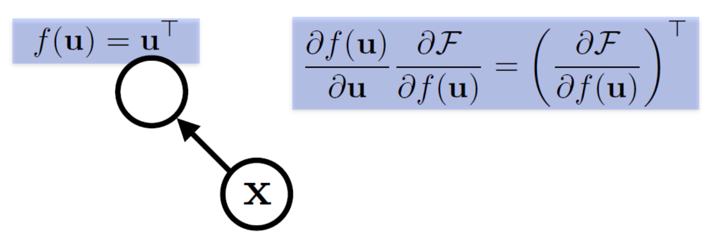
3. 返回更新的参数向量

}

神经网络的实现问题

- 神经网络计算特点与实现要求
  - 数据多（样本和参数）
  - 计算过程繁杂，需要记录数据间的计算关系
  - 需要明了地表示正向计算和反向的梯度计算
  - 因此需要有效组织数据，明确计算关系

- 计算图（有向图）
  - 计算图是一种数据结构、描述计算过程
  - 节点-表示标量、向量、矩阵、张量
  - 边-表示函数依赖（一条边代表一个函数变量）
  - 非叶节点-是沿着入边进入的节点的函数；该函数明确，因此正向和反向计算明晰

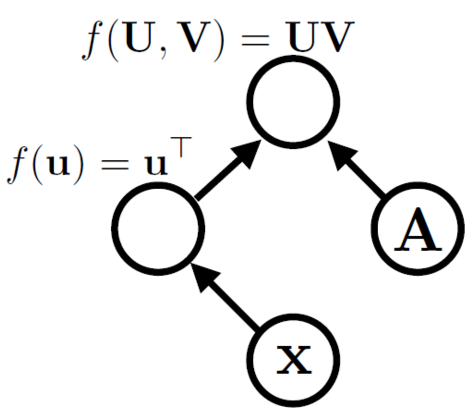


计算图简例

- 表达如下公式

$$\mathbf{x}^\top \mathbf{A}$$

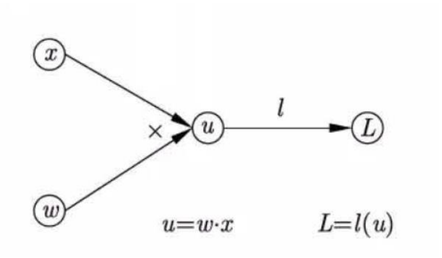
- 其计算图如下
  - 每个节点对应的函数可以是无变量、单变量、双变量和多变量
  - 通常用到的是单变量和双变量

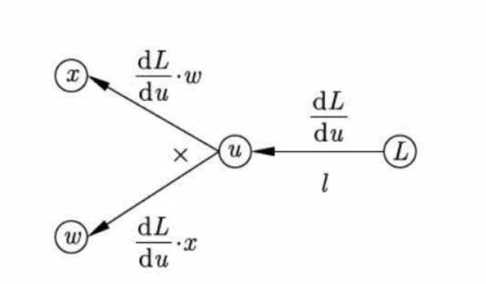


利用计算图解决神经网计算例子

- 正向计算：从起点开始，沿着有向边进行
- 反向计算：从终点开始计算梯度，逆着有向边进行

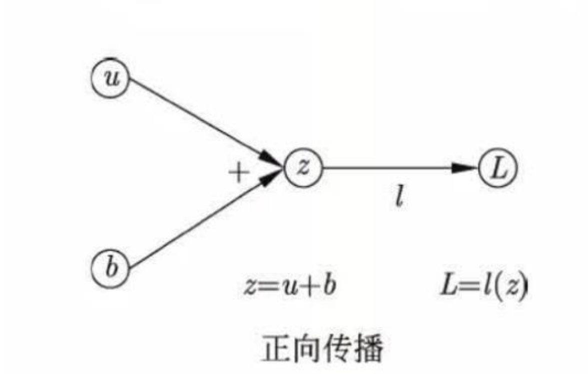
- 乘法

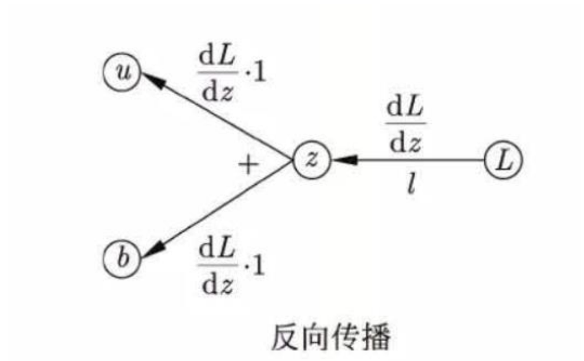




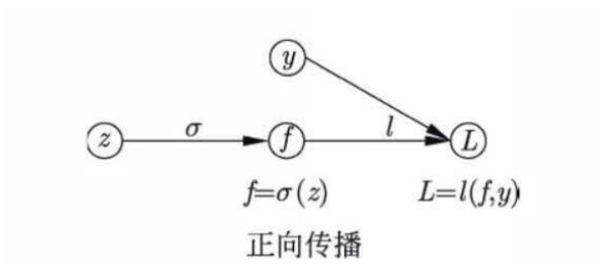
利用计算图解决神经网计算例子

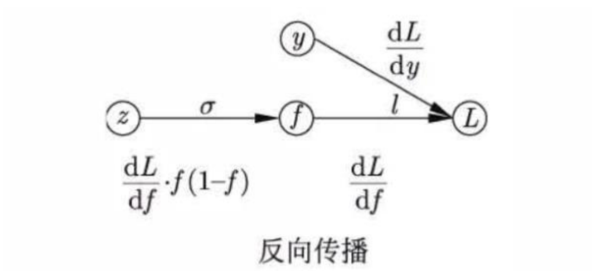
- 加法





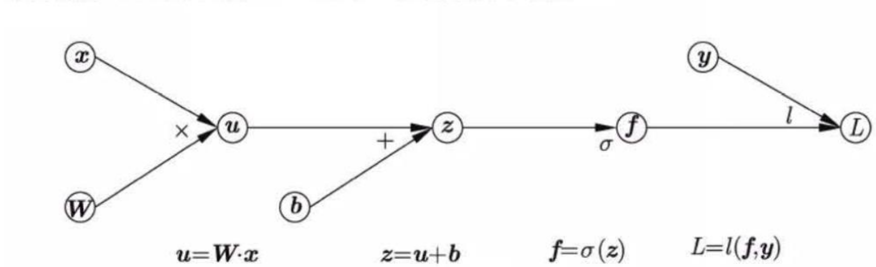
- S型函数



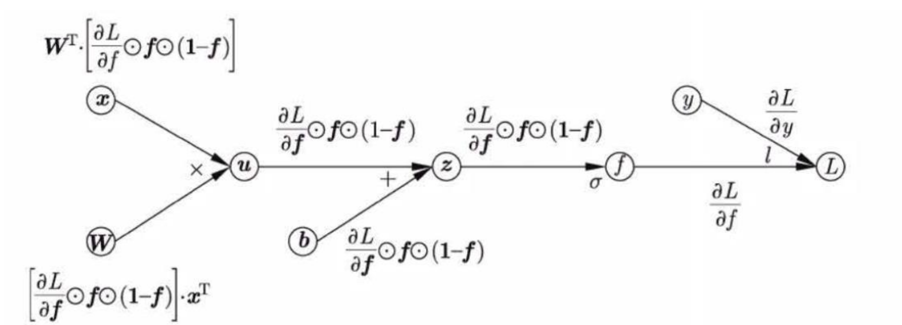


利用计算图解决神经网计算例子

- 一层神经网络的学习，x和y表示



正向传播



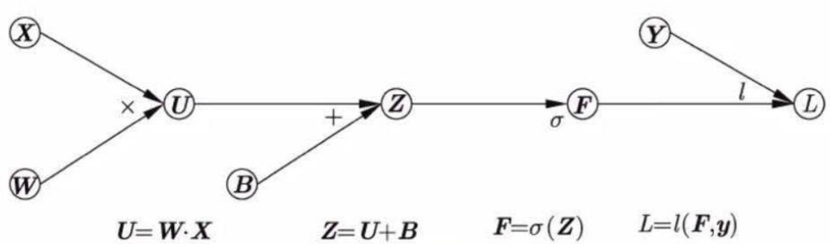
反向传播

利用计算图解决神经网计算例子

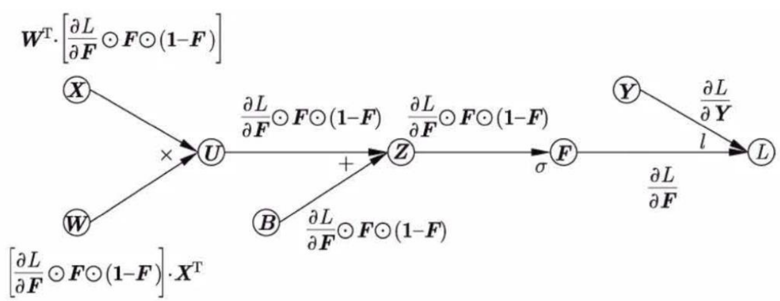
- 一层神经网络的小批量样本学习

$$F = \sigma(W \cdot X + B)$$

$$L = l(F, Y)$$



正向传播



反向传播

算法实现中的问题

- 梯度消失和梯度爆炸
  - 通常由于层数过多、前面几层的梯度有时会接近于0（消失）或者接近无穷（爆炸），无法使程序运行下去
  - 分析

（1）反向传播，首先计算误差向量

$$\boldsymbol{\delta}^{(t-1)} = \left\{ \operatorname{diag}\left(\frac{\partial a}{\partial \boldsymbol{z}^{(t-1)}} \right) \cdot \boldsymbol{W}^{(t)\mathrm{T}} \right\} \cdot \boldsymbol{\delta}^{(t)}$$

之后计算梯度

$$\begin{cases} \nabla_{\boldsymbol{W}^{(t)}} L = \boldsymbol{\delta}^{(t)} \cdot \boldsymbol{h}^{(t-1)\mathrm{T}} \\ \nabla_{\boldsymbol{b}^{(t)}} L = \boldsymbol{\delta}^{(t)} \end{cases}$$

误差间的关系可以写成

$$\boldsymbol{\delta}^{(t-1)} = \boldsymbol{U} \cdot \boldsymbol{\delta}^{(t)} \qquad \boldsymbol{U} = \boldsymbol{V} \cdot \operatorname{diag}(\boldsymbol{\lambda}) \cdot \boldsymbol{V}^{-1}$$

特征值分解（或SVD分解），根据特征值连乘后可以为0或者接近无穷

（2）激活函数

$$\boldsymbol{\delta}^{(t-1)} = \left\{ \boxed{\operatorname{diag}\left(\frac{\partial a}{\partial \boldsymbol{z}^{(t-1)}} \right)} \cdot \boldsymbol{W}^{(t)\mathrm{T}} \right\} \cdot \boldsymbol{\delta}^{(t)}$$

算法实现中的问题

- 批量归一化

$$\delta^{(t-1)} = \left\{ \boxed{\mathrm{diag}\left(\frac{\partial a}{\partial z^{(t-1)}}\right)} \cdot W^{(t)^{\mathrm{T}}} \right\} \cdot \delta^{(t)}$$

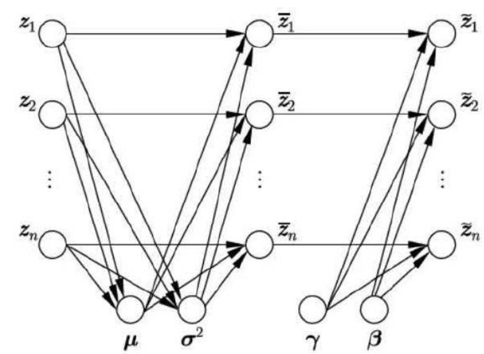
  - 在每个批量样本上，对前馈神经网络的每一层（非输出）的净输入或者输入进行归一化，然后再训练神经网络
  - 当前发现：如果输入向量每一维的值都在(-1,1)之间，那么

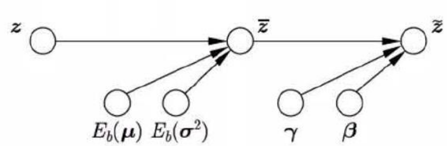
$$\mu = \frac{1}{n}\sum_{j=1}^{n} z_j$$

$$\sigma^2 = \frac{1}{n-1}\sum_{j=1}^{n}(z_j - \mu)^2$$

$$\bar{z}_j = \frac{z_j - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad j = 1, 2, \cdots, n$$

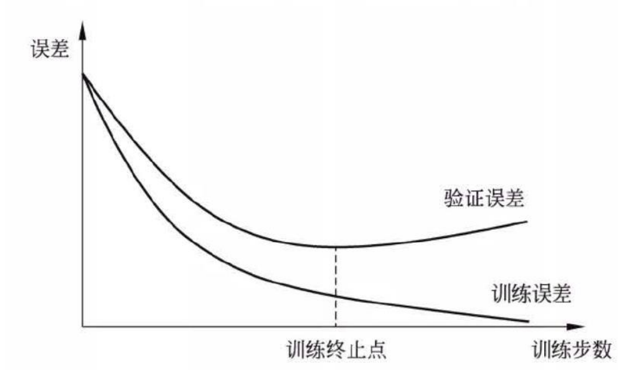
$$\tilde{z}_j = \gamma \odot \bar{z}_j + \beta, \quad j = 1, 2, \cdots, n$$





前馈网络学习的正则化

- 早停法-将数据分为训练集、验证集和测试集
  - 对比决策树构建时的提前结束方案



前馈网络学习的正则化

- 暂退法（dropout）-每一层的神经元