深度学习第六章

李建扣

2016 年 12 月 19 日

深度前馈网络

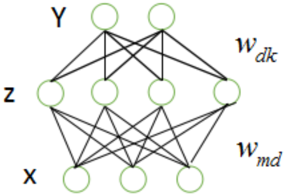
- 简介
- 实例：学习XOR
- 基于梯度的学习
- 网络设计
- 反向传播

- 深度前馈网络、前馈神经网络、多层感知机
- 前馈神经网络通常由不同函数复合在一起，由输入层、隐层、输出层构成

$$f(x) = f_3(f_2(f_1(x))) = \sigma(w_3^T\sigma(w_2^T\sigma(w_1^Tx))) \tag{1}$$

- 卷积神经网络是一种特殊的前馈网络，循环网络的基础也是前馈网络
- 构建机器学习算法：建模、损失函数、优化过程

深度前馈网络-简介



- 层表示向量到向量的函数，由许多并行的操作单元组成
- Logistic 回归和线性回归：线性、凸问题，$f(x) = w^T\phi(x)$
- 深度模型，$f(x) = f(x, \theta, w) = w^T\phi(x, \theta)$

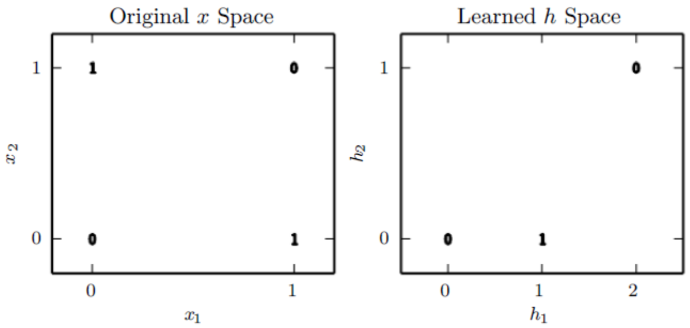


Figure: Deep learning 图6.1

深度前馈网络-学习XOR

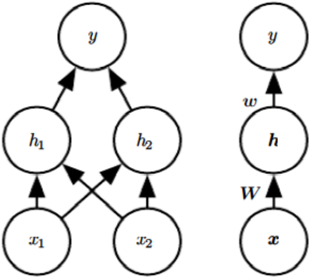


Figure: Deep learning 图6.2

深度前馈网络-学习XOR

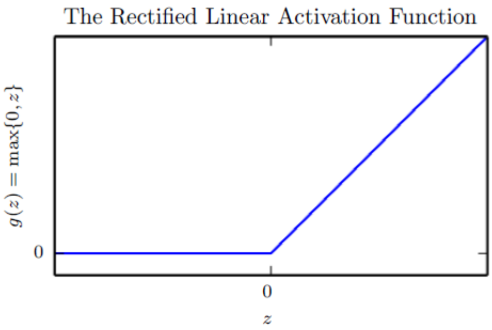


Figure: Deep learning 图6.3

深度前馈网络-学习XOR

$$f(x,\theta) = w^T max\{0, W^T x + c\} \tag{2}$$

其中

$$W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{3}$$

$$c^T = \begin{bmatrix} 0 & -1 \end{bmatrix} \tag{4}$$

$$w^T = \begin{bmatrix} 1 & -2 \end{bmatrix} \tag{5}$$

深度前馈网络-学习XOR

- 仿射变换

$$XW+C=\begin{bmatrix}0&0\\0&1\\1&0\\1&1\end{bmatrix}\begin{bmatrix}1&1\\1&1\end{bmatrix}+\begin{bmatrix}0&-1\\0&-1\\0&-1\\0&-1\end{bmatrix}=\begin{bmatrix}0&-1\\1&0\\1&0\\2&1\end{bmatrix} \tag{6}$$

- 非线性变换

$$\begin{bmatrix}0&0\\1&0\\1&0\\2&1\end{bmatrix} \tag{7}$$

深度前馈网络-学习XOR

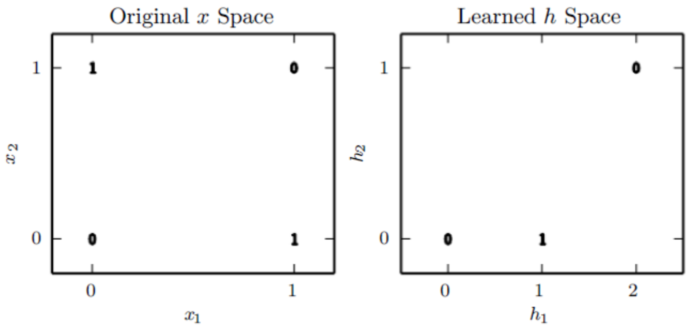
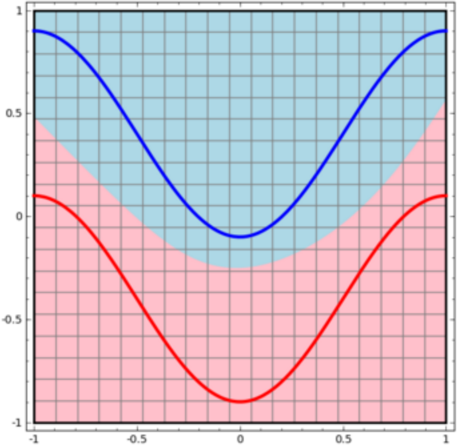


Figure: Deep learning 图6.1

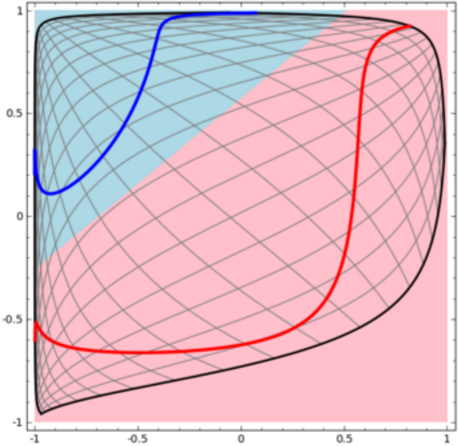
深度前馈网络-学习XOR

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{8}$$

深度前馈网络-实例



深度前馈网络-实例

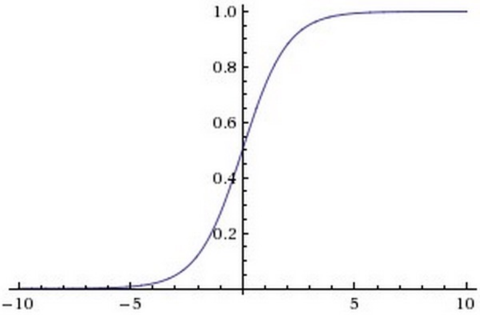


深度前馈网络-基于梯度的学习

- 似然函数与损失函数

$$J(\theta) = -\frac{1}{N}\sum_{n=1}^{N} \log p_{model}(y_n|x_n) = -E_{p_{data}} \log p_{model}(y|x) \quad (9)$$

- 交差熵：任何一个由负对数似然函数组成的损失都是定义在训练集上的经验分布和定义在模型上的概率分布之间的交叉熵



- 饱和：变得非常平
- 损失函数梯度必须足够大和有足够的预测性

深度前馈网络-输出层

- 回归-Gaussian-线性单元
- 二分类-Bernoulli-sigmoid单元
- 多分类-Multinoulli-softmax单元

深度前馈网络-输出层-线性单元

- 输出层

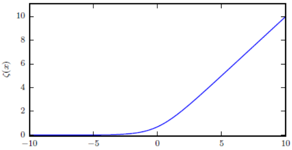
$$y = W^T h + b \tag{10}$$

- 输出层被用来产生条件高斯分布的均值

$$p(t|x) = N(t; y, I) \tag{11}$$

- 最大化对数似然函数等价于最小化均方误差

深度前馈网络-输出层-Sigmoid单元



- 似然函数：$p(y) = \sigma(z)^y(1-\sigma(z))^{1-y} = \sigma((2y-1)z)$
- 损失函数：$J(\theta) = -\log p(y|x) = \zeta((1-2y)z)$
- 当模型得到正确答案时，softplus饱和；
  当答案错误时，$(1-2y)z = |z|$,梯度不收缩；
- 考虑均方误差，损失函数会在任何$\sigma(z)$饱和时饱和，
  $\sigma' = \sigma(1-\sigma)$；

深度前馈网络-输出层-Softmax单元

- 线性层

$$z = W^T h + b \tag{12}$$

- Softmax函数最常用作分多类器的输出；

$$s(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)} \tag{13}$$

$$p(y|x) = \prod_{i=1}^{K} s(z)_i^{y_i} \tag{14}$$

$$\log p(y|x) = z_i - \log \sum_j exp(z_j) \approx z_i - \max_j z_j \tag{15}$$

其中$y_i = 1$

深度前馈网络-隐单元-修正线性单元

- 隐单元：接受输入向量x，计算仿射变换$z = W^T x + b$，然后使用一个作用于每个元素的非线性函数$g(z)$
- 修正线性单元(relu)，隐单元的默认选择

$$g(z) = max(0, z) \tag{16}$$

- 扩展原则：行为更接近线性、模型更容易优化
  - 绝对值修正：$g(z) = |z|$
  - 渗漏线性单元：$g(z) = max(0, z) + 0.01min(0, z)$
  - 参数化修改线性单元：$g(z) = max(0, z) + \alpha min(0, z)$

深度前馈网络-隐单元-Sigmoid

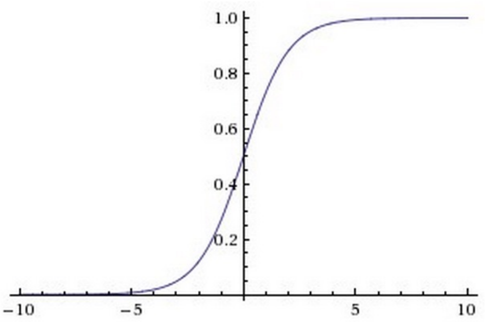


Figure: $\sigma(x) = 1/(1+e^{-x})$，将实数映射到[0,1]区间。主要缺点:在大部分定义域是饱和的，仅当z接近0时对输入敏感，不鼓励用作隐单元，如果使用合适的损失函数抵消饱和性，可以作为输出单元;

深度前馈网络-隐单元-双曲正切函数

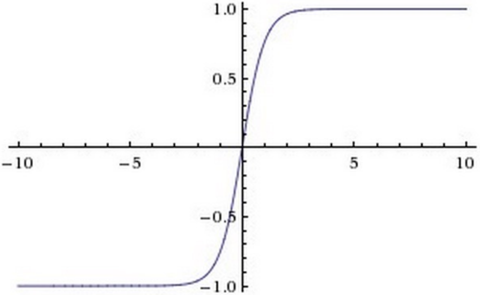


Figure: $tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$，将实数映射到[-1,1]区间，缺点也是当$x$过大或者过小时饱和，通常比sigmoid 函数表现更好，在0附近接近线性模型，$y = w^T \tanh(u^T \tanh(v^T x))$与$y = w^T u^T v^T x$

深度前馈网络-隐单元-其它

- 线性激活函数
- softmax：可以用作一种开关
- softplus：$g(a) = \log(1 + e^a)$，不建议用

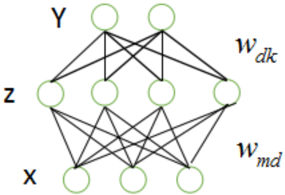
深度前馈网络-结构设计

- 结构：深度，宽度，如何连接
- 普遍近似定理：无论我们试图学习什么函数，一个大的前馈神经网络一定能够表示这个函数
  1. 优化算法可能学不到期望的参数值
  2. 过拟合
- 使用深度的模型能够减少函数所需的单元的数量

深度前馈网络-结构设计

- 主链之外添加额外的属性，例如从第i层到第$i+2$层的连接
- 层与层之间，输入层中的每个单元仅连接到输出层单元的一个子集
- 反馈信息

深度前馈网络-反向传播



- 主要思想：复合函数求导的链式法则

深度前馈网络-反向传播

- E表示平方损失函数、a表示上一层经过仿射变换后的输出、h表示a经过激活函数对应的输出、w表示权重、i表示输入层下标、j表示隐层下标、k表示输出层下标
- 偏导数

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i \tag{17}$$

其中

$$\delta_j = \frac{\partial E_n}{\partial a_j} \qquad z_i = \frac{\partial a_j}{\partial w_{ji}} \tag{18}$$

- 反向传播公式

$$
\begin{aligned}
\delta_j &= \frac{\partial E_n}{\partial a_j} \\
&= \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j} \\
&= \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial z_j}\frac{\partial z_j}{\partial a_j} \\
&= h'(a_j)\sum_k w_{kj}\delta_k
\end{aligned} \tag{19}
$$

深度前馈网络-反向传播

1. 初始化参数w；
2. 前向计算所有激活函数的输入和输出值；
3. 根据公式(20)反向传播δ得到每个神经元的δ；
4. 根据公式(21)计算偏导数；

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{20}$$

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{21}$$

深度前馈网络-反向传播

- 损失函数梯度
- Jacobian矩阵
- Hessian矩阵

小结

- 负对数损失函数
- 输出层设计
- 隐层设计
- 结构设计
- 反向传播

谢谢大家!