# NN提要

## 对比线性模型

线性模型下，回归和分类问题都可以表达为：

$y(\mathbf{x}, \mathbf{w}) = f(\sum_{j=1}^{M} w_j \phi_j(x))$

其中，$f(\cdot)$ 是非线性激活函数，$\phi_j(x)$ 为基函数（basis function）。

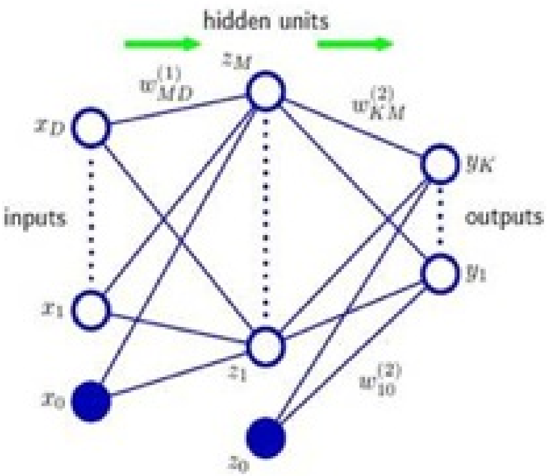
因为$\phi$ 的**形式是固定**的，所以容量和灵活性受限；

神经网络：

$\phi$ 的**形式可变**，具体由通过训练后的w决定。所以NN又称万能函数拟合器。

## 基础形式

以一个三层网络作为说明：




假设输出层是sigmoid，那么：

$y_k(\mathbf{x}, \mathbf{w}) = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)})$

其中所有的w都需要在训练过程中不断修正。

从某种角度理解，**w的值决定了非线性基函数$\phi$ 的形式。**

## 输出层和损失函数

输出层的选择需要根据任务的具体定义决定，例如：

二分类： sigmoid

多分类： softmax

回归： identity

而，不同的输出层的形式，选用的损失函数的形式一般也略有区别，

“损失”可以理解为模型当前的预测值和实际值的偏离程度，常见的有：

欧式距离： $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}||y(\mathbf{x_n},\mathbf{w}) - \mathbf{t_n}||^2$

交叉熵（负对数似然）： $E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn}\, ln y_k(\mathbf{x_n},\mathbf{w})$

**典型的，欧式距离适用于假设噪声符合高斯分布的回归问题；交叉熵适用于衡量任意概率分布的相似性，例如用来评价多分类问题的性能。**

## 训练

训练的目的是**找到合理的w**，使得w表示的函数能够比较好地体现数据集的映射关系，并且期望能够具有较好的泛化性能。

损失函数E（w）度量的是模型的预测值和实际值的偏离程度，那么，目标就是找到使得E（w）尽可能小的w。

由于神经网络的高度非线性，它的形式常常是**非凸的，没有解析解。**

梯度下降法作为一种常见的优化方法，可用作神经网络的训练：

$w^{\tau+1} = w^{\tau} - \eta \bigtriangledown E(w^{\tau})$

其中$\eta$ 是学习率。

### 反向传播算法

为了方便描述，首先把 $y_k(\mathbf{x},\mathbf{w}) = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)})$

中的bias $w_{j0}$ 进行合并，令$a_j = \sum_i w_{ji} z_i$。

其中，对于输入层，$z_i = x_i$，对于隐藏层和输出层，z是上一层的输出，即$z_j = h(a_j)$

链式法则：$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}$

令 $\delta_j = \frac{\partial E_n}{\partial a_j}$，

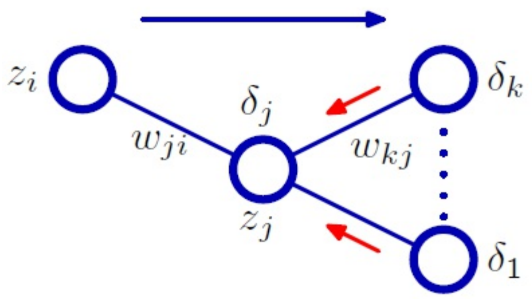
而 $\frac{\partial a_j}{\partial w_{ji}} = z_i$

那么 $\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$（1）。

因为$z_i$已知，那么关键就是如何求解每一层的$\delta_j$

对于输出层，$\delta_j = \sigma'(a_j)E'(w)$（2）

对于隐藏层和输入层，需要从后往前逐层推导：




$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j}$

其中，$\frac{\partial E_n}{\partial a_k} = \delta_k$，$\frac{\partial a_k}{\partial a_j} = h'(a_j)w_{kj}$

所以，得到递推式：

$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$ **（3）**

**结合（1）（2）（3）就可以逐层向前计算得到各个w的偏导，取反即为梯度。**