深度学习概论

韩雅妮

XiDian University

在人工智能的各个分支中，发展最为迅速的就是深度学习。深度学习，一般就是指训练神经网络。

只要给定足够多的训练样本x,y，神经网路就能很好的拟合出一个函数来建立x和y之间的映射关系。

蓝色曲线作为预测函数，是ReLU函数(Recitified Linear Unit,修正线性单元)。该函数的特点是，前一部分输出为0，后部分是直线。

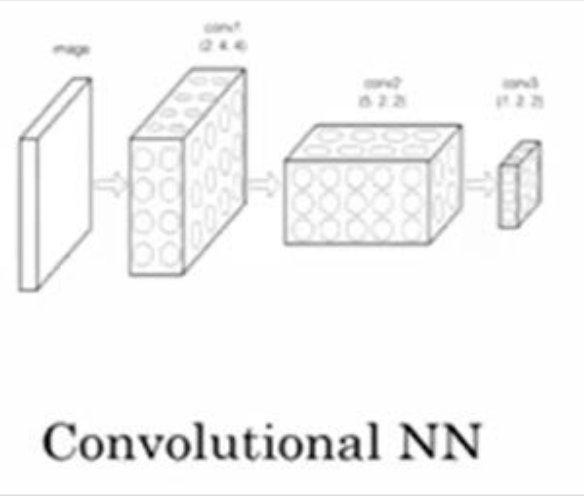
监督学习

Input(x)	Output (y)	Application
Home features	Price	Real Estate
Ad, user info	Click on ad? (0/1)	Online Advertising
Image	Object (1,...,1000)	Photo tagging
Audio	Text transcript	Speech recognition
English	Chinese	Machine translation

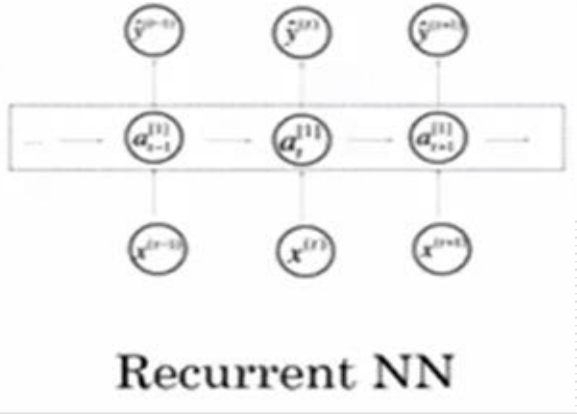
神经网路分类

（1）标准神经网络

（2）卷积神经网络



（3）循环神经网络



数据

数据分为结构化数据和非结构化数据。

（1）结构化数据：

基于数据库的数据，每一个特征都有明确的定义；

Structured Data

Size	#bedrooms	…	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

（2）非结构化数据：

音频，图片，文本等。

Unstructured Data

Audio

Image

Four scores and seven years ago...

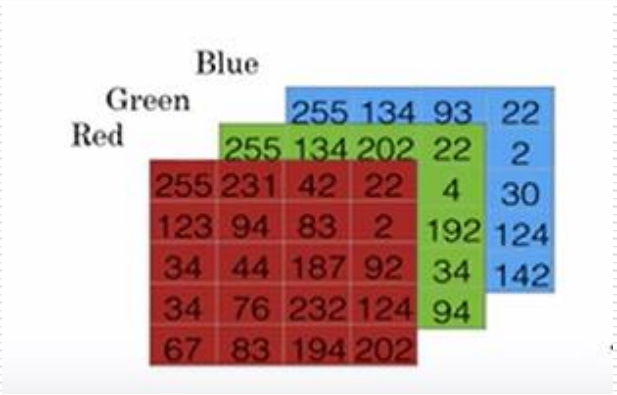
Text

神经网路基础

二元分类

例：判断一张图片上是否为猫。

分析：用二元分类解决。当输出1时，是猫，输出0时，不是猫。



假设图片大小为64*64，在计算机中，存3个64*64的矩阵，代表RGB。定义一个与该图像相对应的向量x，将RGB中的像素值放入x中，x的长度为3*64*64=12288。

目标： 给定输入特征向量x与一张图片对应，希望求是猫的概率： $\hat{y} = P(y=1|x)$ ， $0 \le \hat{y} \le 1$

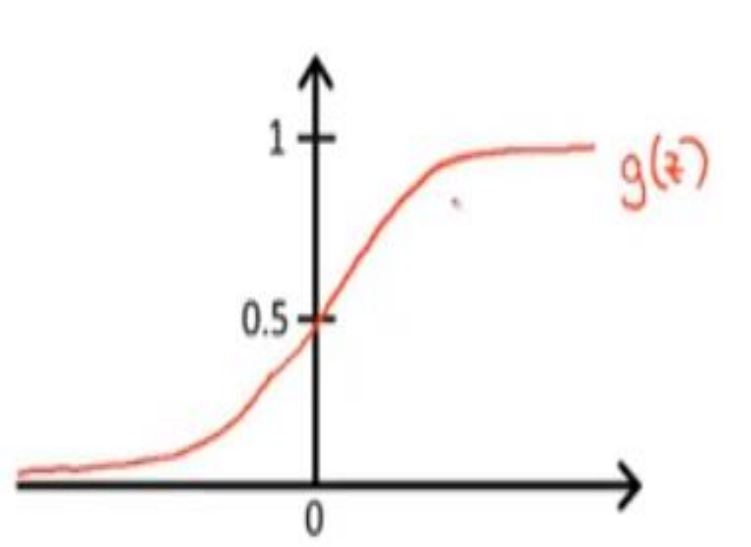
Logistic回归

$\hat{y} = \sigma(w^T x + b)$

其中σ为sigmoid函数，

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

目标：学习参数w和b。



Logistic回归代价函数

单一样本的损失函数：

$$L(\hat{y},y) = -(y\log(\hat{y})+(1-y)\log(1-\hat{y}))$$

m个样本：$(x^{(1)}, y^{(1)}),\ldots, (x^{(m)}, y^{(m)})$

整体的代价函数：

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m}(y^{(i)}log(\hat{y}^{(i)})+(1-y^{(i)})(log(1-\hat{y}^{(i)}))]$$

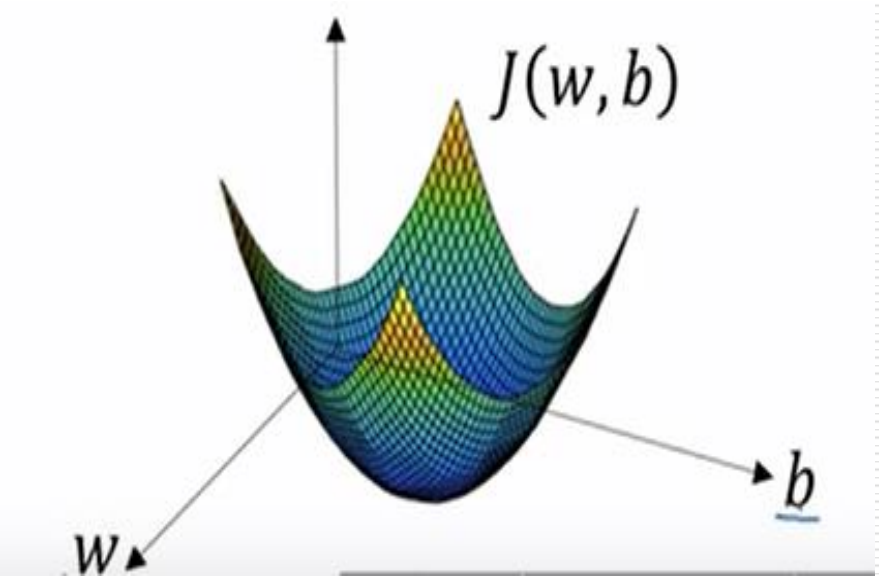
希望找到w和b来最小化$J(w,b)$，使用梯度下降算法来调整这两个参数。

梯度下降算法

迭代更新参数w和b：

$$w := w - \alpha \frac{\partial}{\partial w} J(w, b)$$

$$b := b - \alpha \frac{\partial}{\partial b} J(w, b)$$



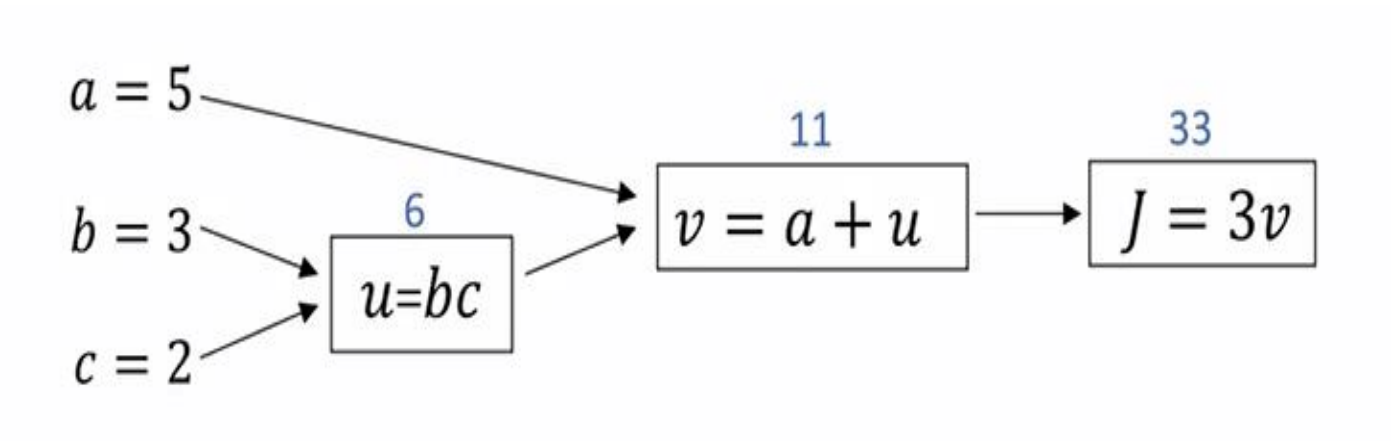
计算图

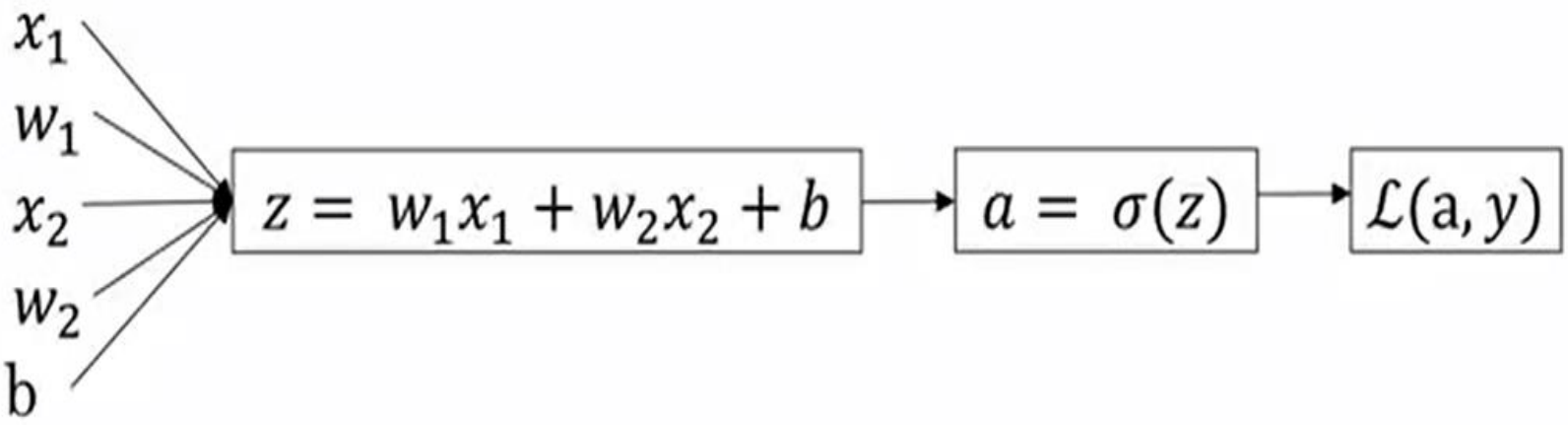
例：J(a,b,c) = 3(a+bc)

分为3步：(1) u = bc;

(2) v = a + u;

(3) J = 3v.





Logistic回归：调整w和b，使损失L最小。

反向传播：

（1）计算损失函数L对a的导数：

$$da = \frac{dL(a,y)}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

（2）$dz = \frac{dL(a,y)}{dz} = \frac{dL(a,y)}{da}\frac{da}{dz} = a - y;$

（3）反向传播最后一步，算出需要改变w和b多少。

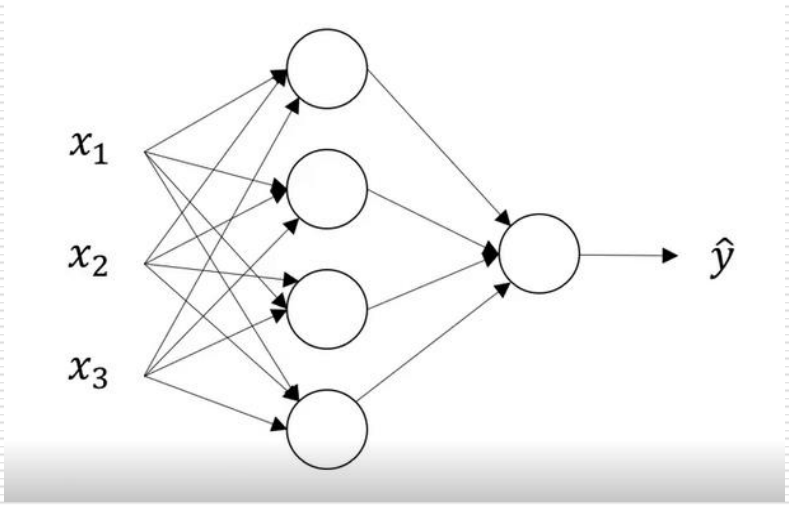
$$dw_1 = x_1 dz;$$

$$dw_2 = x_2 dz;$$

$$db = dz.$$

浅层神经网路

如右图为双层神经网路（不算输入层）
输入层：x_1，x_2，x_3
隐藏层：使用训练集进行训练，已知输入和输出，中间结点的真实值未知，因此称为隐藏层。



$a^{[0]} = x$ 输入特征

$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$ 有参数$w^{[1]}$,$b^{[1]}$, $w^{[1]}$为4*3维矩阵。 4个隐藏单元，3个输入单元。

$\hat{y} = a^{[2]}$

对于隐藏层的第一个节点：

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

第二个节点：

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

向量化：

$$Z^{[1]} = W^{[1]}\mathrm{x} + b^{[1]}$$

$$A^{[1]} = \sigma(Z^{[1]})$$

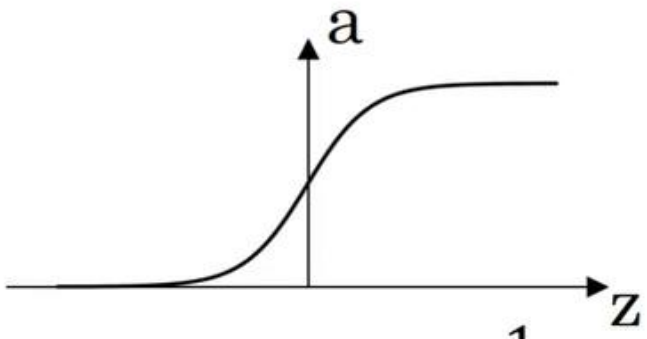
$$Z^{[2]} = W^{[2]}\ A^{[1]} + b^{[2]}$$

$$A^{[2]} = \sigma(Z^{[2]})$$

激活函数

1. simgoid函数：

函数值介于0和1之间。

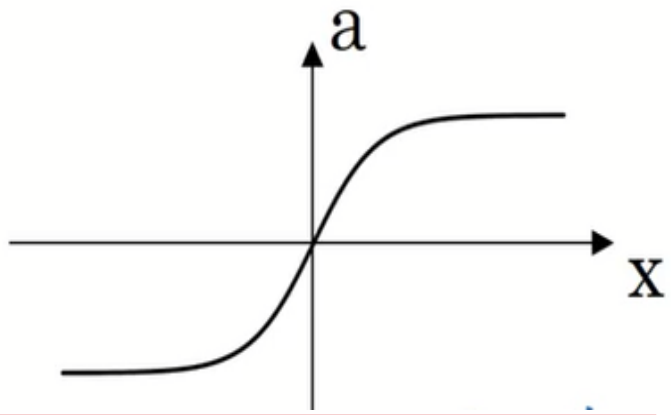


2. tanh函数：

函数值介于-1和1之间。

$a = \tanh(x)$

$$= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$



对比：

tanh函数值介于-1和1之间，激活函数的平均值更接近0，有类似数据中心化的效果，为下一层的学习更方便。

缺点：

当z非常大或者非常小时，导数的梯度(函数斜率)就很小，接近于0，拖慢梯度下降算法。

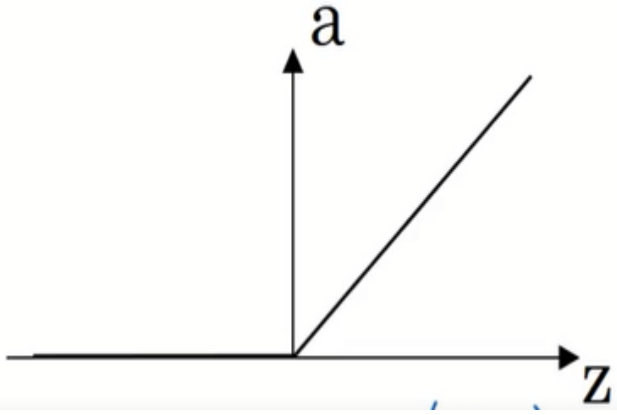
3. ReLU函数：

Recitified Linear Unit,修正线性单元。

a = max(z, 0)

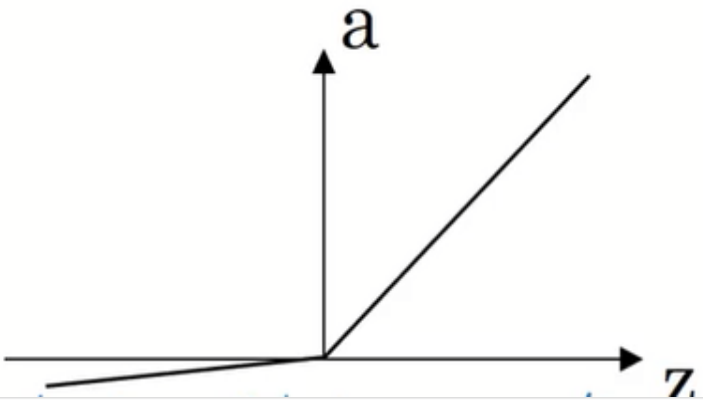
z为正时，导数为1；

z为负时，导数为0。（缺点）



4. 带泄露的ReLU函数：

a = max(0.01z, z)



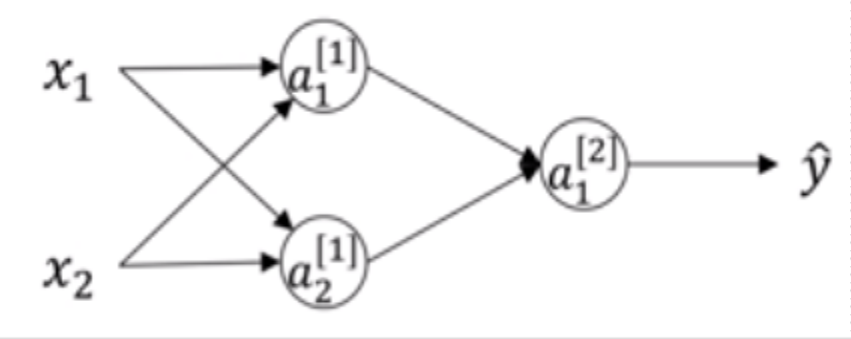
激活函数的选择

1. 若输出值为0和1，做二元分类时，simgoid函数适合作输出层的激活函数，然后其他所有单元都用ReLU函数。

2. 如果不确定隐藏层用哪个函数，就用ReLU函数。因为ReLU函数没有斜率接近于0时减缓学习速度的效应，因此神经网络的学习速度会快很多。

随机初始化权重

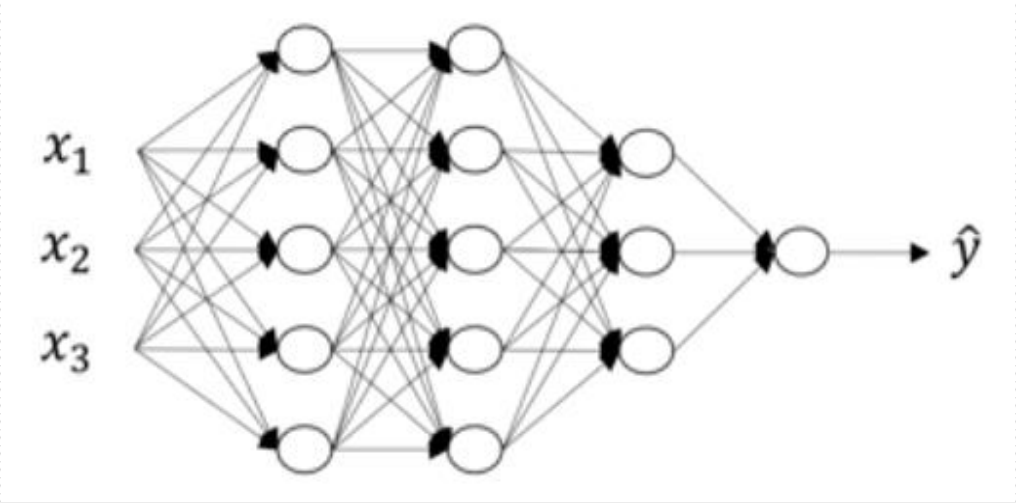
假设$w^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$



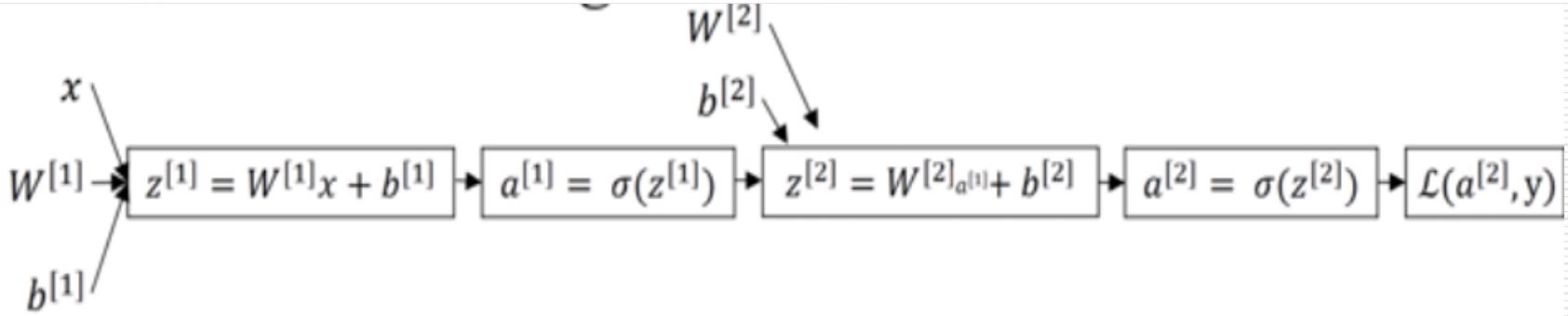
当给神经网络中输入任何样本时， $a_1^{[1]}$和$a_2^{[1]}$是一样的，这两个隐藏单元就完全一样，对称了，意味着节点计算完全一样的函数，在这种情况下，多个隐藏层没有意义。

深层神经网路

4层神经网路（有3个隐藏层）



参数和超参数



参数：$\boldsymbol{w}^{[1]}$，$\boldsymbol{b}^{[1]}$，$\boldsymbol{w}^{[2]}$，$\boldsymbol{b}^{[2]}$，…

超参数：学习率α，循环次数，隐藏层数L，隐藏层单元数，激活函数等。

控制w和b的所有参数都称为超参数。

Thanks