1. 非线性回归

在两层神经网络之间，必须有激活函数连接，从而加入非线性因素，提高神经网络的能力。

1.1 激活函数

1.1.1 激活函数的作用

1. 给神经网络增加非线性因素。
2. 把公式1的计算结果压缩到 $[0, 1]$ 之间，便于后面的计算。

1.1.2 激活函数的 基本性质

- 非线性：线性的激活函数和没有激活函数一样；
- 可导性：做误差反向传播和梯度下降，必须要保证激活函数的可导性；
- 单调性：单一的输入会得到单一的输出，较大值的输入得到较大值的输出。

1.1.3 总结

1. 神经网络最后一层不需要激活函数
2. 激活函数只用于连接相邻的两层神经网络

1.2 挤压式激活函数

这一类函数的特点是，当输入值域的绝对值较大的时候，其输出在两端是饱和的，都具有S形的函数曲线以及压缩输入值域的作用，所以叫挤压型激活函数，又可以叫饱和型激活函数。

1.2.1 Logistic函数（对率函数）

对数几率函数（Logistic Function，简称对率函数）。

很多文字材料中通常把激活函数和分类函数混淆在一起说，有一个原因是：在二分类任务中最后一层使用的对率函数与在神经网络层与层之间连接的Sigmoid激活函数，是同样的形式。所以它既是激活函数，又是分类函数，是个特例。这个在之前的内容里面是存在的（二分类里面）

公式

$$Sigmoid(z) = \frac{1}{1+e^{-z}} \to a \qquad (1)$$

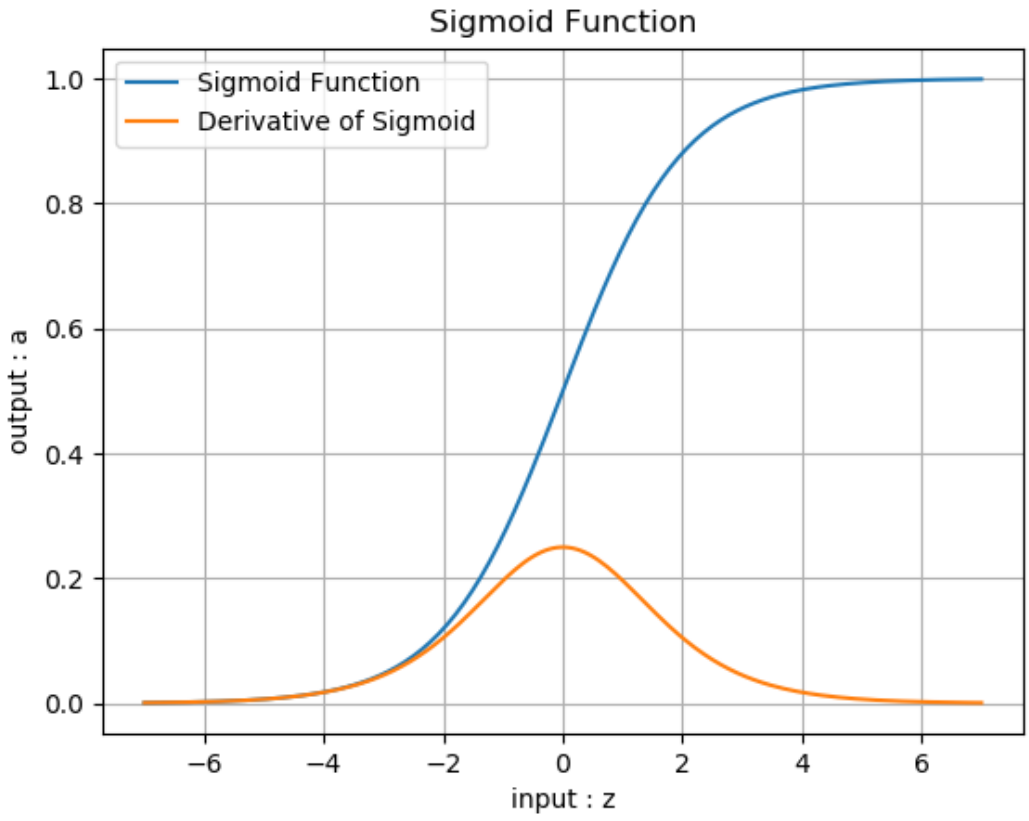
导数

$$Sigmoid'(z) = a(1-a) \qquad (2)$$

值域

- 输入值域：$(-\infty, \infty)$
- 输出值域：$(0, 1)$
- 导数值域：$(0, 0.25]$

函数图像



优点

从函数图像来看，Sigmoid函数的作用是将输入压缩到 $(0, 1)$ 这个区间范围内，

从数学上来看，Sigmoid函数对中央区的信号增益较大，对两侧区的信号增益小，在信号的特征空间映射上，有很好的效果。

从神经科学上来看，中央区酷似神经元的兴奋态，两侧区酷似神经元的抑制态，因而在神经网络学习方面，可以将重点特征推向中央区，
将非重点特征推向两侧区。

Sigmoid函数在这里就起到了如何把一个数值转化成一个通俗意义上的“把握”的表示。z坐标值越大，经过Sigmoid函数之后的结果就越接近1，把握就越大。

缺点

指数计算代价大。

反向传播时梯度消失：从梯度图像中可以看到，Sigmoid的梯度在两端都会接近于0，根据链式法则，如果传回的误差是δ，那么梯度传递函数是$\delta \cdot a'$，而a'这时接近零，也就是说整体的梯度也接近零。这就出现梯度消失的问题，并且这个问题可能导致网络收敛速度比较慢。

1.2.2 Tanh函数（双曲正切函数）

公式

$$Tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = (\frac{2}{1+e^{-2z}} - 1) \to a \qquad (3)$$

即

$$Tanh(z) = 2 \cdot Sigmoid(2z) - 1 \qquad (4)$$

导数

$Tanh'(z) = (1 + a)(1 - a)$

值域

- 输入值域：$(-\infty, \infty)$
- 输出值域：$(-1, 1)$
- 导数值域：$(0, 1)$

函数图像

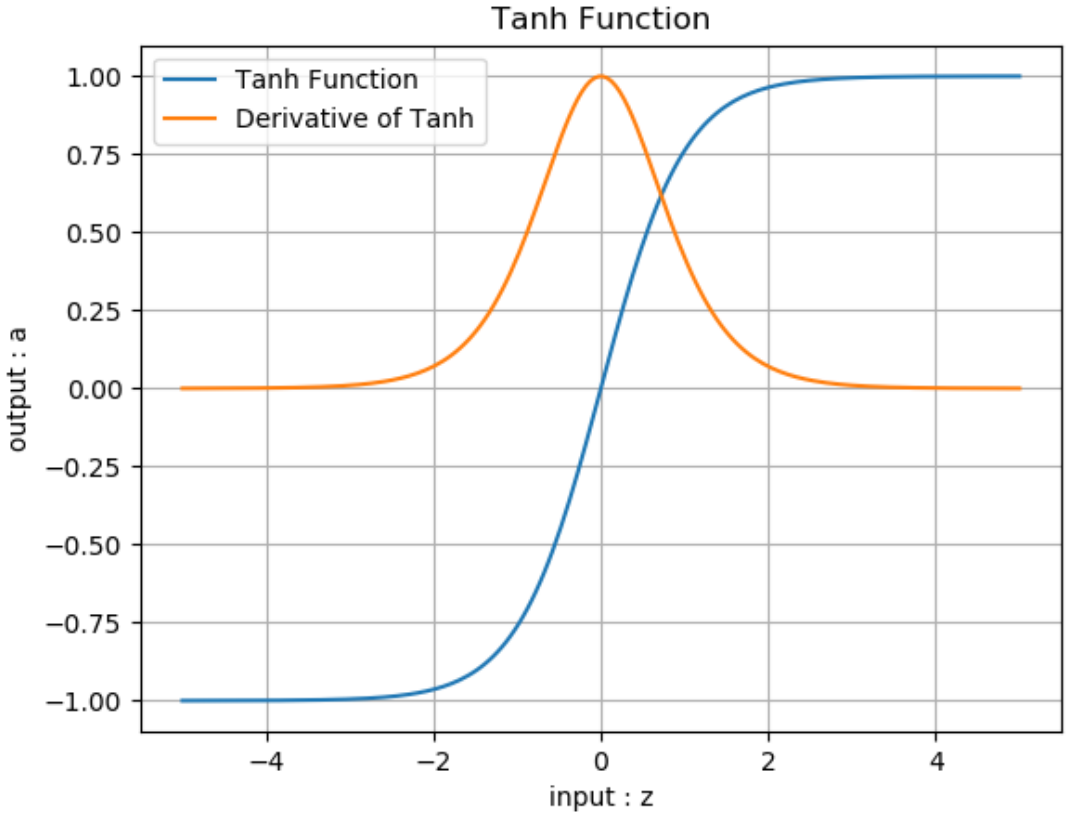


图8-4 双曲正切函数图像

优点

具有Sigmoid的所有优点。和Sigmoid非常相像，但是比起Sigmoid，Tanh减少了一个缺点，**就是他本身是零均值的**，也就是说，在传递过程中，输入数据的均值并不会发生改变，

缺点

exp指数计算代价大。梯度消失问题仍然存在。

1.3 半线性激活函数

1.3.1 ReLU函数

Rectified Linear Unit，修正线性单元，线性整流函数，斜坡函数。

公式

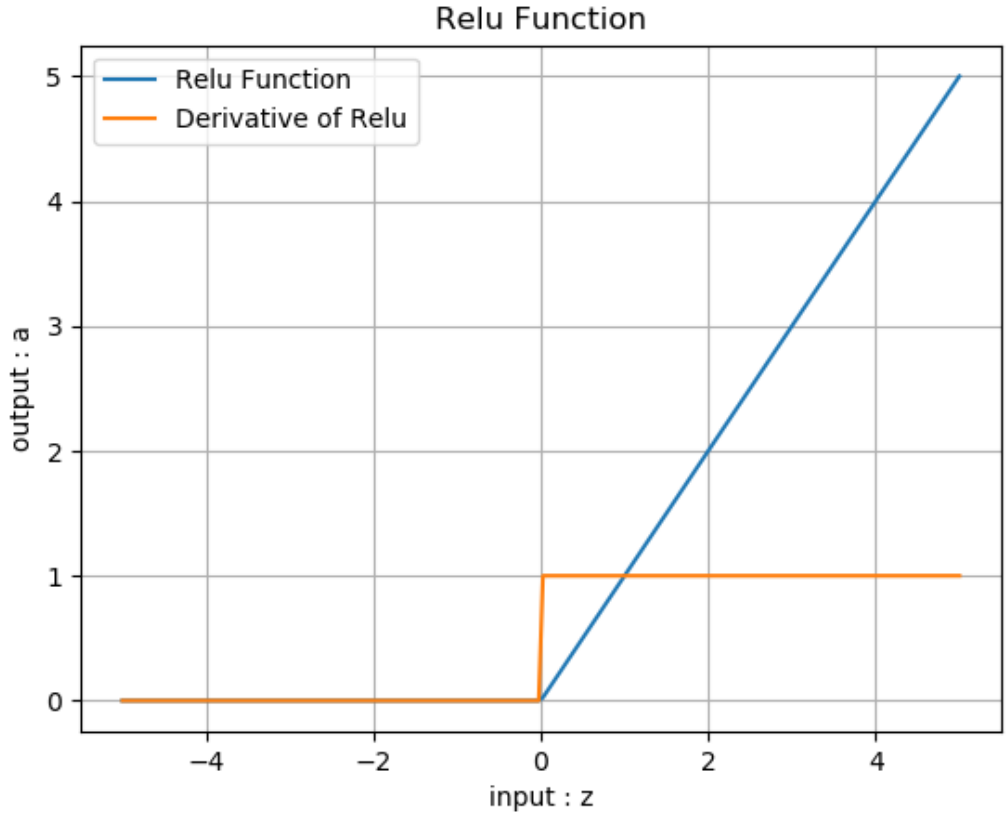
$$ReLU(z) = max(0, z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

导数

$$ReLU'(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

值域

- 输入值域：$(-\infty, \infty)$
- 输出值域：$(0, \infty)$
- 导数值域：$0, 1$



优点

- 反向导数恒等于1，更加有效率的反向传播梯度值，收敛速度快；
- 避免梯度消失问题；
- 计算简单，速度快；
- 活跃度的分散性使得神经网络的整体计算成本下降。

缺点

无界。

梯度很大的时候可能导致的神经元“死”掉。就是输入小于零时，ReLU回传的梯度是零，从而导致了后面的不更新。

1.3.2 Leaky ReLU函数

LReLU，带泄露的线性整流函数。

公式

$$LReLU(z) = \begin{cases} z & z \geq 0 \\ \alpha \cdot z & z < 0 \end{cases}$$

导数

$$LReLU'(z) = \begin{cases} 1 & z \geq 0 \\ \alpha & z < 0 \end{cases}$$

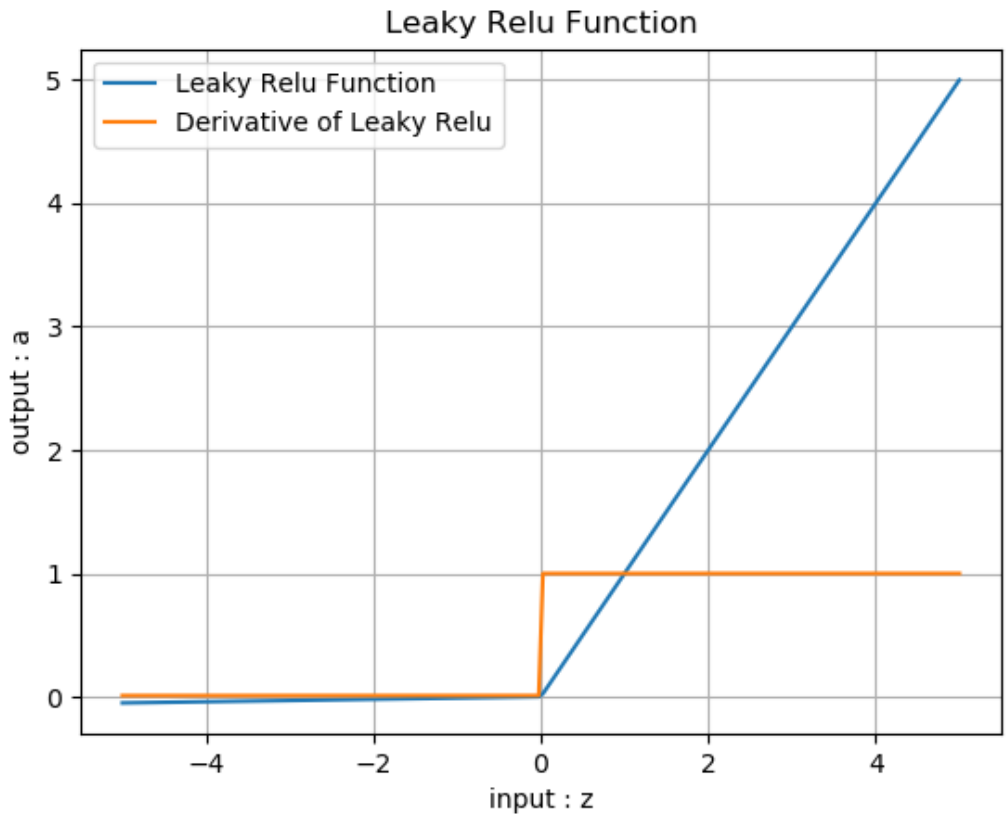
值域

输入值域：$(-\infty, \infty)$

输出值域：$(-\infty, \infty)$

导数值域：$\alpha, 1$

函数图像



优点

继承了ReLU函数的优点。

Leaky ReLU同样有收敛快速和运算复杂度低的优点，而且由于给了$z < 0$时一个比较小的梯度α,使得$z < 0$时依旧可以进行梯度传递和更新，可以在一定程度上避免神经元"死"掉的问题。

1.3.3 Softplus函数

公式

$$Softplus(z) = \ln(1 + e^z)$$

导数

$$Softplus'(z) = \frac{e^z}{1+e^z}$$

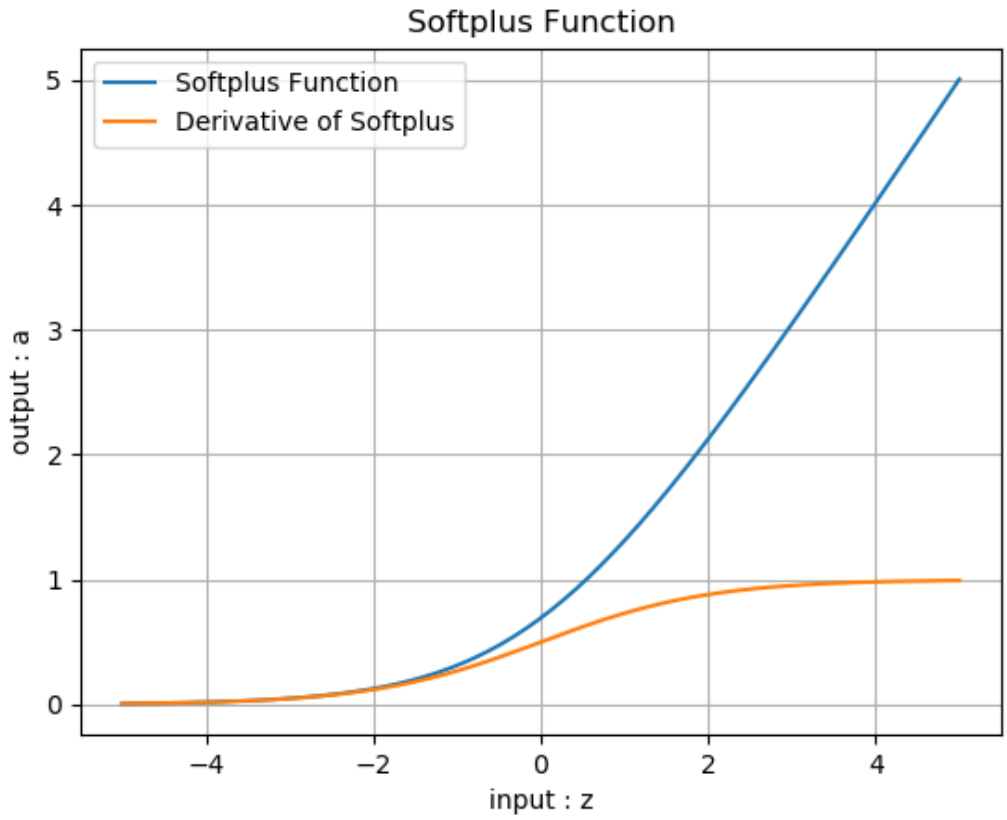
值域

输入值域：$(-\infty, \infty)$

输出值域：$(0, \infty)$

导数值域：$(0, 1)$

函数图像



1.3.4 ELU函数

公式

$$ELU(z) = \begin{cases} z & z \geq 0 \\ \alpha(e^z - 1) & z < 0 \end{cases}$$

导数

$$ELU'(z) = \begin{cases} 1 & z \geq 0 \\ \alpha e^z & z < 0 \end{cases}$$

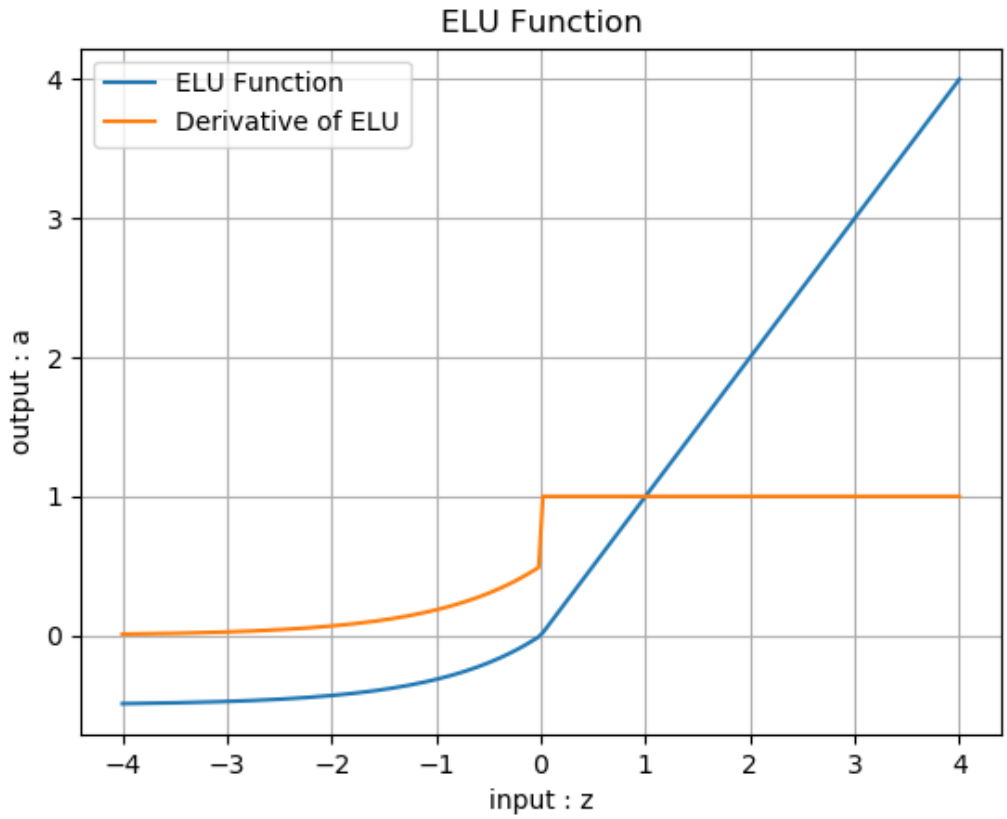
值域

输入值域：$(-\infty, \infty)$

输出值域：$(-\alpha, \infty)$

导数值域：$(0, 1]$

函数图像



2. 单入单出的双层神经网络 - 非线性回归

2.1 定义解释

即自变量X和因变量Y之间不是线性关系。常用的传统的处理方法有线性迭代法、分段回归法、迭代最小二乘法等。在神经网络中，解决这类问题的思路非常简单，就是使用带有一个隐层的两层神经网络。

2.2 回归模型的评估标准

回归问题主要是求值，评价标准主要是看求得值与实际结果的偏差有多大

2.2.1 平均绝对误差

MAE（Mean Abolute Error）。

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|a_i - y_i| \qquad (1)$$

对异常值不如均方差敏感，类似中位数。

2.2.2 绝对平均值率误差

MAPE（Mean Absolute Percentage Error）。

$$MAPE = \frac{100}{m}\sum_{i=1}^{m}\left|\frac{a_i - y_i}{y_i}\right| \qquad (2)$$

2.2.3 和方差

SSE（Sum Squared Error）。

$$SSE = \sum_{i=1}^{m}(a_i - y_i)^2 \qquad (3)$$

得出的值与样本数量有关系，假设有1000个测试样本，得到的值是120；如果只有100个测试样本，得到的值可能是11，我们不能说11就比120要好。

2.2.4 均方差

MSE（Mean Squared Error）。

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(a_i - y_i)^2 \qquad (4)$$

就是实际值减去预测值的平方再求期望，由于MSE计算的是误差的平方，所以它对异常值是非常敏感的，因为一旦出现异常值，MSE指标会变得非常大。MSE越小，证明误差越小。

2.2.5 均方根误差

RMSE（Root Mean Squard Error）。

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(a_i - y_i)^2} \qquad (5)$$

是均方差开根号的结果，其实质是一样的，只不过对结果有更好的解释。

2.2.6 R平方

R-Squared。

我们通常用概率来表达一个准确率，比如89%的准确率。

$$R^2 = 1 - \frac{\sum(a_i - y_i)^2}{\sum(\bar{y}_i - y_i)^2} = 1 - \frac{MSE(a,y)}{Var(y)} \qquad (6)$$

R平方是多元回归中的回归平方和（分子）占总平方和（分母）的比例，它是度量多元回归方程中拟合程度的一个统计量。R平方值越接近1，表明回归平方和占总平方和的比例越大，回归线与各观测点越接近，回归的拟合程度就越好。

- 如果结果是0，说明模型跟瞎猜差不多；
- 如果结果是1，说明模型无错误；
- 如果结果是0-1之间的数，就是模型的好坏程度；
- 如果结果是负数，说明模型还不如瞎猜。

2.3 多项式回归法正弦曲线 + 复合函数曲线

多项式回归确实可以解决复杂曲线拟合问题，但是代价有些高，我们训练了一百万次，才得到初步满意的结果。

2.4 测试和验证

2.4.1 训练集

Training Set，用于模型训练的数据样本。

2.4.2 验证集

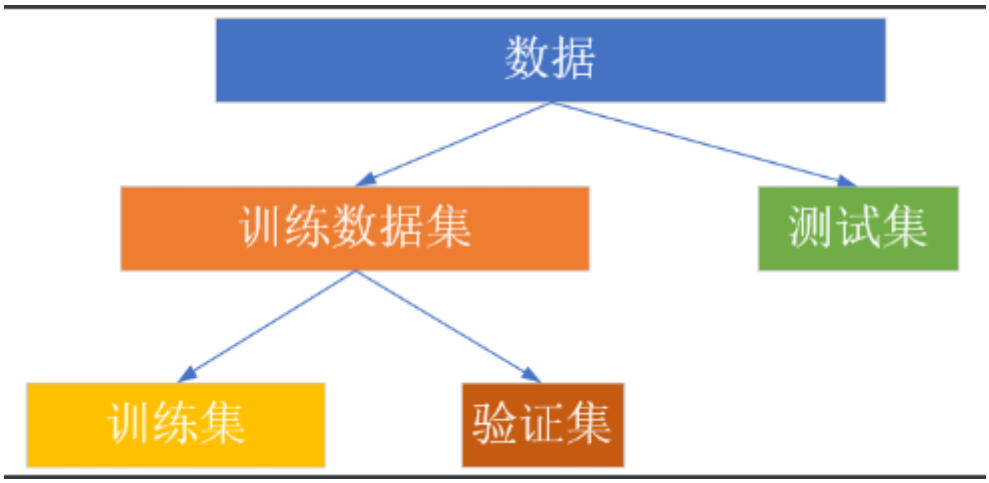
Validation Set，或者叫做Dev Set，是模型训练过程中单独留出的样本集，它可以用于调整模型的超参数和用于对模型的能力进行初步评估。

在神经网络中，验证数据集用于：

- 寻找最优的网络深度
- 或者决定反向传播算法的停止点
- 或者在神经网络中选择隐藏层神经元的数量
- 在普通的机器学习中常用的交叉验证（Cross Validation）就是把训练数据集本身再细分成不同的验证数据集去训练模型。

2.4.3 测试集

Test Set，用来评估最终模型的泛化能力。但不能作为调参、选择特征等算法相关的选择的依据。



2.4.4 交叉验证

2.4.4.1 传统的机器学习

在传统的机器学习中，我们经常用交叉验证的方法，比如把数据分成10份，$V_1 \sim V_{10}$，其中 $V_1 \sim V_9$ 用来训练，V_{10} 用来验证。然后用 $V_2 \sim V_{10}$ 做训练，V_1 做验证.....如此我们可以做10次训练和验证，大大增加了模型的可靠性。

2.4.4.2 神经网络/深度学习

比如在神经网络中，训练时到底迭代多少次停止呢？或者我们设置学习率为多少何时呢？或者用几个中间层，以及每个中间层用几个神经元呢？如何正则化？这些都是超参数设置，都可以用验证集来解决。

2.4.5 留出法

亦即从训练数据中保留出验证样本集，主要用于解决过拟合情况，这部分数据不用于训练

伪代码描述

```
for each epoch
    shuffle
    for each iteraion
        获得当前小批量数据
        前向计算
        反向传播
        更新梯度
        if is checkpoint
```

```
            用当前小批量数据计算训练集的loss值和accuracy值并记录
            计算验证集的loss值和accuracy值并记录
            如果loss值不再下降，停止训练
            如果accuracy值满足要求，停止训练
        end if
    end for
end for
```

2.5 双层神经网络实现非线性回归

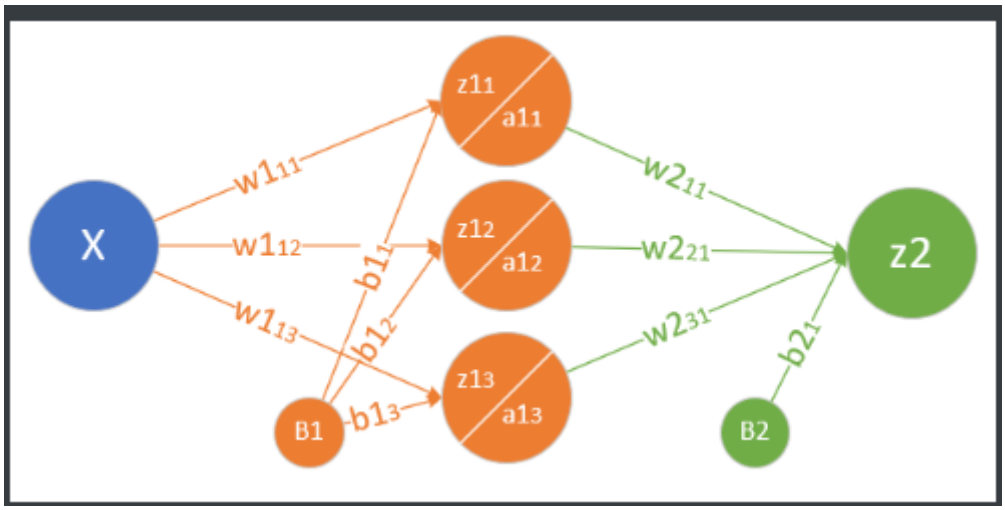
2.5.1 万能近似定理

万能近似定理(universal approximation theorem) [1]，是深度学习最根本的理论依据。它证明了在给定网络具有足够多的隐藏单元的条件下，配备一个线性输出层和一个带有任何“挤压”性质的激活函数（如Sigmoid激活函数）的隐藏层的前馈神经网络，能够以任何想要的误差量近似任何从一个有限维度的空间映射到另一个有限维度空间的Borel可测的函数。

万能近似定理其实说明了理论上神经网络可以近似任何函数。但实践上我们不能保证学习算法一定能学习到目标函数。即使网络可以表示这个函数，学习也可能因为两个不同的原因而失败：

1. 用于训练的优化算法可能找不到用于期望函数的参数值；
2. 训练算法可能由于过拟合而选择了错误的函数。

2.5.2 定义神经网络结构



输入层

输入层就是一个标量x值，如果是成批输入，则是一个矢量或者矩阵，但是特征值数量总为1，因为只有一个横坐标值做为输入。

$$X = (x)$$

权重矩阵W1/B1

$$W1 = (w1_{11} \quad w1_{12} \quad w1_{13})$$

$$B1 = (b1_1 \quad b1_2 \quad b1_3)$$

隐层

我们用3个神经元：

$$Z1 = (z1_1 \quad z1_2 \quad z1_3)$$

$$A1 = (a1_1 \quad a1_2 \quad a1_3)$$

权重矩阵W2/B2

W2的尺寸是3x1，B2的尺寸是1x1。

$$W2 = \begin{pmatrix} w2_{11} \\ w2_{21} \\ w2_{31} \end{pmatrix}$$

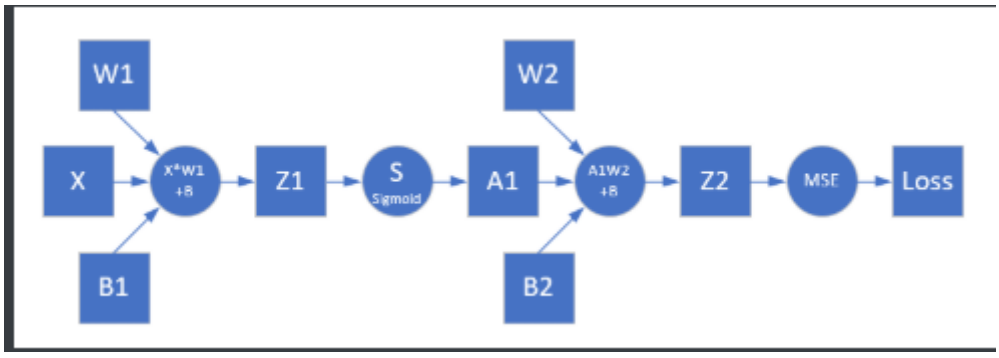
$$B2 = (b2_1)$$

输出层

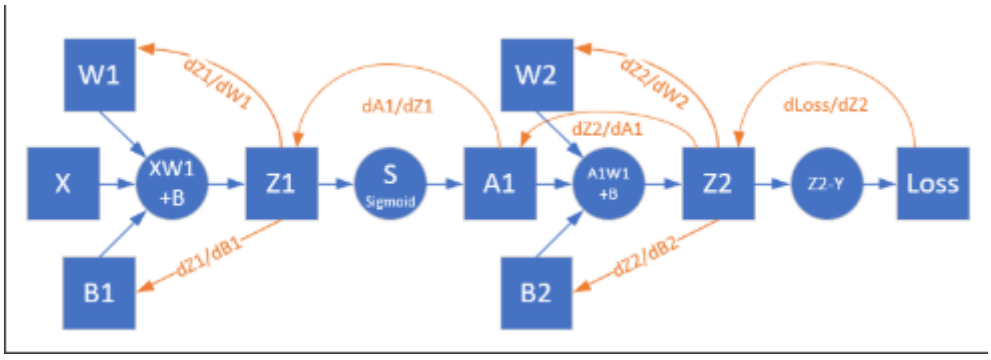
由于我们只想完成一个拟合任务，所以输出层只有一个神经元，尺寸为1x1：

$$Z2 = (z2_1)$$

2.5.3 前向计算



2.5.4 反向传播



2.6 非线性回归的工作原理

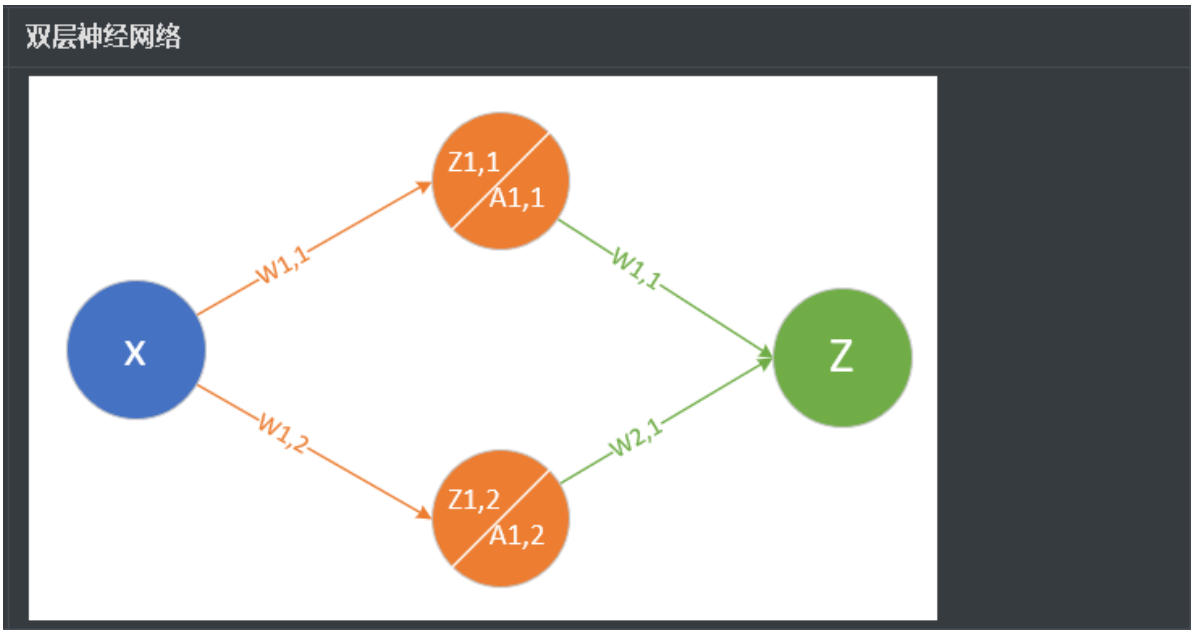
2.6.1 多项式为何能拟合曲线

单层神经网络的多项式回归法，需要x, x^2, x^3三个特征值，组成如下公式来得到拟合结果：

$$z = x \cdot w_1 + x^2 \cdot w_2 + x^3 \cdot w_3 + b \qquad (1)$$

本来一维的特征只能得到线性的结果，但是三维的特征就可以得到非线性的结果，这就是多项式拟合的原理。

2.6.2 神经网络的非线性拟合工作原理



通过线性变换的方式，把x变成了两部分：z_{11}/a_{11}，z_{12}/a_{12}，然后再通过一次线性变换把两者组合成为Z，这种方式和多项式回归非常类似：

1. 隐层把x拆成不同的特征，根据问题复杂度决定神经元数量，神经元的数量相当于特征值的数量；
2. 隐层通过激活函数做一次非线性变换；
3. 输出层使用多变量线性回归，把隐层的输出当作输入特征值，再做一次线性变换，得出拟合结果。

与多项式回归不同的是，不需要指定变换参数，而是从训练中学习到参数，这样的话权重值不会大得离谱。

2.6.3 比较多项式回归和双层神经网络解法

	多项式回归	双层神经网络
特征提取方式	特征值的高次方	线性变换拆分
特征值数量级	高几倍的数量级	数量级与原特征值相同
训练效率	低，需要迭代次数多	高，比前者少好几个数量级

2.7 参数调优

超参数优化（Hyperparameter Optimization）主要存在两方面的困难：

1. 超参数优化是一个组合优化问题，无法像一般参数那样通过梯度下降方法来优化，也没有一种通用有效的优化方法。
2. 评估一组超参数配置（Configuration）的时间代价非常高，从而导致一些优化方法（比如演化算法）在超参数优化中难以应用。

对于超参数的设置，比较简单的方法有人工搜索、网格搜索和随机搜索。

2.7.1 可调的参数

参数	缺省值	是否可调	注释
输入层神经元数	1	No	
隐层神经元数	4	Yes	影响迭代次数
输出层神经元数	1	No	
学习率	0.1	Yes	影响迭代次数
批样本量	10	Yes	影响迭代次数
最大epoch	10000	Yes	影响终止条件,建议不改动
损失门限值	0.001	Yes	影响终止条件,建议不改动
损失函数	MSE	No	
权重矩阵初始化方法	Xavier	Yes	

表中的参数，最终可以调节的其实只有三个：

- 隐层神经元数
- 学习率
- 批样本量

超参数	目标	作用	副作用
学习率	调至最优	低的学习率会导致收敛慢，高的学习率会导致错失最佳解	容易忽略其它参数的调整
隐层神经元数量	增加	增加数量会增加模型的表示能力	参数增多、训练时间增长
批大小	有限范围内尽量大	大批量的数据可以保持训练平稳，缩短训练时间	可能会收敛速度慢

2.7.2 网格搜索

类似于下面的这种组合形式，找到最优的

	eta=0.1	eta=0.3	eta=0.5	eta=0.7
ne=2	0.63	0.68	0.71	0.73
ne=4	0.86	0.89	0.91	0.3
ne=8	0.92	0.94	0.97	0.95
ne=12	0.69	0.84	0.88	0.87

2.7.3 随机搜索