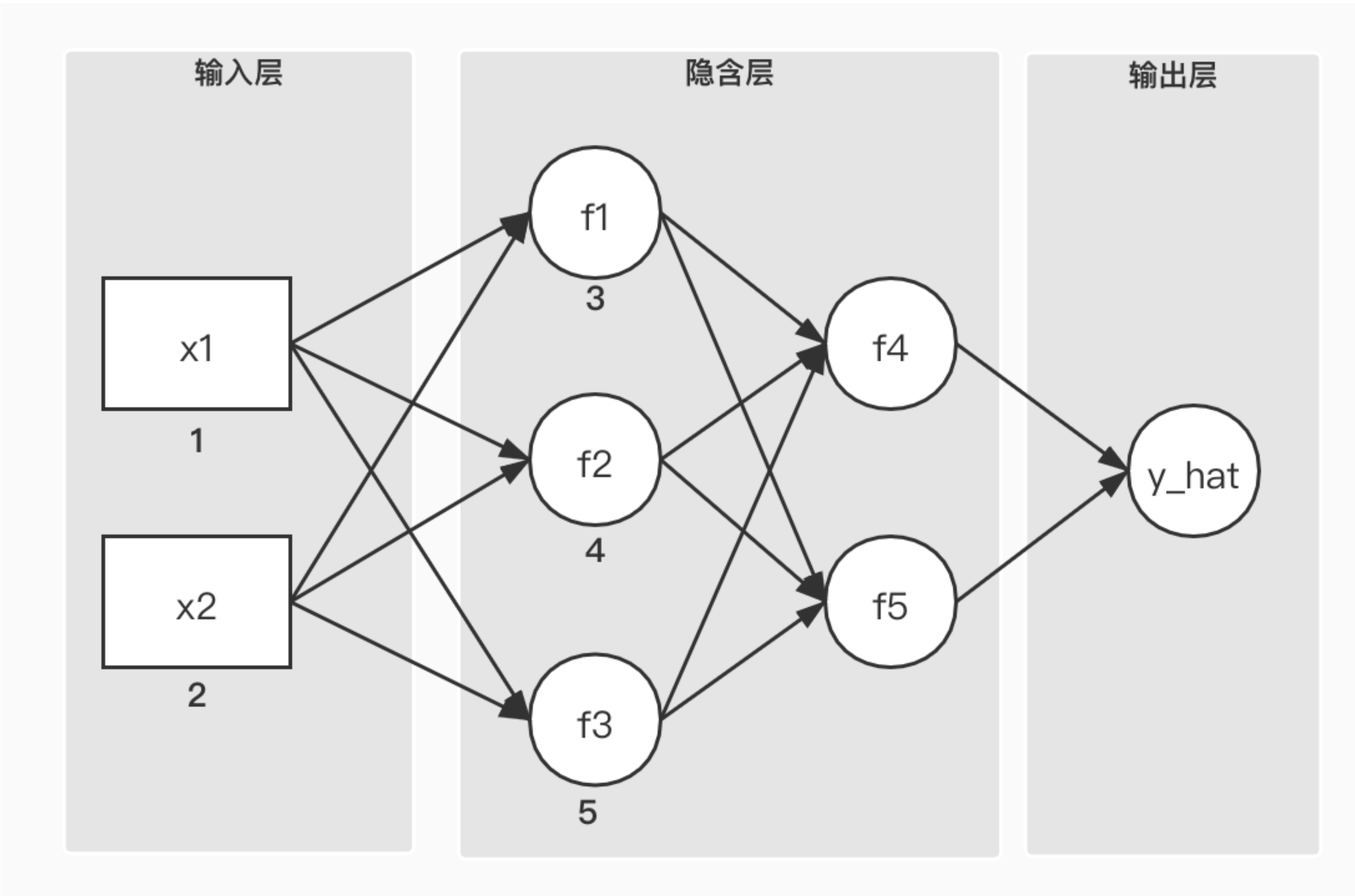
BP算法

神经网络介绍



上图为含有一个输入层，2个隐含层和一个输出层的简单神经网络(介绍的神经网络将不包含偏置项)。

输入/输出/激活函数

整个神经网络的输入为$\boldsymbol{x}=[x_1,x_2]^T$,输出为$\hat{y}$(对应y_hat)。
对于隐含层来说，每一个神经元都有一个标量输入值和标量输出值。对于第一层隐含层，输入为

$$a^1=\begin{bmatrix}a_1^1\\a_2^1\\a_3^1\end{bmatrix}$$

经过**激活函数** $\boldsymbol{f}()$ 输出为

$$o^1=\begin{bmatrix} o_1^1 \\ o_2^1 \\ o_3^1 \end{bmatrix}=\begin{bmatrix} f_1(a_1^1) \\ f_2(a_2^1) \\ f_3(a_3^1) \end{bmatrix}$$

在提供的c++代码中,

$$f(x)=\frac{1}{1+e^{-x}}$$

即Logistic函数。

权重

若上层输出向量为$\boldsymbol{o}_{l-1}$,这层权重为$\boldsymbol{W}_l$，则有$\boldsymbol{W}_l\boldsymbol{o}_{l-1}=\boldsymbol{a}_l$.
l层的权重矩阵的行数等于l层神经元的个数，列数等于上一层输出向量的纬度(上一层神经元的个数)。
距离来说，第一层隐含层的输入a^1，是由输入的$\boldsymbol{x}$加权得到,其中权重矩阵为

$$\boldsymbol{W}=\begin{bmatrix} w_{13} & w_{23} \\ w_{14} & w_{24} \\ w_{15} & w_{25} \end{bmatrix}$$

正向传播

根据前一部分，有2个推导的等式，即

$$\begin{aligned} \boldsymbol{a}_l &= \boldsymbol{W}_l\boldsymbol{o}_{l-1} \\ \boldsymbol{o}_l &= \boldsymbol{f}(\boldsymbol{a}_l) \end{aligned}$$

通过这2个等式我们可以根据输入的$\boldsymbol{x}$(即第一层的$\boldsymbol{o}_0$)，以$\boldsymbol{o}_0,\boldsymbol{a}_1,\boldsymbol{o}_1,\boldsymbol{a}_2,\boldsymbol{o}_2,\ldots,\boldsymbol{o}_n$正向的计算出神经网络的输出.对于上述神经网络可以写成:

$$f(W_3f(W_2f(W_1\boldsymbol{x})))=\boldsymbol{o}_3=\hat{y}$$

详细的描述(以上述神经网络为例，其中上标表示纬度，下标表示层)

$$W_1^{(3,2)} o_0^{(2,1)} \rightarrow a_1^{(3,1)} \rightarrow f() \rightarrow o_1^{(3,1)} \rightarrow W_2^{(2,3)} o_1 \rightarrow a_2^{(2,1)} \rightarrow f() \rightarrow o_2^{(2,1)}$$
$$\rightarrow W_3^{(2,1)} o_2 \rightarrow a_3^{(1,1)} \rightarrow f() \rightarrow \hat{y}$$

BP(误差反向传播)

一次输入通过正向传播即可得到输出，但是仅限于在训练好的神经网络下才有意义。BP算法则是经典的训练网络的算法，概括来说是一种更新权重的方法。其权重的更新公式为：

$$\boldsymbol{W}_l = \boldsymbol{W}_l - \eta \frac{\partial L}{\partial \boldsymbol{W}_l}$$

其中η为设定的学习率常数，L为定义的损失函数，在提供的代码中，采用的是均方误差函数，即

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

向量形式为

$$C = \frac{1}{2}\|\boldsymbol{o}_L - \boldsymbol{y}\|^2$$

权值矩阵的偏导

为更新权值矩阵，需要计算$\frac{\partial L}{\partial \boldsymbol{W}_l}$。由于

$$\begin{aligned} \boldsymbol{o}_L &= f(\boldsymbol{a}_l) \\ \boldsymbol{a}_l &= \boldsymbol{W}_l \boldsymbol{o}_{l-1} = g(\boldsymbol{W}_l) \end{aligned}$$

则$\boldsymbol{o}_l = f(g(\boldsymbol{W}_l))$，因此$L()$看做关于$W_l$的复合函数，根据链式求导法则

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}_l} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{a}_l} \frac{\partial \boldsymbol{a}_l}{\partial \boldsymbol{W}_l}$$

定义$\xi_l = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{a}_l}$为误差，右边$\frac{\partial \boldsymbol{a}_l}{\partial \boldsymbol{W}_l} = \frac{\partial \boldsymbol{W}_l \boldsymbol{o}_{l-1}}{\partial \boldsymbol{W}_l} = \boldsymbol{o}_{l-1}^T$ (矩阵求导公式)
现在计算ξ,采用类似数学归纳法的思想，找ξ_l和ξ_{l+1}的关联。
则有

$$
\begin{aligned}
\xi_l &= \frac{\partial L}{\partial a_l} \\
&= \frac{\partial L}{\partial o_l}\frac{\partial o_l}{\partial a_l} \\
&= \frac{\partial L}{\partial a_{l+1}}\frac{\partial a_{l+1}}{\partial o_l}\frac{\partial o_l}{\partial a_l}
\end{aligned}
$$

现在计算每项

$$
\begin{gathered}
\frac{\partial L}{\partial a_{l+1}} = \xi_{l+1} \\
\frac{\partial a_{l+1}}{\partial o_l} = \frac{\partial W_{l+1} o_l}{\partial o_l} = W_{l+1}^T \\
\frac{\partial o_l}{\partial a_l} = \frac{\partial f(a_l)}{\partial a_l} = f'(a_l)
\end{gathered}
$$

注意矩阵的乘法顺序

$$
\begin{bmatrix} \xi_l^1 \\ \xi_l^2 \end{bmatrix} = \begin{bmatrix} f' & 0 \\ 0 & f' \end{bmatrix} \begin{bmatrix} w & w \\ w & w \\ w & w \end{bmatrix}^T \begin{bmatrix} \xi_{l+1}^1 \\ \xi_{l+1}^2 \\ \xi_{l+1}^3 \end{bmatrix}
$$

即描述为l层神经元的误差项是由l+1神经元的误差项乘以l+1层的权重，再乘以l层激活函数的导数(梯度)得到的。

总结

1. 遍历所有训练样本$(\boldsymbol{x}_n, y_n)$，进行一次正向传播，依次求出$\boldsymbol{o}_0, \boldsymbol{a}_1, \boldsymbol{o}_1, \boldsymbol{a}_2, \boldsymbol{o}_2, \boldsymbol{a}_3, \boldsymbol{o}_3, \dots, \boldsymbol{o}_L$
2. 根据公式$\xi_L = \boldsymbol{f}'(\boldsymbol{a}_L)(\hat{y} - \boldsymbol{y})$求最后一层的误差
3. 根据$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}_l} = \boldsymbol{\xi}_l \boldsymbol{o}_{l-1}^T, \quad \xi_l = f'(a_l) W_{l+1}^T \xi_{l+1}, \quad \boldsymbol{W}_l = \boldsymbol{W}_l - \eta \frac{\partial L}{\partial \boldsymbol{W}_l}$更新权重$W_l$（从l-1层到第一层）