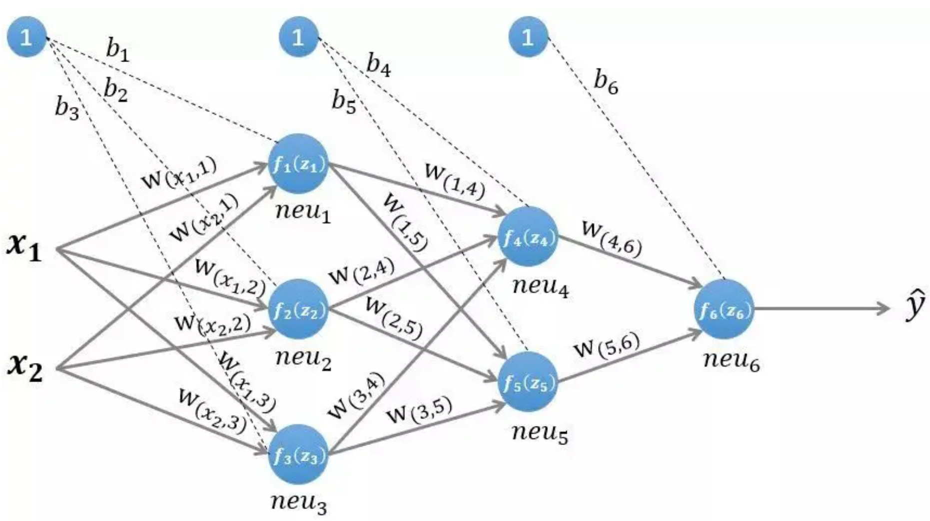
反向传播算法

jerrychenly 关注 IP属地: 江苏
2019.06.26 15:54:00 字数 1,089 阅读 284

反向传播算法（Backpropagation Algorithm，简称BP算法）是深度学习的重要思想基础，本文将介绍该算法的原理。



上图是一个简单的神经网络，我们用它来实现二分类。我们给它一个输入样本（x_1，x_2），通过前向运算得到输出 $\hat{y}$，输出值 $\hat{y}$ 的值域为[0,1]，例如 $\hat{y}$ 的值越接近0，代表该样本是"0"类的可能性越大，反之是"1"类的可能性大。

一、首先我们来看看前向传播的过程：

输入的样本为：$\vec{a} = (x_1, x_2)$

第一层网络参数为：

$$W^{(1)}=\begin{bmatrix} w_{(x_1,1)}, w_{(x_2,1)} \\ w_{(x_1,2)}, w_{(x_2,2)} \\ w_{(x_1,3)}, w_{(x_2,3)} \end{bmatrix}, \quad b^{(1)}=[b_1, b_2, b_3]$$

第二层网络参数为：

$$W^{(2)}=\begin{bmatrix} w_{(1,4)}, w_{(2,4)}, w_{(3,4)} \\ w_{(1,5)}, w_{(2,5)}, w_{(3,5)} \end{bmatrix}, \quad b^{(2)}=[b_4, b_5]$$

第三层网络参数为：

$$W^3=[w_{(4,6)}, w_{(5,6)}], \quad b^{(3)}=[b_6]$$

1、第一层隐藏层：

第一层有三个神经元 neu_1， neu_2， neu_3 。该层的输入为：

$z^{(1)} = W^{(1)} * (\vec{a})^T + (b^1)^T$ ，故此可得：

$$z_2 = w_{x_1,2} * x_1 + w_{x_2,2} * x_2 + b_2$$

$$z_3 = w_{x_1,3} * x_1 + w_{x_2,3} * x_2 + b_3$$

假设第一层的激活函数为f(x)（上图中的激活函数都标了一个下标，一般情况下，同一层的激活函数都一样，不同层可以选择不同的激活函数），那么第一层的输出为：$f_1(z_1)$，$f_2(z_1)$，$f_3(z_1)$。

2、第二层隐藏层：

第二层有两个神经元neu_4，neu_5。该层的输入为：$\mathbf{z}^{(2)} = W^{(2)} * [\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3]^T + (b^2)^T$，即第二层的输入是第一层的输出乘以第二层的权重，再加上第二层的偏置，所以第二层两个神经元的输入为：

$$\mathrm{z}_4 = w_{1,4} * \mathrm{z}_1 + w_{2,4} * \mathrm{z}_2 + w_{3,4} * \mathrm{z}_3 + b_4$$

$$\mathrm{z}_5 = w_{1,5} * \mathrm{z}_1 + w_{2,5} * \mathrm{z}_2 + w_{3,5} * \mathrm{z}_3 + b_5$$

所以第二层的输出为：$f_4(z_4)$和$f_5(z_5)$。

3、输出层：

输出层只有一个神经元neu_6。该层的输入为：$z^{(3)} = W^{(3)} * [z_4, z_5]^T + (b^3)^T$ 即：$z_6 = w_{4,6} * \mathrm{z}_4 + w_{5,6} * \mathrm{z}_5 + b_6$。

因为该网络要解决一个二分类的问题，所以输出层的激活函数也可以使用一个sigmoid型函数，神经网络最后的输出为：$f_6(z_6)$。

二、反向传播的过程

上面我们已经知道了数据沿着神经网络前向传播的过程，现在我们来看看反向传播的过程。反向传播算法会对特定样本的预测输出和理想输出进行比较，然后确定网络的每个权重的更新幅度。假设我们使用随机梯度下降的方式来学习神经网络的参数，损失函数定义为$L(y,\hat{y})$，其中y是该样本的真实列表。使用梯度下降进行参数学习，我们需要计算出损失函数关于神经网络中各层参数（权重w和偏置b）的偏导数。

假设我们要对第k层隐藏层的参数$W^{(k)}$和$b^{(k)}$求偏导数，即求$\frac{\partial L(y,\hat{y})}{\partial W^{(k)}}$和$\frac{\partial L(y,\hat{y})}{\partial b^{(k)}}$。假设$z^{(k)}$代表第k层神经元的输入，即$z^{(k)} = W^{(k)} * n^{(k-1)} + b^{(k)}$，其中$n^{(k-1)}$为前一层神经元的输出，根据链式法则有：

$$\frac{\partial L(y,\hat{y})}{\partial W^{(k)}} = \frac{\frac{\partial L(y,\hat{y})}{\partial z^{(k)}} * \partial z^{(k)}}{\partial W^{(k)}}$$

$$\frac{\partial L(y,\hat{y})}{\partial b^{(k)}} = \frac{\frac{\partial L(y,\hat{y})}{\partial z^{(k)}} * \partial z^{(k)}}{\partial b^{(k)}}$$

。

因此，我们只需要分别计算偏导数$\frac{\partial L(y,\hat{y})}{\partial z^{(k)}}$，$\frac{\partial z^{(k)}}{\partial W^{(k)}}$和$\frac{\partial z^{(k)}}{\partial b^{(k)}}$。

1、计算偏导数 $\frac{\partial z^{(k)}}{}$ 和 $\frac{\partial z^{(k)}}{}$

$$\frac{\partial z}{\partial W^{(k)}}=\begin{bmatrix} \vdots \\ \frac{\partial\left(W_{m:}^{(k)}*n^{(k-1)}+b^{(k)}\right)}{\partial W^{(k)}} \end{bmatrix} \underset{=}{\text{初等变换}} \left(n^{(k-1)}\right)^T$$

上式中，$W_{m:}^{(k)}$ 代表第k层神经元的权重矩阵 $W^{(k)}$ 的第m行，$W_{mn}^{(k)}$ 代表第k层神经元的权重矩阵 $W^{(k)}$ 的第m行中的第n列。

偏置b是一个常数项，因此偏导数的计算也很简单：

$$\frac{\partial z^{(k)}}{\partial b^{(k)}}=\begin{bmatrix} \frac{\partial\left(W_{1:}^{(k)}*n^{(k-1)}+b_1\right)}{\partial b_1} & \cdots & \frac{\partial\left(W_{1:}^{(k)}*n^{(k-1)}+b_1\right)}{\partial b_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial\left(W_{m:}^{(k)}*n^{(k-1)}+b_m\right)}{\partial b_1} & \cdots & \frac{\partial\left(W_{m:}^{(k)}*n^{(k-1)}+b_m\right)}{\partial b_m} \end{bmatrix}$$

得到计算结果是单位矩阵。

2、计算偏导数 $\frac{\partial L(y,\hat{y})}{\partial z^{(k)}}$

偏到数 $\frac{\partial L(y,\hat{y})}{\partial z^{(k)}}$ 又称为误差项（也称"灵敏度"），一般用 δ 表示，例如 $\delta^{(1)}=\frac{\partial L(y,\hat{y})}{\partial z^{(1)}}$ 是第一层神经元的误差项，其值的大小代表了第一层神经元对于最终总误差的影响大小。

根据前向计算，我们知道第k+1层的输入与第k层输出的关系为：
$\mathbf{z}^{(k+1)}=W^{(k+1)}*n^{(k)}+b^{k+1}$

又因为 $n^{(k)}=f_k\left(z^{(k)}\right)$，根据链式法则，我们可以得到 $\delta^{(k)}$ 为：

$$\begin{aligned}
\delta^{(k)} &= \frac{\partial \mathrm{L}(\mathrm{y},\hat{\mathrm{y}})}{\partial z^{(k)}} \\
&= \frac{\partial n^{(k)}}{\partial z^{(k)}} * \frac{\partial z^{(k+1)}}{\partial n^{(k)}} * \frac{\partial \mathrm{L}(\mathrm{y}\hat{\mathrm{y}})}{\partial z^{(k+1)}} \\
&= \frac{\partial n^{(k)}}{\partial z^{(k)}} * \frac{\partial z^{(k+1)}}{\partial n^{(k)}} * \delta^{(k+1)} \\
&= f_k'\left(z^{(k)}\right) * \left(\left(W^{(k+1)}\right)^T * \delta^{(k+1)}\right)
\end{aligned}$$

由上式我们可以知道，第k层神经元的误差项 $\delta^{(k)}$ 是由第k+1层的误差项乘以第k+1层的权重，再乘以第k层激活函数的导数(梯度)得到的。**这就是误差的反向传播。**

到这里，我们可以分别计算出损失函数关于权重和偏置的偏导数了：

$$\frac{\partial L(y,\hat{y})}{\partial W^{(k)}}=\frac{\frac{\partial L(y,\hat{y})}{\partial z^{(k)}}*\partial z^k}{\partial W^k}=\delta^k*\left(n^{(k-1)}\right)^T$$

$$\frac{\partial L(y,\hat{y})}{\partial b^{(k)}}=\frac{\frac{\partial L(y,\hat{y})}{\partial z^{(k)}}*\partial z^k}{\partial b^k}=\delta^k$$

有了上面两个偏导数，我们就能够利用随机梯度下降算法来更新参数。