深度神经网络

深度神经网络(Deep Neural Networks, DNN)是深度学习的基础。假设 DNN 的总层数为 L(包括输入层)，第 $l(l=1,2,\ldots,L)$层的神经元个数为 k_l，第 l 层的输出向量为$\boldsymbol{a}_{k_l}^{l}$，第 l-1 层到 l 层的权重矩阵为$\boldsymbol{W}_{k_l\times k_{l-1}}^{l}$，第 l 层的偏倚向量为$\boldsymbol{b}_{k_l}^{l}$，因此 DNN 的前向传播过程为：

(1).初始化$\boldsymbol{a}_{k_1}^{1}=\boldsymbol{x}$，$\boldsymbol{x}$ 表示输入层的样本向量；

(2).for l = 2 to L，计算：

$$\boldsymbol{a}_{k_l}^{l}=f\left(\boldsymbol{z}_{k_l}^{l}\right)=f\left(\boldsymbol{W}_{k_l\times k_{l-1}}^{l}\boldsymbol{a}_{k_{l-1}}^{l-1}+\boldsymbol{b}_{k_l}^{l}\right)$$

上式中，最后输出的结果为$\boldsymbol{a}_{k_L}^{L}$，f 表示的是激活函数，$\boldsymbol{z}_{k_l}^{l}$ 表示第 l 层的输入向量。

有了 DNN 的前向传播过程，可以通过通过反向传播算法来更新各层参数的值。假设损失函数为 J，根据输出层的激活函数和损失函数类型可以分为如下三种：

(1).输出层的激活函数为 *softmax*，损失函数为交叉熵损失函数，当 $l=L$ 时，则：

$$\boldsymbol{\delta}_{k_L}^{L}=\frac{\partial J}{\partial \boldsymbol{z}_{k_L}^{L}}=softmax\left(\boldsymbol{z}_{k_L}^{L}\right)-\boldsymbol{y}_{k_L}$$

上式中，$\boldsymbol{y}_{k_L}$ 表示输出层的标签；

(2).输出层的损失函数为 $J=\frac{1}{2}\left\|\boldsymbol{a}_{k_L}^{L}-\boldsymbol{y}_{k_L}\right\|^2=\frac{1}{2}\left\|f\left(\boldsymbol{z}_{k_L}^{L}\right)-\boldsymbol{y}_{k_L}\right\|^2$，当 $l=L$ 时，则：

$$\boldsymbol{\delta}_{k_L}^{L}=\frac{\partial J}{\partial \boldsymbol{z}_{k_L}^{L}}=\left(f\left(\boldsymbol{z}_{k_L}^{L}\right)-\boldsymbol{y}_{k_L}\right)\odot f'\left(\boldsymbol{z}_{k_L}^{L}\right)$$

上式中，$\boldsymbol{y}_{k_L}$ 表示输出层的输出值；

(3).输出层只有一个神经元，激活函数为 *sigmoid*，损失函数为交叉熵损失函数，当 $l=L$ 时，则：

$$\boldsymbol{\delta}_{k_L}^{L}=\frac{\partial J}{\partial \boldsymbol{z}_{k_L}^{L}}=sigmoid\left(\boldsymbol{z}_{k_L}^{L}\right)-\boldsymbol{y}_{k_L}$$

上式中，由于输出层只有一个神经元，因此，$\boldsymbol{y}_{k_L}$ 表示 1 或者 0；

对于以上三种情况都有当 l=2 to L-1 时，则：

$$\boldsymbol{\delta}_{k_l}^{l}=\frac{\partial J}{\partial \boldsymbol{z}_{k_l}^{l}}=\frac{\partial J}{\partial \boldsymbol{z}_{k_L}^{L}}\frac{\partial \boldsymbol{z}_{k_L}^{L}}{\partial \boldsymbol{z}_{k_{L-1}}^{L-1}}\cdots\frac{\partial \boldsymbol{z}_{k_{l+2}}^{l+2}}{\partial \boldsymbol{z}_{k_{l+1}}^{l}}\frac{\partial \boldsymbol{z}_{k_{l+1}}^{l+1}}{\partial \boldsymbol{z}_{k_l}^{l}}=\boldsymbol{\delta}_{k_{l+1}}^{l+1}\frac{\partial \boldsymbol{z}_{k_{l+1}}^{l+1}}{\partial \boldsymbol{z}_{k_l}^{l}}$$

又因为$\boldsymbol{z}_{k_{l+1}}^{l+1}=\boldsymbol{W}_{k_{l+1}\times k_l}^{l+1}f\left(\boldsymbol{z}_{k_l}^{l}\right)+\boldsymbol{b}_{k_{l+1}}^{l+1}$，所以，

$$\boldsymbol{\delta}_{k_l}^{l}=\boldsymbol{\delta}_{k_{l+1}}^{l+1}\frac{\partial \boldsymbol{z}_{k_{l+1}}^{l+1}}{f\left(\boldsymbol{z}_{k_l}^{l}\right)}\odot\frac{f\left(\boldsymbol{z}_{k_l}^{l}\right)}{\partial \boldsymbol{z}_{k_l}^{l}}=\left(\boldsymbol{W}_{k_{l+1}\times k_l}^{l+1}\right)^{T}\boldsymbol{\delta}_{k_{l+1}}^{l+1}\odot\frac{f\left(\boldsymbol{z}_{k_l}^{l}\right)}{\partial \boldsymbol{z}_{k_l}^{l}}=\left(\left(\boldsymbol{W}_{k_{l+1}\times k_l}^{l+1}\right)^{T}\boldsymbol{\delta}_{k_{l+1}}^{l+1}\right)\odot f'\left(\boldsymbol{z}_{k_l}^{l}\right)$$

因此，损失函数 J 对 $\boldsymbol{W}_{k_l\times k_{l-1}}^{l}$ 和 $\boldsymbol{b}_{k_l}^{l}$ 的偏导为

$$\frac{\partial J}{\partial \boldsymbol{W}_{k_l\times k_{l-1}}^{l}}=\frac{\partial J}{\partial \boldsymbol{z}_{k_l}^{l}}\frac{\partial \boldsymbol{z}_{k_l}^{l}}{\boldsymbol{W}_{k_l\times k_{l-1}}^{l}}=\boldsymbol{\delta}_{k_l}^{l}\left(\boldsymbol{a}_{k_{l-1}}^{l-1}\right)^{T}$$

$$\frac{\partial J}{\partial \boldsymbol{b}_{k_l}^{l}}=\frac{\partial J}{\partial \boldsymbol{z}_{k_l}^{l}}\frac{\partial \boldsymbol{z}_{k_l}^{l}}{\boldsymbol{b}_{k_l}^{l}}=\boldsymbol{\delta}_{k_l}^{l}$$

最后，根据如下公式更新 RNN 模型中的参数：

$$\theta_{update}=\theta-\alpha\frac{\partial L}{\partial \theta}$$

上式中，θ 代表的是模型参数 $\boldsymbol{W}_{k_l\times k_{l-1}}^{l}$ 和 $\boldsymbol{b}_{k_l}^{l}$，α 代表学习率，θ_{update} 代表更新完以后的参数。

但是，在 DNN 中会出现梯度消失和梯度爆炸的现象。出现梯度消失，是因为更新参数 $\boldsymbol{W}_{k_l\times k_{l-1}}^{l}$ 和 $\boldsymbol{b}_{k_l}^{l}$ 时，都与前面的模型参数连乘，又因为激活函数的导数在 0 到 1 之间，经过多次迭代相乘，极容易出现梯度消失。出现梯度爆炸，是因为更新参数 $\boldsymbol{W}_{k_l\times k_{l-1}}^{l}$ 和 $\boldsymbol{b}_{k_l}^{l}$ 时，当激活函数的导数不是特别小同时模型参数特别大，就会出现梯度爆炸，但是梯度爆炸的概率比较小。

注：

1. 本文用 $\odot$ 表示 Hadamard 积，对于两个维度相同的向量 $\boldsymbol{a}=\left[a_1,a_2,\ldots,a_m\right]^{T}$ 和 $\boldsymbol{b}=\left[b_1,b_2,\ldots,b_m\right]^{T}$，则 $\boldsymbol{a}\odot\boldsymbol{b}=\left[a_1b_1,a_2b_2,\ldots,a_mb_m\right]^{T}$；

2.假设两个维度相同的向量 $\boldsymbol{y}=\left[y_1,y_2,\ldots,y_m\right]^{T}$ 和 $\boldsymbol{x}=\left[x_1,x_2,\ldots,x_m\right]^{T}$ 满足如下公式：

$$\boldsymbol{y}=f\left(\boldsymbol{x}\right)\Leftrightarrow\begin{bmatrix}y_1\\y_2\\\vdots\\y_m\end{bmatrix}=\begin{bmatrix}f\left(x_1\right)\\f\left(x_2\right)\\\vdots\\f\left(x_m\right)\end{bmatrix}$$

则：

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}=\begin{bmatrix}f'\left(x_1\right)\\f'\left(x_2\right)\\\vdots\\f'\left(x_m\right)\end{bmatrix}$$

3.假设两个维度相同的向量 $\boldsymbol{y}=\left[y_1,y_2,\ldots,y_m\right]^{T}$ 和 $\boldsymbol{x}=\left[x_1,x_2,\ldots,x_n\right]^{T}$ 满足如下公式：

$$\boldsymbol{y} = \boldsymbol{A}_{m\times n}\boldsymbol{x}$$

则：

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \boldsymbol{A}_{m\times n}^{T}$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{A}_{m\times n}} = \boldsymbol{x}^{T}$$