深度学习反向传播

一：回归与分类

回归与分类是机器学习中两类最重要的问题。其核心区别在于：

- 回归问题预测的是连续值
  - 房价预测，身高预测等。
- 分类问题预测的是离散值
  - 如常见的二分类问题 $y_i \in \{0,1\}$,垃圾邮件分类，肿瘤的良性与恶性区分等。

1.1 sigmoid函数

考虑这样一条直线 $y = \mathbf{w}^{\mathbf{T}}\mathbf{x} + b$，将其进行映射。有如下两种方式：

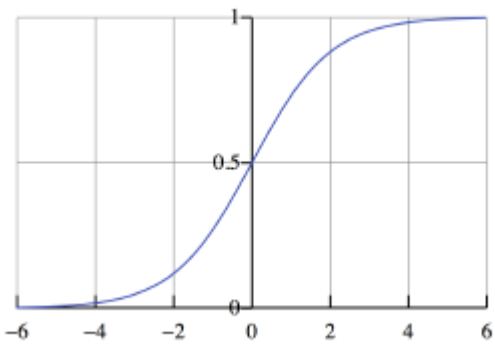
- 非线性映射1：

$$z = \begin{cases} 0 & y < 0 \\ 0.5 & y = 0 \\ 1 & y > 0 \end{cases}$$

缺点：这种函数不是连续函数，不方便进行求导。

- 非线性映射2：

$$z = \frac{1}{1 + e^{-y}}$$



当y趋于正无穷时，$z=1$;当y趋于负无穷时，$z=0$.这样就把一个$(-\infty,+\infty)$的区间映射到$(0,1)$之间，又因为$(0,1)$可以看作概率值，因此，我们可以把z看作是值取1的概率。如果output>0.5，则最终结果判定为1。如果output<0.5，则最终结果判定为0。

1.2 sigmoid函数求导

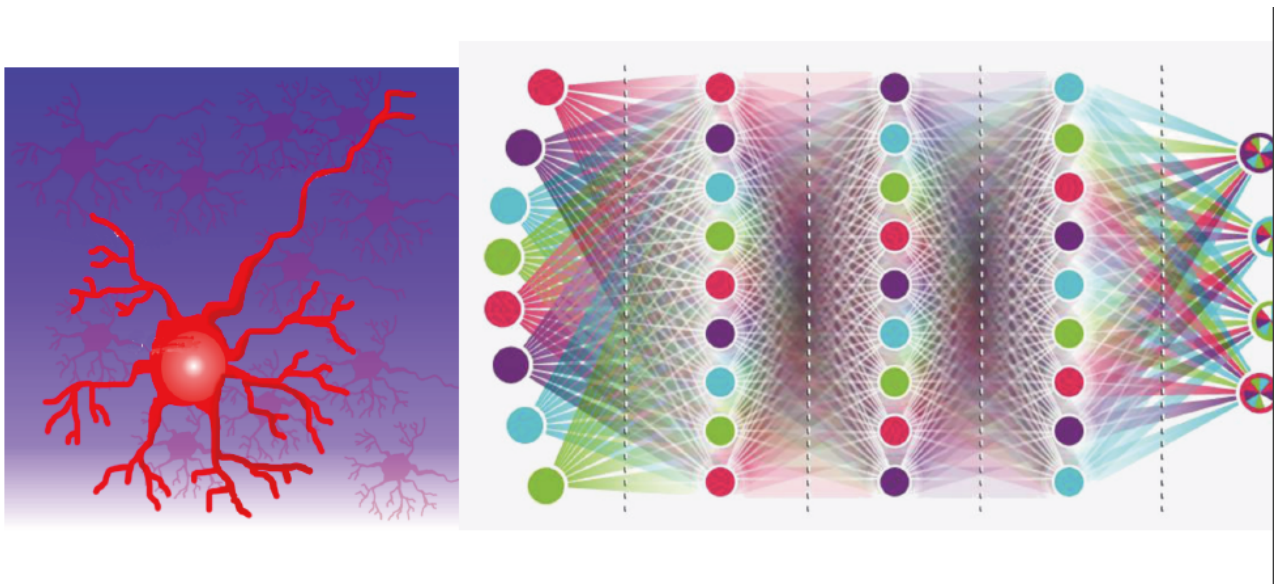
$$\sigma(x)=\frac{1}{1+e^{-x}}$$

- $$\begin{aligned}\frac{d\sigma(x)}{dx}&=(\frac{1}{1+e^{-x}})'=-\frac{-e^{-x}}{(1+e^{-x})^2}=\frac{1+e^{-x}-1}{(1+e^{-x})^2}\\&=\frac{1}{1+e^{-x}}-(\frac{1}{1+e^{-x}})^2\\&=\sigma(x)(1-\sigma(x))\end{aligned}$$

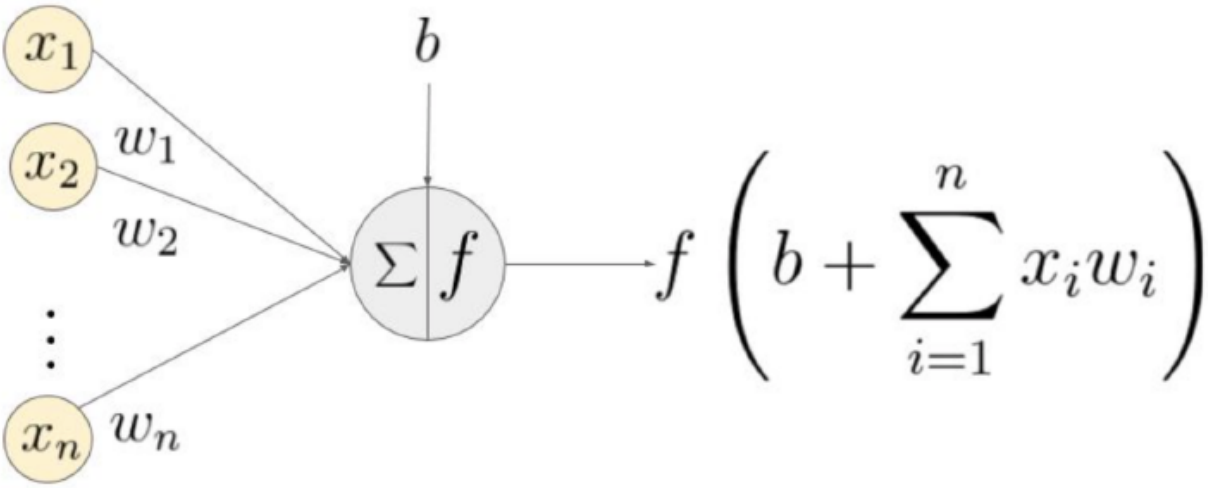
二：神经网络

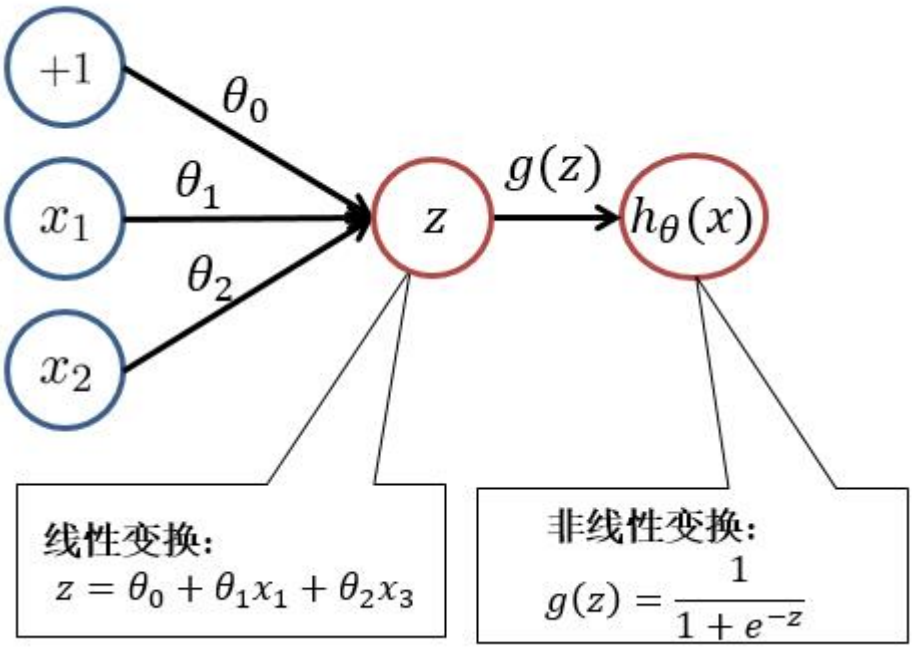
2.1 架构

包含输入层，隐藏层和输出层。此次我们探讨的神经网络架构中，每层内的节点之间是不连接的，每层的节点只和下一层的节点实现全连接。

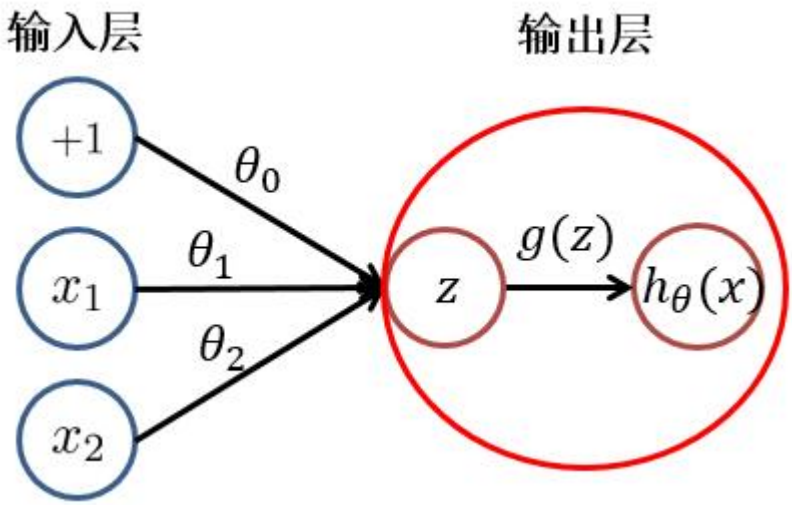


2.2 神经元介绍





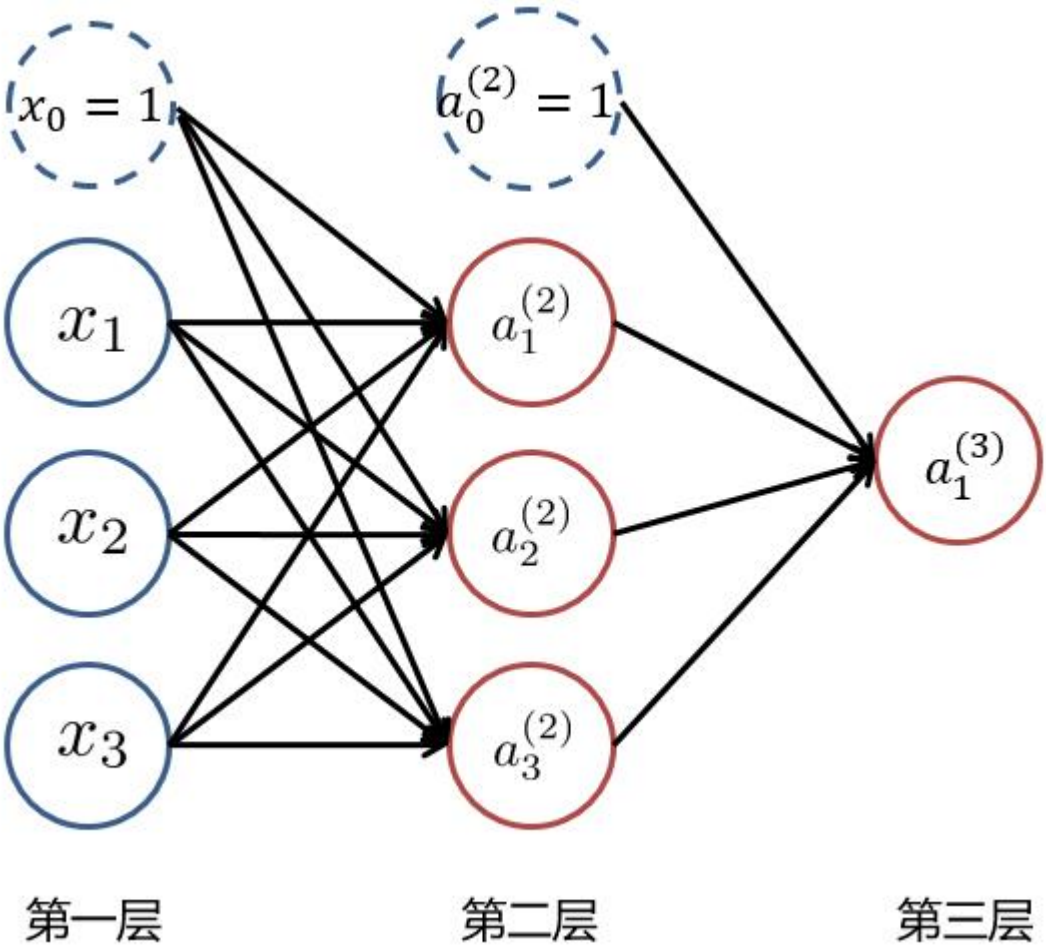
将**线性变换（求和）**与**非线性变换**集成在一个神经元中，便如下图所示。



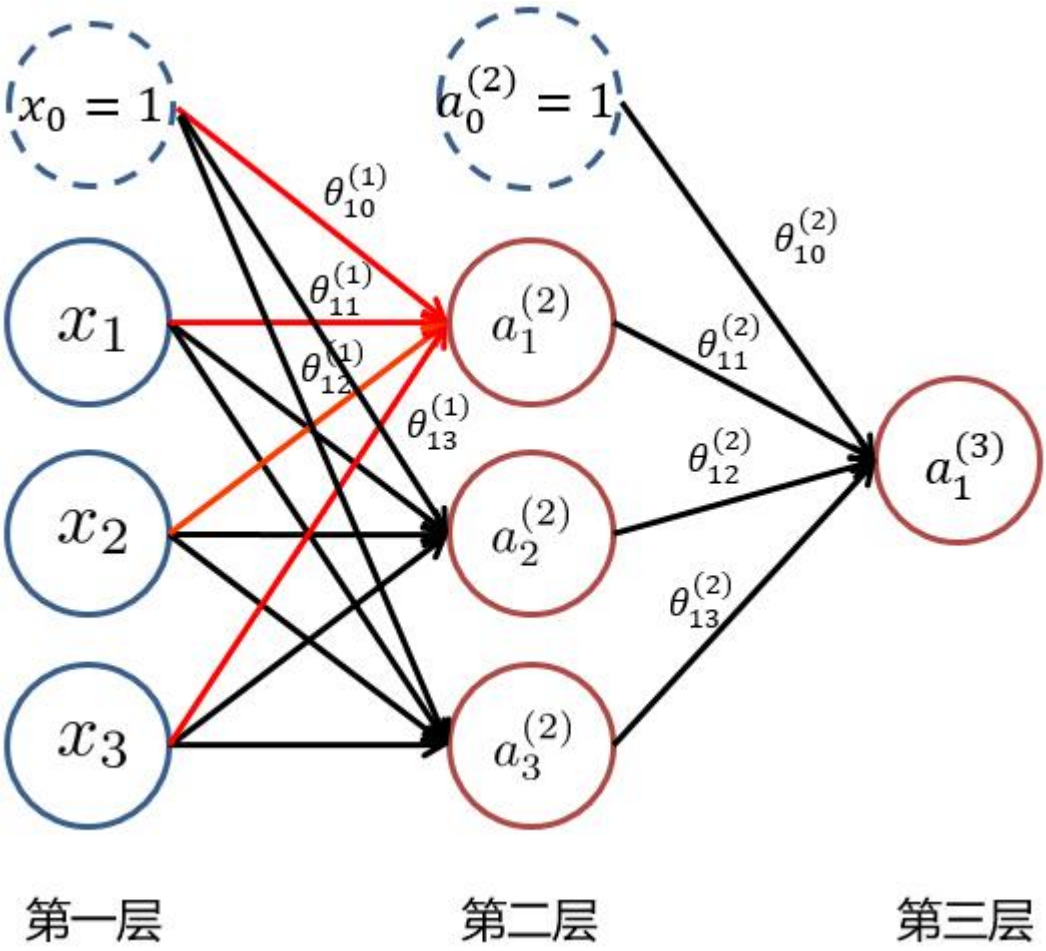
注：图中θ与w的含义是一样的，只是不同的表示符号。

上图是用来介绍神经元内部计算的示意图。对于每个输入的x特征，会分配一个权重w，然后进行求和，最后会加上偏置b。即首先计算$\mathbf{w}^{\mathrm{T}}\mathbf{x}+b$，然后将计算结果带入激活函数$f$中，在本次课中，我们选择使用sigmoid函数作为激活函数。但是需要注意的是激活函数不一定非要是sigmoid函数。

2.3 神经网络介绍



其中$x_i(i=1,2,3)$为输入层的值，$a_i^{(k)}(k=1,2,3...,K\ ;i=1,2,3...,N_k)$，表示第$k$中，第$i$个神经元的激活值，$N_k$表示第$k$层神经元的个数，当$k=1$时，即为输入层，即$a_i^{(1)}=x_i$，$x_0=1$与$a_0^{(2)}=1$为偏置项。隐藏层或输出层的每个神经元的激活值都由上一层经过类似逻辑回归的计算得到，具体可参考下图：



使用$\theta_{ji}^{(k)}$ 来表示第 k层的参数（边权），其中下标 jj表示第 $k+1$层的第j个神经元，i表示第k层的第i个神经元。于是我们可以计算出隐藏层的三个激活值：

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3)$$

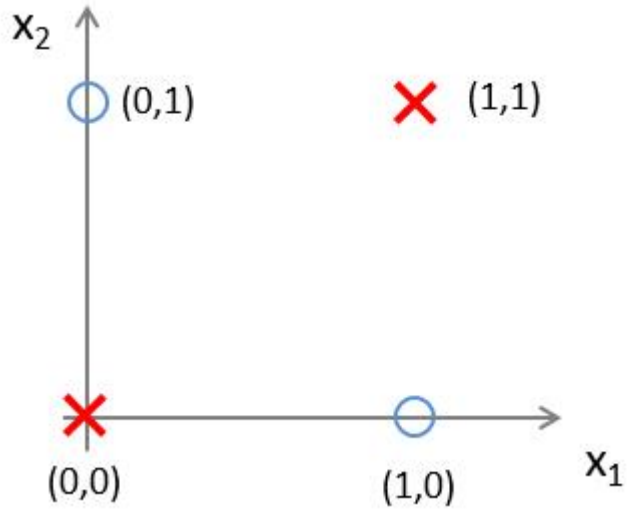
再将隐藏层的三个激活值以及偏置项（$a_0^{(2)}, a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$）用来计算输出层神经元的激活值。

$$a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$$

其中$g(z)$为非线性变换函数（或称激活函数）。

2.4 神经网络进行非线性切分的原理

有这样一组样本，如下图：



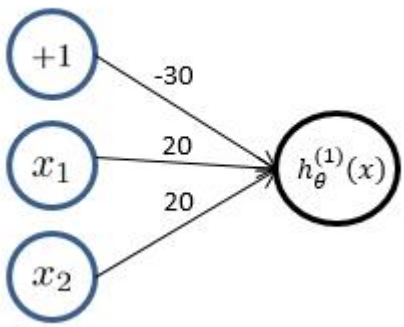
x_1	x_2	$h_\theta(x)$
0	0	0
0	1	1
1	0	1
1	1	0

仔细观察，我们发现，如果想要对上述样本进行分类，则很难找到一条线性分隔边界。观察上表中的输入输出值，我们发现分类结果与输出值是异或关系。而逻辑回归可以通过改变参数实现“与”，“或”，“非”的操作。如下所示：

（1）**逻辑回归的“与”操作**，假设模型参数如下：

$$h_\theta^{(1)}(x) = g(-30 + 20x_1 + 20x_2) = \frac{1}{1 + e^{-(-30+20x_1+20x_2)}}$$

对应结构如下图所示：

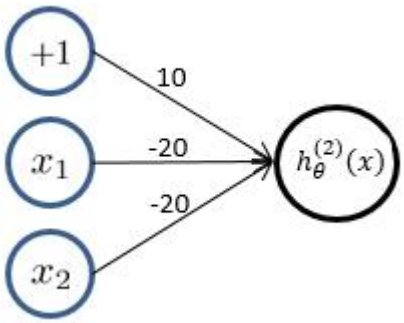


x_1	x_2	$h_\theta^{(1)}(x)$
0	0	0
0	1	0
1	0	0
1	1	1

（2） 逻辑回归实现**逻辑“或非”操作**，假设模型函数如下：

$$h_\theta^{(2)}(x) = g(-10 - 20x_1 - 20x_2) = \frac{1}{1 + e^{-(10-20x_1-20x_2)}}$$

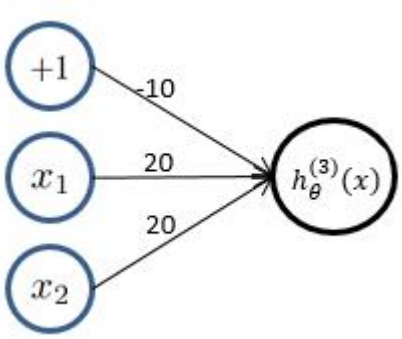
对应结构如下：



x_1	x_2	$h_\theta^{(2)}(x)$
0	0	1
0	1	0
1	0	0
1	1	0

（2）**逻辑回归**实现**逻辑“或”操作**，假设模型函数如下：

$$h_\theta^{(3)}(x) = g(-10 + 20x_1 + 20x_2) = \frac{1}{1 + e^{-(10+20x_1+20x_2)}}$$



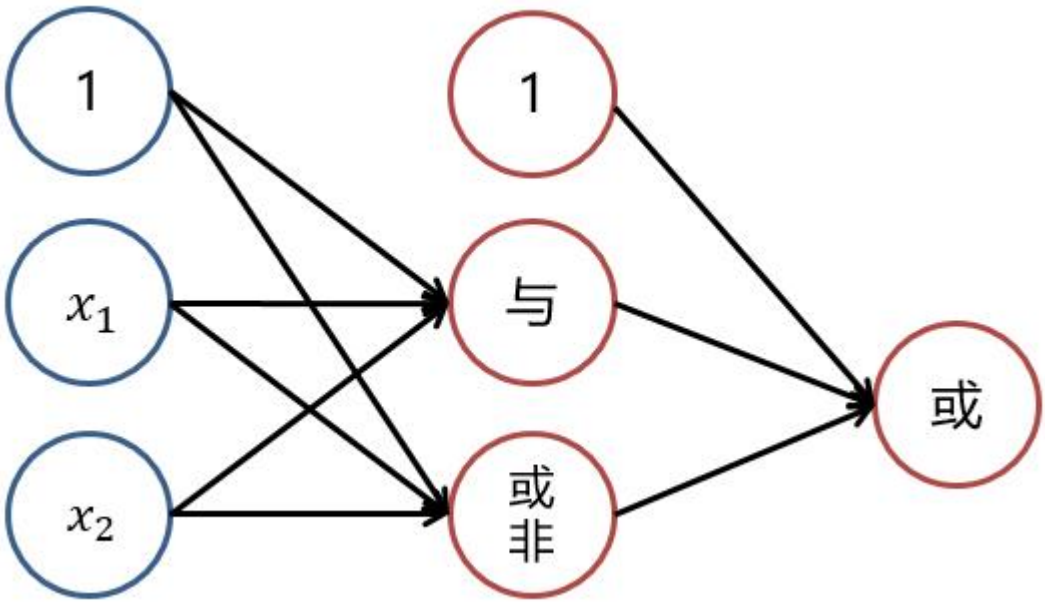
x_1	x_2	$h_\theta^{(3)}(x)$
0	0	0
0	1	1
1	0	1
1	1	1

仔细观察上述三组结果，可以发现，如果将$h_\theta^{(1)}(x)$和$h_\theta^{(2)}(x)$进行**逻辑"或"**的组合，我们便可以得到如下**异或**结果：

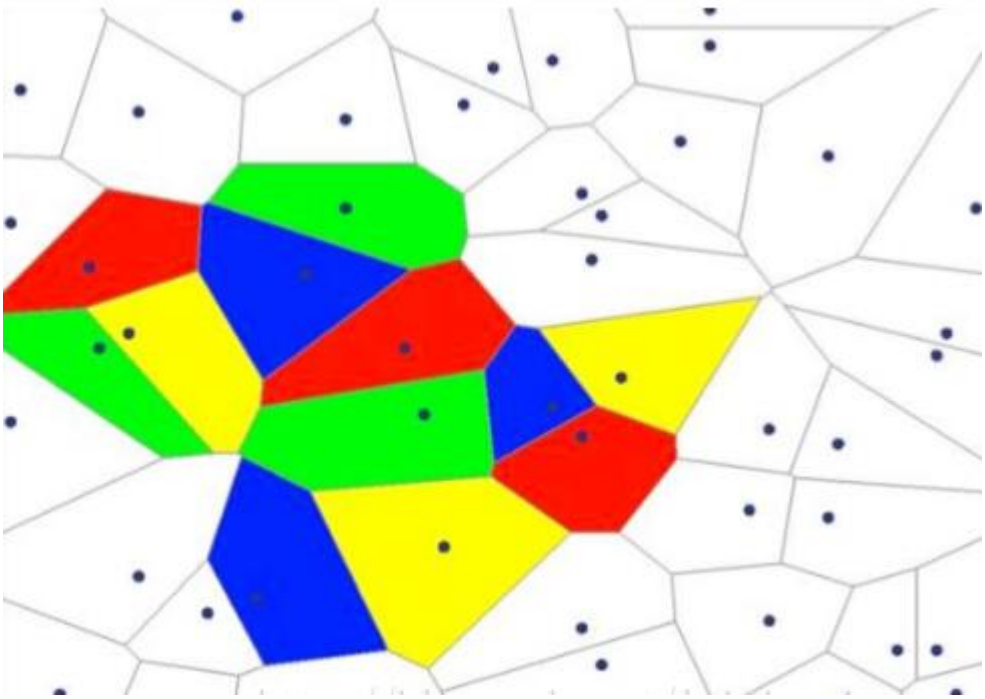
$h_\theta^{(1)}(x)$	$h_\theta^{(2)}(x)$	$h_\theta(x)$
0	1	1
0	0	0
0	0	0
1	0	1

也就是说，将三个逻辑回归的操作，进行叠加，便可以实现**异或**操作，从而对上面的例子进行非线性分类。换句话说，就是说**可以将线性分类器进行组合，从而实现非线性分类。**

x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1



线性分类器的逻辑与和逻辑或的**组合**可以完美的对平面样本进行分类。



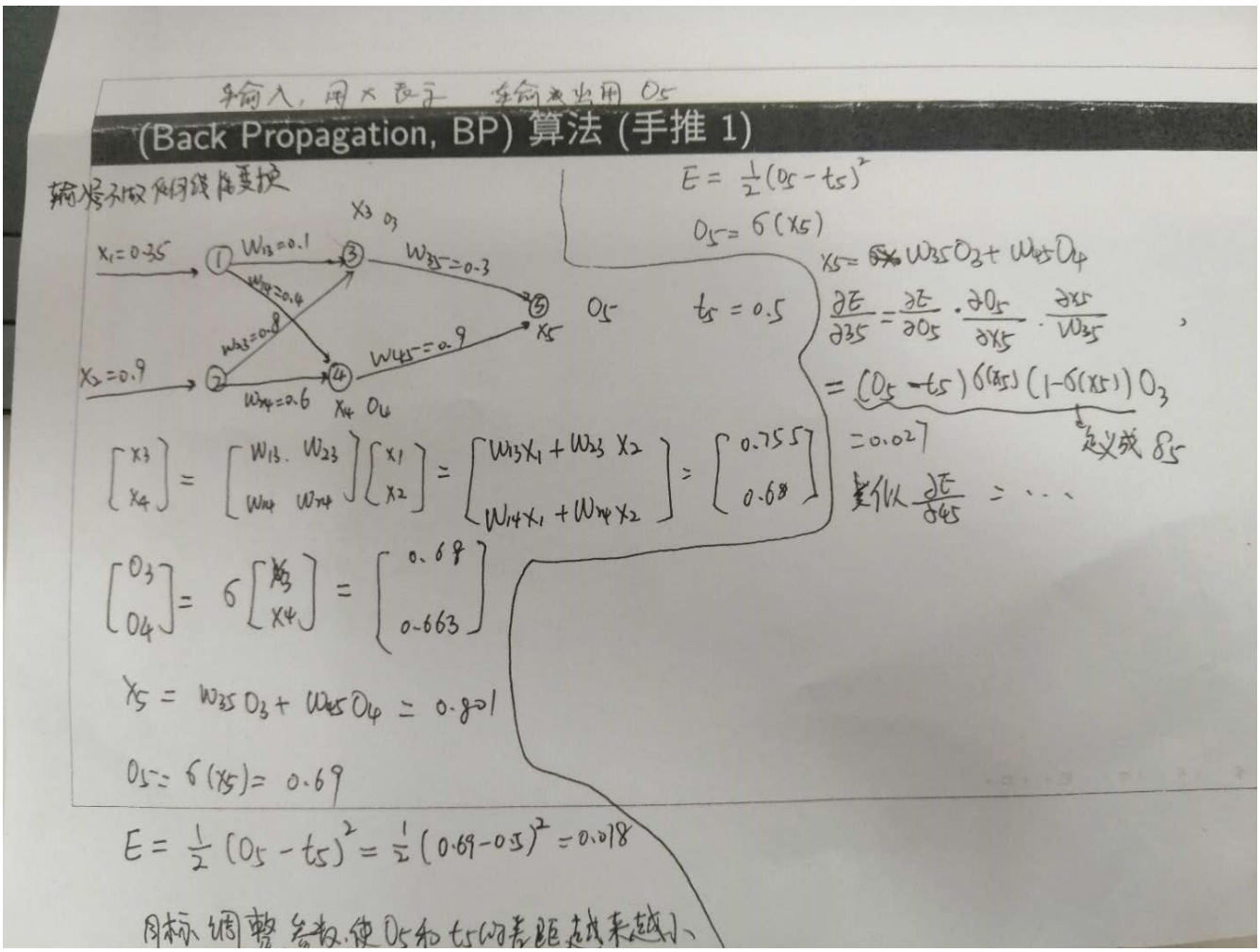
三：BP算法推导

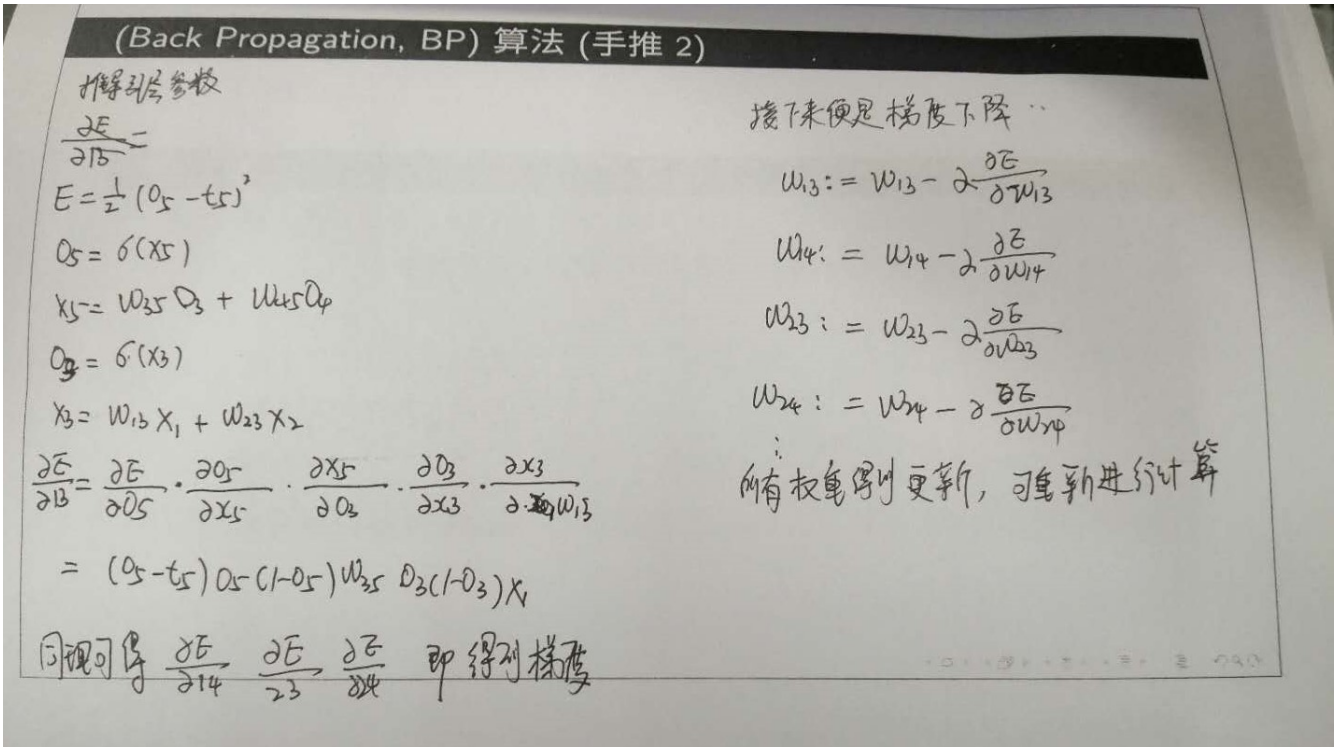
BP算法的思想为：学习过程由**信号的正向传播(求损失)**与**误差的反向传播(误差回传)**两个过程组成。

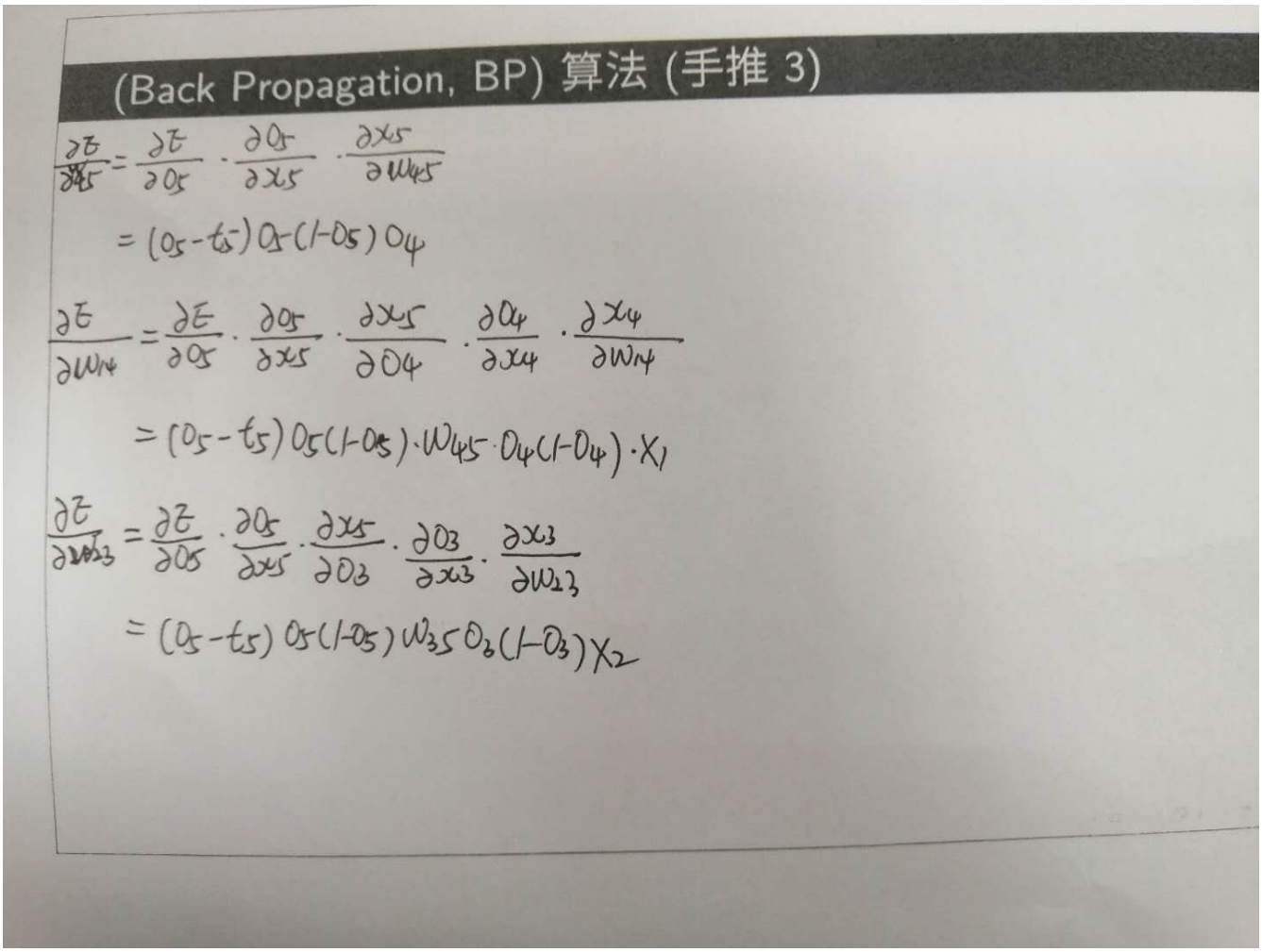
1) 正向传播FP(求损失).在这个过程中,我们根据输入的样本,给定的初始化权重值W和偏置项的值b, 计算最终输出值以及输出值与实际值之间的损失值.如果损失值不在给定的范围内则进行反向传播的过程; 否则停止W,b的更新.

2) 反向传播BP(回传误差).将输出以某种形式通过隐层向输入层逐层反传,并将误差分摊给各层的所有单元，从而获得各层单元的误差信号,此误差信号即作为修正各单元权值的依据。

3.1 手推过程




(Back Propagation, BP) 算法 (手推 2)
推导引号参数
$\frac{\partial E}{\partial b}=$
$E=\frac{1}{2}(o_5-t_5)^2$
$o_5=\sigma(x_5)$
$x_5=w_{35}o_3+w_{45}o_4$
$o_3=\sigma(x_3)$
$x_3=w_{13}x_1+w_{23}x_2$
$\frac{\partial E}{\partial b}=\frac{\partial E}{\partial o_5}\cdot\frac{\partial o_5}{\partial x_5}\cdot\frac{\partial x_5}{\partial o_3}\cdot\frac{\partial o_3}{\partial x_3}\cdot\frac{\partial x_3}{\partial w_{13}}$
$=(o_5-t_5)o_5(1-o_5)w_{35}o_3(1-o_3)x_1$
同理可得 $\frac{\partial E}{\partial w_{14}}$ $\frac{\partial E}{\partial w_{23}}$ $\frac{\partial E}{\partial w_{24}}$ 即得到梯度
接下来便是梯度下降：
$w_{13}:=w_{13}-\alpha\frac{\partial E}{\partial w_{13}}$
$w_{14}:=w_{14}-\alpha\frac{\partial E}{\partial w_{14}}$
$w_{23}:=w_{23}-\alpha\frac{\partial E}{\partial w_{23}}$
$w_{24}:=w_{24}-\alpha\frac{\partial E}{\partial w_{24}}$
所有权重得到更新，可继续进行计算


(Back Propagation, BP) 算法 (手推 3)
$\frac{\partial E}{\partial w_{45}}=\frac{\partial E}{\partial o_5}\cdot\frac{\partial o_5}{\partial x_5}\cdot\frac{\partial x_5}{\partial w_{45}}$
$=(o_5-t_5)o_5(1-o_5)o_4$
$\frac{\partial E}{\partial w_{14}}=\frac{\partial E}{\partial o_5}\cdot\frac{\partial o_5}{\partial x_5}\cdot\frac{\partial x_5}{\partial o_4}\cdot\frac{\partial o_4}{\partial x_4}\cdot\frac{\partial x_4}{\partial w_{14}}$
$=(o_5-t_5)o_5(1-o_5)\cdot w_{45}\cdot o_4(1-o_4)\cdot x_1$
$\frac{\partial E}{\partial w_{23}}=\frac{\partial E}{\partial o_5}\cdot\frac{\partial o_5}{\partial x_5}\cdot\frac{\partial x_5}{\partial o_3}\cdot\frac{\partial o_3}{\partial x_3}\cdot\frac{\partial x_3}{\partial w_{23}}$
$=(o_5-t_5)o_5(1-o_5)w_{35}o_3(1-o_3)x_2$

3.2 数学化总结

3.2.1 符号表示

- x_j^l:第l层第j个节点的输出;
- $w_{i,j}^l$:从层$l-1$中的节点i到层l中节点j的权重;
- $\sigma(x)=\frac{1}{1+e^{-x}}$
- θ_j^l:第l层第j个节点的偏置;
- O_j^l:第l层第j个节点的输出;
- t_j:目标值
- $E=\frac{1}{2}\sum_{k\in K}(O_k-t_k)^2$

3.2.2 输出层计算

目标:计算$\frac{\partial E}{\partial w_{j,k}}$

$$\begin{aligned}\frac{\partial E}{\partial w_{j,k}}&=\frac{\partial}{\partial O_k}\frac{1}{2}\sum_{k\in K}(O_k-t_k)^2\times\frac{\partial}{\partial x_k}O_k\times\frac{\partial}{\partial w_{j,k}}x_k\\&=(O_k-t_k)\times\frac{\partial}{\partial x_k}\sigma(x_k)\times O_j\\&=(O_k-t_k)\times\sigma(x_k)(1-\sigma(x_k))\times O_j\\&=\underbrace{(O_k-t_k)O_k(1-O_k)}_{\delta_k}O_j\end{aligned}$$

其中:$\delta_k=(O_k-t_k)O_k(1-O_k)$

3.2.3 隐层计算

$$\begin{aligned}\frac{\partial E}{\partial w_{i,j}}&=O_j(1-O_j)O_i\sum_{k\in K}(O_k-t_k)O_k(1-O_k)w_{j,k}\\&=O_iO_j(1-O_j)\sum_{k\in K}\delta_k w_{j,k}\\&=O_i\delta_j\end{aligned}$$

其中:$\delta_j=O_j(1-O_j)\sum_{k\in K}\delta_k w_{j,k}$

四:参考文章

1. https://www.cnblogs.com/lliuye/p/9183914.html
2. https://blog.csdn.net/qq_32241189/article/details/80305566

非常感谢博主的无私分享!