1. 线性分类

1.1 概念理解

分类问题在很多资料中都称之为逻辑回归，Logistic Regression，其原因是使用了线性回归中的线性模型，加上一个Logistic二分类函数，共同构造了一个分类器。

做二分类时，我们一般用Sigmoid函数做分类函数，那么和Sigmoid函数长得特别像的双曲正切函数能不能做分类函数呢？答案是可以的

Softmax函数是多分类问题的分类函数，通过对它的分析，可以学习多分类的原理、实现、以及可视化结果，从而理解神经网络的工作方式。

1.2 多入单出的单层神经网路 - 线性二分类

1.2.1 二分类函数

对率函数Logistic Function，即可以做为激活函数使用，又可以当作二分类函数使用。根据不同的任务区分激活函数和分类函数这两个概念，在二分类任务中，叫做Logistic函数，而在作为激活函数时，叫做Sigmoid函数。

- Logistic函数公式

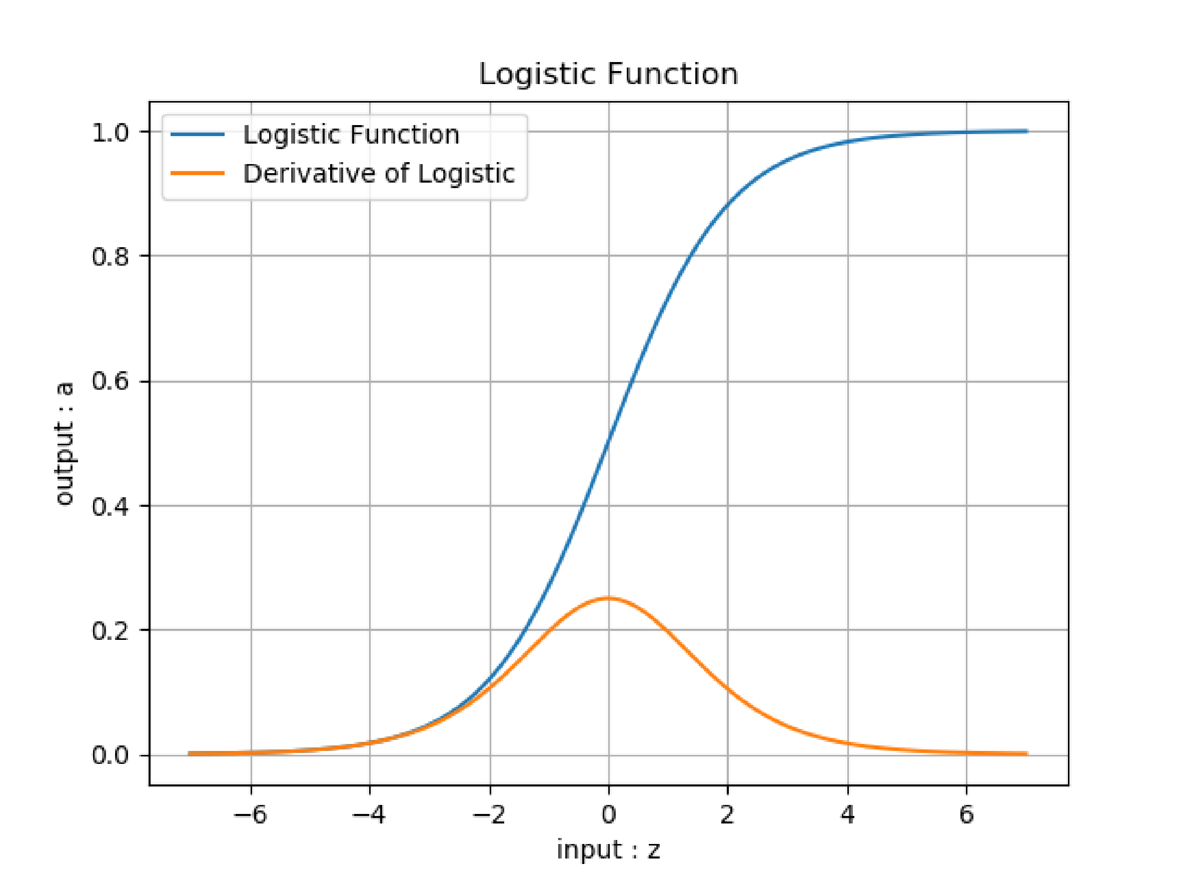
$Logistic(z) = \frac{1}{1+e^{-z}}$

以下记 $a = Logistic(z)$。

- 导数

$Logistic'(z) = a(1-a)$

图像：



1.2.1 正向传播

1.2.1.1 矩阵运算

$$z = x \cdot w + b \tag{1}$$

1.2.1.2 分类计算

$$a = Logistic(z) = \frac{1}{1+e^{-z}} \tag{2}$$

1.2.1.3 损失函数计算

二分类交叉熵损失函数：

$$loss(w,b) = -[y \ln a + (1-y)\ln(1-a)] \tag{3}$$

1.2.3 反向传播

1.2.3.1 求损失函数对 a 的偏导

$$\frac{\partial loss}{\partial a} = -\left[\frac{y}{a} - \frac{1-y}{1-a}\right] = \frac{a-y}{a(1-a)} \tag{4}$$

1.2.3.2 求 a 对 z 的偏导

$$\frac{\partial a}{\partial z} = a(1-a) \tag{5}$$

1.2.3.3 求误差 $loss$ 对 z 的偏导

使用链式法则链接公式4和公式5：

$$\frac{\partial loss}{\partial z} = \frac{\partial loss}{\partial a} \frac{\partial a}{\partial z} = \frac{a-y}{a(1-a)} \cdot a(1-a) = a-y \tag{6}$$

1.2.4 线性二分类原理

通过均方差函数误差反向传播的方法，不断矫正拟合直线的角度（Weights）和偏移（Bias），因为均方差函数能够准确地反映出当前的拟合程度。那么在线性分类中，我们能不能采取类似的方法呢？

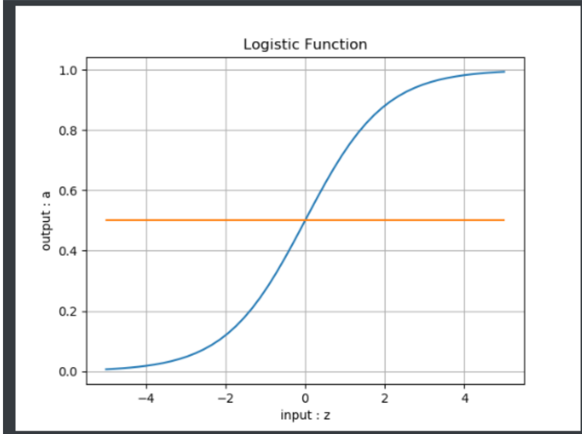
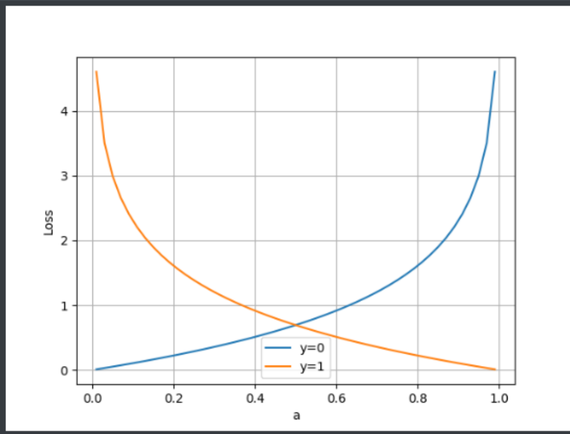
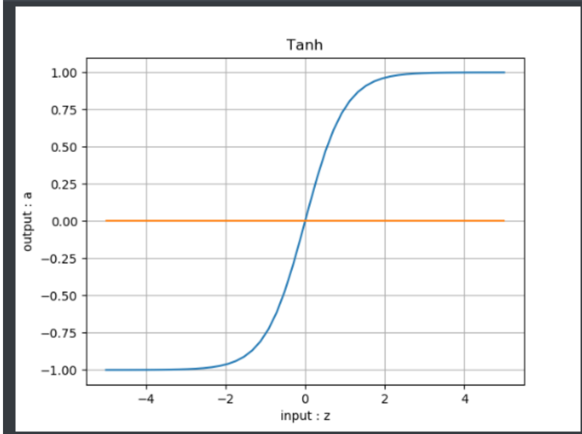
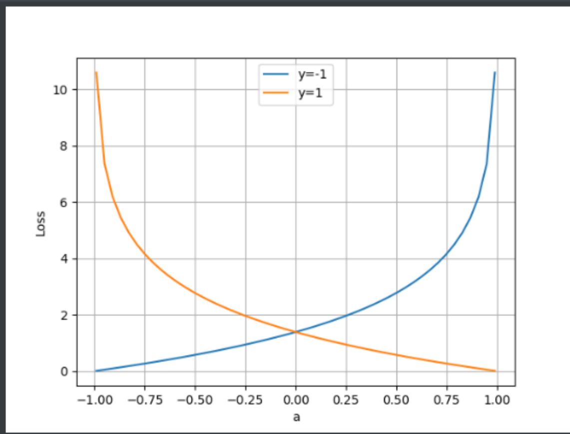
线性分类，试图在含有两种样本的空间中划出一条分界线，让双方截然分开，就好像是中国象棋的棋盘中的楚河汉界一样。与线性回归相似的地方是，两者都需要划出那条“直线”来，但是不同的地方也很明显，见表6-4。

表6-4 线性回归和线性分类的比较

	线性回归	线性分类
相同点	需要在样本群中找到一条直线	需要在样本群中找到一条直线
不同点	用直线来拟合所有样本，使得各个样本到这条直线的距离尽可能最短	用直线来分割所有样本，使得正例样本和负例样本尽可能分布在直线两侧

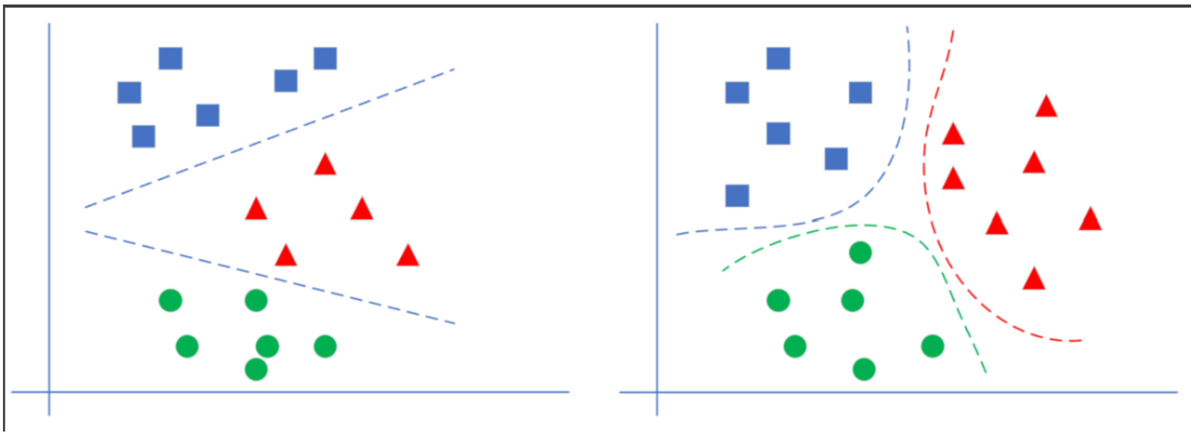
可以看到线性回归中的目标--“距离最短”，还是很容易理解的，但是线性分类的目标--“分布在两侧”，

可以用 用双曲正切函数做二分类函数，通过修改前向计算和反向传播函数，并且修改损失函数，为了增加准确度，修改样本数据标签值，

<table>
<tr><th>分类函数</th><th>交叉熵函数</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>输出值域a在(0,1)之间，分界线为a=0.5，标签值为y=0/1</td><td>y=0为负例，y=1为正例，输入值域a在(0,1)之间，符合对率函数的输出值域</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>输出值域a在(-1,1)之间，分界线为a=0，标签值为y=-1/1</td><td>y=-1为负例，y=1为正例，输入值域a在(-1,1)之间，符合双曲正切函数的输出值域</td></tr>
</table>

1.3 多入多出的单层神经网路 - 线性多分类

1.3.1 线性多分类和非线性多分类的区别



左侧为线性多分类，右侧为非线性多分类。它们的区别在于不同类别的样本点之间是否可以用一条直线来互相分割。对神经网络来说，线性多分类可以使用单层结构来解决，而分线性多分类需要使用双层结构。

1.3.2 多分类函数

对于三分类问题，我们要求 z 不是一个标量，而是一个向量，

1.3.2.1 引入Softmax

Softmax加了个"soft"来模拟max的行为，但同时又保留了相对大小的信息。

$$a_j = \frac{e^{z_j}}{\sum\limits_{i=1}^{m} e^{z_i}} = \frac{e^{z_j}}{e^{z_1} + e^{z_2} + \dots + e^{z_m}}$$

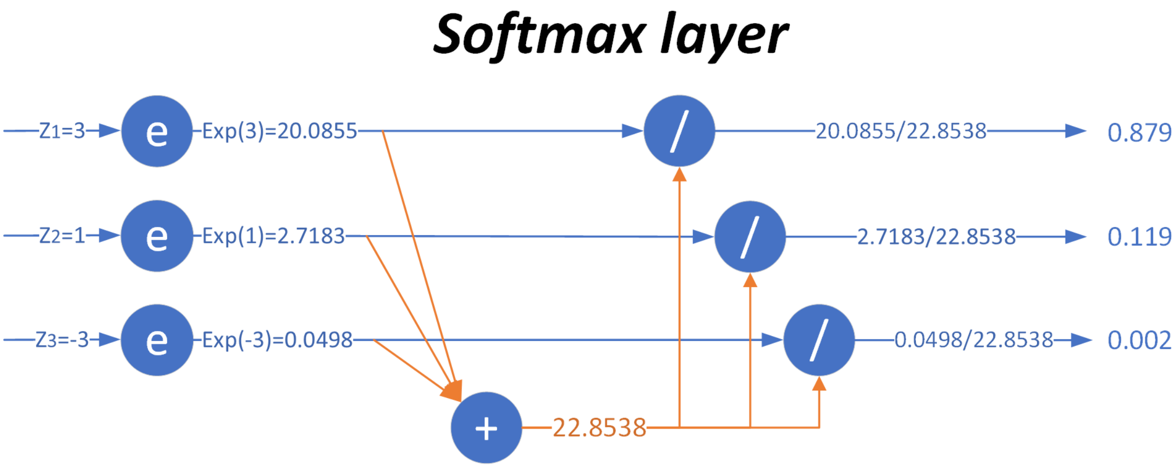
上式中:

- z_j 是对第 j 项的分类原始值，即矩阵运算的结果
- z_i 是参与分类计算的每个类别的原始值
- m 是总分类数
- a_j 是对第 j 项的计算结果

假设 $j = 1, m = 3$，上式为:

$a_1 = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$

用下图来形象地说明这个过程。



Softmax工作过程

当输入的数据$[z_1, z_2, z_3]$是$[3, 1, -3]$时，按照图示过程进行计算，可以得出输出的概率分布是$[0.879, 0.119, 0.002]$。

对比MAX运算和Softmax的不同，下表 MAX运算和Softmax的不同

输入原始值	MAX计算	Softmax计算
$[3, 1, -3]$	$[1, 0, 0]$	$[0.879, 0.119, 0.002]$

也就是说，在（至少）有三个类别时，通过使用Softmax公式计算它们的输出，比较相对大小后，得出该样本属于第一类，因为第一类的值为0.879，在三者中最大。注意这是对一个样本的计算得出的数值，而不是三个样本，亦即Softmax给出了某个样本分别属于三个类别的概率。

它有两个特点:

1. 三个类别的概率相加为1
2. 每个类别的概率都大于0

1.3.2.2 Softmax的反向传播工作原理

在使用Softmax之后，我们得到的值是 $a = [0.879, 0.119, 0.002]$，用 $a - y$：

$a - y = [-0.121, 0.119, 0.002]$

再来分析这个信息：

- 第一项-0.121是奖励给该类别0.121，因为它做对了，但是可以让这个概率值更大，最好是1
- 第二项0.119是惩罚，因为它试图给第二类0.119的概率，所以需要这个概率值更小，最好是0
- 第三项0.002是惩罚，因为它试图给第三类0.002的概率，所以需要这个概率值更小，最好是0

这个信息是完全正确的，可以用于反向传播。Softmax先做了归一化，把输出值归一到[0,1]之间，这样就可以与标签值的0或1去比较，并且知道惩罚或奖励的幅度。

1.3.2.3 正向传播

矩阵运算

$$z = x \cdot w + b \tag{1}$$

分类计算

$$a_j = \frac{e^{z_j}}{\sum\limits_{i=1}^m e^{z_i}} = \frac{e^{z_j}}{e^{z_1} + e^{z_2} + \dots + e^{z_m}} \tag{2}$$

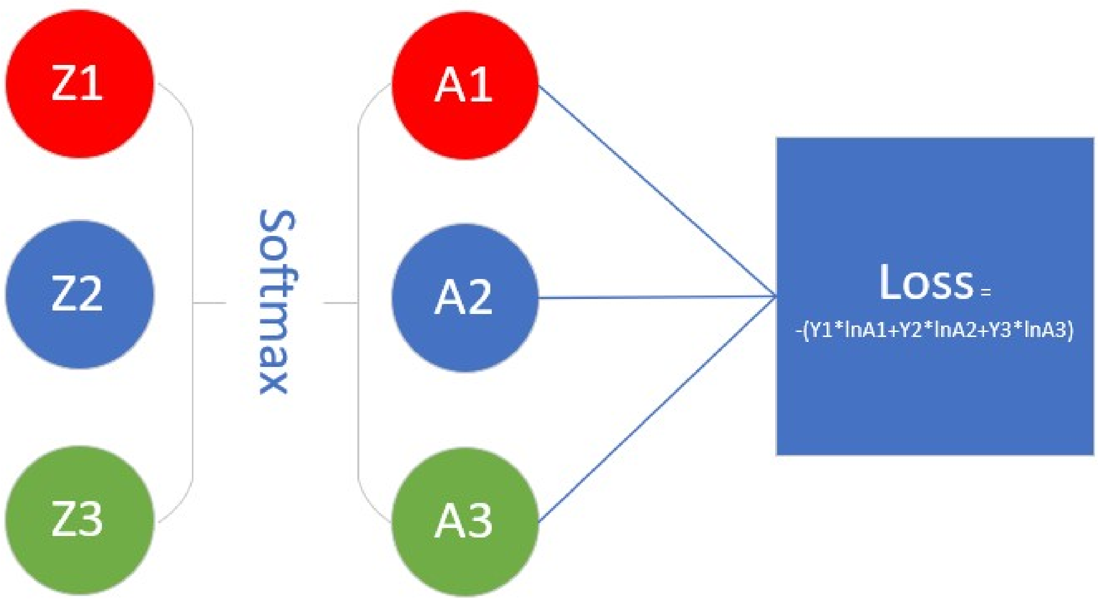
损失函数计算

计算单样本时，m是分类数：

$$loss(w,b) = -\sum_{i=1}^m y_i \ln a_i \tag{3}$$

计算多样本时，m是分类数，n是样本数：

$$J(w,b) = -\sum_{j=1}^n \sum_{i=1}^m y_{ji} \log a_{ji} \tag{4}$$



Softmax在神经网络结构中的示意图