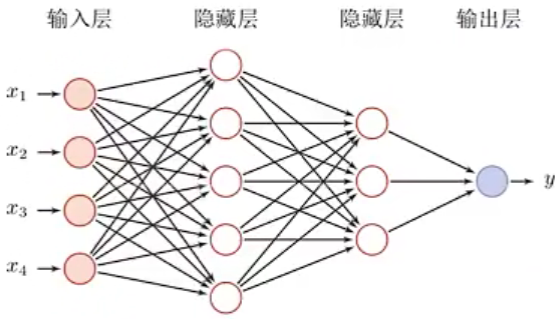
前馈神经网络

前馈神经网络(全连接神经网络、多层感知机)

- 各个神经元分别属于不同的层,层内无连接
- 相邻两层之间的神经元全部两两连接
- 整个网络中无反馈,信号从输入层向输出层单向传播,可用一个有向无环图表示



神经网络的层数不计入输入层(或者说输入层为第0层),上图为三层的神经网络.

神经网络的符号表示:

记号	含义
L	神经网络的层数
M_l	第l层神经元的个数
$f_l(\cdot)$	第l层神经元的激活函数
$W^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$	第$l-1$层到l层的权重矩阵
$b^{(l)} \in \mathbb{R}^{M_l}$	第$l-1$层到l层的偏置
$z^{(l)} \in \mathbb{R}^{M_l}$	第l层神经元的净输出(净活性值) 就是没有经过激活的
$a^{(l)} \in \mathbb{R}^{M_l}$	第l层神经元的输出(活性值) 经过激活函数的

前馈神经网络建模

前馈神经网络通过下面公式进行信息传播

$$
\begin{aligned}
z^{(l)} &= W^{(l)} a^{(l-1)} + b^{(l)} \\
a^{(l)} &= f_l(z^{(l)})
\end{aligned}
\tag{1}
$$

参数学习问题

多分类

如果使用softmax回归分类器,相当于网络的最后一层设置C个神经元,其输出经过softmax函数进行归一化后可以作为每个类的条件概率.

$$
\hat{y} = softmax(z^{(l)}) \tag{2}
$$

采用交叉熵损失函数,对于样本(x, y),其损失函数为

$$
\mathcal{L}(y, \hat{y}) = -y^T log\ \hat{y} \tag{3}
$$

给定训练集为$D = \{(x^{(n)}), y^{(n)}\}_{n=1}^{N}$,将每个样本$x^{(n)}$输入给前馈神经网络,得到网络输出为$\hat{y}^{(n)}$,其在数据集$D$上的结构化风险函数为:

$$\mathcal{R}(W,b)=\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(n)},\hat{y}^{(n)})+\frac{1}{2}\lambda||W||_F^2 \tag{4}$$

梯度下降:

计算$\frac{\partial\mathcal{R}(W,b)}{\partial W^{(l)}},\frac{\partial\mathcal{R}(W,b)}{\partial b^{(l)}}$.

如何计算梯度?

神经网络是一种复杂的复合函数

- 采用链式法则:

$$y=f^5(f^4(f^3(f^2(f^1(x)))))\rightarrow\frac{\partial y}{\partial x}=\frac{\partial f^1}{\partial x}\frac{\partial f^2}{\partial f^1}\cdots\frac{\partial f^5}{\partial f^4} \tag{5}$$

- 反向传播算法

  一种根据前馈神经网络的特点而设计的高效方法.

- 一个更加通用的计算方法

  自动微分

矩阵微积分

- 矩阵微积分是多元微积分的一种表达方式,即使用矩阵和向量来表示因变量每个成分关于自变量每个成分的偏导数.

分母布局

分母布局为求导后是列向量

- 标量关于向量的偏导数

$$\frac{\partial y}{\partial}=\left[\frac{\partial y}{}\quad\cdots\quad\frac{\partial y}{}\right]^T \tag{6}$$

$$\frac{\partial}{\partial x}=\left[\frac{\partial}{\partial x_1},\cdots,\frac{\partial}{\partial x_m}\right] \tag{6}$$

- 向量关于向量的偏导数

$$\frac{\partial f(X)}{\partial x}=\begin{bmatrix}\frac{\partial y_1}{\partial x_1},\cdots,\frac{\partial y_n}{x_1}\\ \vdots,\ddots,\vdots\\ \frac{\partial y_1}{x_m},\cdots,\frac{\partial y_n}{x_m}\end{bmatrix}\in\mathbb{R}^{M\times N} \tag{7}$$

链式法则:

(1) 若$x\in\mathbb{R},y=g(x)\in\mathbb{R}^M,z=f(y)\in\mathbb{R}^N$,则

$$\frac{\partial z}{\partial x}=\frac{\partial y}{\partial x}\frac{\partial z}{\partial y}\quad\in\mathbb{R}^{1\times N} \tag{8}$$

(2) 若$x\in\mathbb{R}^M$,$y=g(x)\in\mathbb{R}^K,z=f(y)\in\mathbb{R}^N$,则

$$\frac{\partial z}{\partial x}=\frac{\partial y}{\partial x}\frac{\partial z}{\partial y}\in\mathbb{R}^{M\times N} \tag{9}$$

(3)若$X\in\mathbb{R}^{M\times N}$为矩阵,$y=g(x)\in\mathbb{R}^K,z=f(y)\in\mathbb{R}$,则

$$\frac{\partial z}{\partial x_{ij}}=\frac{\partial y}{\partial x_{ij}}\frac{\partial z}{\partial y}\in\mathbb{R} \tag{10}$$

其实,只要记住一个东西,矩阵的链式法则,往左边乘就可以满足乘法规则.

计算梯度

$$\begin{aligned}\frac{\mathcal{L}(y,\hat{y})}{\partial w_{ij}^{(l)}}&=\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}\frac{\mathcal{L}(y,\hat{y})}{\partial z^{(l)}}\\ \frac{\mathcal{L}(y,\hat{y})}{\partial b^{(l)}}&=\frac{\partial z^{(l)}}{\partial b^{(l)}}\frac{\mathcal{L}(y,\hat{y})}{\partial z^{(l)}}\end{aligned} \tag{11}$$

上式中,只用计算三项即可.

$z^{(l)}=W^{(l)}a^{(l-1)}+b^{(l)}$

- 对于 $\frac{\partial z^{(l)}}{\partial b^{(l)}}=I$
- 对于 $\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}$ 来说,

$$
\begin{aligned}
\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}} &= \left[\frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_m^{(l)}}{\partial w_{ij}^{(l)}}\right] \\
&= \left[0, \cdots, \frac{\partial (w_i^{(l)} a^{(l-1)+b_i^{(l)}})}{\partial w_{ij}^{(l)}}, \cdots, 0\right] \\
&= \left[0, \cdots, a^{l-1}, \cdots, 0\right]
\end{aligned} \tag{12}
$$

- 对于$\frac{\partial \mathcal{L}(y,\hat{y})}{\partial z^{(l)}}$

$$
\begin{aligned}
\frac{\partial \mathcal{L}(y,\hat{y})}{\partial z^{(l)}} &= \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l+1)}}{\partial a^{(l)}} \frac{\partial \mathcal{L}(y,\hat{y})}{\partial z^{(l+1)}} \\
&= diag(f_l'(z^{(l)})) \cdot (W^{(l+1)})^T \cdot \frac{\partial \mathcal{L}(y,\hat{y})}{\partial z^{(l+1)}} \\
&= f_1'(z^{(l)}) \odot ((W^{(l+1)})^T) \frac{\partial \mathcal{L}(y,\hat{y})}{\partial z^{(l+1)}} \in \mathbb{R}^{M_l}
\end{aligned} \tag{13}
$$

  其中⊙为逐元素乘法.

  从上式中可以看出:误差项可以由上一项给出,即可以迭代到最后一层,只要计算$\frac{\partial \mathcal{L}}{\partial z^{(l)}}$就可以通过迭代出来梯度.

> 以上内容就叫做反向传播.

反向传播存在的问题,就是梯度需要自己人工计算,如果模型复杂,计算量会很大.

因此一般现在采用计算图和自动微分的方法.即将复杂函数拆分成简单函数,单个计算前进,单个计算梯度反作用回来.(就是pytorch里面的方法.)