1 神经网络

1.1 神经网络概述

1.1.1 适应数据的可变基

前面章节中的回归方法在数学上可以描述为

$$y(\boldsymbol{x}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})\right)$$

基函数 $\phi_j(x)$ 的选择是固定的, 当数据维度较高时会导致基函数的数目过于庞大, 导致维度灾难.

为了解决这一问题, 可以引入适应基的概念, 即固定基函数的数目, 但允许基函数本身随训练数据进行调整.

定义 1.1 适应基 形如 $\phi_j(\boldsymbol{x}, \boldsymbol{p}_j)$ 的基函数被称作**适应基 (Adaptive Basis Functions)**, 它不仅依赖于输入变量 $\boldsymbol{x}$, 还依赖于一组可调参数 $\boldsymbol{p}_j$. 通过调整这些参数, 可以使基函数更好地适应训练数据.

1.1.2 神经网络的数学模型

神经网络可以被描述为一个 $N+1$ 层的网络, 每层包含若干个节点 (神经元). $n=1$ 的层表示输入层, 节点数目即输入 $\boldsymbol{x}$ 的分量数目 (有时会额外增加一个为 1 的分量表示截距项), $n=N+1$ 的层表示输出层, 节点数目即输出 $\boldsymbol{y}$ 的分量数目; 中间的层称为隐藏层. 每个节点包含刺激值 $z_i^{(k)}$ 和响应值 $a_i^{(k)}$.

神经网络的迭代方式如下: 对于第 $k+1$ 层的每个节点 j, 它的刺激值 $z_j^{(k+1)}$ 由上一层的所有节点的响应值 $a_i^{(k)}$ 加权得到, 然后通过一个激活函数 $h(\cdot)$ 得到响应值 $a_j^{(k+1)}$, 即

$$z_j^{(k+1)} = \sum_i w_{ij}^{(k)} a_i^{(k)}, \quad a_j^{(k+1)} = h\left(z_j^{(k+1)}\right)$$

其中 $w_{ij}^{(k)}$ 表示第 k 层的节点 i 到第 $k+1$ 层的节点 j 的连接权重.

神经网络的功能依赖于激活函数 $h(\cdot)$ 的选择, 常用的激活函数包括 Sigmoid 函数, ReLU 函数和 tanh 函数等. 通过一定的办法求出连接权重 $w_{ij}^{(k)}$ 后, 神经网络就可以对输入数据进行映射, 完成各种任务.

可以看出, 神经网络的每一层都相当于一次逻辑回归, 因此神经网络可以看作是多层串联的逻辑回归模型. 大多数神经网络都具有多个隐藏层. 深度学习主要指的就是较多隐藏层的神经网络, 称深度神经网络.

1.1.3 万能近似定理

神经网络具有很强的拟合能力. 事实上, 数学上已经证明下面的定理:

定理 1.2 万能近似定理 设 $h: \mathbb{R} \to \mathbb{R}$ 是一个非恒等的、有界的、单调递增的连续函数. 令 $C(\mathbb{R}^n)$ 表示从 $\mathbb{R}^n$ 到 $\mathbb{R}$ 的所有连续函数构成的空间. 对于任意 $f \in C(\mathbb{R}^n)$ 和 $\varepsilon > 0$, 存在一个整数 M, 常数 $v_i, b_i \in \mathbb{R}$ 和向

量 $\boldsymbol{w}_i \in \mathbb{R}^n$ 使得

$$\left| f(\boldsymbol{x}) - \sum_{i=1}^{M} v_i h\left(\boldsymbol{w}_i^T \boldsymbol{x} + b_i\right) \right| < \varepsilon, \quad \forall \boldsymbol{x} \in \mathbb{R}^n$$

翻译成神经网络的语言即: 一个具有单隐藏层的神经网络, 只要隐藏层节点数目足够多, 就可以以任意精度逼近任意连续函数.

1.2 神经网络的训练

1.2.1 误差函数

与线性回归类似, 神经网络的训练目标也是最小化误差函数. 对于回归问题, 常用的误差函数也是均方误差函数:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w}) - \boldsymbol{t}_n\|^2$$

对于分类问题, 常用的误差函数是交叉熵误差函数:

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_k(\boldsymbol{x}_n, \boldsymbol{w})$$

1.2.2 神经网络的梯度: 误差反向传播算法

由前面递推定义的神经网络模型是一个嵌套函数, 因此误差函数关于连接权重 $w_{ij}^{(k)}$ 的梯度可以通过链式法则进行计算. 定义辅助变量

$$\delta_i^{(k)} = \frac{\partial E}{\partial z_i^{(k)}}$$

则

$$\delta_i^{(k)} = \sum_j \frac{\partial E}{\partial z_j^{(k+1)}} \frac{\partial z_j^{(k+1)}}{\partial a_i^{(k)}} \frac{\partial a_i^{(k)}}{\partial z_i^{(k)}} = h'\left(z_i^{(k)}\right) \sum_j \delta_j^{(k)} w_{ij}^{(k)}$$

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \frac{\partial E}{\partial z_j^{(k+1)}} \frac{\partial z_j^{(k+1)}}{\partial w_{ij}^{(k)}} = \delta_j^{(k+1)} a_i^{(k)}$$

对于回归问题和分类问题, 输出层 (假定其为 $N+1$ 层) 的 δ 值可由 E 的形式统一得出简单的表达式:

$$\delta_i^{(N+1)} = y_i - t_i = a_j^{(N+1)} - t_n$$

这样, 每次计算先正向逐层迭代计算激活信号 $a_i^{(k)}$, 然后反向逐层迭代计算 $\delta_i^{(k)}$, 最后计算梯度 $\dfrac{\partial E}{\partial w_{ij}^{(k)}}$. 这一过程称为**误差反向传播 (Backpropagation)** 算法.