

# Reactor

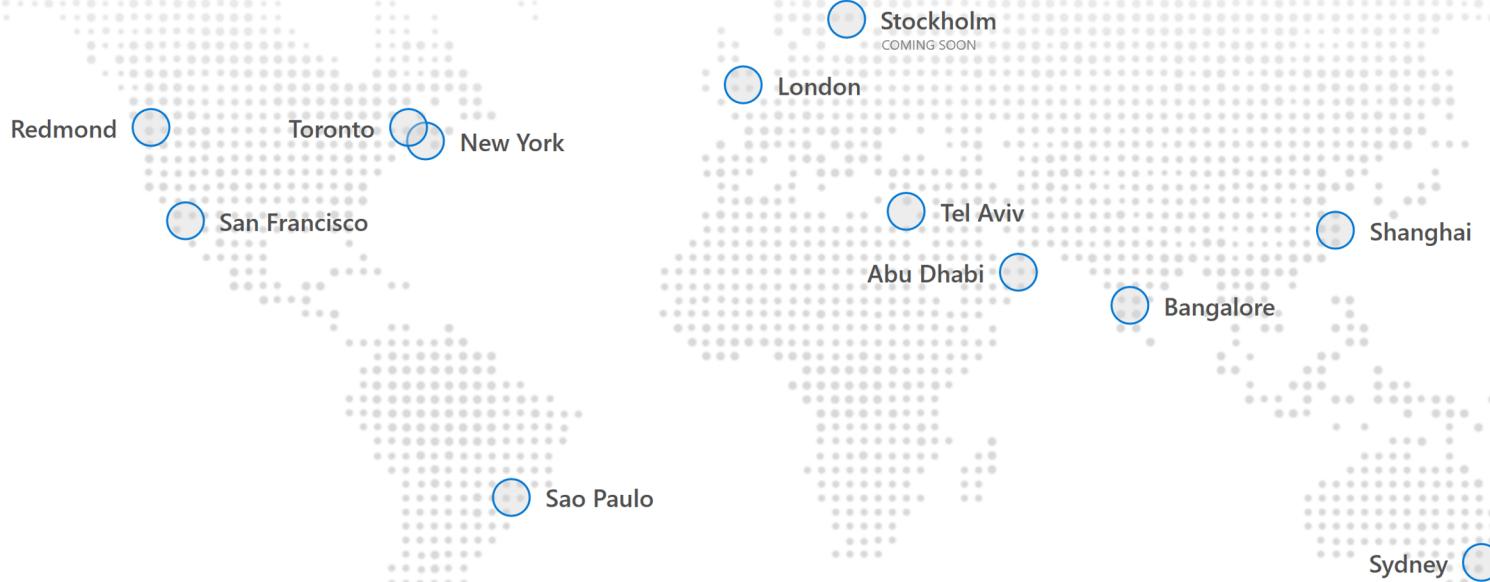
一起学人工智能系列  
- 知识回顾

---

2021-11-02



# Map



# 个人介绍



## Kinfey Lo – (卢建晖)

Microsoft Cloud Advocate

前微软MVP、Xamarin MVP和微软RD，拥有超过10年的云原生、人工智能和移动应用经验，为教育、金融和医疗提供应用解决方案。 Microsoft Ignite , TechEd 会议讲师，Microsoft AI 黑客马拉松教练，目前在微软，为技术人员和不同行业宣讲技术和相关应用场景。



爱编程(Python , C# , TypeScript , Swift , Rust , Go )

专注于人工智能，云原生，跨平台移动开发

Github : <https://github.com/kinfey>

Email : [kinfeylo@microsoft.com](mailto:kinfeylo@microsoft.com) Blog : <https://blog.csdn.net/kinfey>

Twitter : @Ljh8304

# 回顾

经过8周的学习，我们学会了

## 线性

分类

回归

## 非线性

分类

回归

# 回归 vs 分类



# 回归 – 以数学观点出发

回归，指研究一组随机变量( $Y_1, Y_2, \dots, Y_i$ )和另一组( $X_1, X_2, \dots, X_k$ )变量之间关系的统计分析方法，又称多重回归分析。通常 $Y_1, Y_2, \dots, Y_i$ 是因变量， $X_1, X_2, \dots, X_k$ 是自变量。

回归分析是一种数学模型。

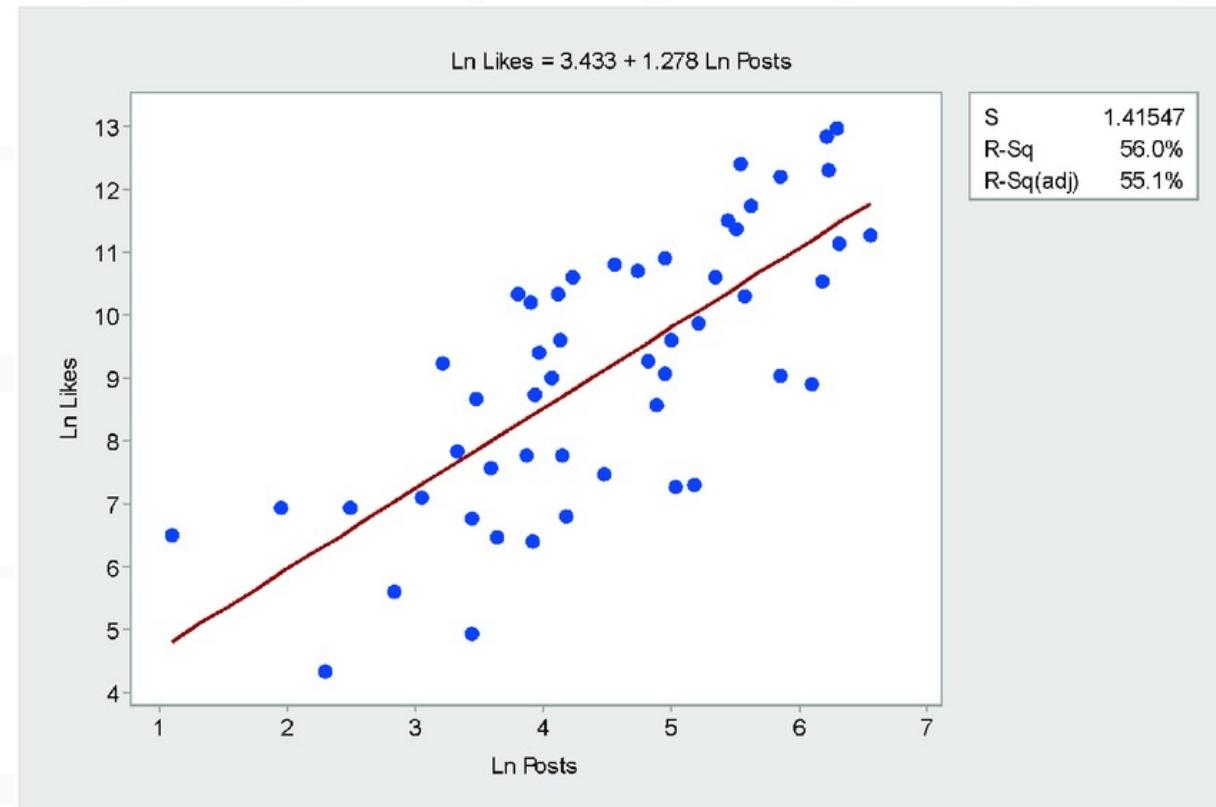
- ①从一组数据出发，确定某些变量之间的定量关系式；即建立数学模型并估计未知参数。通常用最小二乘法。
- ②检验这些关系式的可信任程度。
- ③在多个自变量影响一个因变量的关系中，判断自变量的影响是否显著，并将影响显著的选入模型中，剔除不显著的变量。通常用逐步回归、向前回归和向后回归等方法。
- ④利用所求的关系式对某一过程进行预测或控制。

# 回归

回归主要的种类有：线性回归、曲线回归、二元logistic回归、多元logistic回归。

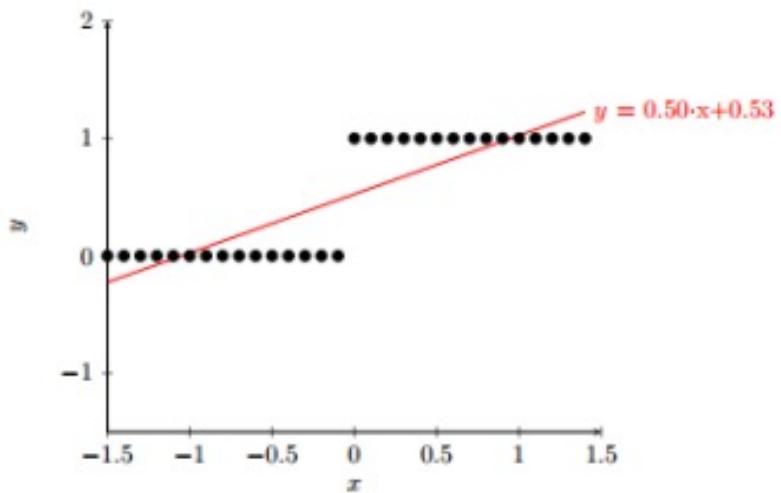
常用的一些应用场景

1. 销售量预测
2. 制造缺陷预测
3. 预测名人的离婚率
4. 预测所在地区的房价



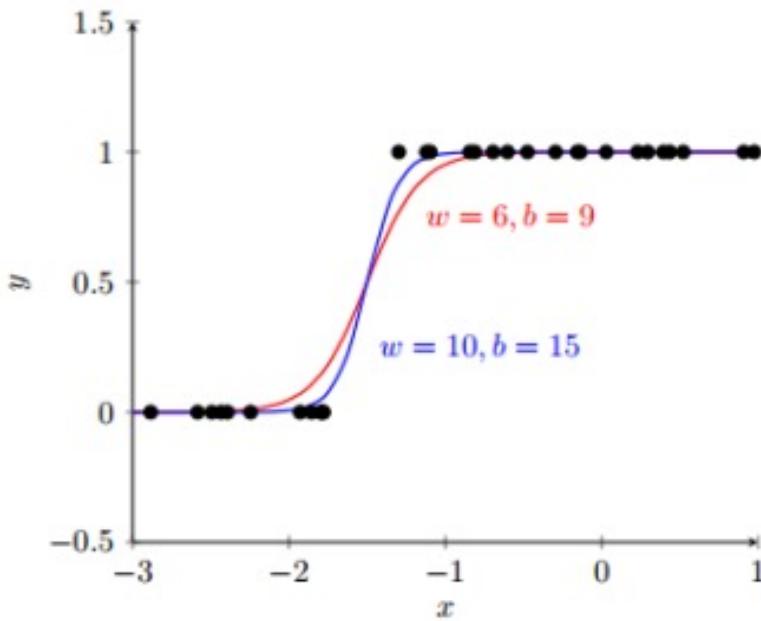
# 逻辑回归 (Logistic Regression)

逻辑回归 ( Logistic Regression ) , 回归给出的结果是事件成功或失败的概率。当因变量的类型属于二值 ( 1/0 , 真/假 , 是/否 ) 变量时 , 我们就应该使用逻辑回归



(a) 线性回归

线性回归使用一条直线拟合样本数据



(b) Logistic 回归

逻辑回归的目标是 “拟合” 0或1两个数值 , 而不是具体连续数值 , 所以称为广义线性模型

# 逻辑回归 (Logistic Regression)

探讨引发疾病的危险因素，并根据危险因素预测疾病发生的概率等。

以胃癌病情分析为例，选择两组人群，一组是胃癌组，一组是非胃癌组，两组人群必定具有不同的体征与生活方式等。因此因变量就为是否胃癌，值为“是”或“否”；自变量就可以包括很多了，如年龄、性别、饮食习惯、幽门螺杆菌感染等。

# 逻辑回归 (Logistic Regression)

自变量既可以是连续的，也可以是分类的。然后通过Logistic回归分析，可以得到自变量的权重，从而可以大致了解到底哪些因素是胃癌的危险因素。同时根据该权值可以根据危险因素预测一个人患癌症的可能性。

逻辑回归的另外一个名字叫做分类器，分为线性分类器和非线性分类器，本章中我们学习线性分类器。而无论是线性还是非线性分类器，又分为两种：二分类问题和多分类问题



# 回归 vs 分类

## Regression (回归)

预测明天的气温是多少度

## Classification (分类)

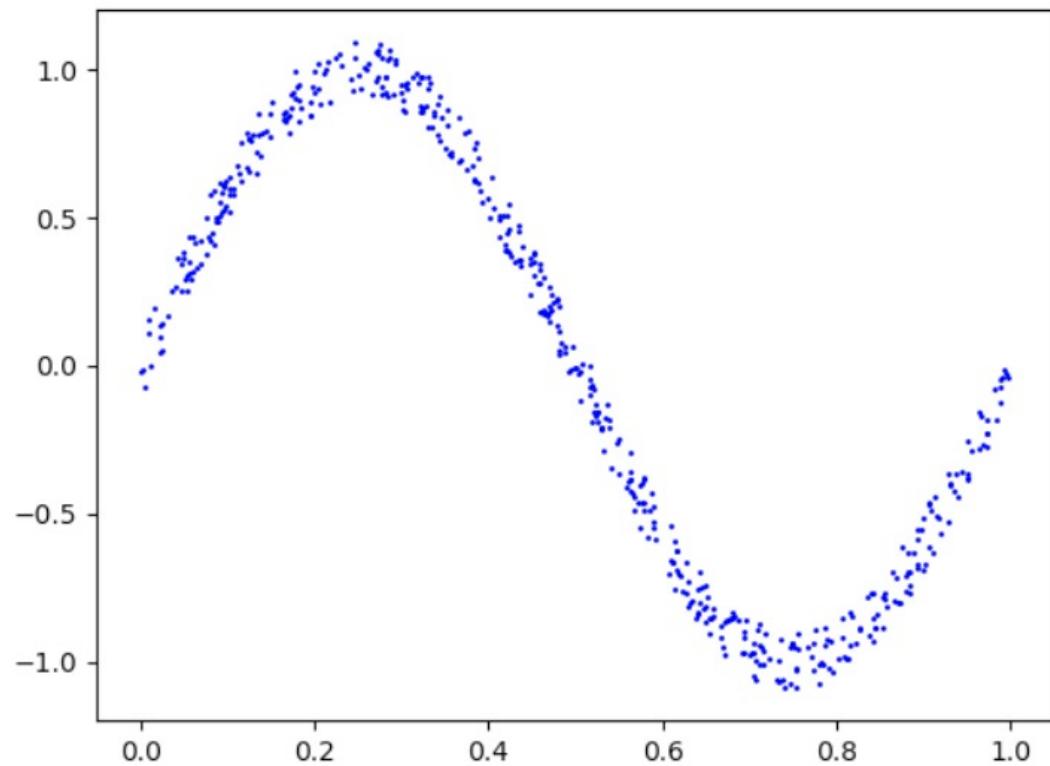
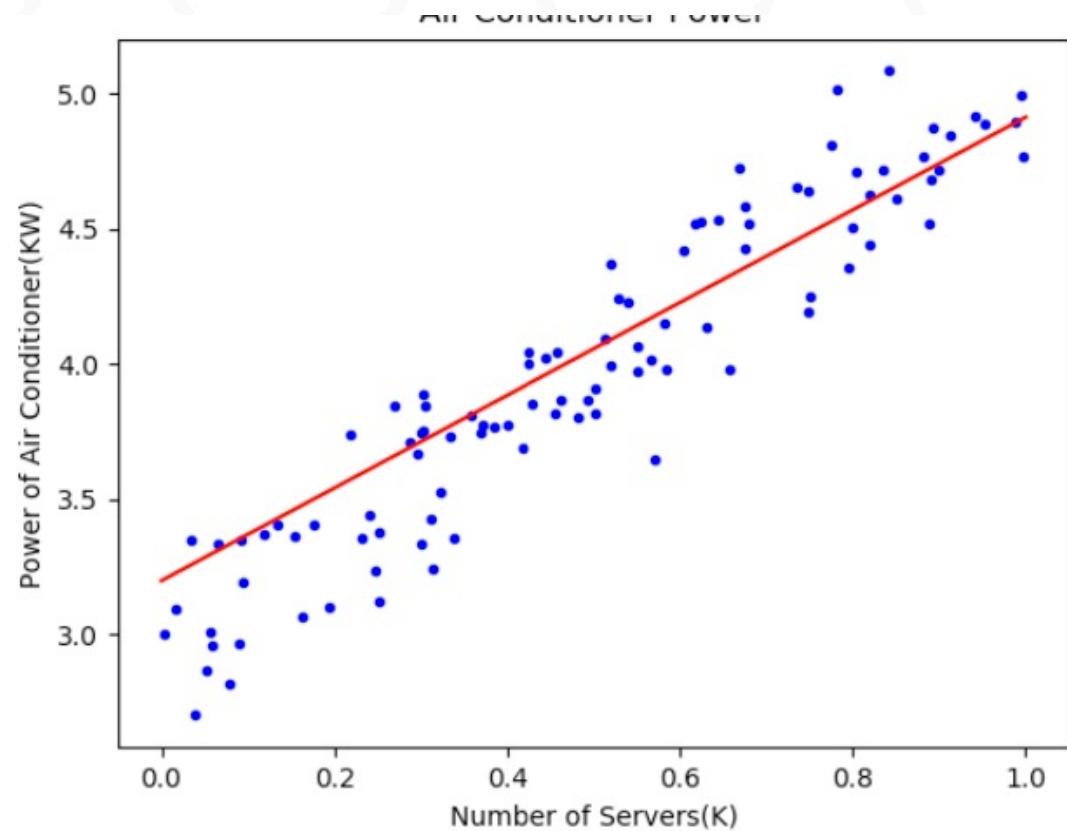
预测明天是阴、晴还是雨

输出	连续数据	离散数据
目的	定量- 找到最佳拟合	定性- 决策边界
评价	拟合度	精度
场景	预测房价, 天气	垃圾邮件, 物品分类

# 线性 vs 非线形



# 线性回归 vs 非线性回归



# 单变量线性回归问题

回归分析是一种数学模型。当因变量和自变量为线性关系时，它是一种特殊的线性模型。

最简单的情形是一元线性回归，由大体上有线性关系的一个自变量和一个因变量组成，模型是：

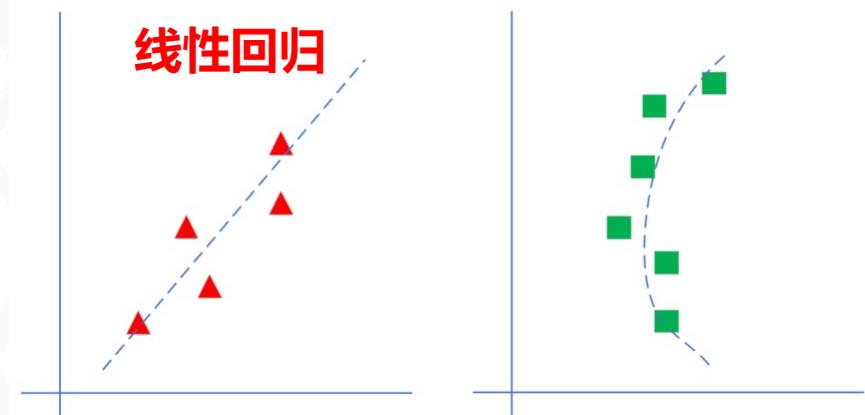
$$y = a + bX + \varepsilon$$

X是自变量，Y是因变量， $\varepsilon$ 是随机误差，a和b是参数，在线性回归模型中，a,b是要通过算法学习出来的

# 线性回归模型

对于线性回归模型，有如下一些概念需要了解：

- 通常假定随机误差  $\epsilon$  的均值为 0，方差为  $\sigma^2$  ( $\sigma^2 > 0$ ,  $\sigma^2$  与  $X$  的值无关)
- 若进一步假定随机误差遵从正态分布，就叫做正态线性模型
- 一般地，若有  $k$  个自变量和 1 个因变量（即公式1中的  $Y$ ），则因变量的值分为两部分：一部分由自变量影响，即表示为它的函数，函数形式已知且含有未知参数；另一部分由其他的未考虑因素和随机性影响，即随机误差
- 当函数为参数未知的线性函数时，称为线性回归分析模型
- 当函数为参数未知的非线性函数时，称为非线性回归分析模型
- 当自变量个数大于 1 时称为多元回归
- 当因变量个数大于 1 时称为多重回归



# 多元线性回归

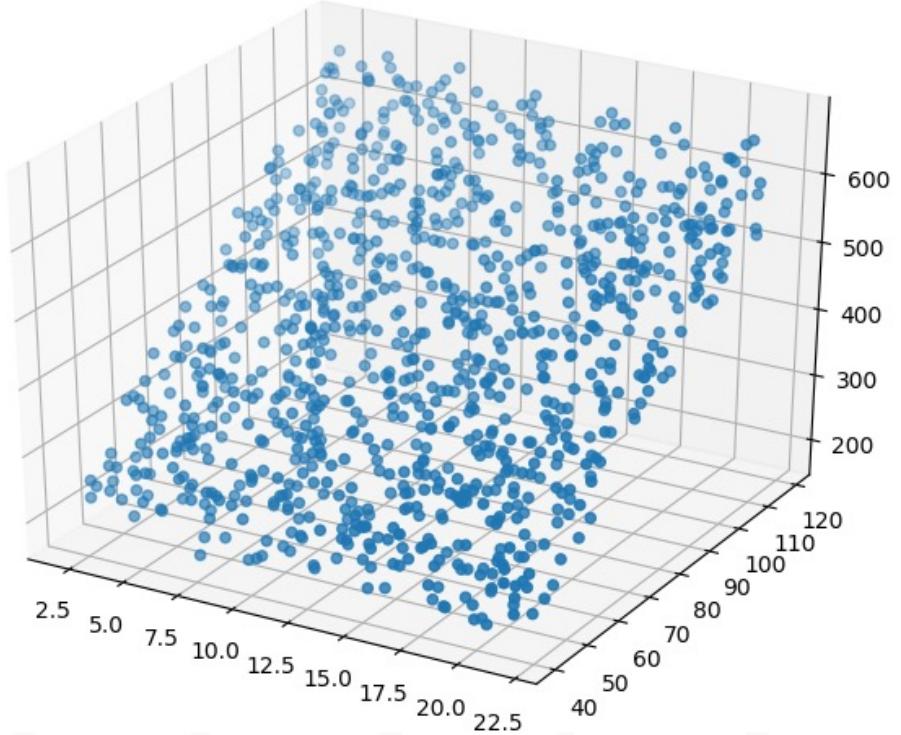
线性回归问题，而且是典型的多元线性回归，即包括两个或两个以上自变量的回归。多元线性回归的函数模型如下：

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_kx_k$$

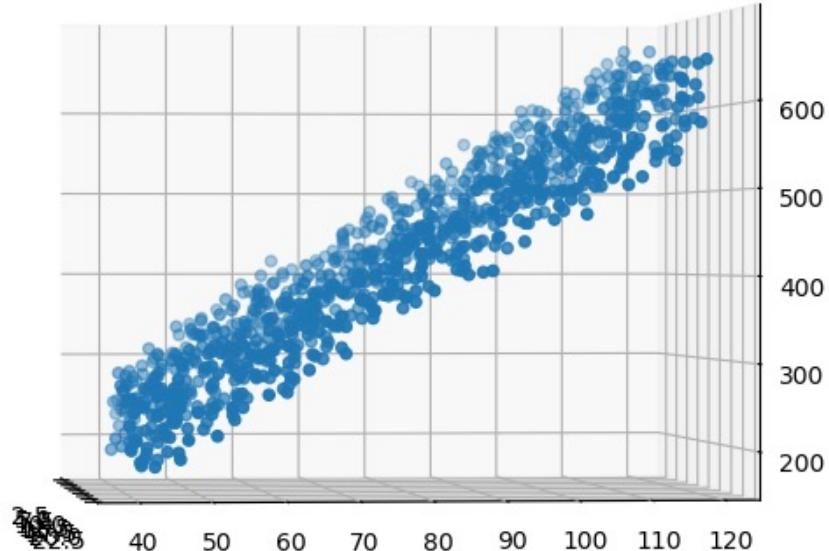
具体化到房价预测问题，上面的公式可以简化成：

$$z = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

# 房价



正向



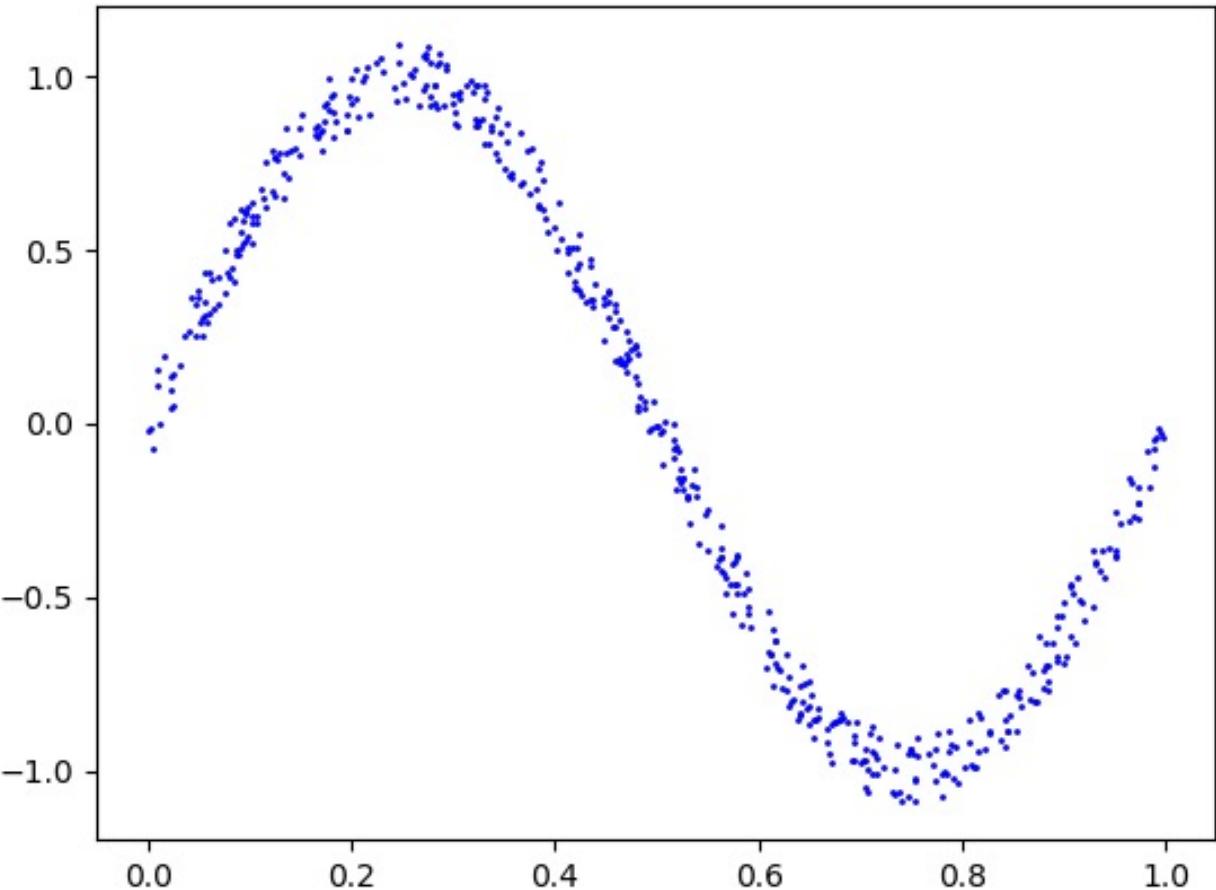
侧向

# 多元线性回归

- 1.自变量对因变量必须有显著的影响，并呈密切的线性相关；
- 2.自变量与因变量之间的线性相关必须是真实的，而不是形式上的；
- 3.自变量之间应具有一定的互斥性，即自变量之间的相关程度不应高于自变量与因变量之因的相关程度；
- 4.自变量应具有完整的统计数据，其预测值容易确定。

# 非线性回归

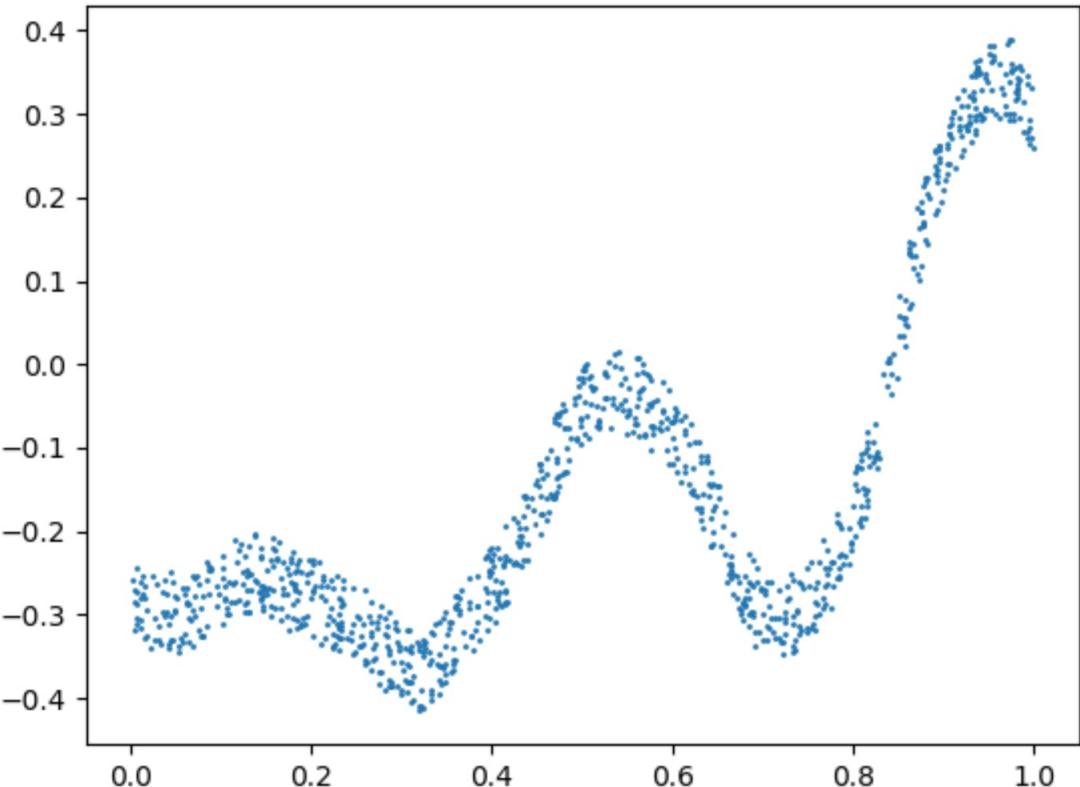
## 正弦曲线



样本	x	y
1	0.1199	0.6108
2	0.0535	0.3832
3	0.6978	0.9496
...	...	...

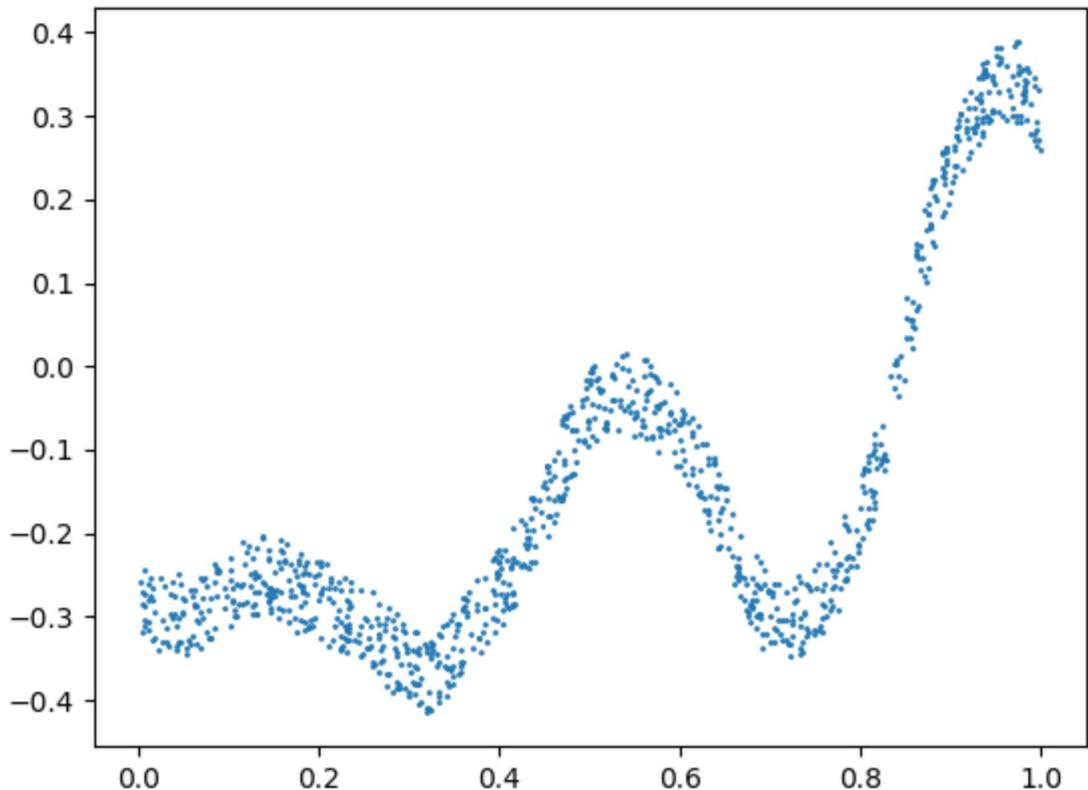
# 非线性回归

## 更复杂的曲线



样本	x	y
1	0.606	-0.113
2	0.129	-0.269
3	0.582	0.027
...	...	...
1000	0.199	-0.281

# 非线性回归



$$y = 0.4x^2 + 0.3x\sin(15x) + 0.01\cos(50x) - 0.3$$

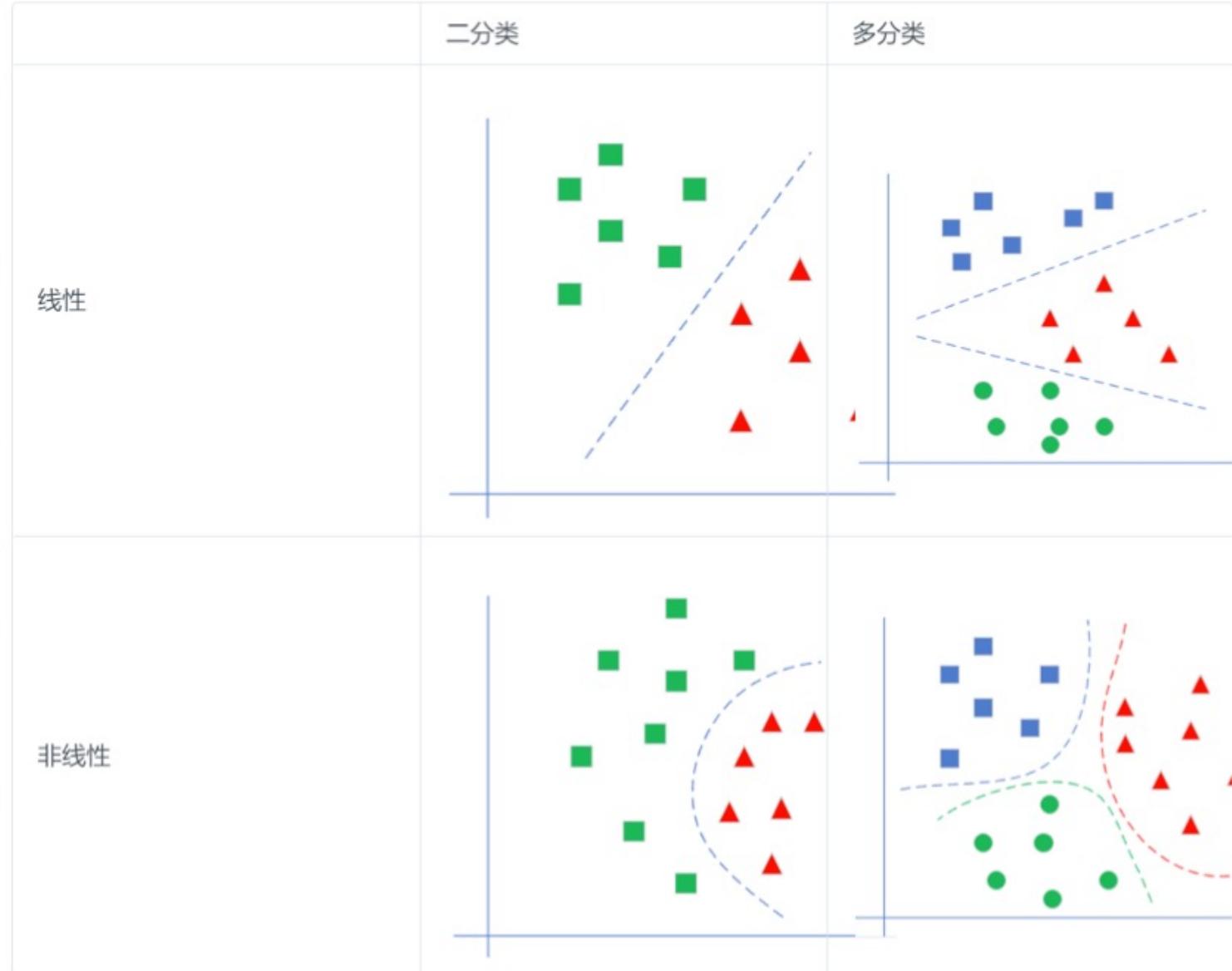
我们特意把数据限制在[0,1]之间，避免做归一化的麻烦。要是觉得这个公式还不够复杂，大家可以用更复杂的公式去自己做试验。

以上问题可以叫做非线性回归，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。

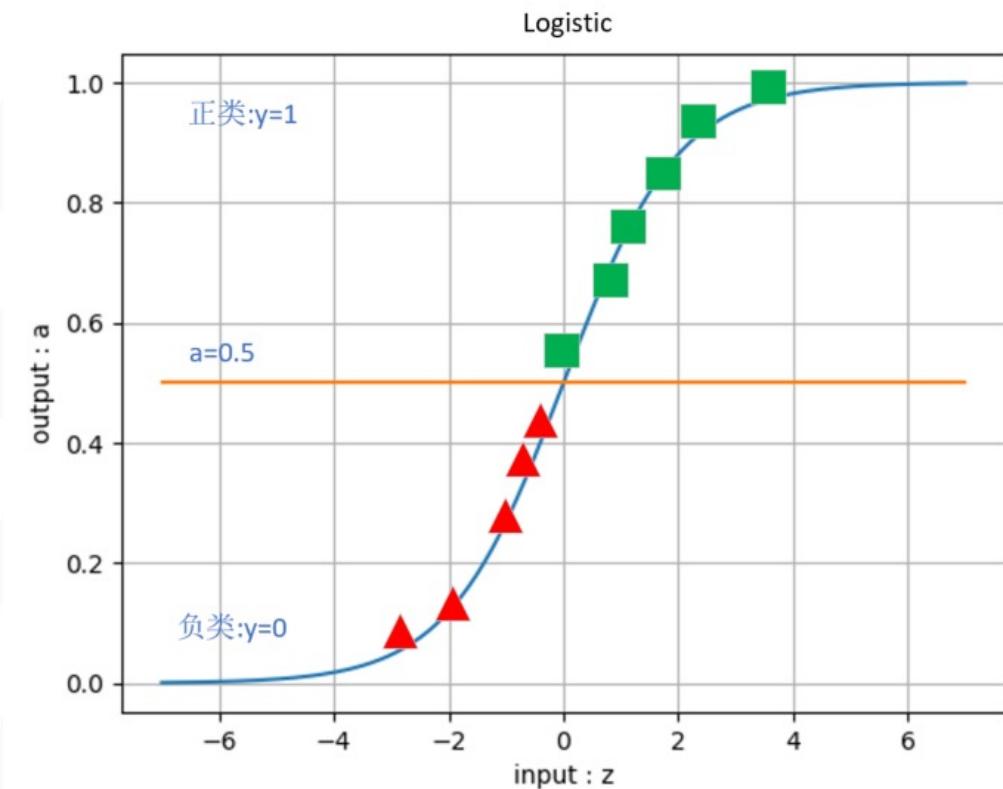
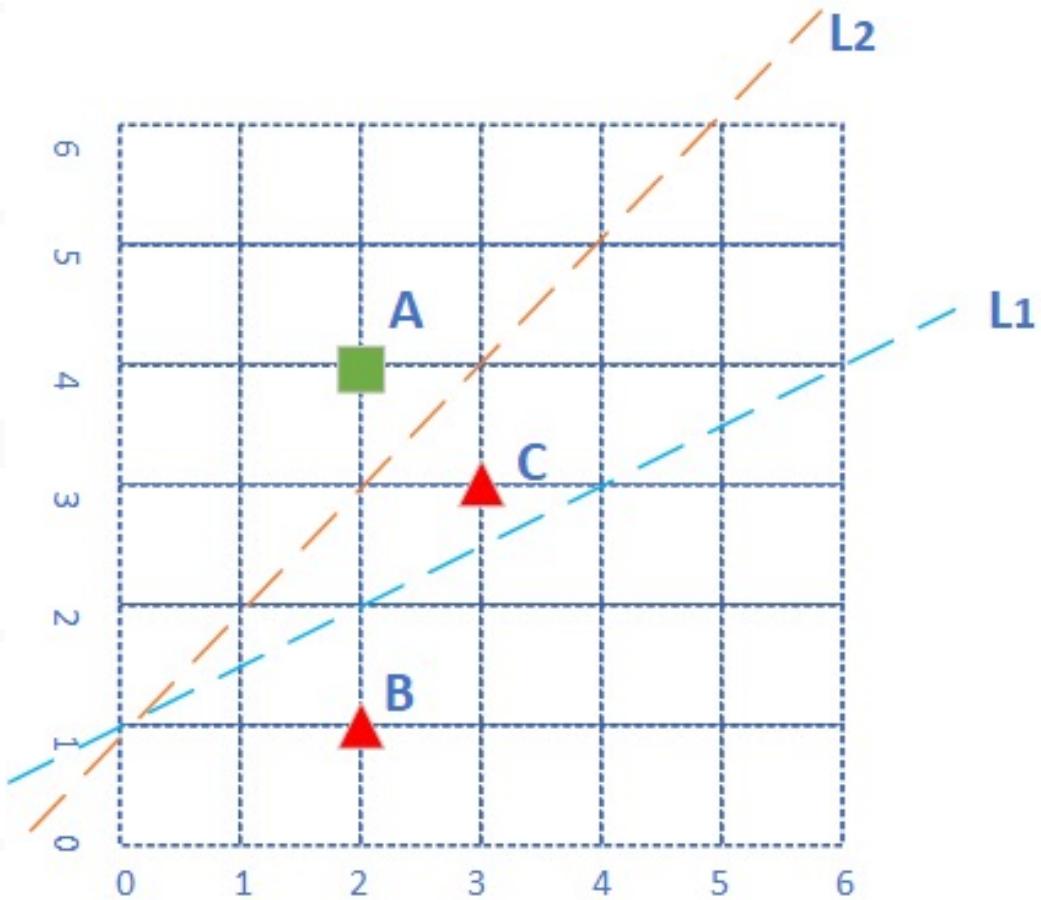
# 线性分类 vs 非线性分类

## 分类的含义：

从复杂程度上分，有线性/非线性之分；  
从样本类别上分，有二分类/多分类之分。

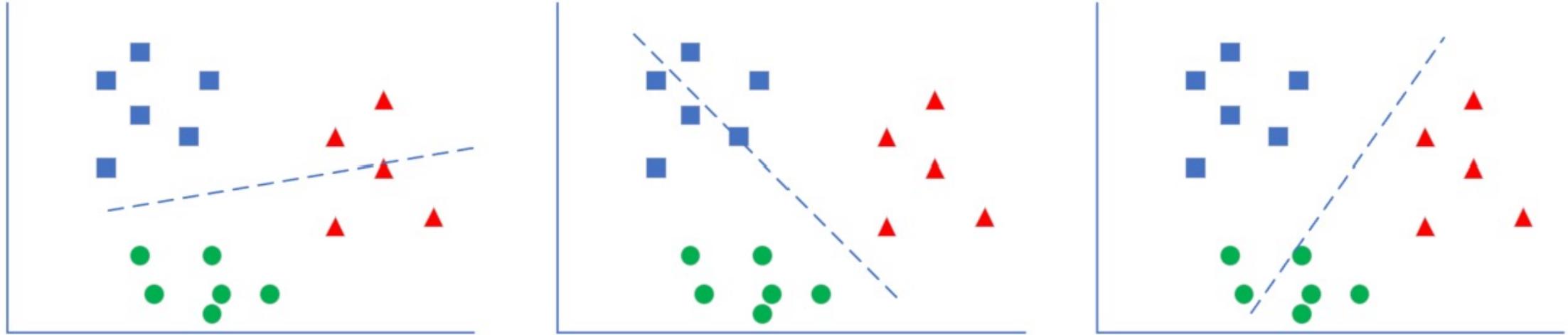


# 线性二分类



# 多分类问题解法 - 一对一边式

每次先只保留两个类别的数据，训练一个分类器。如果一共有N个类别，则需要训练 $C_N^2$ 个分类器。以N=3时举例，需要训练A|B, B|C, A|CA|B, B|C, A|CA|B, B|C, A|C三个分类器。

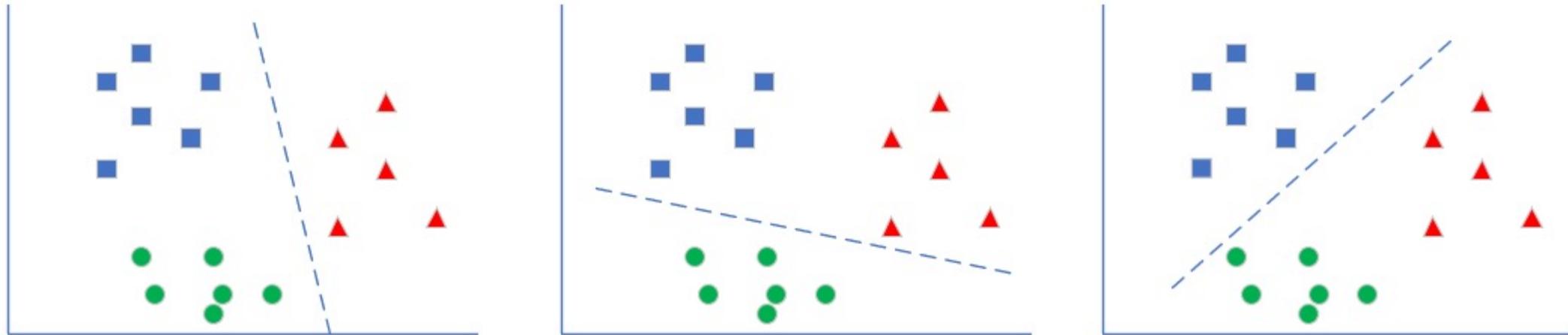


推理时，(A|B)分类器告诉你是A类时，需要到(A|C)分类器再试一下，如果也是A类，则就是A类。如果(A|C)告诉你是C类，则基本是C类了，不可能是B类，不信的话可以到(B|C)分类器再去测试一下。

# 多分类问题解法 - 一对多方式

处理一个类别时，暂时把其它所有类别看作是一类，这样对于三分类问题，可以得到三个分类器

**这种情况是在训练时，把红色样本当作一类，把蓝色和绿色样本混在一起当作另外一类。**



同时调用三个分类器，再把三种结果组合起来，就是真实的结果。比如，第一个分类器告诉你的是“红类”，那么它确实就是红类；如果告诉你是非红类，则需要看第二个分类器的结果，绿类或者非绿类；依此类推。

# 多分类问题解法 – 多对多方式

假设有4个类别ABCD，我们可以把AB算作一类，CD算作一类，训练一个分类器1；再把AC算作一类，BD算作一类，训练一个分类器2。

推理时，第1个分类器告诉你是AB类，第二个分类器告诉你是BD类，则做“与”操作，就是B类。

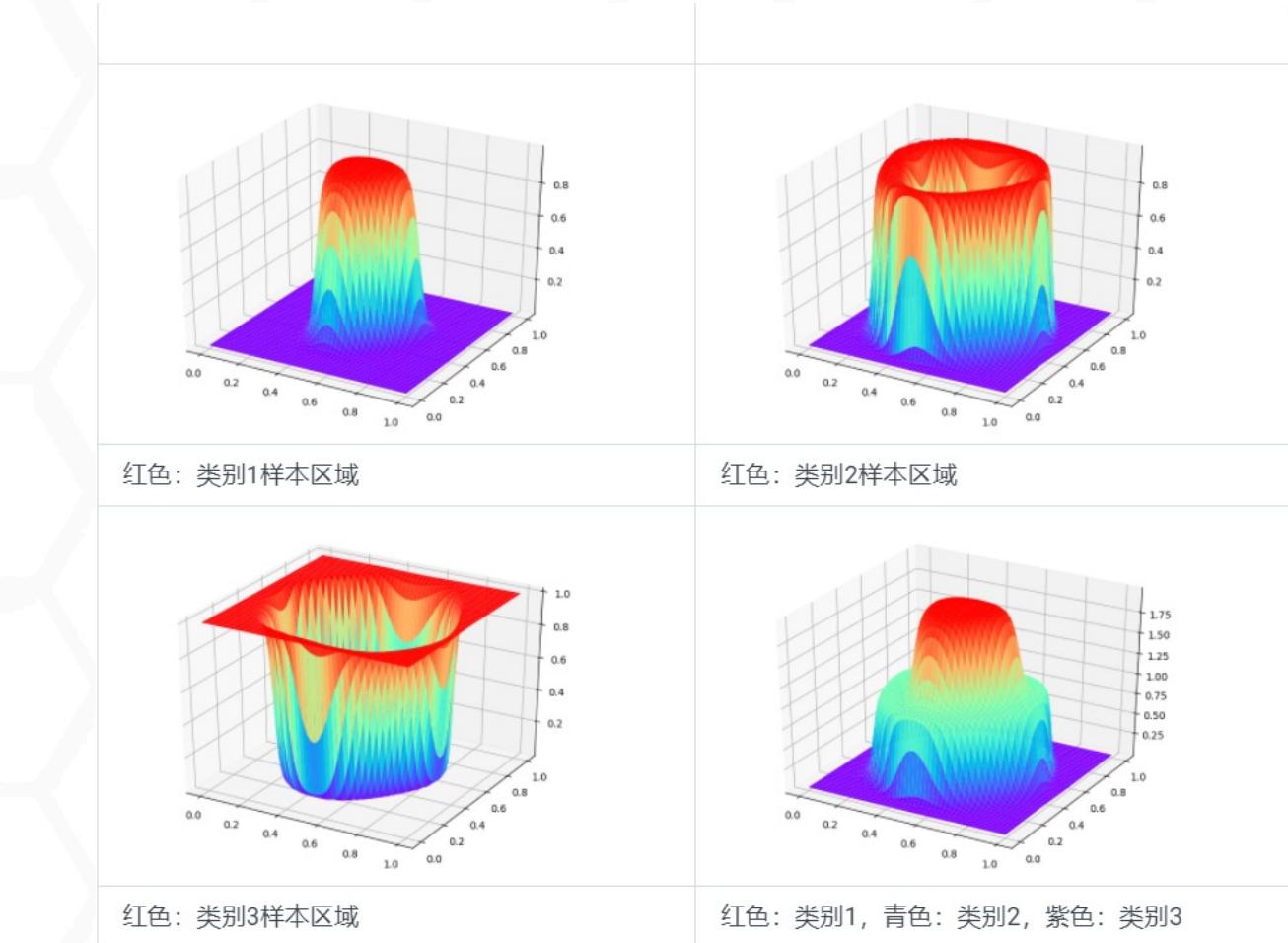
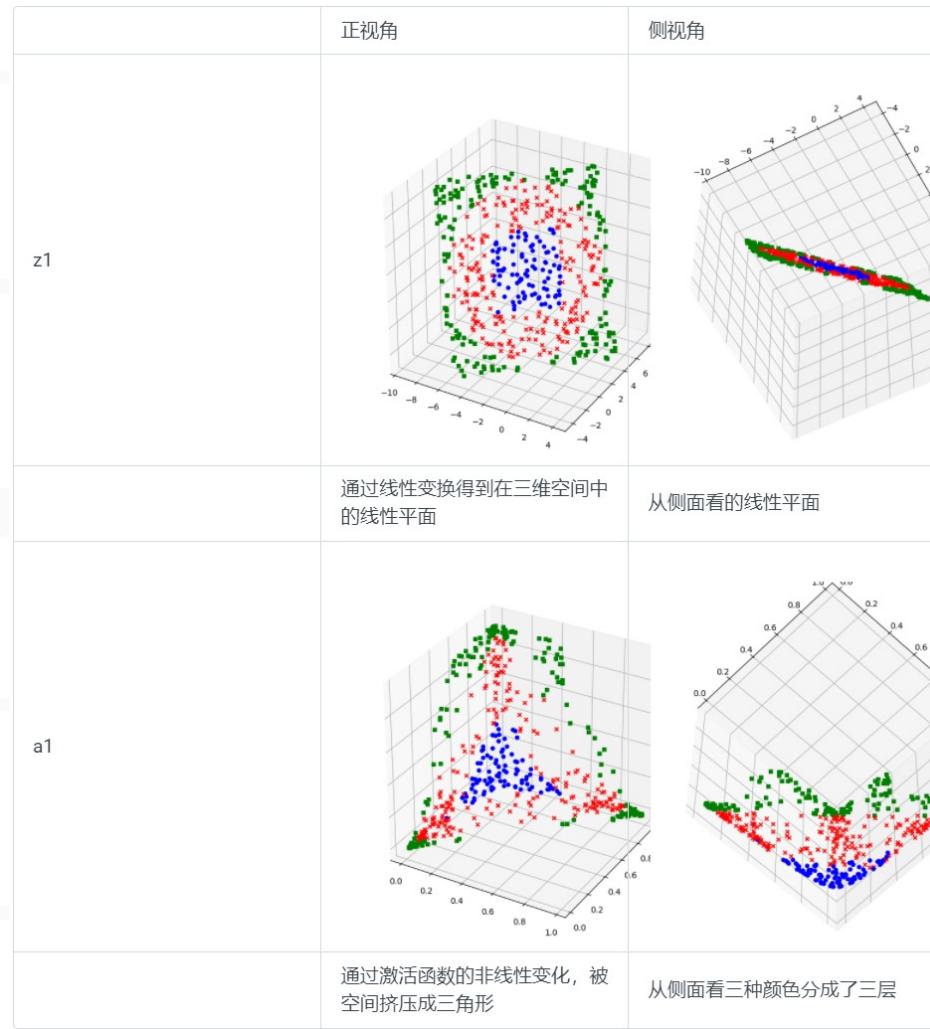
# 多分类与多标签

多分类学习中，虽然有多个类别，但是每个样本只属于一个类别。

有一种情况也很常见，比如一幅图中，既有蓝天白云，又有花草树木，那么这张图片可以有两种标注方法：

- 标注为“风景”，而不是“人物”，属于风景图片，这叫做分类
- 被同时标注为“蓝天”、“白云”、“花草”、“树木”等多个标签，这样的任务不叫作多分类学习，而是“多标签”学习，multi-label learning。我们此处不涉及这类问题。

# 非线性多分类



# 数学公式解题法



# 单变量线性回归问题

回归分析是一种数学模型。当因变量和自变量为线性关系时，它是一种特殊的线性模型。

最简单的情形是一元线性回归，由大体上有线性关系的一个自变量和一个因变量组成，模型是：

$$y = a + bX + \varepsilon$$

X是自变量，Y是因变量， $\varepsilon$ 是随机误差，a和b是参数，在线性回归模型中，a,b是要通过算法学习出来的

# 最小二乘法

最小二乘法，也叫做最小平方法（Least Square），它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。最小二乘法还可用于曲线拟合。其他一些优化问题也可通过最小化能量或最小二乘法来表达。

线性回归试图学得：

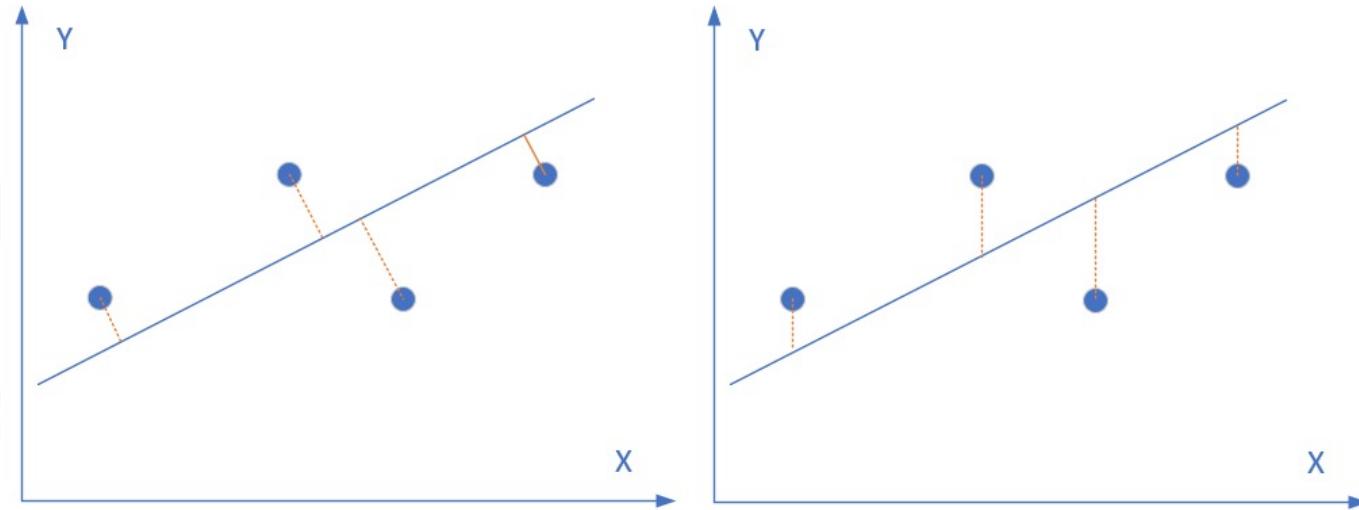
$$z_i = w \cdot x_i + b \rightarrow z_i \approx y_i$$

其中， $x_i$  是样本特征值， $y_i$  是样本标签值， $z_i$  是模型预测值。

如何学得  $w$  和  $b$  呢？均方差(MSE - mean squared error)是回归任务中常用的手段：

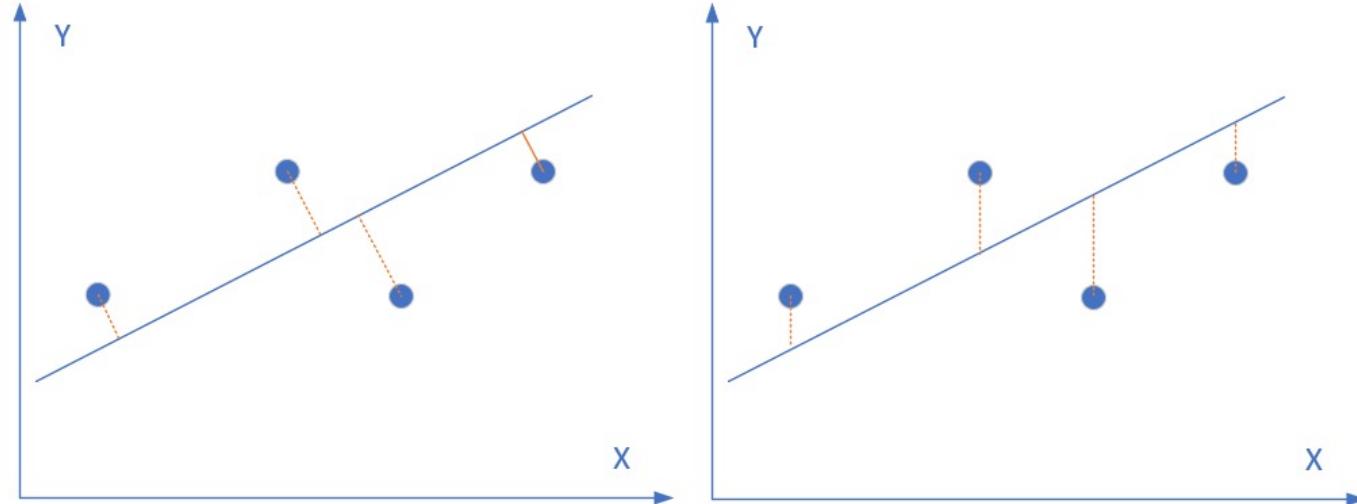
$$J = \sum_{i=1}^m (z(x_i) - y_i)^2 = \sum_{i=1}^m (y_i - wx_i - b)^2$$

# 最小二乘法



圆形点是样本点，直线是当前的拟合结果。如左图所示，我们是要计算样本点到直线的垂直距离，需要再根据直线的斜率来求垂足然后再计算距离，这样计算起来很慢；但实际上，在工程上我们通常使用的是右图的方式，即样本点到直线的竖直距离，因为这样计算很方便，用一个减法就可以了。

# 最小二乘法(MSE, Mean Square Error)



我们选择的参数决定了我们得到的直线相对于我们的训练集的准确程度，模型所预测的值与训练集中实际值之间的差距（红线所指）就是建模误差（modeling error）。

圆形点是样本点，直线是当前的拟合结果。如左图所示，我们是要计算样本点到直线的垂直距离，需要再根据直线的斜率来求垂足然后再计算距离，这样计算起来很慢；但实际上，在工程上我们通常使用的是右图的方式，即样本点到直线的竖直距离，因为这样计算很方便，用一个减法就可以了。

•拟合过程中因为w和b的取值准确度，预测的结果与训练集中的实际值有差距。

# 梯度下降法

在下面的公式中，我们规定  $x$  是样本特征值（单特征）， $y$  是样本标签值， $z$  是预测值，下标  $i$  表示其中一个样本。

## 预设函数 ( Hypothesis Function )

线性函数：

$$z_i = w * x_i + b$$

## 损失函数 ( Loss Function )

均方误差：  $loss_i(w, b) = \frac{1}{2} (Z_i - y_i)^2$

计算  $w$  的梯度

$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial z_i} \frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算  $b$  的梯度

$$\frac{\partial loss}{\partial b} = \frac{\partial loss}{\partial z_i} \frac{\partial z_i}{\partial b} = z_i - y_i$$

与最小二乘法比较可以看到，**梯度下降法和最小二乘法的模型及损失函数是相同的，都是一个线性模型加均方差损失函数**，模型用于拟合，损失函数用于评估效果。

# 多元线性回归

线性回归问题，而且是典型的多元线性回归，即包括两个或两个以上自变量的回归。多元线性回归的函数模型如下：

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_kx_k$$

具体化到房价预测问题，上面的公式可以简化成：

$$z = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

# 多元线性回归

- 1.自变量对因变量必须有显著的影响，并呈密切的线性相关；
- 2.自变量与因变量之间的线性相关必须是真实的，而不是形式上的；
- 3.自变量之间应具有一定的互斥性，即自变量之间的相关程度不应高于自变量与因变量之因的相关程度；
- 4.自变量应具有完整的统计数据，其预测值容易确定。

# 数学方式解决问题

## 多元线性方程式

函数拟合（回归）时，  
我们假设函数为

$b = w_0$

$x$  是一个样本的  $n$  个特征值，如果我们把  $m$  个样本一起计算，将会得到下面这个矩阵

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_kx_k$$

$$H(w, b) = b + x_1w_1 + x_2w_2 + \cdots + x_nw_n$$

$$H(W) = w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \cdots + x_n \cdot w_n$$

$$H(W) = X \cdot W$$

# 数学方式解决问题

$$X = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{pmatrix} * \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

$$H(W) = X \cdot W = Y$$

$$X^\top XW = X^\top Y$$

其中， $X^\top$  是  $X$  的转置矩阵， $X^\top X$  一定是个方阵，并且假设其存在逆矩阵，把它移到等式右侧来：

$$W = (X^\top X)^{-1} X^\top Y$$

# 数学方式解决问题

我们仍然使用均方差损失函数（略去了系数  $\frac{1}{2m}$ ）：

$$J(w, b) = \sum_{i=1}^m (z_i - y_i)^2$$

把  $b$  看作是一个恒等于 1 的 feature，并把  $Z = XW$  计算公式带入，并变成矩阵形式：

$$J(W) = \sum_{i=1}^m \left( \sum_{j=0}^n x_{ij} w_j - y_i \right)^2 = (XW - Y)^\top \cdot (XW - Y)$$

对  $W$  求导，令导数为 0，可得到  $W$  的最小值解：

$$\begin{aligned} \frac{\partial J(W)}{\partial W} &= \frac{\partial}{\partial W} [(XW - Y)^\top \cdot (XW - Y)] \\ &= \frac{\partial}{\partial W} [(W^\top X^\top - Y^\top) \cdot (XW - Y)] \\ &= \frac{\partial}{\partial W} [(W^\top X^\top XW - W^\top X^\top Y - Y^\top XW + Y^\top Y)] \end{aligned}$$

求导后（请参考矩阵/向量求导公式）：

第一项的结果是： $2X^\top XW$  (分母布局, denominator layout)

第二项的结果是： $X^\top Y$  (分母布局方式, denominator layout)

第三项的结果是： $X^\top Y$  (分子布局方式, numerator layout, 需要转置  $Y^\top X$ )

第四项的结果是：0

$$X^\top XW = X^\top Y$$

再令导数为 0：

其中， $X^\top$  是  $X$  的转置矩阵， $X^\top X$  一定是个方阵，并且假设其存在逆矩阵，把它移到等式右侧来：

$$W = (X^\top X)^{-1} X^\top Y$$

# 二分类函数

- 公式

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

- 导数

$$\text{Logistic}'(z) = a(1 - a)$$

- 输入值域

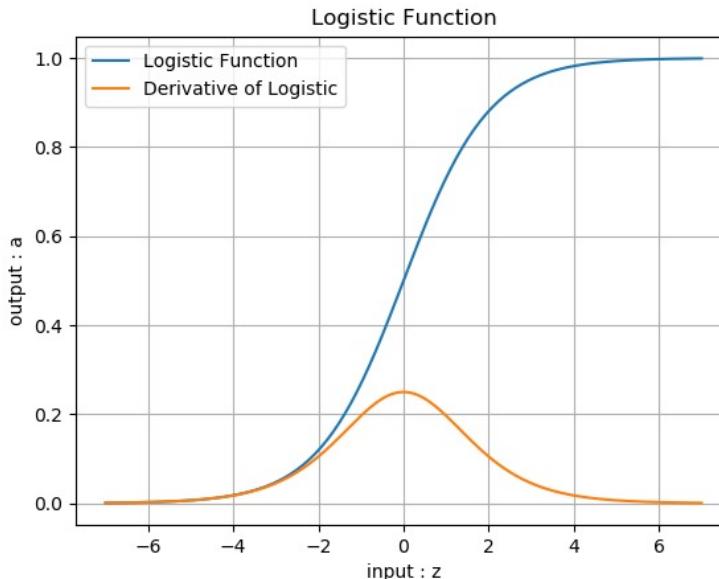
$$(-\infty, \infty)$$

- 输出值域

$$(0, 1)$$

• **input=2时，output=0.88，而0.88>0.5，算作正例**

• **input=-1时，output=0.27，而0.27<0.5，算作负例**



此函数实际上是一个概率计算，它把 $(-\infty, \infty)$ 之间的任何数字都压缩到 $(0, 1)$ 之间，返回一个概率值，这个概率值接近1时，认为是正例，否则认为是负例。

训练时，一个样本 $x$ 在经过神经网络的最后一层的矩阵运算结果作为输入 $z$ ，经过Logistic计算后，输出一个 $(0, 1)$ 之间的预测值。我们假设这个样本的标签值为0属于负类，如果其预测值越接近0，就越接近标签值，那么误差越小，反向传播的力度就越小。

推理时，我们预先设定一个阈值比如0.5，则当推理结果大于0.5时，认为是正类；小于0.5时认为是负类；等于0.5时，根据情况自己定义。阈值也不一定就是0.5，也可以是0.65等等，阈值越大，准确率越高，召回率越低；阈值越小则相反，准确度越低，召回率越高。

# 正向传播

正向传播

矩阵运算

$$z = x \cdot w + b \quad (1)$$

分类计算

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

损失函数计算

二分类交叉熵损失函数：

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

# 反向传播

## 反向传播

求损失函数对a的偏导

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

求a对z的偏导

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

求损失函数loss对z的偏导

使用链式法则链接公式4和公式5:

$$\begin{aligned} \frac{\partial \text{loss}}{\partial z} &= \frac{\partial \text{loss}}{\partial a} \frac{\partial a}{\partial z} \\ &= \frac{a-y}{a(1-a)} \cdot a(1-a) = a - y \end{aligned} \quad (6)$$

# 多样本

## 多样本情况

我们用三个样本做实例化推导：

$$Z = \begin{pmatrix} z_1 & z_2 & z_3 \end{pmatrix}$$
$$A = Logistic(z_1 \ z_2 \ z_3) = \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}$$

$$J(w, b) = - [y_1 \ln a_1 + (1 - y_1) \ln (1 - a_1)]$$
$$- [y_2 \ln a_2 + (1 - y_2) \ln (1 - a_2)]$$
$$- [y_3 \ln a_3 + (1 - y_3) \ln (1 - a_3)]$$

代入公式6结果：

$$\frac{\partial J(w, b)}{\partial z} = \left( \frac{\partial J(w, b)}{\partial z_1} \frac{\partial J(w, b)}{\partial z_2} \frac{\partial J(w, b)}{\partial z_3} \right)$$
$$= (a_1 - y_1 \quad a_2 - y_2 \quad a_3 - y_3)$$
$$= A - Y$$

所以，用矩阵运算时可以简化为矩阵相减的形式： $A - Y$ 。

# 线性多分类

我们依然假设对于一个样本的分类值是用这个线性公式得到的：

$$z = \mathbf{x} \cdot \mathbf{w} + b$$

但是，我们要求  $z$  不是一个标量，而是一个向量。如果是三分类问题，我们就要求  $z$  是一个三维的向量，向量中的每个单元的元素值代表该样本分别属于三个分类的值，这样不就可以了吗？

具体的说，假设  $\mathbf{x}$  是一个  $(1 \times 2)$  的向量，把  $\mathbf{w}$  设计成一个  $(2 \times 3)$  的向量， $\mathbf{b}$  设计成  $(1 \times 3)$  的向量，则  $\mathbf{z}$  就是一个  $(1 \times 3)$  的向量。我们假设  $\mathbf{z}$  的计算结果是  $[3, 1, -3]$ ，这三个值分别代表了样本  $\mathbf{x}$  在三个分类中的数值，下面我们把它转换成概率值。

有的读者可能会有疑问：我们不能训练神经网络让它的  $z$  值直接变成概率形式吗？答案是否定的，因为  $z$  值是经过线性计算得到的，线性计算能力有限，无法有效地直接变成概率值。

# 演算

假设输入值是: [3, 1, -3], 如果取max操作会变成: [1,0,0], 这符合我们的分类需要。但是有两个不足:

1. 分类结果是[1, 0, 0], 只保留的非0即1的信息, 没有各元素之间相差多少的信息, 可以理解是“Hard-Max”
2. max操作本身不可导, 无法用在反向传播中。

所以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。

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

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
- $m$  是总的分类数
- $a_j$  是对第 j 项的计算结果

# 演算

假设 $j=1, m=3$ , 上式为:

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

用图7-5来形象地说明这个过程。

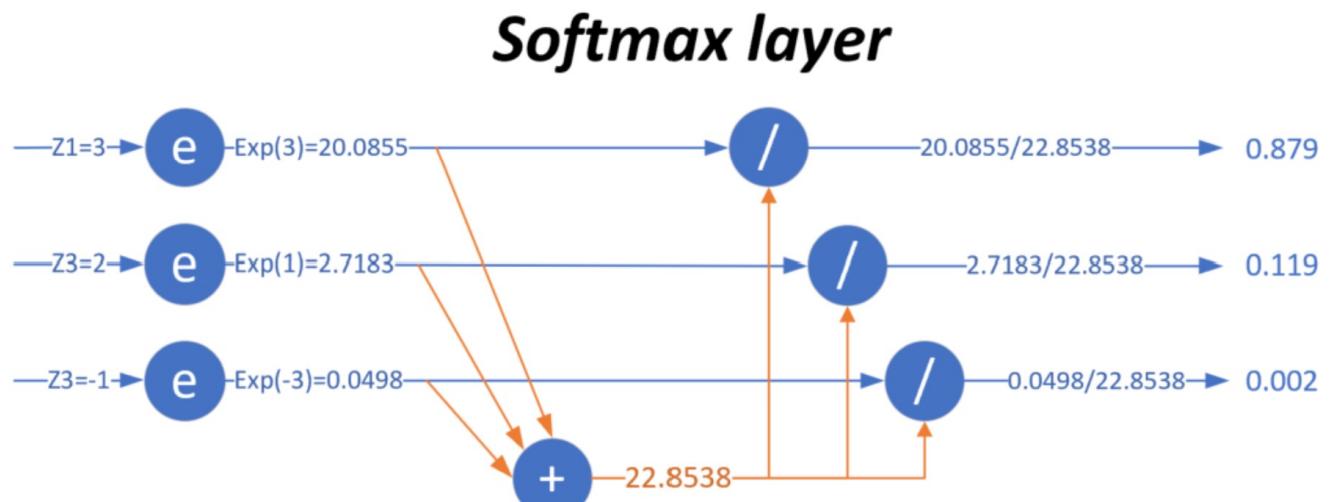


图7-5 Softmax工作过程

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  $[0.879, 0.119, 0.002]$ 。

# MAX与Softmax的对比

输入原始值	(3, 1, -3)
MAX计算	(1, 0, 0)
Softmax计算	(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

# Softmax的工作原理

我们仍假设网络输出的预测数据是 $z=[3, 1, -3]$ ，而标签值是 $y=[1, 0, 0]$ 。在做反向传播时，根据前面的经验，我们会用 $z-y$ ，得到：

$$z - y = [2, 1, -3]$$

这个信息很奇怪：

- 第一项是2，我们已经预测准确了此样本属于第一类，但是反向误差的值是2，即惩罚值是2
- 第二项是1，惩罚值是1，预测对了，仍有惩罚值
- 第三项是-3，惩罚值是-3，意为着奖励值是3，明明预测错误了却给了奖励

所以，如果不使用Softmax这种机制，会存在有个问题：

- $z$ 值和 $y$ 值之间，即预测值和标签值之间不可比，比如 $z[0]=3$ 与 $y[0]=1$ 是不可比的
- $z$ 值中的三个元素之间虽然可比，但只能比大小，不能比差值，比如 $z[0]>z[1]>z[2]$ ，但3和1相差2，1和-3相差4，这些差值是无意义的

# Softmax的工作原理

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。

从继承关系的角度来说，Softmax函数可以视作Logistic函数扩展，比如一个二分类问题：

$$a1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2}} = \frac{1}{1 + e^{z_2 - z_1}}$$

和Logistic函数形式非常像

Logistic函数也是给出了当前样本的一个概率值，只不过是依靠靠近0或靠近1来判断属于正类还是负类

# 正向传播

## 正向传播

### 矩阵运算

$$z = x \cdot w + b \quad (1)$$

### 分类计算

$$a_j = \frac{e^{z_j}}{\sum_{i=1}^m e^{z_i}} = \frac{e^{z_j}}{e^{z_1} + e^{z_2} + \dots + e^{z_m}} \quad (2)$$

### 损失函数计算 $\varphi$

计算单样本时， $m$ 是分类数：

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

计算多样本时， $m$ 是分类数， $n$ 是样本数：

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

# 非线性回归·多项式回归

## 一元一次线性模型

因为只有一项，所以不能称为多项式了。它可以解决单变量的线性回归，我们在前面学习过相关内容。其模型为：

$$z = xw + b \quad (1)$$

## 多元一次多项式

多变量的线性回归，我们在前面也学习过相关内容。其模型为：

$$z = x_1w_1 + x_2w_2 + \dots + x_mw_m + b \quad (2)$$

这里的多变量，是指样本数据的特征值为多个，上式中的  $x_1, x_2, \dots, x_m$  代表了m个特征值。

# 非线性回归·多项式回归

## 一元多次多项式

单变量的非线性回归，比如正弦曲线的拟合问题，很明显不是线性问题，但是只有一个x特征值，所以不满足前两种形式。如何解决这种问题呢？

有一个定理：任意一个函数在一个较小的范围内，都可以用多项式任意逼近。因此在实际工程实践中，有时候可以不管y值与x值的数学关系究竟是什么，而是强行用回归分析方法进行近似的拟合。

那么如何得到更多的特征值呢？对于只有一个特征值的问题，人们发明了一种聪明的办法，就是把特征值的高次方作为另外的特征值，加入到回归分析中，用公式描述：

$$z = xw_1 + x^2w_2 + \dots + x^m w_m + b \quad (3)$$

**上式中x是原有的唯一特征值， $x^m$ 是利用x的m次方作为额外的特征值，这样就把特征值的数量从1个变为m个。**

换一种表达形式，令： $x_1 = x, x_2 = x^2, \dots, x_m = x^m$ ，则：

$$z = x_1 w_1 + x_2 w_2 + \dots + x_m w_m + b \quad (4)$$

可以看到公式4和上面的公式2是一样的，所以解决方案也一样。

# 非线性回归

## 多元多次多项式

多变量的非线性回归，其参数与特征组合繁复，但最终都可以归结为公式2和公式4的形式。

所以，不管是几元几次多项式，我们都可以使用线性回归学到的方法来解决。在用代码具体实现之前，我们先学习一些前人总结的经验。先看一个被经常拿出来讲解的例子，如图9-3所示。

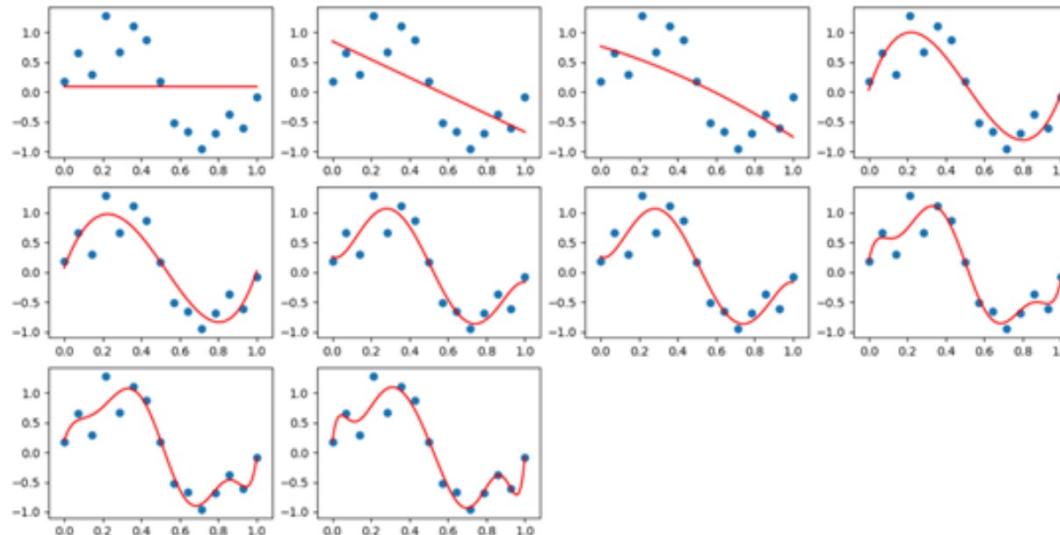
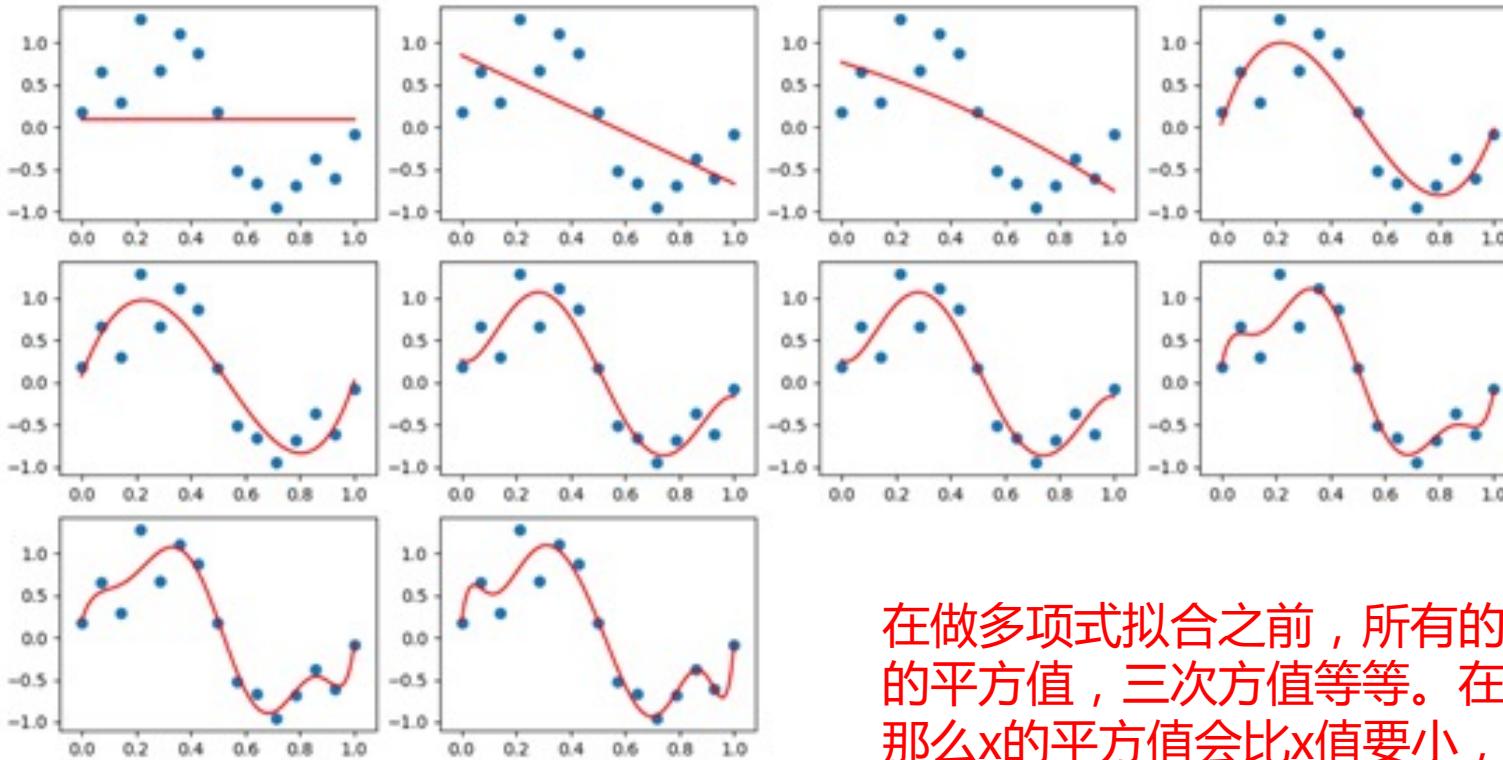


图9-3 对有噪音的正弦曲线的拟合

一堆散点，看上去像是一条带有很多噪音的正弦曲线，从左上到右下，分别是1次多项式、2次多项式……10次多项式，其中：

- 第4、5、6、7图是比较理想的拟合
- 第1、2、3图欠拟合，多项式的次数不够高
- 第8、9、10图，多项式次数过高，过拟合了

# 多项式回归

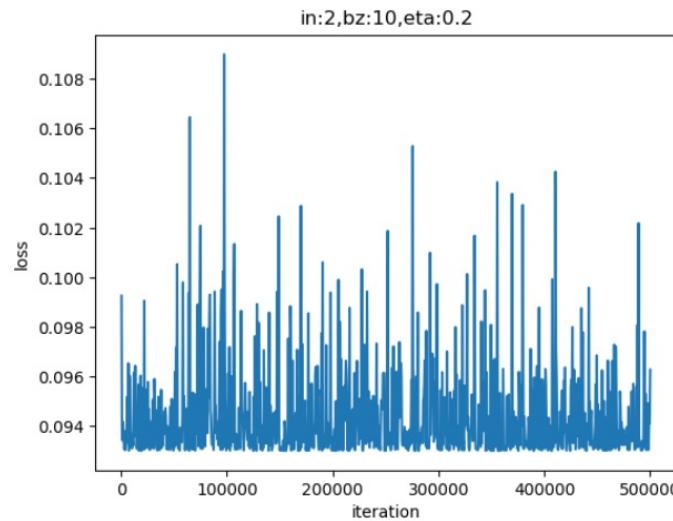


在做多项式拟合之前，所有的特征值都会先做归一化，然后再获得 $x$ 的平方值，三次方值等等。在归一化之后， $x$ 的值变成了 $[0,1]$ 之间，那么 $x$ 的平方值会比 $x$ 值要小， $x$ 的三次方值会比 $x$ 的平方值要小。假设 $x=0.5$ ， $x*x=0.25$ ， $x*x*x=0.125$ ，所以次数越高，权重值会越大，特征值与权重值的乘积才会是一个不太小的数，以此来弥补特征值小的问题。

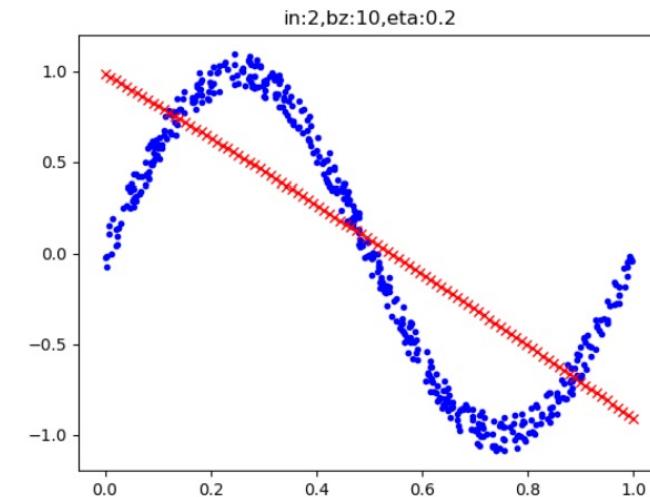
# 非线性回归·场景-用二次多项式拟合

$$z = xw_1 + x^2w_2 + b$$

损失函数值



拟合结果



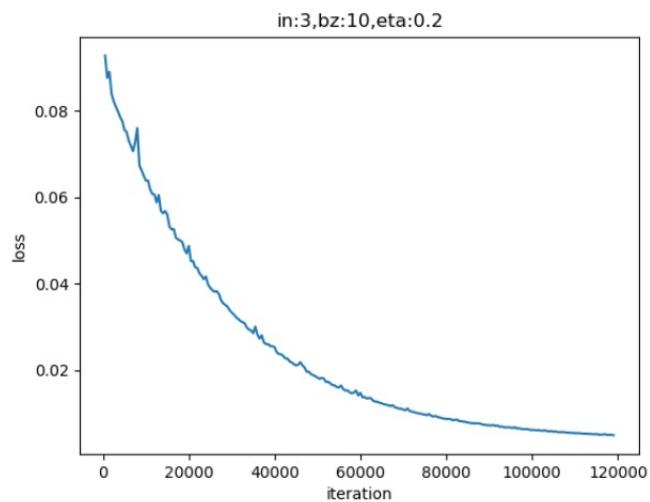
没有任何损失值下降的趋势

拟合情况，只拟合成了一条直线

# 场景-用三次多项式拟合

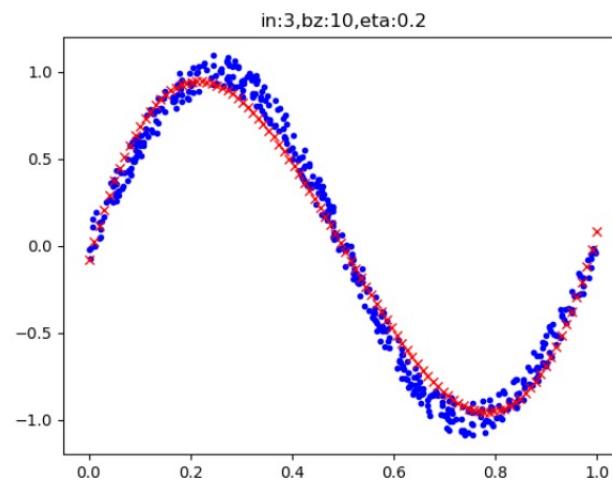
$$z = xw_1 + x^2w_2 + x^3w_3 + b$$

损失函数值



损失函数值下降得很平稳

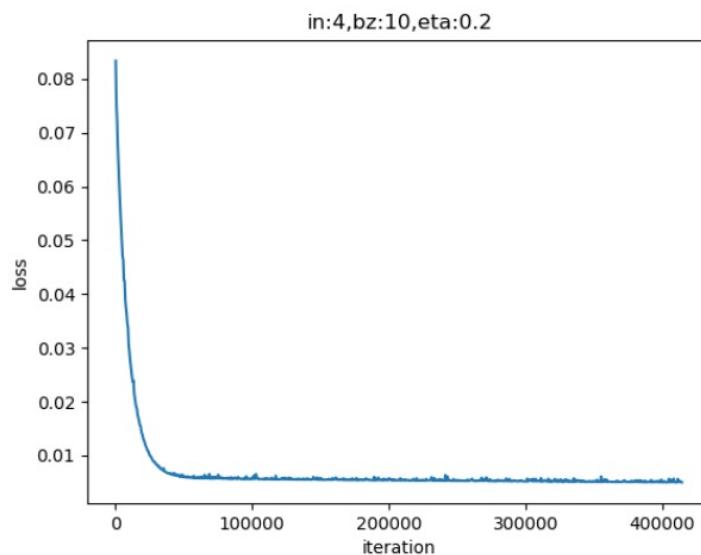
拟合结果



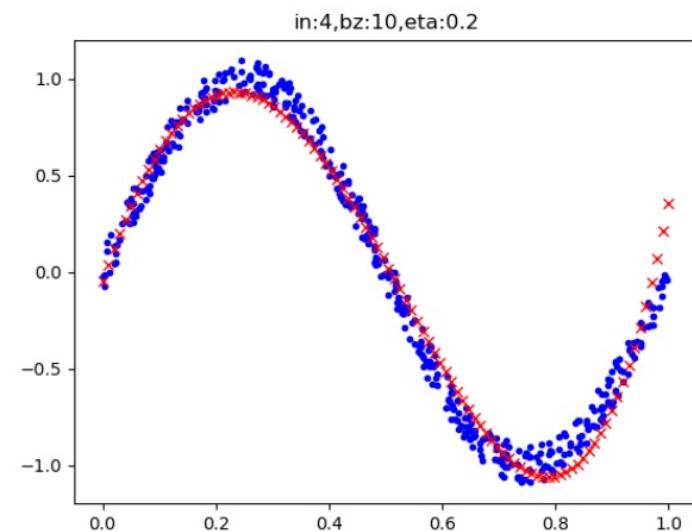
拟合的结果也很令人满意，虽然红色线没有严丝合缝地落在蓝色样本点内

# 场景-用四次多项式拟合

损失函数值



拟合结果



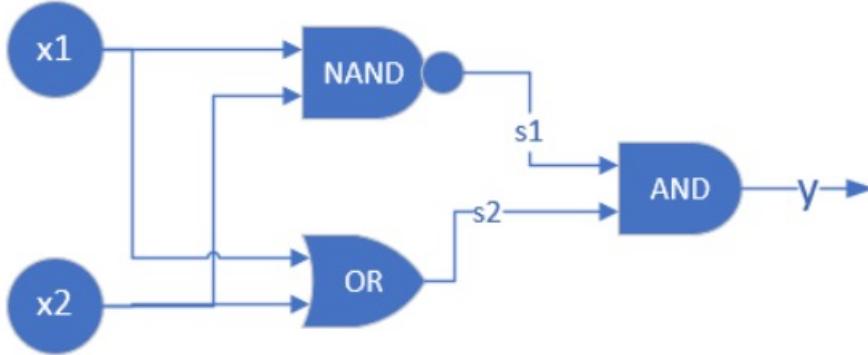
# 场景-结果比较

多项式次数	迭代数	损失函数值
2	10000	0.095
3	2380	0.005
4	8290	0.005

1. 二次多项式的损失值在下降了一定程度后，一直处于平缓期，不再下降，说明网络能力到了一定的限制，直到10000次迭代也没有达到目的；
2. 损失值达到0.005时，四项式迭代了8290次，比三次多项式的2380次要多很多，说明四次多项式多出的一个特征值，没有给我们带来什么好处，反而是增加了网络训练的复杂度。

**多项式次数并不是越高越好，对不同的问题，有特定的限制，需要在实践中摸索，并无理论指导。**

# 非线性的可能性



与、与非、或、或非

样本与计算

x1	x2	$s_1 = x_1 \text{ NAND } x_2$	$s_2 = x_1 \text{ OR } x_1$	$y = s_1 \text{ AND } s_2$
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	0	1	0

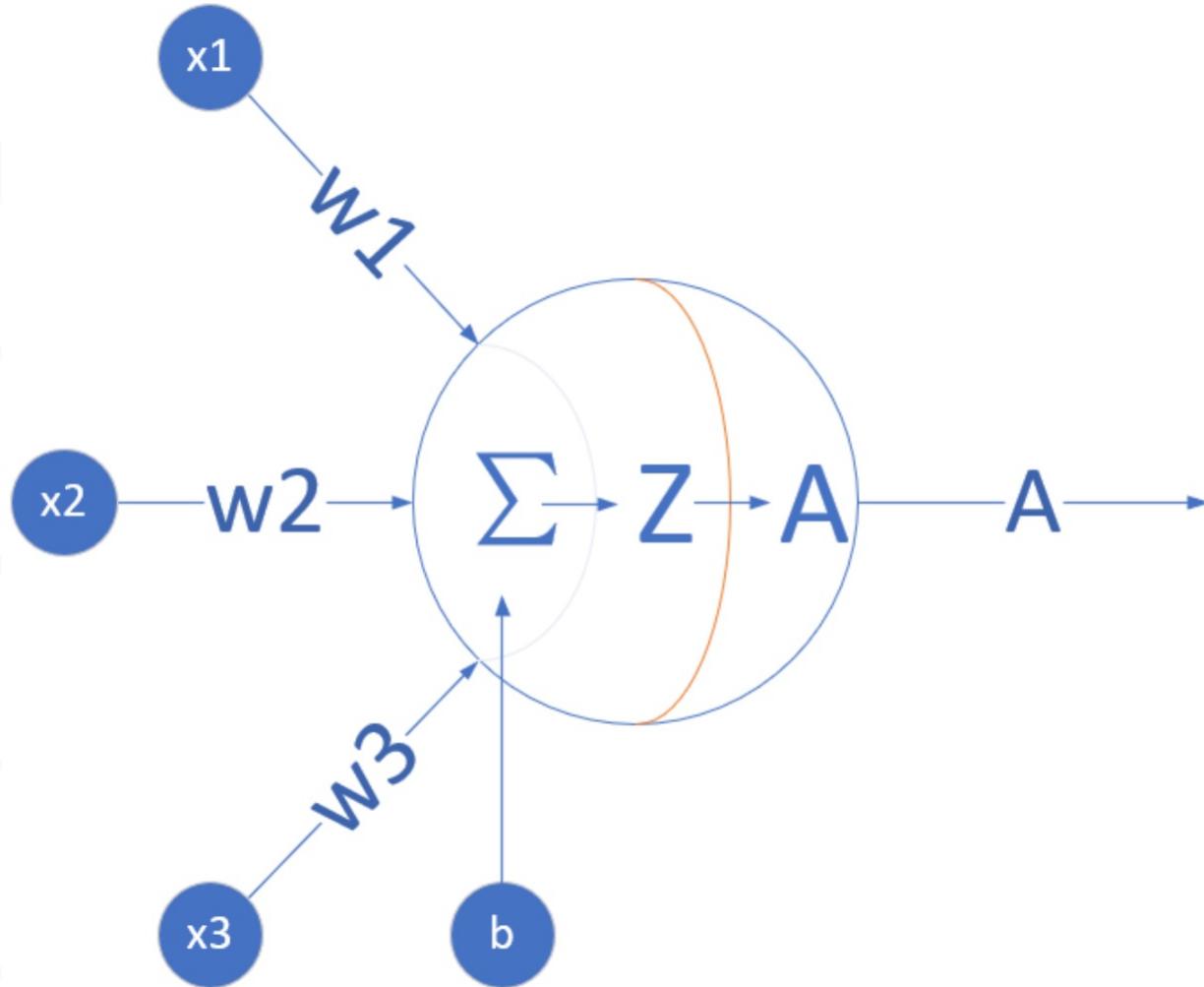
可以看到y的输出与 $x_1, x_2$ 的输入相比，就是异或逻辑了。所以，实践证明两层逻辑电路可以解决问题。另外，我们在第四步中学习了非线性回归，使用双层神经网络可以完成一些神奇的事情，比如复杂曲线的拟合，只需要6、7个参数就搞定了。

1	2	3	4
0	0	1	1
0	1	0	1
1	1	1	0
0	1	1	1
0	1	1	0

# 神经网络解题



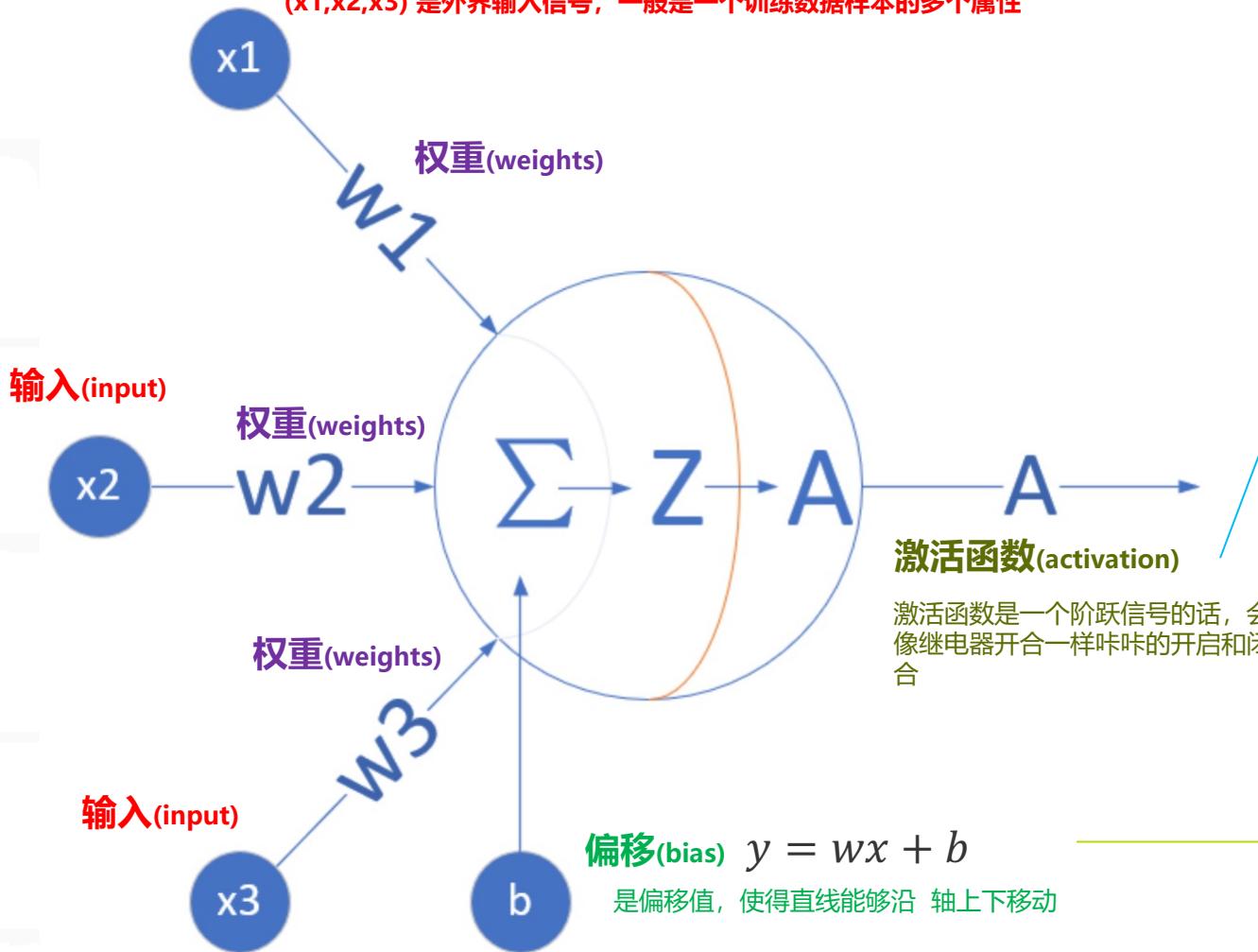
# 生物学中的神经网络



在生物神经网络中，每个神经元与其他神经元相连，当它兴奋时，就会像相邻的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过了一个阈值，那么它就会被激活（兴奋），向其他神经元发送化学物质。把许多这样的神经元按照一定的层次结构连接起来，我们就构建了一个神经网络

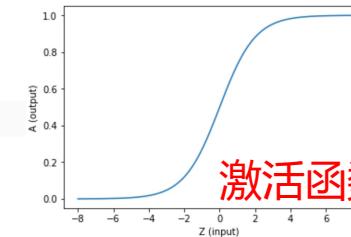
# 计算机科学的神经网络

输入(input)  
( $x_1, x_2, x_3$ ) 是外界输入信号，一般是一个训练数据样本的多个属性



## 激活函数(activation)

求和之后, 神经细胞已经处于兴奋状态了, 已经决定要向下一个神经元传递信号了, 但是要传递多强烈的信号, 要由激活函数来确定:



激活函数是有渐变过程的

$$A = \sigma(Z)$$
$$\text{偏移(bias)} = \sum_{i=1}^m (w_i x_i) + b = W X + b$$

在脑神经细胞中, 一定是输入信号的电平/电流大于某个临界值时, 神经元细胞才会处于兴奋状态, 这个  $b$  实际就是那个临界值

$$w_1 x_1 + w_2 x_2 + w_3 x_3 \geq t$$

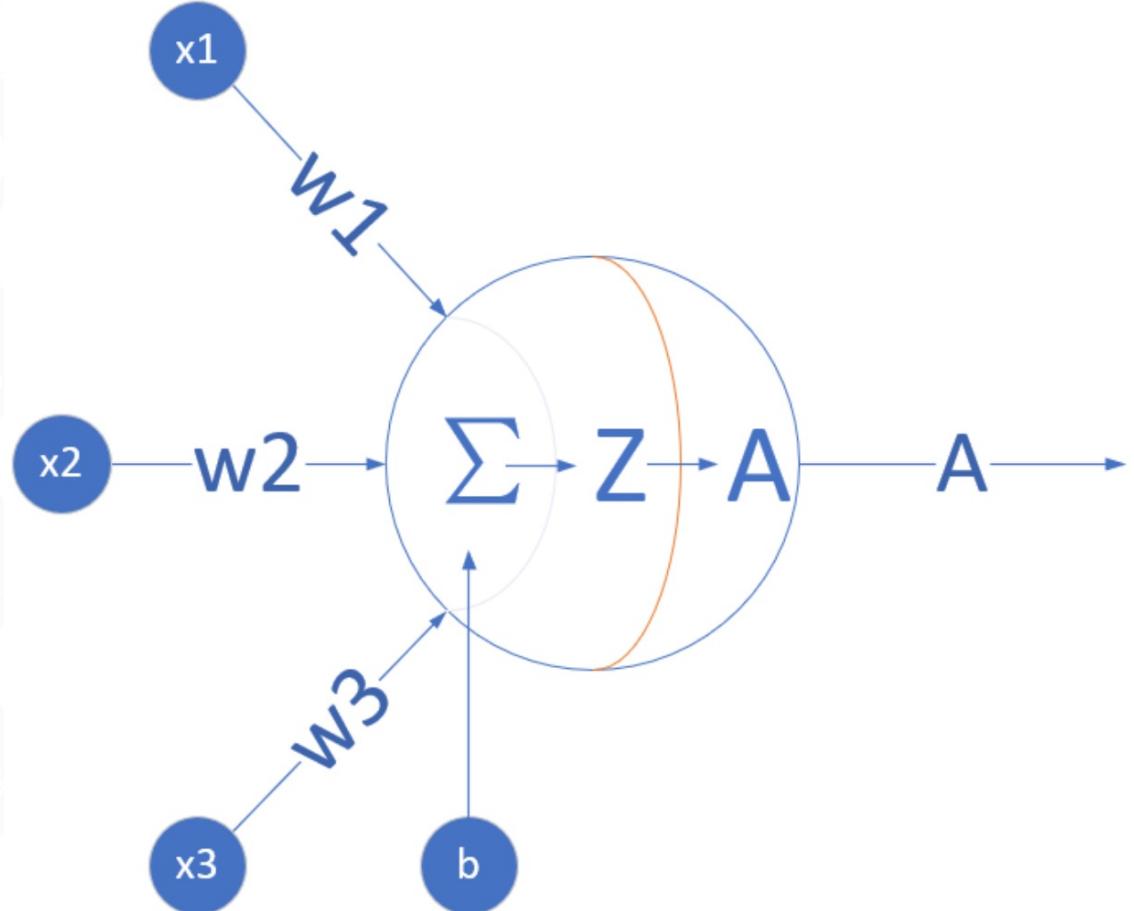
$$w_1 x_1 + w_2 x_2 + w_3 x_3 - t \geq 0$$

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + b \geq 0$$

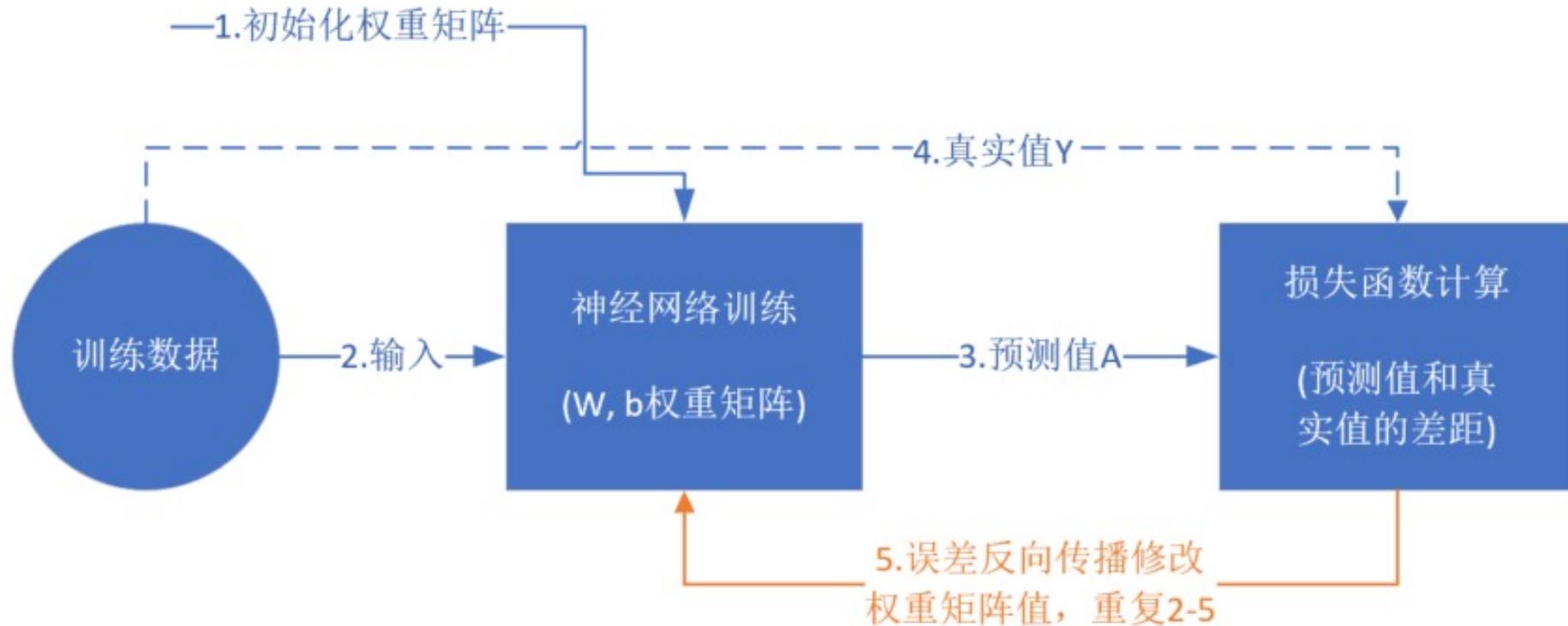
$$Z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

# 神经网络的一些细节

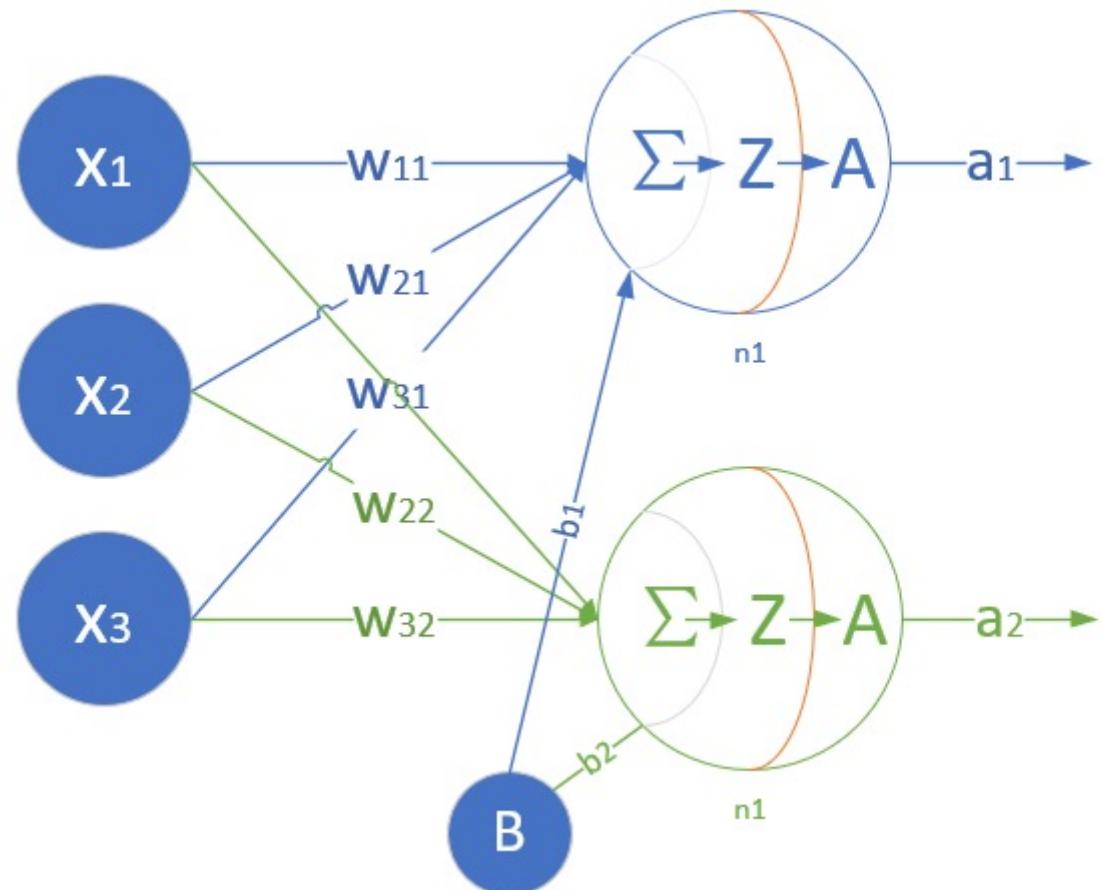
- 一个神经元可以有多个输入
- 一个神经元只能有一个输出，这个输出可以同时输入给多个神经元。
- 一个神经元的  $w$  的数量和输入的数量一致。
- 一个神经元只有一个  $b$ 。
- $w$  和  $b$  有人为的初始值，在训练过程中被不断修改。
- $A$  可以等于  $Z$ ，即激活函数不是必须有的。
- 一层神经网络中的所有神经元的激活函数必须一致。



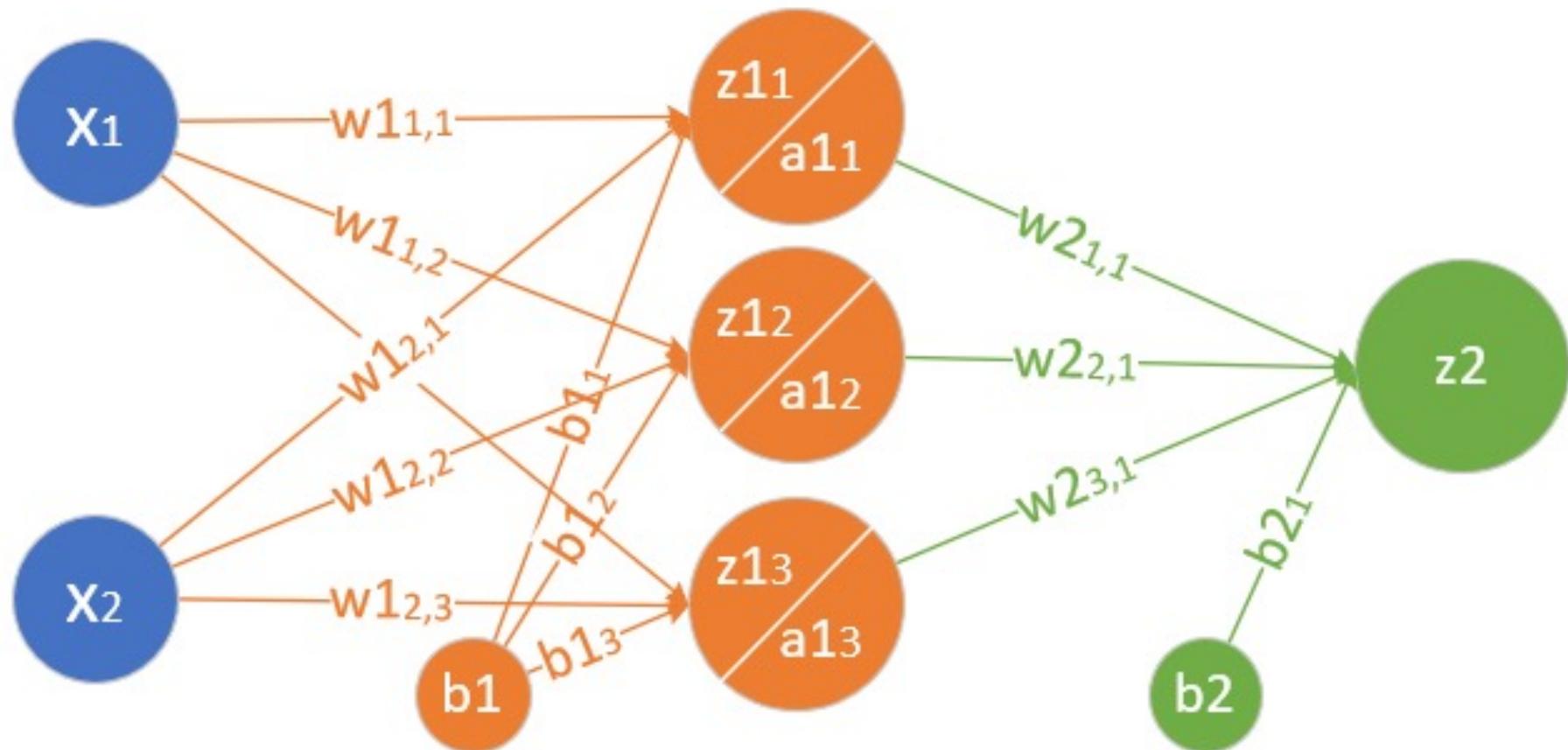
# 解剖神经网络训练过程



# 一个单层神经网络



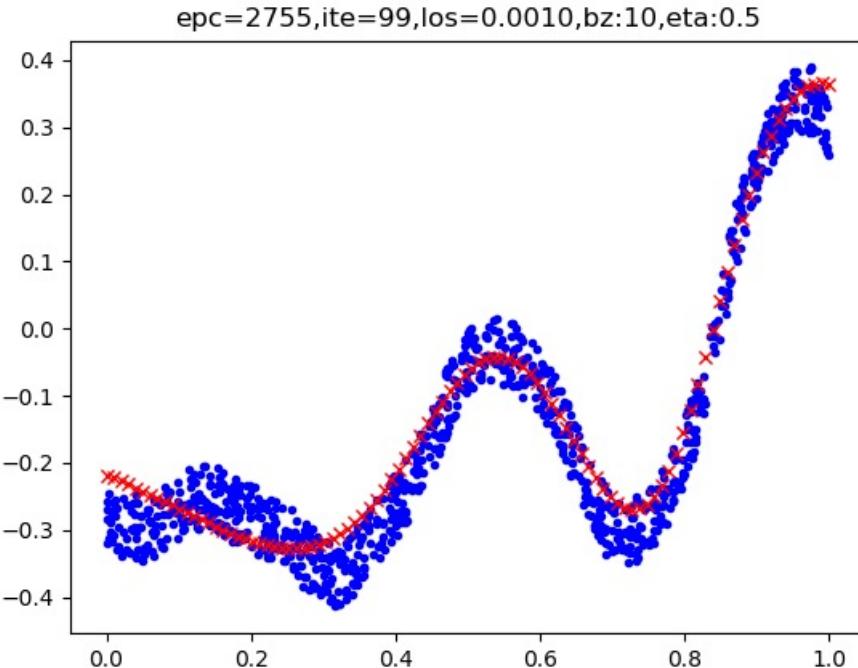
# 两层神经网络(矩阵运算)



# 神经网络的主要功能

## 回归 ( Regression ) 或者叫做拟合 ( Fitting )

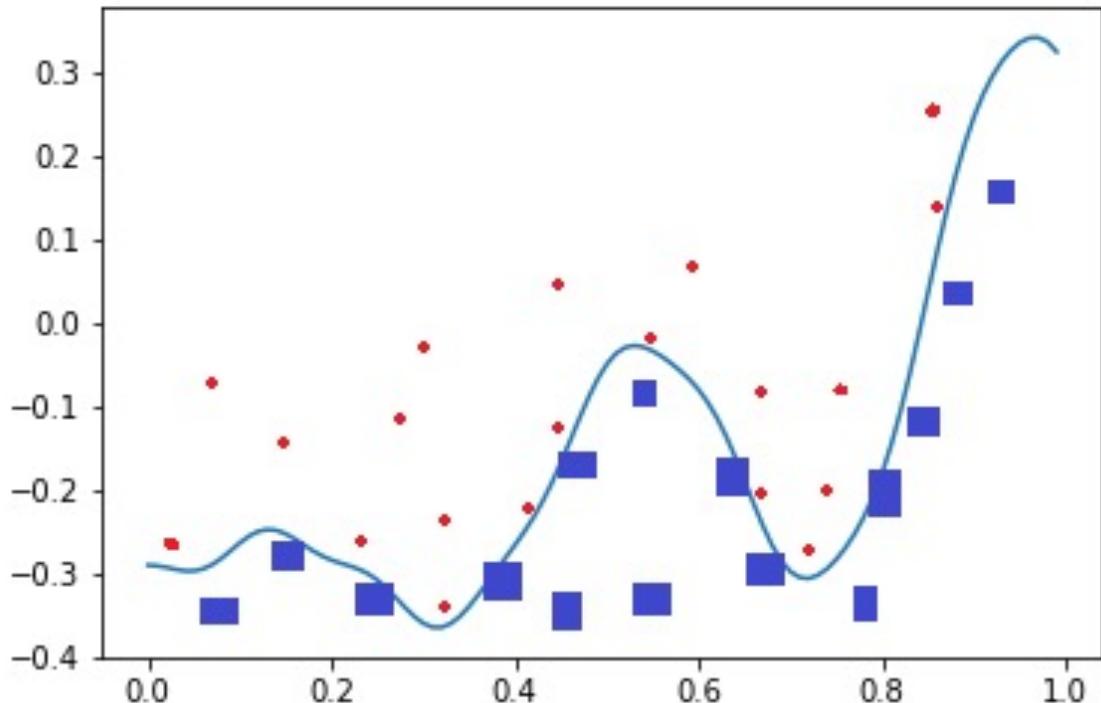
单层的神经网络能够模拟一条二维平面上的直线，从而可以完成线性分割任务。而理论证明，两层神经网络可以无限逼近任意连续函数。



所谓回归或者拟合，其实就是给出x值输出y值的过程，并且让y值与样本数据形成的曲线的距离尽量小，可以理解为是对样本数据的一种骨架式的抽象。

# 神经网络的主要功能

## 分类 (Classification)

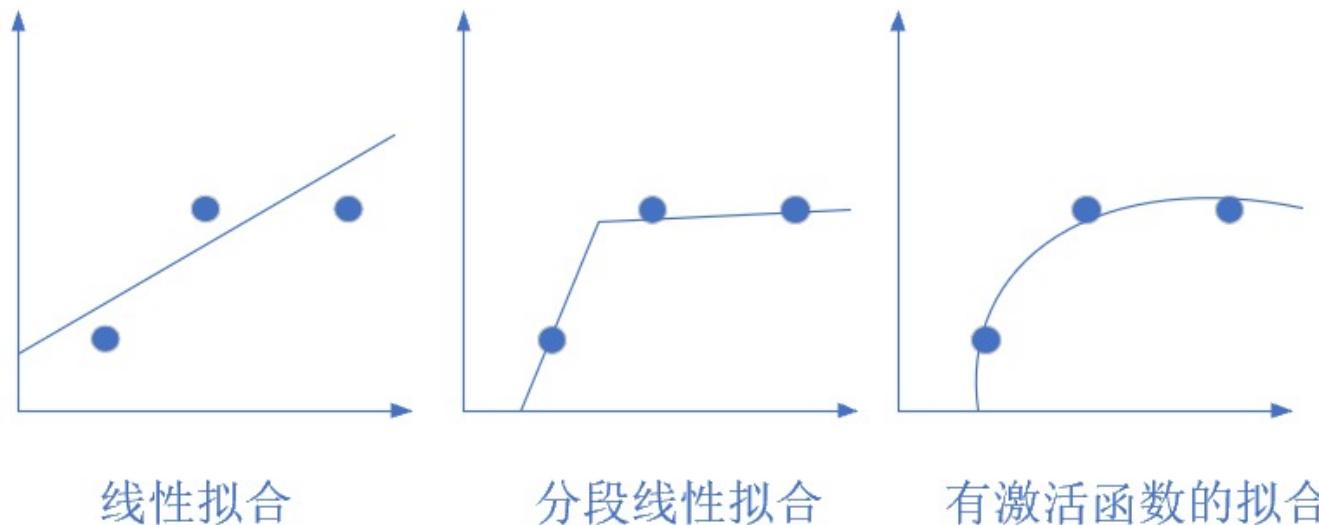


使用一个两层的神经网络可以得到一个非常近似的结果，使得分类误差在满意的范围之内。图中那条淡蓝色的曲线，本来并不存在，是通过神经网络训练出来的分界线，可以比较完美地把两类样本分开，所以分类可以理解为是对两类或多类样本数据的边界的抽象。

# 再来谈谈激活函数

我们不运用激活函数的话，则输出信号将仅仅是一个简单的线性函数。线性函数一个一级多项式。线性方程是很容易解决的，但是它们的复杂性有限，并且从数据中学习复杂函数映射的能力更小。一个没有激活函数的神经网络将只不过是一个线性回归模型罢了，不能解决现实世界中的大多数非线性问题。

**没有激活函数，我们的神经网络将无法学习和模拟其他复杂类型的数据**

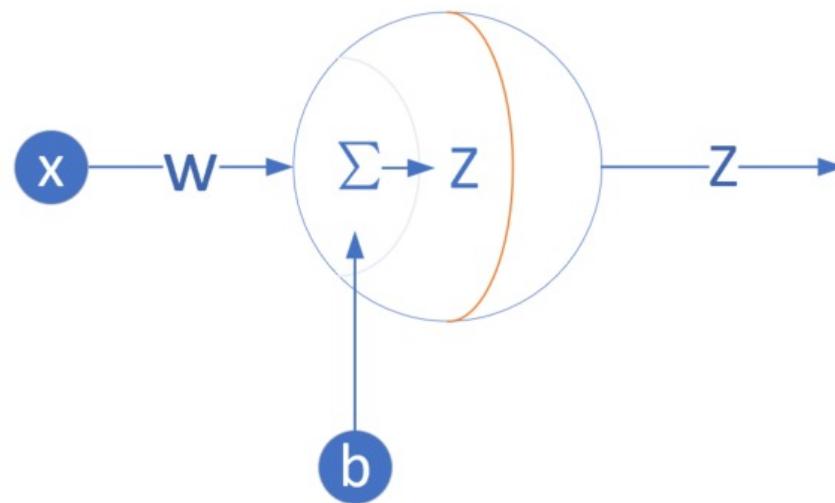


最左侧的是线性拟合，中间的是分段线性拟合，右侧的是曲线拟合，只有当使用激活函数时，才能做到完美的曲线拟合。

# 一元线性回归

## 输入层

此神经元在输入层只接受一个输入特征，经过参数 w,b 的计算后，直接输出结果。这样一个简单的“网络”，只能解决简单的一元线性回归问题，而且由于是线性的，我们不需要定义激活函数，这就大大简化了程序，而且便于大家循序渐进地理解各种知识点。严格来说输入层在神经网络中并不能称为一个层



计算  $w$  的梯度

$$\frac{\partial \text{loss}}{\partial w} = \frac{\partial \text{loss}}{\partial z_i} \frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算  $b$  的梯度

$$\frac{\partial \text{loss}}{\partial b} = \frac{\partial \text{loss}}{\partial z_i} \frac{\partial z_i}{\partial b} = z_i - y_i$$

## 权重 $w,b$

因为是一元线性问题，所以  $w,b$  都是标量。

## 输出层

输出层 1 个神经元，线性预测公式是：

$$z_i = w * x_i + b$$

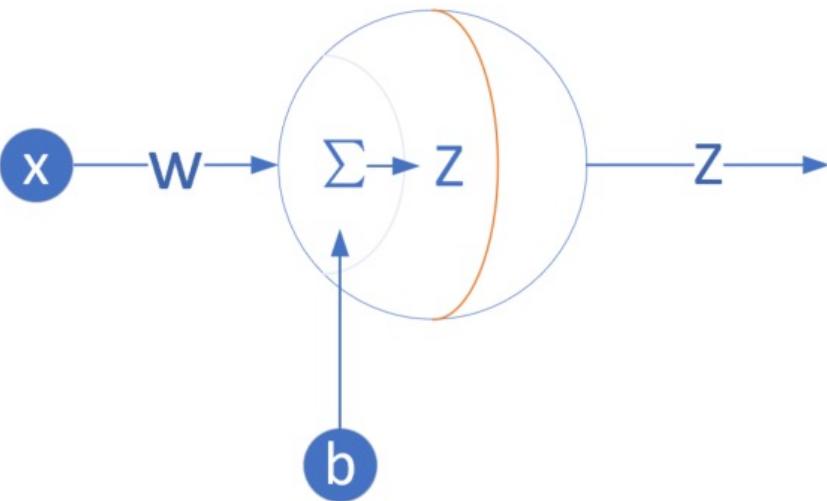
$z$  是模型的预测输出， $y$  是实际的样本标签值，下标  $i$  为样本。

## 损失函数

因为是线性回归问题，所以损失函数使用均方差函数。

$$\text{loss}_i(w, b) = \frac{1}{2} (Z_i - y_i)^2$$

# 多元线性回归



计算  $w$  的梯度

$$\frac{\partial \text{loss}}{\partial w} = \frac{\partial \text{loss}}{\partial z_i} \frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算  $b$  的梯度

$$\frac{\partial \text{loss}}{\partial b} = \frac{\partial \text{loss}}{\partial z_i} \frac{\partial z_i}{\partial b} = z_i - y_i$$

## 输入层

此神经元在输入层只接受一个输入特征，经过参数  $w, b$  的计算后，直接输出结果。这样一个简单的“网络”，只能解决简单的一元线性回归问题，而且由于是线性的，我们不需要定义激活函数，这就大大简化了程序，而且便于大家循序渐进地理解各种知识点。严格来说输入层在神经网络中并不能称为一个层

## 权重 $w, b$

因为是一元线性问题，所以  $w, b$  都是标量。

## 输出层

输出层 1 个神经元，线性预测公式是：

$$z_i = w * x_i + b$$

$z$  是模型的预测输出， $y$  是实际的样本标签值，下标  $i$  为样本。

## 损失函数

因为是线性回归问题，所以损失函数使用均方差函数。

$$\text{loss}_i(w, b) = \frac{1}{2} (Z_i - y_i)^2$$

# 多元线性回归

初次使用神经网络，一定有水土不服的地方。最小二乘法可以得到数学解析解，所以它的结果是可信的。梯度下降法和神经网络法实际是一回事儿，只是梯度下降没有使用神经元模型而已。所以，接下来我们研究一下如何调整神经网络的训练过程，先从最简单的梯度下降的三种形式说起。

在下面的说明中，我们使用如下假设，以便简化问题易于理解：

1. 使用可以解决本章的问题的线性回归模型，即  $z = x \cdot w + b$ ；

2. 样本特征值数量为1，即  $x, w, b$  都是标量；

3. 使用均方差损失函数。

计算  $w$  的梯度：

$$\frac{\partial \text{loss}}{\partial w} = \frac{\partial \text{loss}}{\partial z_i} \frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算  $b$  的梯度：

$$\frac{\partial \text{loss}}{\partial b} = \frac{\partial \text{loss}}{\partial z_i} \frac{\partial z_i}{\partial b} = z_i - y_i$$

# 多元线性回归-单样本随机梯度法-SGD(Stochastic Gradient Descent)

SGD(Stochastic Gradient Descent)

样本访问示意图如图4-7所示。



图4-7 单样本访问方式

## 计算过程

假设一共100个样本，每次使用1个样本：

---

*repeat{ for i = 1, 2, 3, ..., 100{       $z_i = x_i \cdot w + b$        $dw = x_i \cdot (z_i - y_i)$        $db = z_i - y_i$        $w = w - \eta \cdot dw$  }*

---

## 特点

- 训练样本：每次使用一个样本数据进行一次训练，更新一次梯度，重复以上过程。
- 优点：训练开始时损失值下降很快，随机性大，找到最优解的可能性大。
- 缺点：受单个样本的影响最大，损失函数值波动大，到后期徘徊不前，在最优解附近震荡。不能并行计算。

# 多元线性回归-小批量样本梯度下降-Mini-Batch Gradient Descent

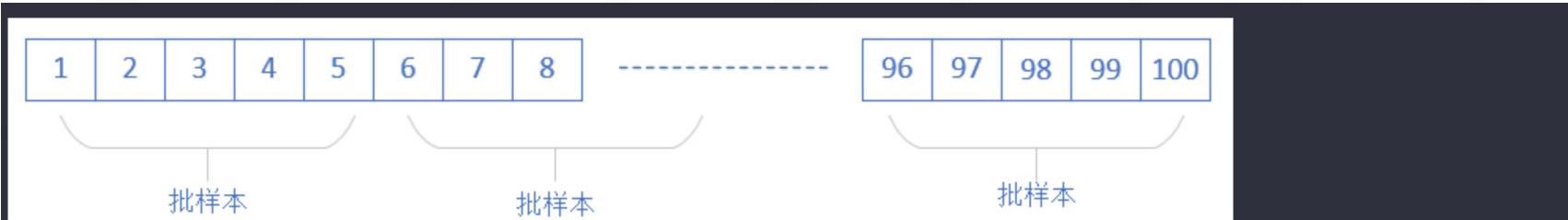


图4-8 小批量样本访问方式

## 计算过程

假设一共100个样本，每个小批量5个样本：

```
repeat{  for  i = 1,6,11,...,96{       $z_i = x_i \cdot w + b$        $z_{i+1} = x_{i+1} \cdot w + b$       ...       $z_{i+4} = x_{i+4} \cdot w + b$       dw
```

上述算法中，循环体中的前5行分别计算了  $z_i, z_{i+1}, \dots, z_{i+4}$ ，可以换成一次性的矩阵运算。

## 特点

- 训练样本：选择一小部分样本进行训练，更新一次梯度，然后再选取另外一小部分样本进行训练，再更新一次梯度。
- 优点：不受单样本噪声影响，训练速度较快。
- 缺点：batch size的数值选择很关键，会影响训练结果。

# 多元线性回归-全批量样本梯度下降-Full Batch Gradient Descent

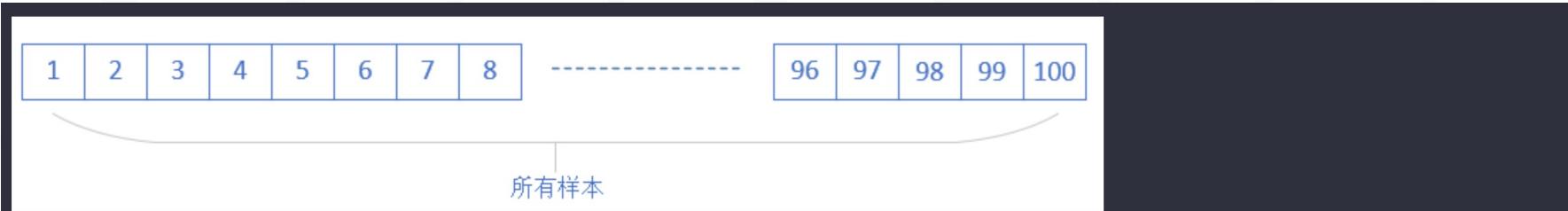


图4-9 全批量样本访问方式

## 计算过程

假设一共100个样本，每次使用全部样本：

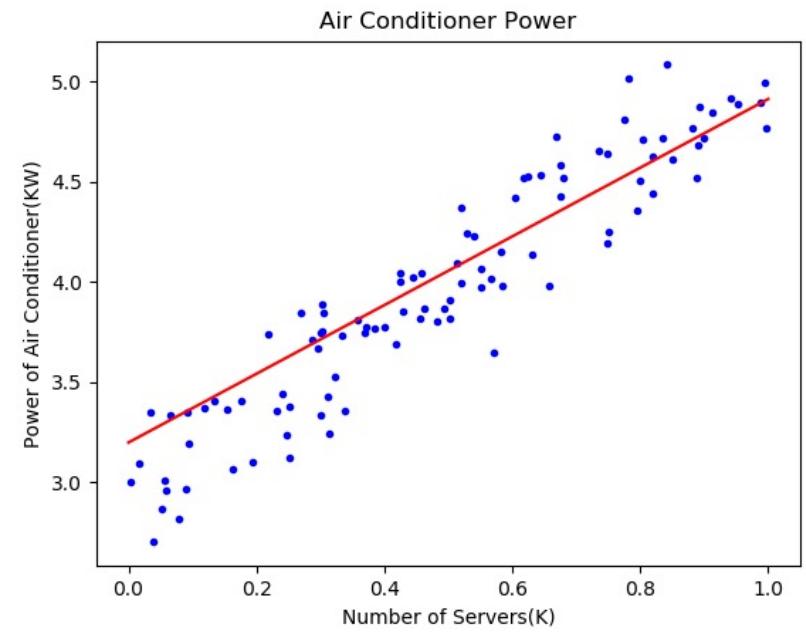
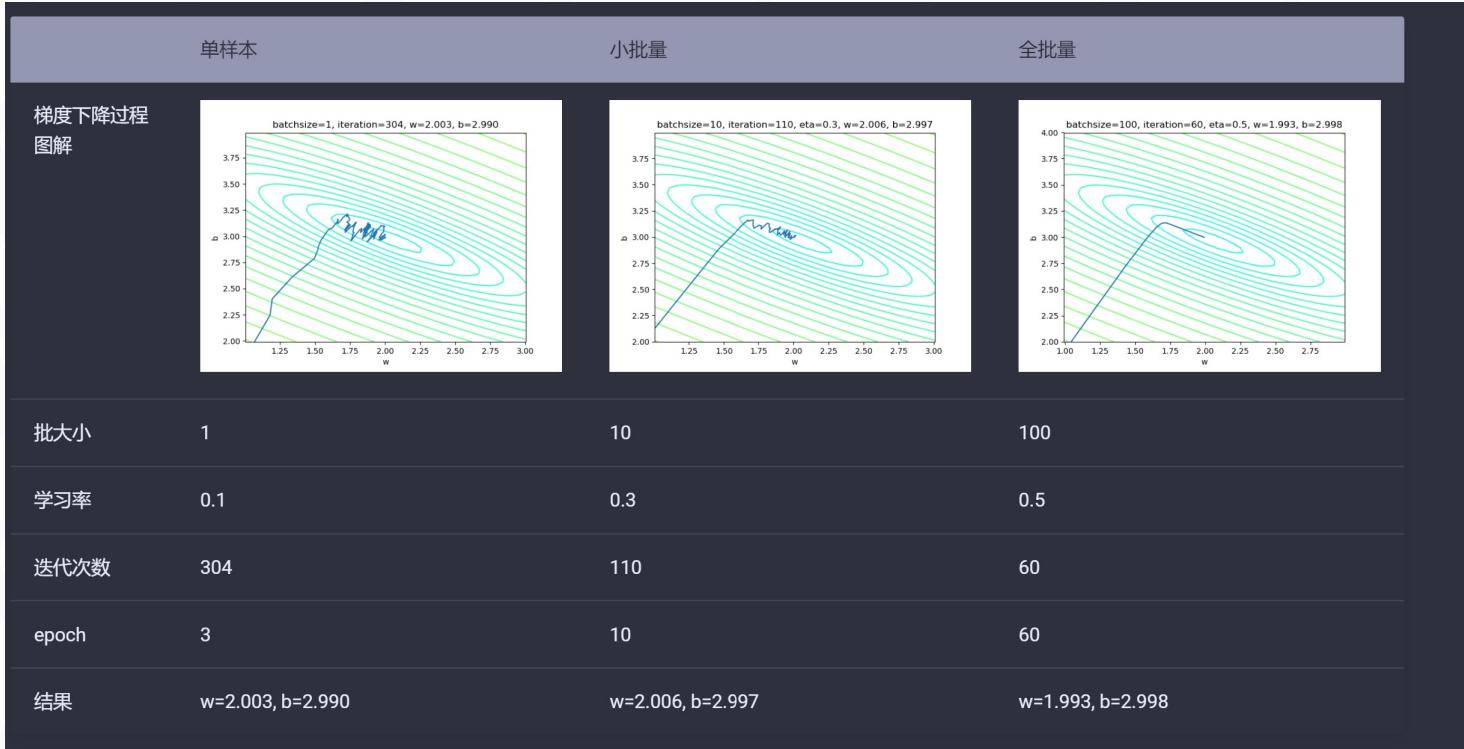
```
repeat{  z1 = x1 · w + b  z2 = x2 · w + b  ...  z100 = x100 · w + b  dw = 1/100 ∑100i=1 xi · (zi - yi)  db = 1/100 ∑100i=1 (
```

上述算法中，循环体中的前100行分别计算了  $z_1, z_2, \dots, z_{100}$ ，可以换成一次性的矩阵运算。

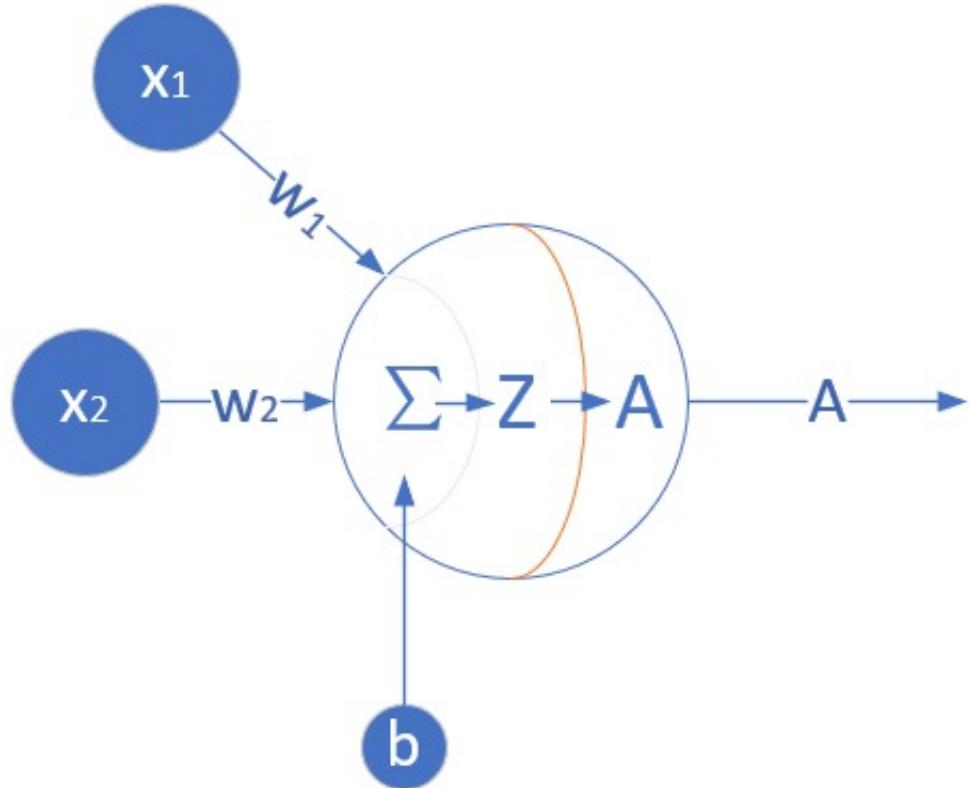
## 特点

- 训练样本：每次使用全部数据集进行一次训练，更新一次梯度，重复以上过程。
- 优点：受单个样本的影响最小，一次计算全体样本速度快，损失函数值没有波动，到达最优点平稳。方便并行计算。
- 缺点：数据量较大时不能实现（内存限制），训练过程变慢。初始值不同，可能导致获得局部最优解，并非全局最优解。

# 三种方式比较

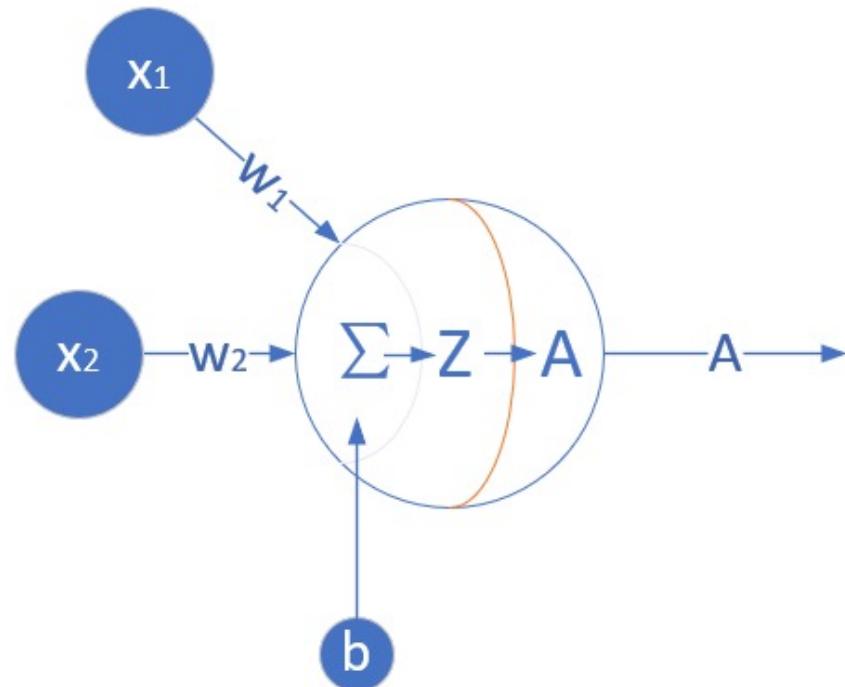


# 神经网络实现线性二分类



- 从视觉上判断是线性可分的，所以我们使用单层神经网络即可；
- 输入特征是经度和纬度，所以我们在输入层设置两个输入 $X1=经度$ ， $X2=维度$ ；
- 最后输出的是一个二分类，分别是楚汉地盘，可以看成非0即1的二分类问题，所以我们只用一个输出单元就可以了。

# 神经网络实现线性二分类



## 输入层

输入经度  $x_1$  和纬度  $x_2$  两个特征:

$$X = \begin{pmatrix} x_1 & x_2 \end{pmatrix}$$

## 权重矩阵

输入是2个特征，输出一个数，则  $W$  的尺寸就是  $2 \times 1$ :

$$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

$B$  的尺寸是  $1 \times 1$ ，行数永远是 1，列数永远和  $W$  一样。

$$B = (b_1)$$

## 输出层

$$\begin{aligned} z &= x \cdot w + b = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \\ &= x_1 \cdot w_1 + x_2 \cdot w_2 + b \end{aligned} \tag{1}$$

$$a = Logistic(z) \tag{2}$$

## 损失函数

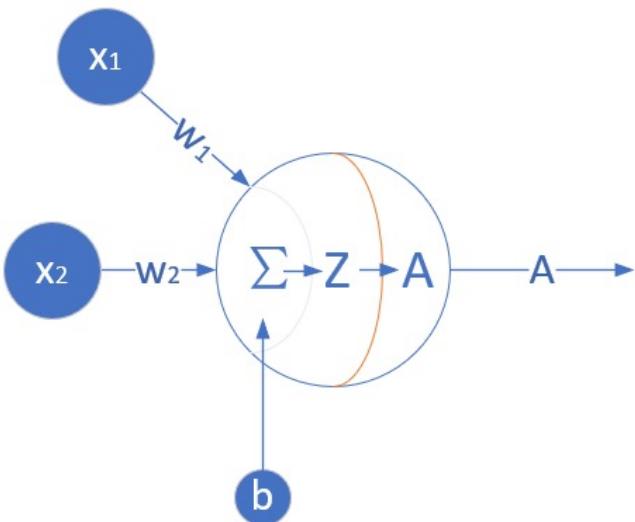
二分类交叉熵损失函数:

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

# 神经网络实现线性二分类

## 反向传播

我们在[上一节](#)已经推导了loss对z的偏导数，结论为  $A - Y$ 。接下来，我们求loss对w的导数。本例中，w的形式是一个2行1列的向量，所以求w的偏导时，要对向量求导：



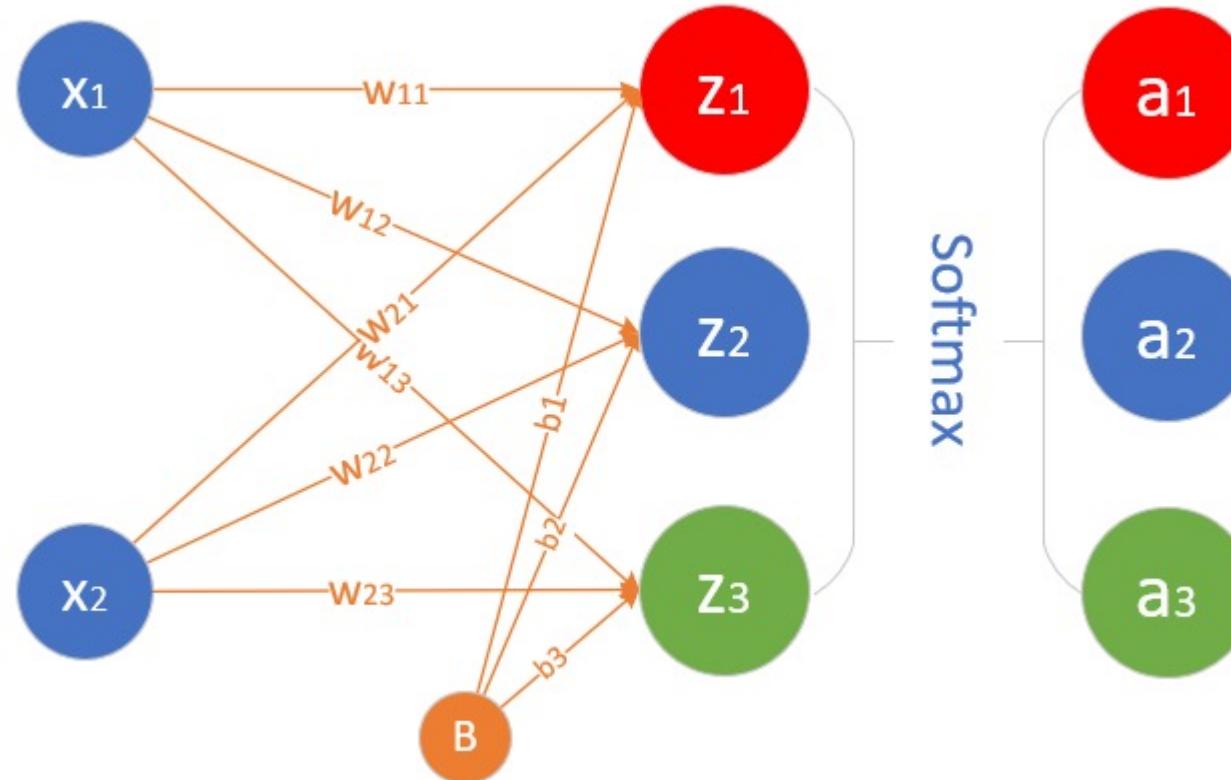
$$\begin{aligned}\frac{\partial \text{loss}}{\partial w} &= \begin{pmatrix} \frac{\partial \text{loss}}{\partial w_1} \\ \frac{\partial \text{loss}}{\partial w_2} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial \text{loss}}{\partial z} \frac{\partial z}{\partial w_1} \\ \frac{\partial \text{loss}}{\partial z} \frac{\partial z}{\partial w_2} \end{pmatrix} = \begin{pmatrix} (a - y)x_1 \\ (a - y)x_2 \end{pmatrix} \\ &= (x_1 x_2)^T (a - y)\end{aligned}\quad (4)$$

上式中  $x_1, x_2$  是一个样本的两个特征值。如果是多样本的话，公式4将会变成其矩阵形式，以3个样本为例：

$$\begin{aligned}\frac{\partial J(w,b)}{\partial w} &= \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix}^T \begin{pmatrix} a_1 - y_1 \\ a_2 - y_2 \\ a_3 - y_3 \end{pmatrix} \\ &= X^T(A - Y)\end{aligned}\quad (5)$$

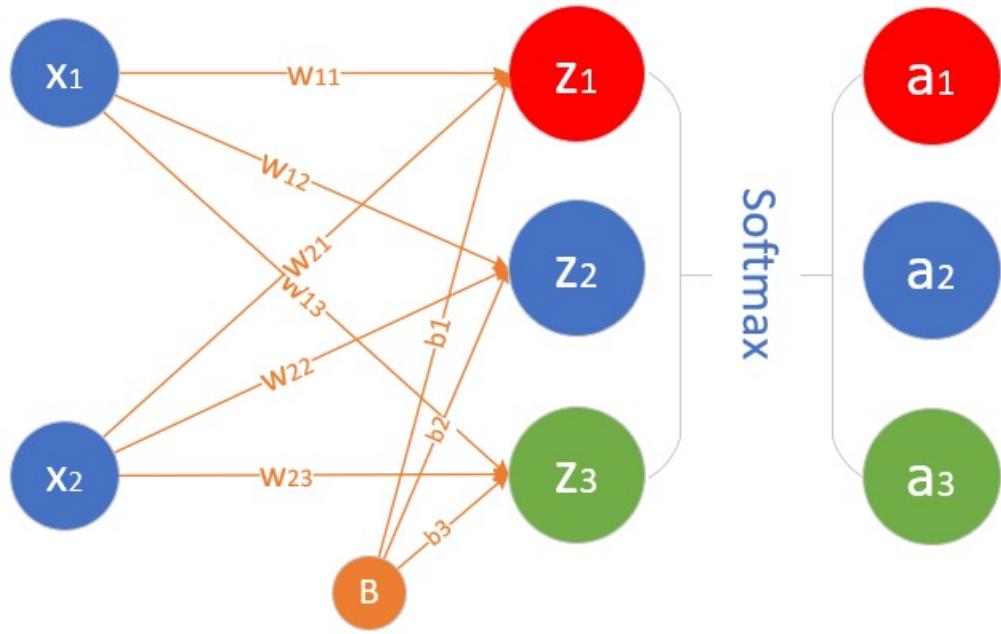
# 线性多分类

与前面的单层网络不同的是，输出层还多出来一个Softmax分类函数，这是多分类任务中的标准配置，可以看作是输出层的激活函数，并不单独成为一层，与二分类中的Logistic函数一样。



只有输入层和输出层，由于输入层不算在内，所以是一层网络

# 线性多分类



## 输入层

输入经度  $x_1$  和纬度  $x_2$  两个特征:

$$x = (x_1 \quad x_2)$$

## 权重矩阵

$W$  权重矩阵的尺寸, 可以从前往后看, 比如: 输入层是2个特征, 输出层是3个神经元, 则  $W$  的尺寸就是  $2 \times 3$ 。

$$w = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

$B$  的尺寸是  $1 \times 3$ , 列数永远和神经元的数量一样, 行数永远是1。

$$B = (b_1 \quad b_2 \quad b_3)$$

## 输出层

输出层三个神经元, 再加上一个Softmax计算, 最后有  $A_1, A_2, A_3$  三个输出, 写作:

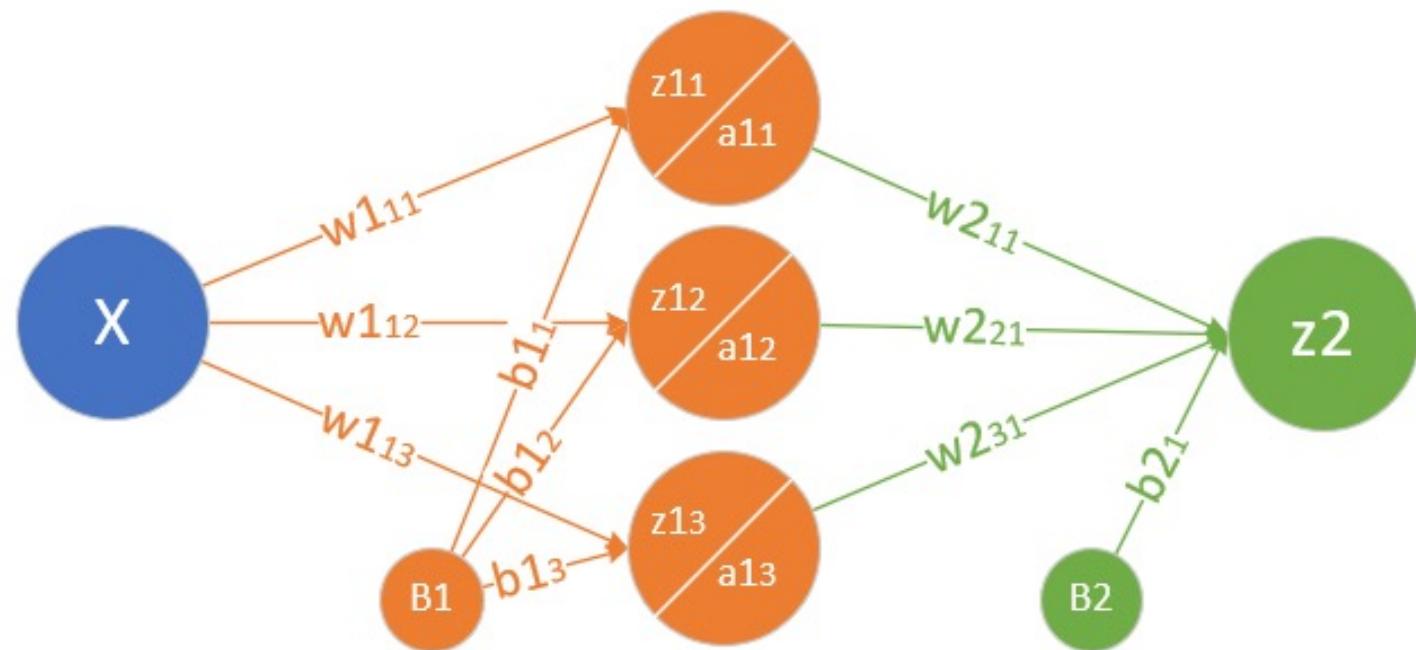
$$Z = (z_1 \quad z_2 \quad z_3)$$

$$A = (a_1 \quad a_2 \quad a_3)$$

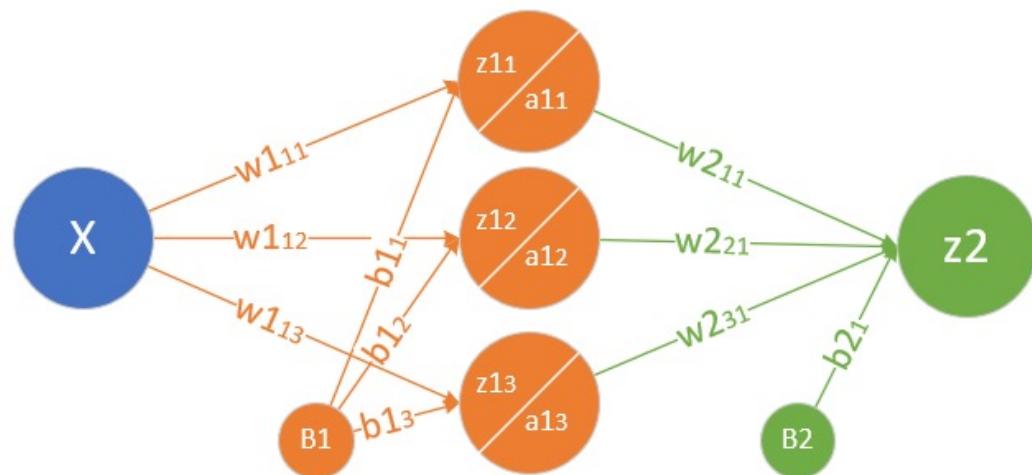
其中,  $Z = X \cdot W + B$ ,  $A = \text{Softmax}(Z)$

# 非线形回归-定义神经网络

根据万能近似定理的要求，我们定义一个两层的神经网络，输入层不算，一个隐藏层，含3个神经元，一个输出层。



# 非线形回归-定义神经网络



## 权重矩阵W1/B1

$$W1 = (w_{111} \quad w_{112} \quad w_{113})$$

$$B1 = (b_{11} \quad b_{12} \quad b_{13})$$

## 隐层

我们用3个神经元：

$$Z1 = (z_{11} \quad z_{12} \quad z_{13})$$

$$A1 = (a_{11} \quad a_{12} \quad a_{13})$$

## 权重矩阵W2/B2

W2的尺寸是 $3 \times 1$ , B2的尺寸是 $1 \times 1$ 。

$$W2 = \begin{pmatrix} w_{11}^2 \\ w_{21}^2 \\ w_{31}^2 \end{pmatrix}$$

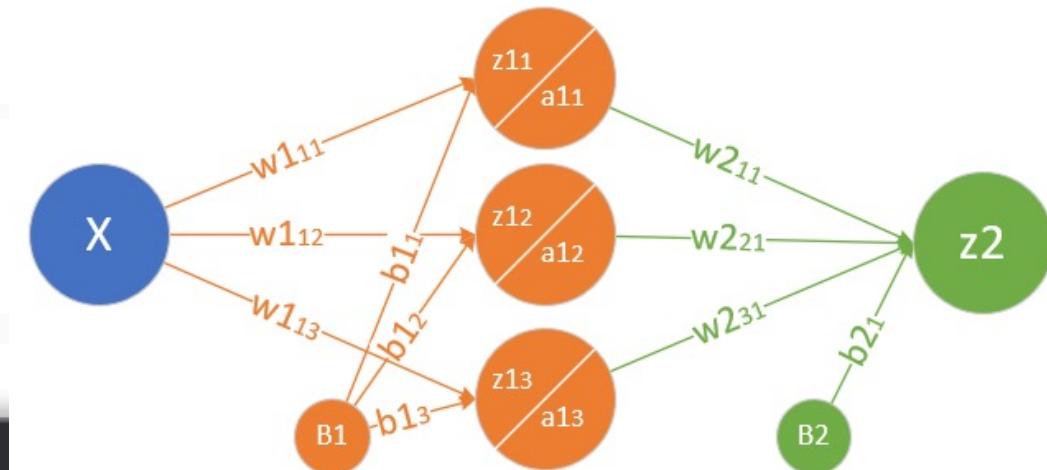
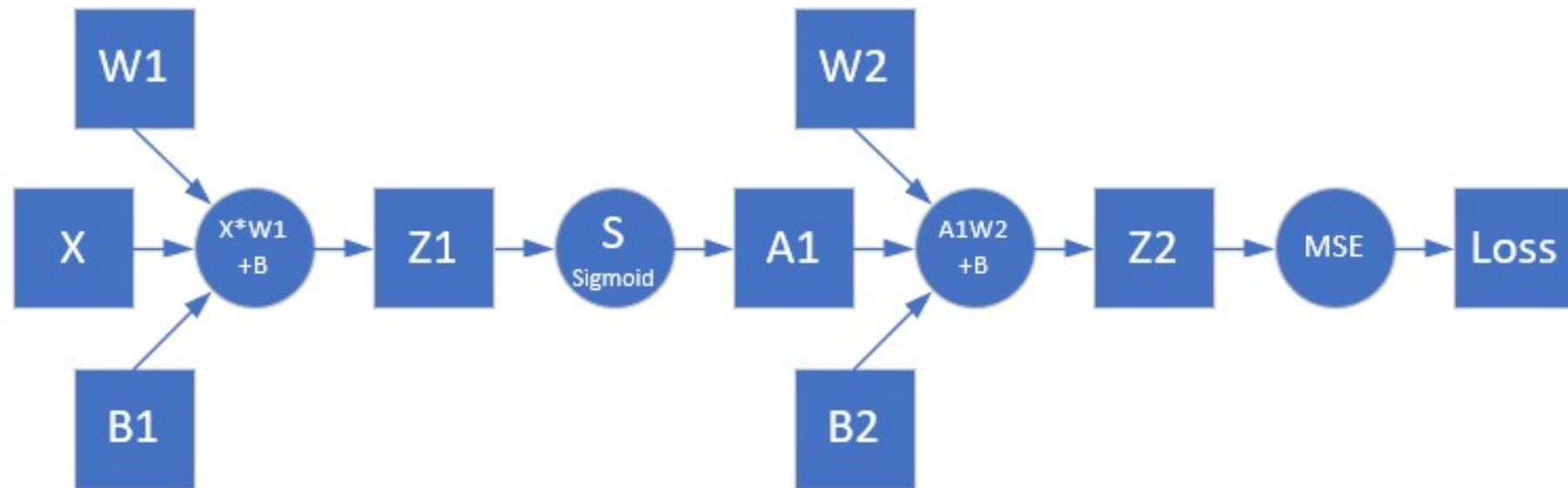
$$B2 = (b_1^2)$$

## 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 $1 \times 1$ ：

$$Z2 = (z_1^2)$$

# 定义神经网络 - 前向计算



## 隐层

- 线性计算

$$z_1^1 = x \cdot w_{11}^1 + b_1^1$$

$$z_2^1 = x \cdot w_{12}^1 + b_2^1$$

$$z_3^1 = x \cdot w_{13}^1 + b_3^1$$

矩阵形式:

$$Z1 = X \cdot W1 + B1 \quad (1)$$

- 激活函数

$$a_1^1 = Sigmoid(z_1^1)$$

$$a_2^1 = Sigmoid(z_2^1)$$

$$a_3^1 = Sigmoid(z_3^1)$$

矩阵形式:

$$A1 = Sigmoid(Z1) \quad (2)$$

## 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：

$$z2 = a_1^1 w_{11}^2 + a_2^1 w_{21}^2 + a_3^1 w_{31}^2 + b_1^2$$

矩阵形式

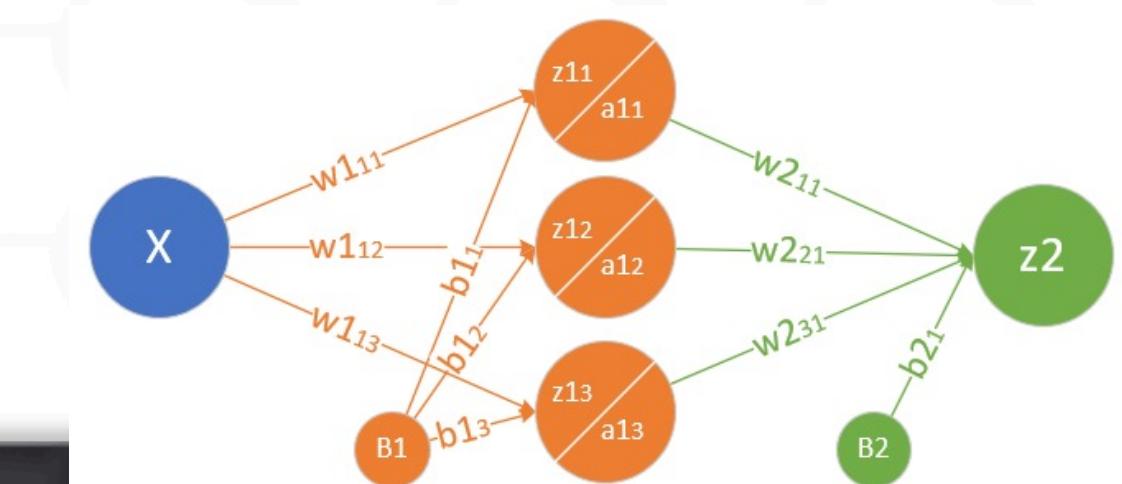
$$Z2 = A1 \cdot W2 + B2 \quad (3)$$

## 损失函数

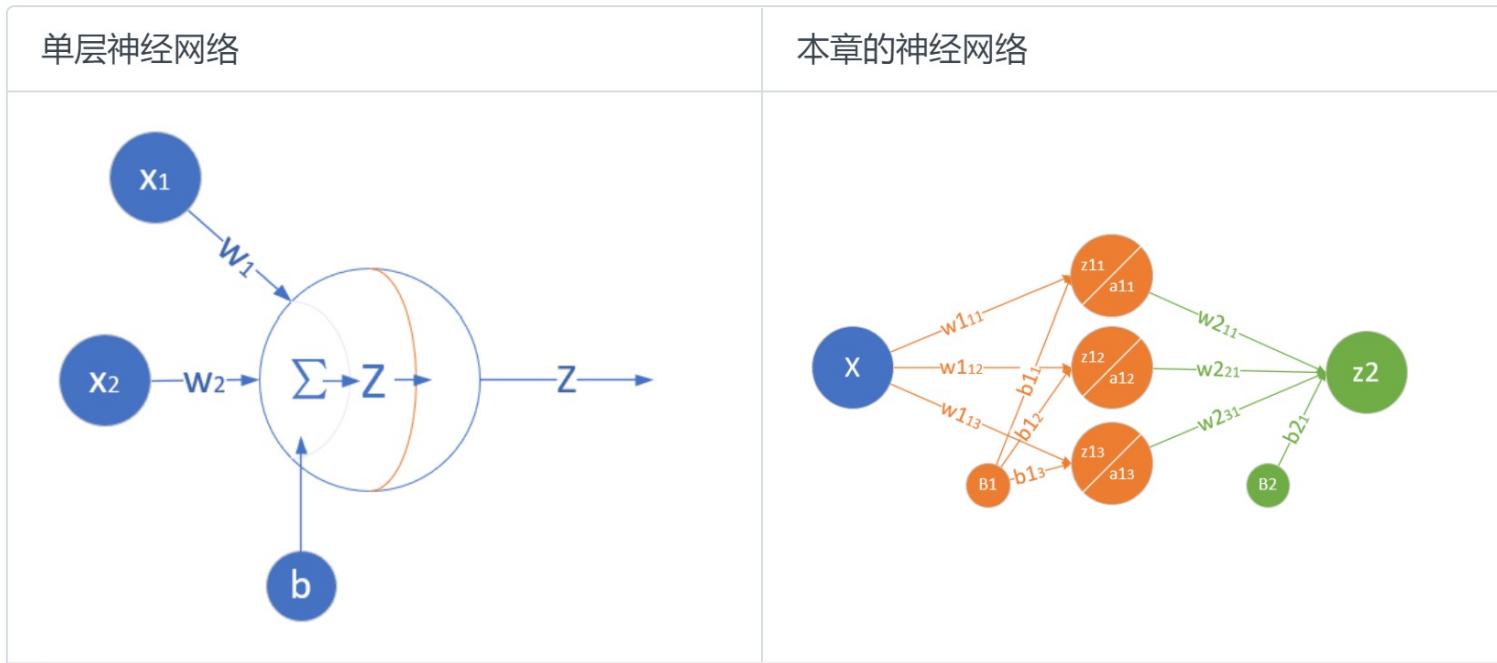
均方差损失函数：

$$loss(w, b) = \frac{1}{2}(z2 - y)^2 \quad (4)$$

其中， $z2$ 是预测值， $y$ 是样本的标签值。



# 非线形回归-定义神经网络 - 反向计算



本章使用了真正的“网络”，而单层神经网络其实只是一个神经元而已。再看本章的网络的右半部分，从隐层到输出层的结构，和单层的神经元结构一模一样，只是输入为3个特征，而左图中输入为两个特征。比较正向计算公式的话，也可以得到相同的结论。这就意味着反向传播的公式应该也是一样的。

# 非线形回归-定义神经网络-反向计算

## 求损失函数对输出层的反向误差

根据公式4：

$$\frac{\partial \text{loss}}{\partial z_2} = z_2 - y \rightarrow dZ2 \quad (5)$$

## 求W2的梯度

根据公式3和W2的矩阵形状，把标量对矩阵的求导分解到矩阵中的每一元素：

$$dW2 = \frac{\partial \text{loss}}{\partial W2} = \begin{pmatrix} \frac{\partial \text{loss}}{\partial z_2} \frac{\partial z_2}{\partial w_{11}^2} \\ \frac{\partial \text{loss}}{\partial z_2} \frac{\partial z_2}{\partial w_{21}^2} \\ \frac{\partial \text{loss}}{\partial z_2} \frac{\partial z_2}{\partial w_{31}^2} \end{pmatrix} = \begin{pmatrix} dZ2 \cdot a_1^1 \\ dZ2 \cdot a_2^1 \\ dZ2 \cdot a_3^1 \end{pmatrix}$$

$$= (a_1^1 \quad a_2^1 \quad a_3^1)^T \cdot dZ2 = A1^T \cdot dZ2 \rightarrow dW2 \quad (6)$$

与单层神经网络相比，除了把X换成A以外，其它的都一样。对于输出层来说，A就是它的输入，也就相当于X。

## 求B2的梯度

$$\frac{\partial \text{loss}}{\partial B2} = dZ2 \rightarrow dB2 \quad (7)$$

与单层神经网络相比，除了把X换成A以外，其它的都一样。对于输出层来说，A就是它的输入，也就相当于X。



# 非线性二分类-定义神经网络结构

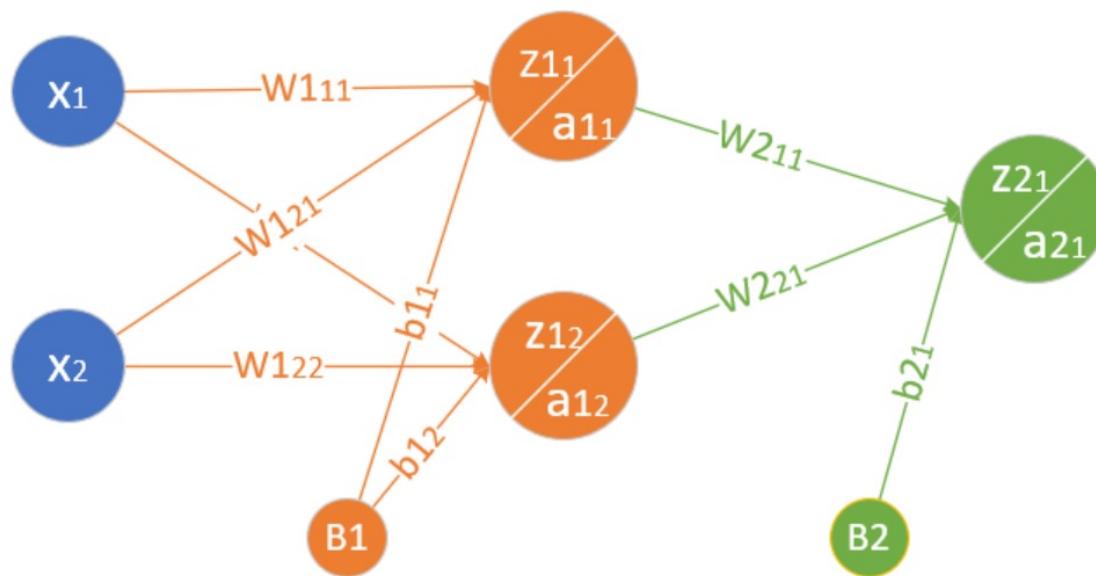


图10-6 非线性二分类神经网络结构图

- 隐层 $2 \times 2$ 的权重矩阵 $W1$

$$X = (x_1 \ x_2)$$

$$W1 = \begin{pmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{pmatrix}$$

- 隐层 $1 \times 2$ 的偏移矩阵 $B1$

$$B1 = (b_1^1 \ b_2^1)$$

- 隐层由两个神经元构成

$$Z1 = (z_{11} \ z_{12})$$

$$A1 = (a_{11} \ a_{12})$$

- 输出层 $2 \times 1$ 的权重矩阵 $W2$

$$W2 = \begin{pmatrix} w_{11}^2 \\ w_{21}^2 \end{pmatrix}$$

- 输出层 $1 \times 1$ 的偏移矩阵 $B2$

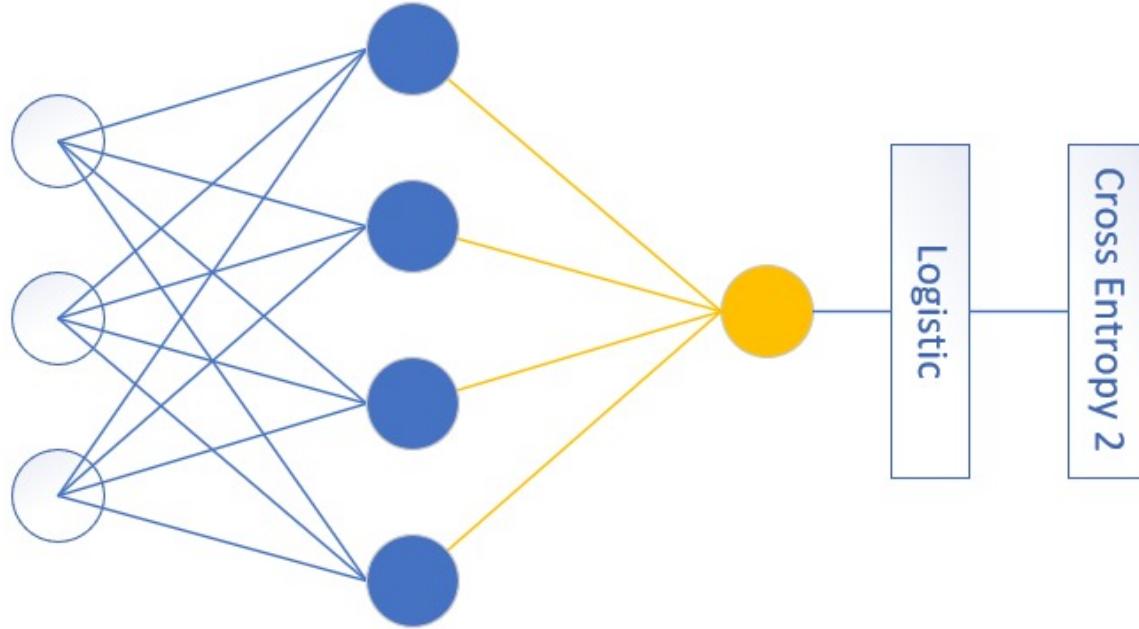
$$B2 = (b_1^2)$$

- 输出层有一个神经元使用Logistic函数进行分类

$$Z2 = (z_1^2)$$

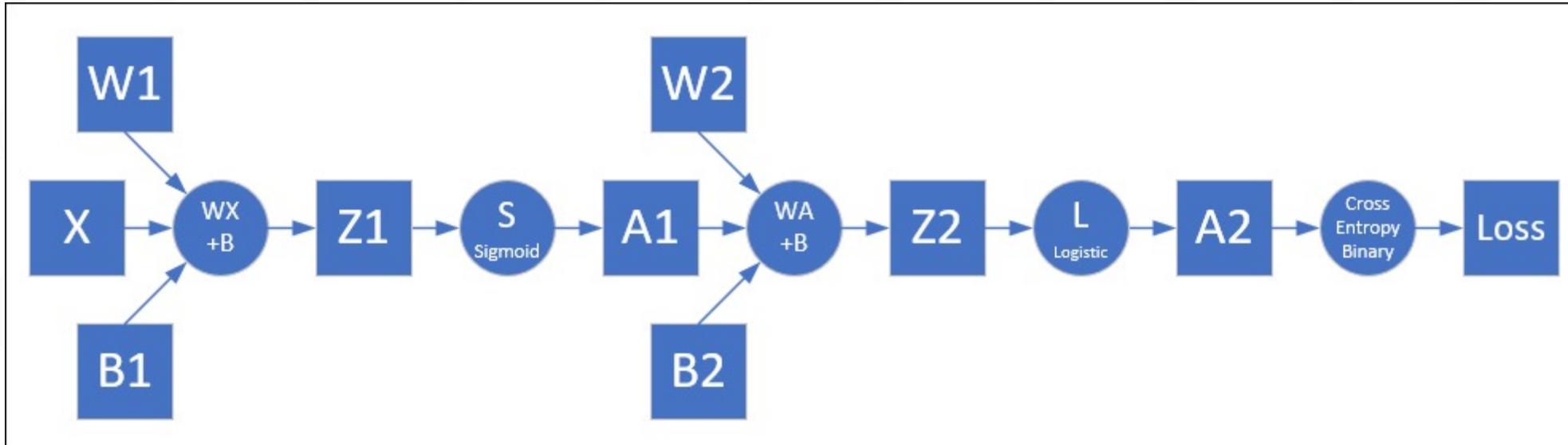
$$A2 = (a_1^2)$$

# 定义神经网络结构 - 二分类的双层神经网络



输入特征值可以有很多，隐层单元也可以有很多，输出单元只有一个，且后面要接Logistic分类函数和二分类交叉熵损失函数。

# 定义神经网络结构 - 前向计算



## 第一层

- 线性计算

$$z_1^1 = x_1 w_{11}^1 + x_2 w_{21}^1 + b_1^1$$

$$z_2^1 = x_1 w_{12}^1 + x_2 w_{22}^1 + b_2^1$$

$$Z1 = X \cdot W1 + B1$$

- 激活函数

$$a_1^1 = Sigmoid(z_1^1)$$

$$a_2^1 = Sigmoid(z_2^1)$$

$$A1 = (a_1^1 \ a_2^1)$$

## 第二层

- 线性计算

$$z_1^2 = a_1^1 w_{11}^2 + a_2^1 w_{21}^2 + b_1^2$$

$$Z2 = A1 \cdot W2 + B2$$

- 分类函数

$$a_1^2 = Logistic(z_1^2)$$

$$A2 = Logistic(Z2)$$

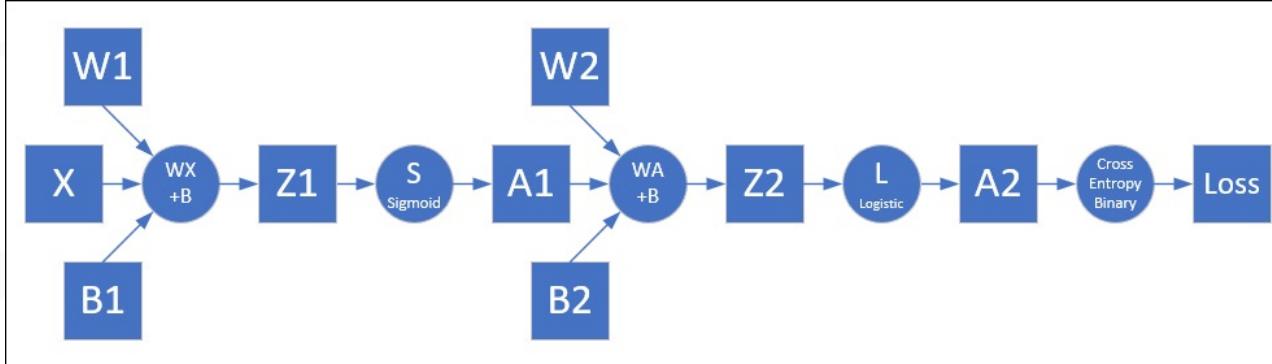
## 损失函数

我们把异或问题归类成二分类问题，所以使用二分类交叉熵损失函数：

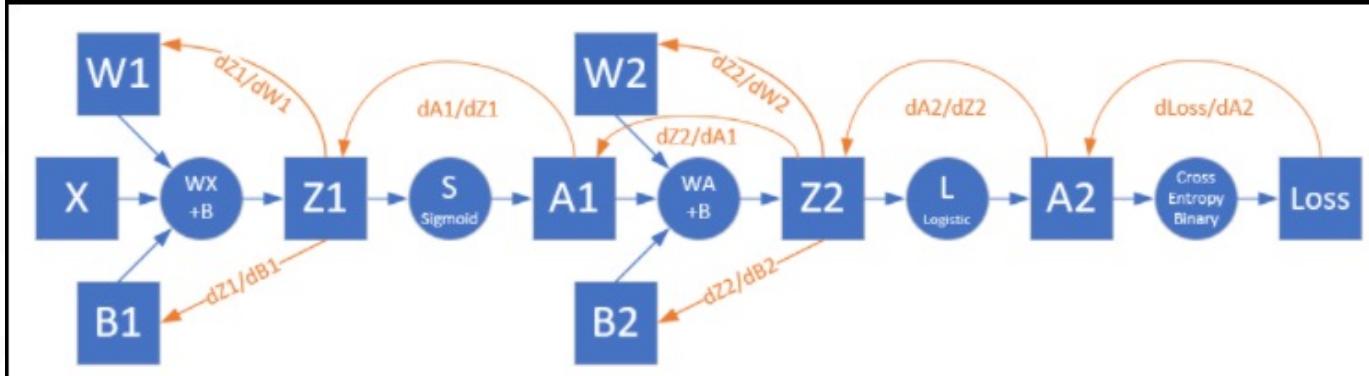
$$loss = -Y \ln A2 + (1 - Y) \ln(1 - A2) \quad (12)$$

在二分类问题中，Y、A2都是一个单一的数值，而非矩阵，但是为了前后统一，我们可以把它们看作是一个1x1的矩阵。

# 定义神经网络结构 - 前向计算



# 定义神经网络结构 – 反向计算



求损失函数对输出层的反向误差

对损失函数求导，可以得到损失函数对输出层的梯度值，即图10-9中的Z2部分。

根据公式12，求A2和Z2的导数（此处A2、Z2、Y可以看作是标量，以方便求导）：

$$\frac{\partial \text{loss}}{\partial Z2} = \frac{\partial \text{loss}}{\partial A2} \frac{\partial A2}{\partial Z2} = \frac{A2 - Y}{A2(1 - A2)} \cdot A2(1 - A2) = A2 - Y \rightarrow dZ2 \quad (13)$$

求W2和B2的梯度

$$\frac{\partial \text{loss}}{\partial W2} = \begin{pmatrix} \frac{\partial \text{loss}}{\partial w_{11}} \\ \frac{\partial \text{loss}}{\partial w_{21}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \text{loss}}{\partial Z2} \frac{\partial Z2}{\partial w_{11}} \\ \frac{\partial \text{loss}}{\partial Z2} \frac{\partial Z2}{\partial w_{21}} \end{pmatrix}$$

$$= \begin{pmatrix} dZ2 \cdot a_1^1 \\ dZ2 \cdot a_2^1 \end{pmatrix} = \begin{pmatrix} a_1^1 \\ a_2^1 \end{pmatrix} dZ2$$

$$= A1^T \cdot dZ2 \Rightarrow dW2 \quad (14)$$

$$\frac{\partial \text{loss}}{\partial B2} = dZ2 \Rightarrow dB2 \quad (15)$$

求损失函数对隐层的反向误差

$$\frac{\partial \text{loss}}{\partial A1} = \begin{pmatrix} \frac{\partial \text{loss}}{\partial a_1^1} & \frac{\partial \text{loss}}{\partial a_2^1} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial \text{loss}}{\partial Z2} \frac{\partial Z2}{\partial a_1^1} & \frac{\partial \text{loss}}{\partial Z2} \frac{\partial Z2}{\partial a_2^1} \end{pmatrix}$$

$$= (dZ2 \cdot w_{11}^0 \quad dZ2 \cdot w_{21}^0)$$

$$= dZ2 \cdot (w_{11}^0 \quad w_{21}^0)$$

$$= dZ2 \cdot W2^T$$

$$\frac{\partial A1}{\partial Z1} = A1 \odot (1 - A1) \Rightarrow dA1 \quad (17)$$

所以最后到达z1的误差矩阵是：

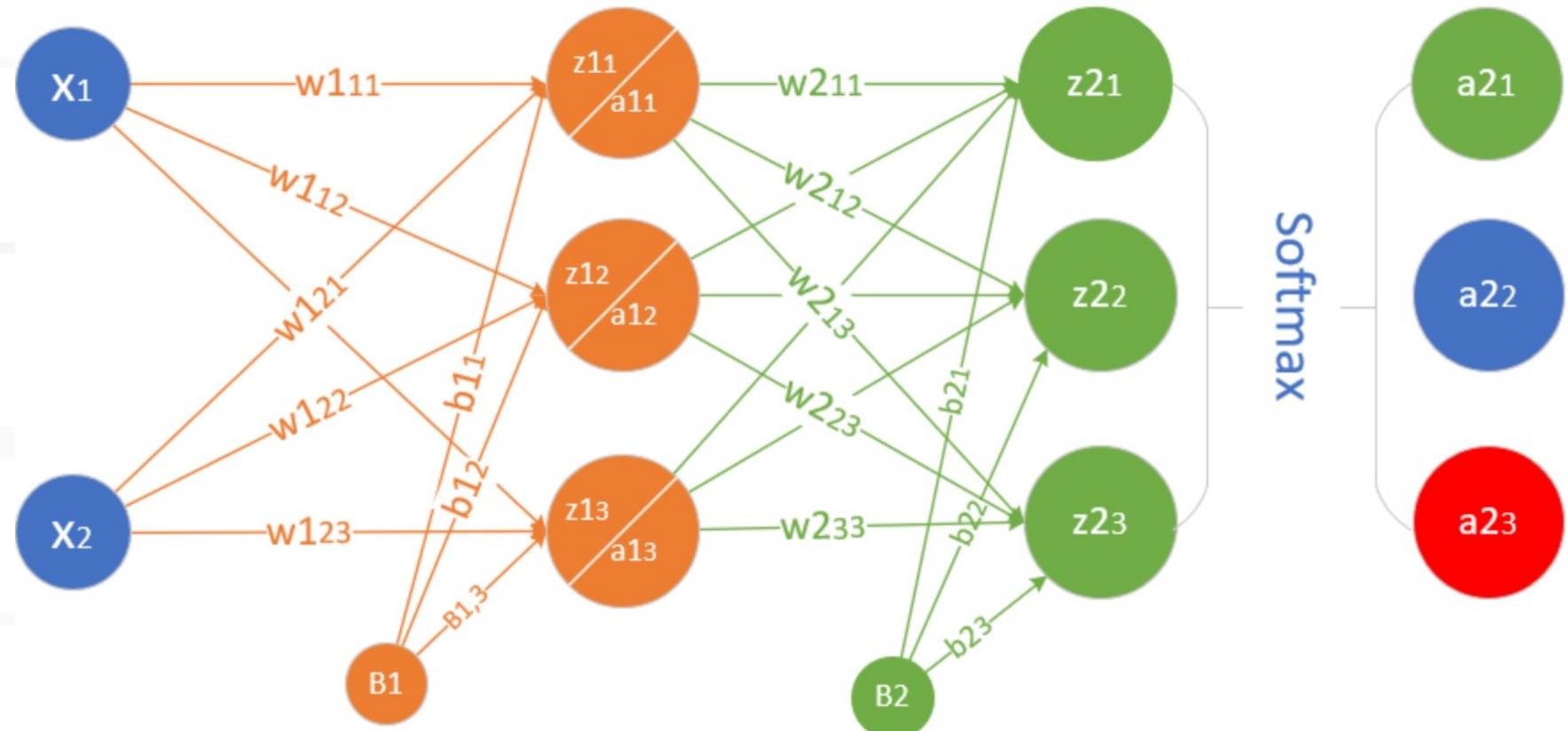
$$\frac{\partial \text{loss}}{\partial Z1} = \frac{\partial \text{loss}}{\partial A1} \frac{\partial A1}{\partial Z1} = dZ2 \cdot W2^T \odot dA1 \rightarrow dZ1 \quad (18)$$

有了dZ1后，再向前求W1和B1的误差，我们直接列在下面了：

$$dW1 = X^T \cdot dZ1 \quad (19)$$

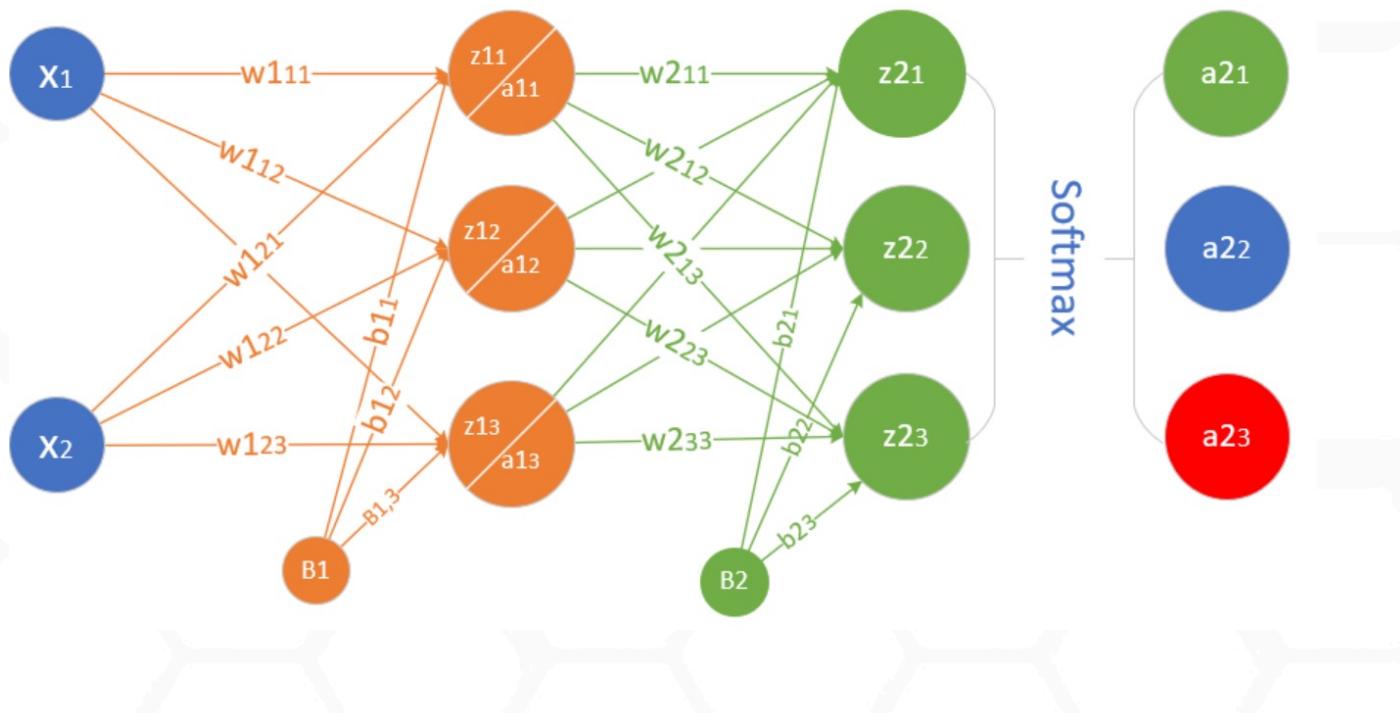
$$dB1 = dZ1 \quad (20)$$

# 非线性多分类-神经网络结构



- 输入层两个特征值  $x_1, x_2$

# 非线性多分类-神经网络结构



- 隐层2x3的权重矩阵  $W1$

$$W1 = \begin{pmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \end{pmatrix}$$

- 隐层1x3的偏移矩阵  $B1$

$$B1 = \begin{pmatrix} b_1^1 & b_2^1 & b_3^1 \end{pmatrix}$$

- 隐层由3个神经元构成
- 输出层3x3的权重矩阵  $W2$

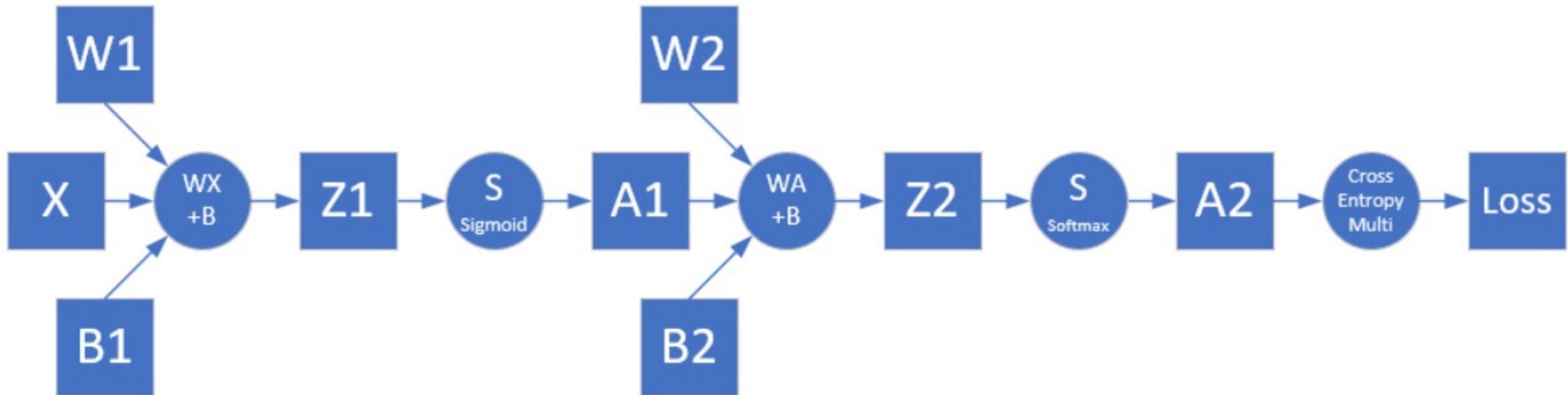
$$W2 = \begin{pmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \\ w_{31}^2 & w_{32}^2 & w_{33}^2 \end{pmatrix}$$

- 输出层1x1的偏移矩阵  $B2$

$$B2 = \begin{pmatrix} b_1^2 & b_2^2 & b_3^2 \end{pmatrix}$$

- 输出层有3个神经元使用Softmax函数进行分类

# 非线性多分类 - 前向计算



# 非线性多分类 - 前向计算

## 第一层

- 线性计算

$$z_1^1 = x_1 w_{11}^1 + x_2 w_{21}^1 + b_1^1$$

$$z_2^1 = x_1 w_{12}^1 + x_2 w_{22}^1 + b_2^1$$

$$z_3^1 = x_1 w_{13}^1 + x_2 w_{23}^1 + b_3^1$$

$$Z1 = X \cdot W1 + B1$$

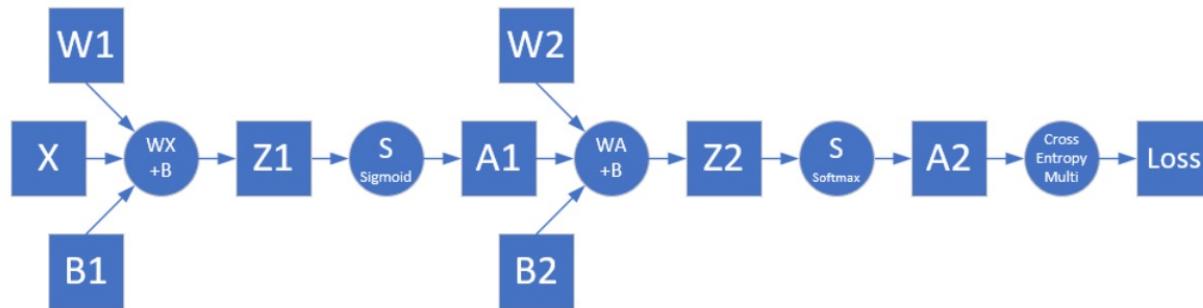
- 激活函数

$$a_1^1 = Sigmoid(z_1^1)$$

$$a_2^1 = Sigmoid(z_2^1)$$

$$a_3^1 = Sigmoid(z_3^1)$$

$$A1 = Sigmoid(Z1)$$



# 非线性多分类 - 前向计算

- 线性计算

$$z_1^2 = a_1^1 w_{11}^2 + a_2^1 w_{21}^2 + a_3^1 w_{31}^2 + b_1^2$$

$$z_2^2 = a_1^1 w_{12}^2 + a_2^1 w_{22}^2 + a_3^1 w_{32}^2 + b_2^2$$

$$z_3^2 = a_1^1 w_{13}^2 + a_2^1 w_{23}^2 + a_3^1 w_{33}^2 + b_3^2$$

$$Z2 = A1 \cdot W2 + B2$$

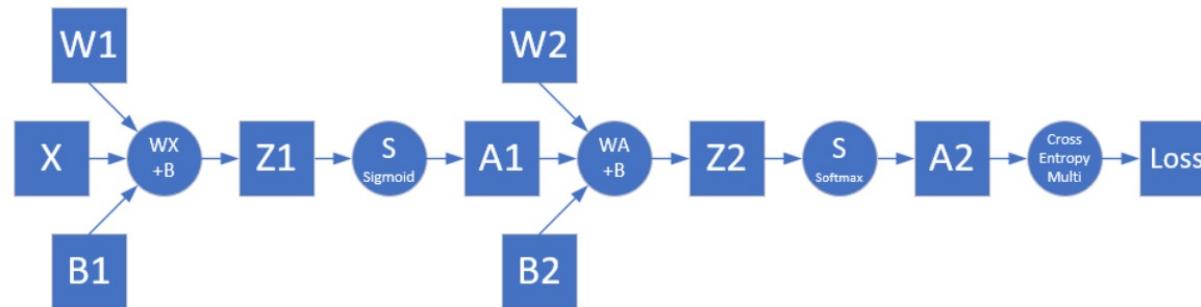
- 分类函数

$$a_1^2 = \frac{e^{z_1^2}}{e^{z_1^2} + e^{z_2^2} + e^{z_3^2}}$$

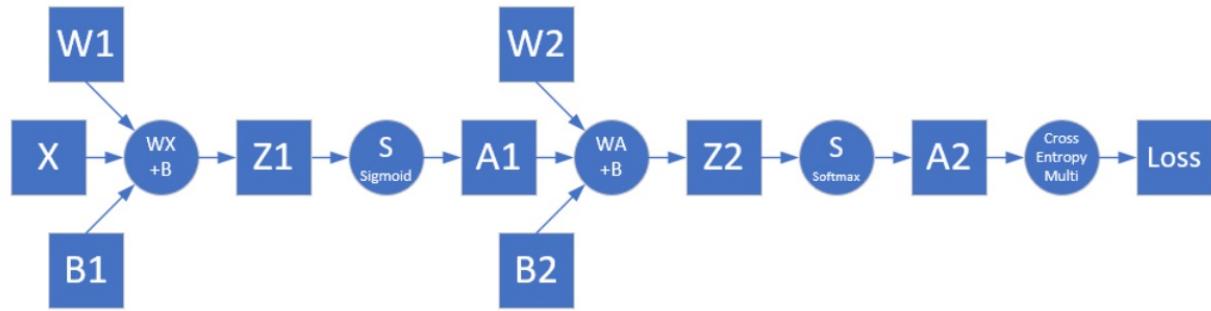
$$a_2^2 = \frac{e^{z_2^2}}{e^{z_1^2} + e^{z_2^2} + e^{z_3^2}}$$

$$a_3^2 = \frac{e^{z_3^2}}{e^{z_1^2} + e^{z_2^2} + e^{z_3^2}}$$

$$A2 = Softmax(Z2)$$



# 非线性多分类 - 前向计算



## 损失函数

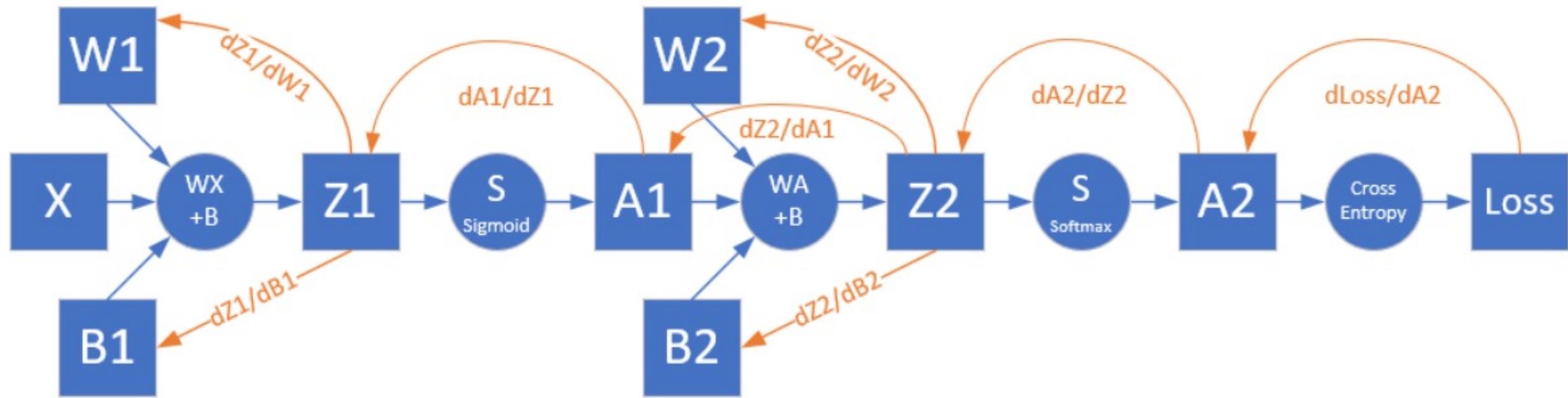
使用多分类交叉熵损失函数：

$$loss = -(y_1 \ln a_1^2 + y_2 \ln a_2^2 + y_3 \ln a_3^2)$$

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^n y_{ij} \ln(a_{ij}^2)$$

m为样本数，n为类别数。

# 非线性多分类 – 反向计算



# Q&A





# Reactor

## Thank You!