沈征宇 18302010009

代码基本架构

本网络实现了可伸缩易调整的网络结构，例如可以灵活设置层数、神经元个数、学习率等

无论是回归网络，还是分类网络，都可以将各参数作为构造函数的参数传入构造方法

回归拟合sinx

因为sigmoid值域(0,1)，而sinx值域(-1,1)

需要经过函数映射，将g(sinx)值域限制在(0,1)

可以用到g(x)=(x+1)/2，很好地改变了值域，经过反函数运算，可以实现拟合sinx

输入层神经元数：1，代表一个输入值

输出层神经元数：1，代表一个输出值

隐藏层神经元数：10

精度限制：0.00001

学习率：2.0

流程：

先获得指定规模数据点位训练集，做gx映射，再将它作为参数传给神经网络进行训练，具体是先固定输入到隐藏层的权矩阵，计算误差对隐藏层到输出层的权矩阵的偏导，利用梯度下降法修正隐藏层到输出层的权矩阵；再固定隐藏层到输出层的权矩阵，计算误差对输入到隐藏层的权矩阵的偏导，利用梯度下降法修正输入到隐藏层的权矩阵

对于每一个数据点位，如果结果值与期望值的差大于指定精度，神经网络会利用它进行网络的修正；否则成功次数++

当所有数据点位被使用，称为一个epoch

当一个epoch中所有点位的期望值和结果值误差都可接受，即训练成功率100%，或者达到了最大训练次数，那么可以停止训练；否则重新根据训练集进行校正

最后获得测试集，计算误差、准确率

分类手写汉字

预处理和特征提取

采用逐像素特征提取的方法，遍历所有像素点，提取数字样本的特征向量。

将像素点RGB值低位为0xffffff的特征值设为0（白色255,255,255），反之设为1。

归一化的图像生成一个28x28的布尔矩阵，依次取每列元素，转化为784x1的列矩阵，作为输入图像的特征向量。

在getSamples方法中，新建一个二维数组，第一维维度是所有的图片张数（=类数*每类下的图片），第二维维度是图片像素数+分类数；

第一个字在二维数组上的一维维度是0,12,24

即每12个一维index指示了“博学笃志切问近思自由无用”

每一个二维维度上，前imageSizeximageSize(28x28)个元素，是这张图片的特征向量，一行append在前一行之后，后category(12)个元素，指示了这个特征向量属于哪个分类

提取了特征之后，就转化为利用BP神经网络实现多分类问题

BP神经网络

根据ppt471页提到，有以下小技巧：

边的随机赋权重：[-1,+1]/sqrt(number of units)

每个结点的bias设置：最后一层[-0.2,0.2]，否则[0,-1]

隐藏层激活函数使用sigmoid

输出层使用softmax回归，即每一个输出结点的结果是该元素的指数与所有元素指数和的比值

用Loss对yk求导，得到梯度是yk-1，它是更新的梯度

损失函数选择交叉熵，将tk*lnyk求和，目标类tk为1，其它为0，yk是经过softmax算出来的结果

learning rate 一开始设置为0.2，之后设置为0.01，这是因为学习效率直接影响着网络收敛的速度，以及网络能否收敛。学习效率设置偏小可以保证网络收敛，但是收敛较慢；反之则有可能使网络训练不收敛，影响识别效果。因此可以在误差快速下降后放缓学习效率，增强模型稳定性。

不同网络结构、网络参数的实验比较

回归和分类损失函数不同

线性回归我们采用均方误差（MSE）作为损失函数，而逻辑回归（分类算法）采用交叉熵（Cross Entropy）。

损失函数是用于评价当前拟合结果和期望结果有多大程度的不一致

估计sinx中，损失函数取

$$Error = \frac{\sum_{i=1}^{n}(d_i - O_i)^2}{2}$$

这是因为估计sinx是回归问题，回归主要解决的是对具体数值的预测，解决回归问题的神经网络一般只有一个输出节点，这个节点的输出值就是预测值

分类中，损失函数取

$$Error = -\sum_{i=1}^{n} t_i lny(x_i)$$

交叉熵描述了两个不同的概率分布p和q的差异程度，两个分布差异越大，则交叉熵的差异越大。softmax回归的交叉熵损失函数只计算真实类别对应的预测概率的损失，而不考虑其他的预测概率损失（非真实分类对应的y_i为0）.

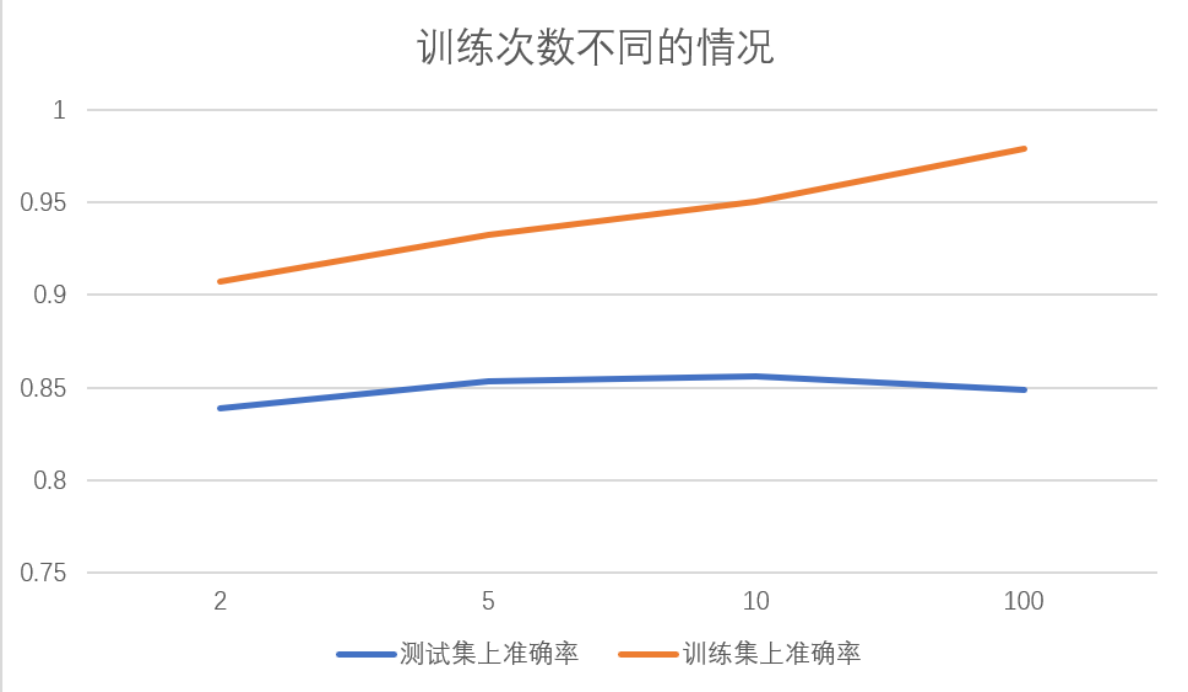
损失函数需要保证输出值和期望值的差距呈现下降趋势，并尽可能小。而交叉熵函数让期望结果的概率值最大，让较大的更大，让较小的更小。

汉字分类中不同网络参数的实验比较

在有明显趋势的实验数据中，绘制了图表

训练次数不同

以下实验数据皆为5次取均值，其它条件相同，后续比较也建立在此基础之上



训练次数	测试集上准确率	训练集上准确率
2	0.8387	0.9071
5	0.8535	0.9322
10	0.8562	0.9502
100	0.8486	0.9788

2次可能较少，还没有调整足够。

100次过多，在训练集上出现了过拟合的情况。

隐藏层神经元个数不同

隐藏层神经元个数	测试准确率	
64	0.8461	
128	0.8515	
256	0.8556	

结合训练效率和测试准确度来看，选择128

初始learning rate不同

注：这里的learning rate指 epoch 1 的learning rate

初始learning rate	测试准确率	
0.1	0.8504	
0.2	0.8534	
0.25	0.8563	

实际上初始learning rate只影响第一次调整

（因为采用了第二次及以后再调整不变）

当训练次数足够，影响不显著

采用0.2

后续learning rate不同

注：这里的learning rate指 epoch 1以后的learning rate

后续learning rate	测试准确率	
0.005	0.8504	
0.01	0.8629	
0.015	0.8622	

当学习率设置的过小时，收敛过程将变得十分缓慢。而当学习率设置的过大时，梯度可能会在最小值附近来回震荡，甚至可能无法收敛。

采用0.01

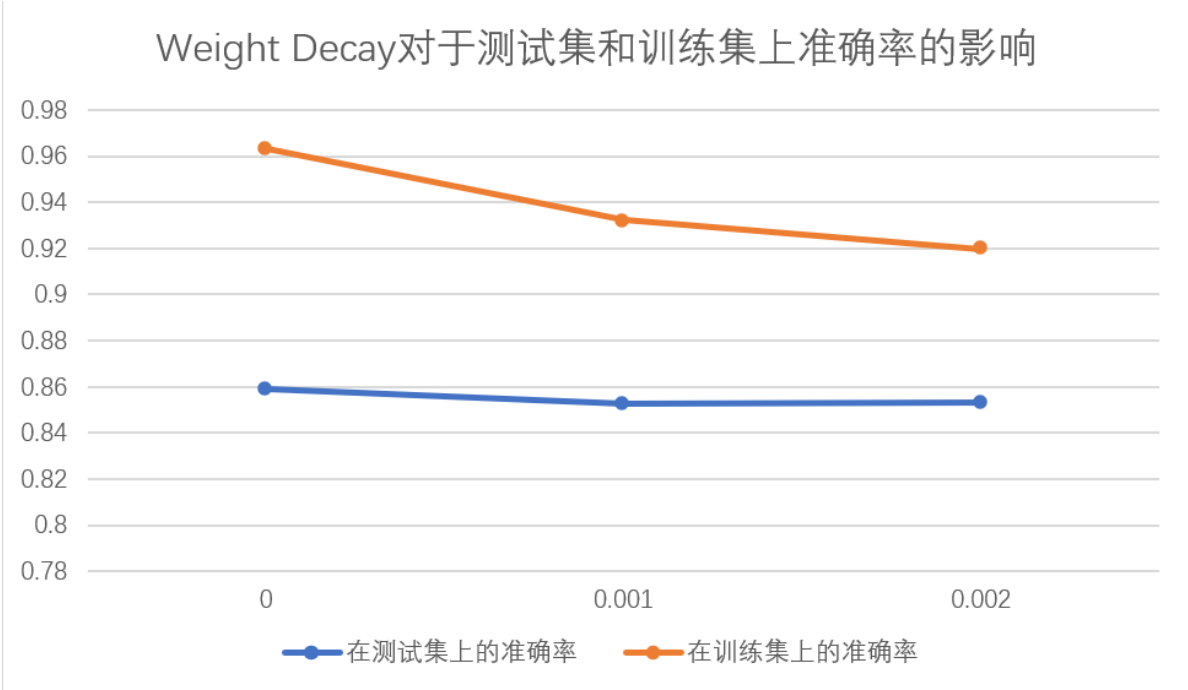
训练集和测试集比例不同

训练集占比	测试准确率	
0.95	0.9038	
0.8	0.8522	
0.7	0.8591	
0.6	0.8434	

测试集太小，样本数不够，准确率并不真实

采用7：3

weight decay



weight decay	测试准确率	训练集上准确率
0.001	0.8525	0.9322
0.002	0.8532	0.9198
0	0.8591	0.9633

可以看出，weight decay会影响在训练集上的准确率，因为weight decay的作用是防止数据在训练集上过拟合，更好地泛化模型，在lab中测试集选取的样本数较少，所以效果不显著

采用0

outlayer初始bias的不同

out layer bias	测试准确率	
(-0.2,0.2)	0.8571	
(-0.5,0.5)	0.8584	
(-1,1)	0.8525	

选择(-0.2,0.2)，兼顾学习效率和测试精度

各Link初始weight不同

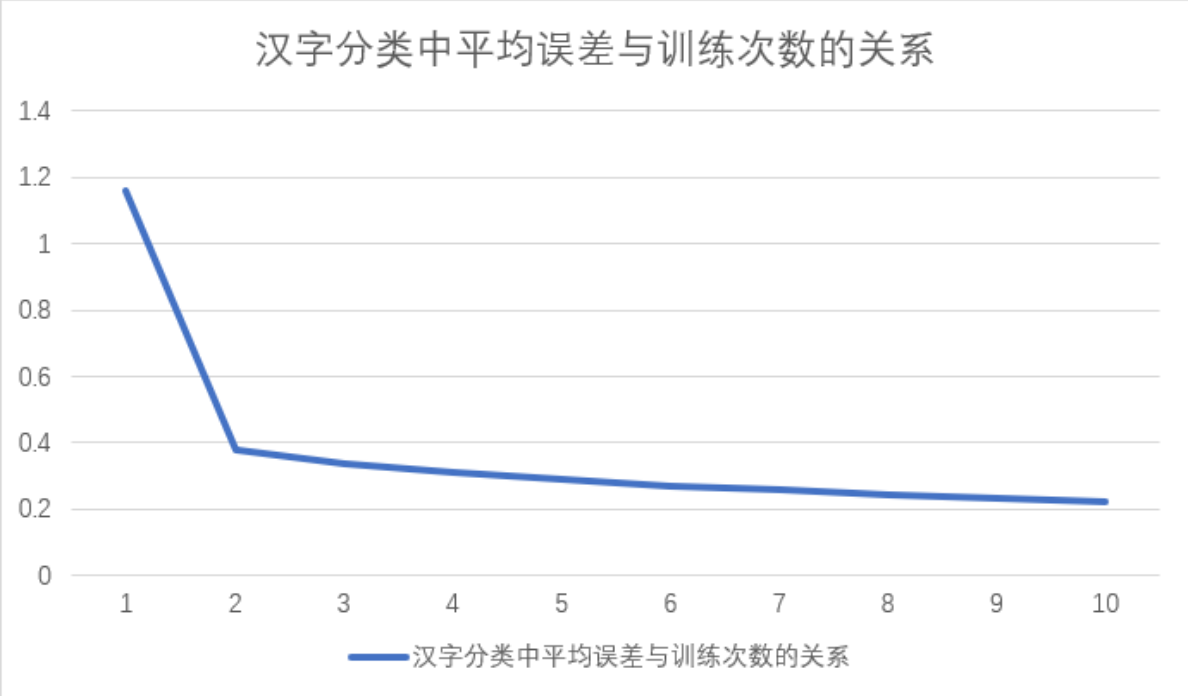
weight	测试准确率	
(-1,1)/numOfSourceLayer	0.8454	
(-1,1)/Math.sqrt(numOfSourceLayer)	0.8525	

可以看出没有显著差别

最后选择(-1,1)/Math.sqrt(numOfSourceLayer)

最佳实践

隐藏神经元个数128, 隐藏层层数1, 初始学习率0.2,第二次及以后学习率0.01,误差阈值0.5, 训练次数10

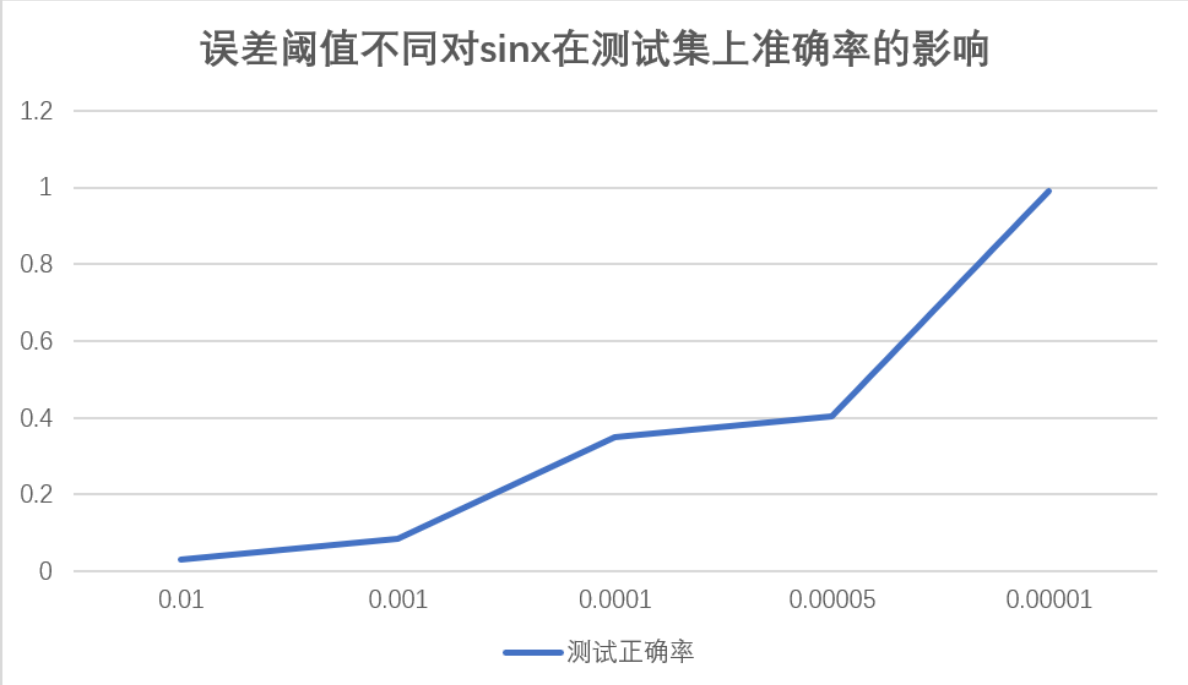


拟合sin函数中不同网络参数的实验比较

对sin函数中，大多数参数通过足量的训练可以达到相同效果

误差阈值不同

10次实验取均值

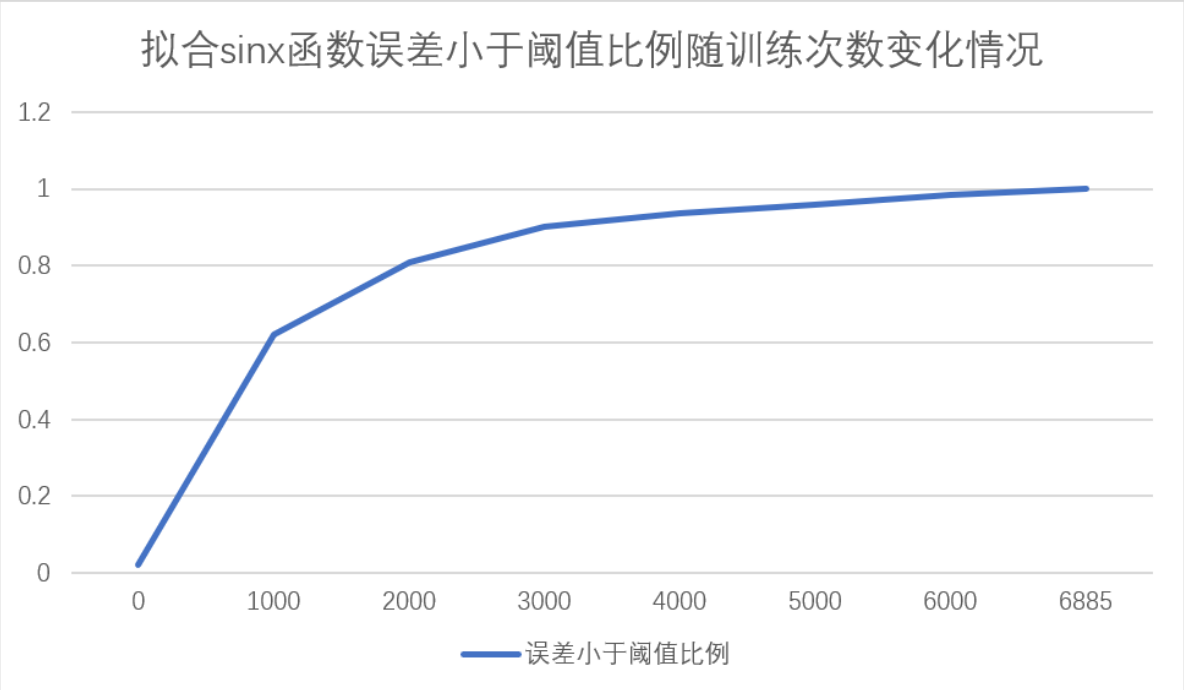


误差阈值	测试正确率
0.01	0.0297
0.001	0.0864
0.0001	0.3511
0.00005	0.4038
0.00001	0.9922

由此看来，需要取得合适的阈值，才能使模型有足够的训练次数

最佳实践

误差阈值0.01，隐藏层层数1，隐藏层神经元数目10，误差阈值0.00001，学习率2.0



对反向传播算法的理解

整个BP神经网络我认为是通过样本得到结果，再将结果和期望值得到差异，反向更新整个网络，最终达到拟合结果，可以预测的过程。

总体网络结构，输入层，隐藏层，输出层，bias可视作输入层及隐藏层中的一个神经元，值为1，它与下一层的每一个结点的Link weight即是bias，这样做可以转化求和公式为线性（看起来）

具体步骤：

1. 初始化整个神经网络，包括，初始化各结点，联结各层并为每条边赋权值
2. 前向传播，得到预测结果
3. 和期望值进行比较
4. 利用数学推导的结果，更新weight
5. 更新bias

以下是学习课程时，记录的本次lab基础知识，资料源于互联网，个人进行总结

发展历史

单个神经元、感知机

BP神经网络

深度学习

深度学习

神经网络：输入层、隐藏层、输出层

层与层之间的神经元有连接，层内互相没有，连接的神经元有对应的权重

隐藏层多——深度学习，隐藏层神经元的个数是自己确定的（如512）

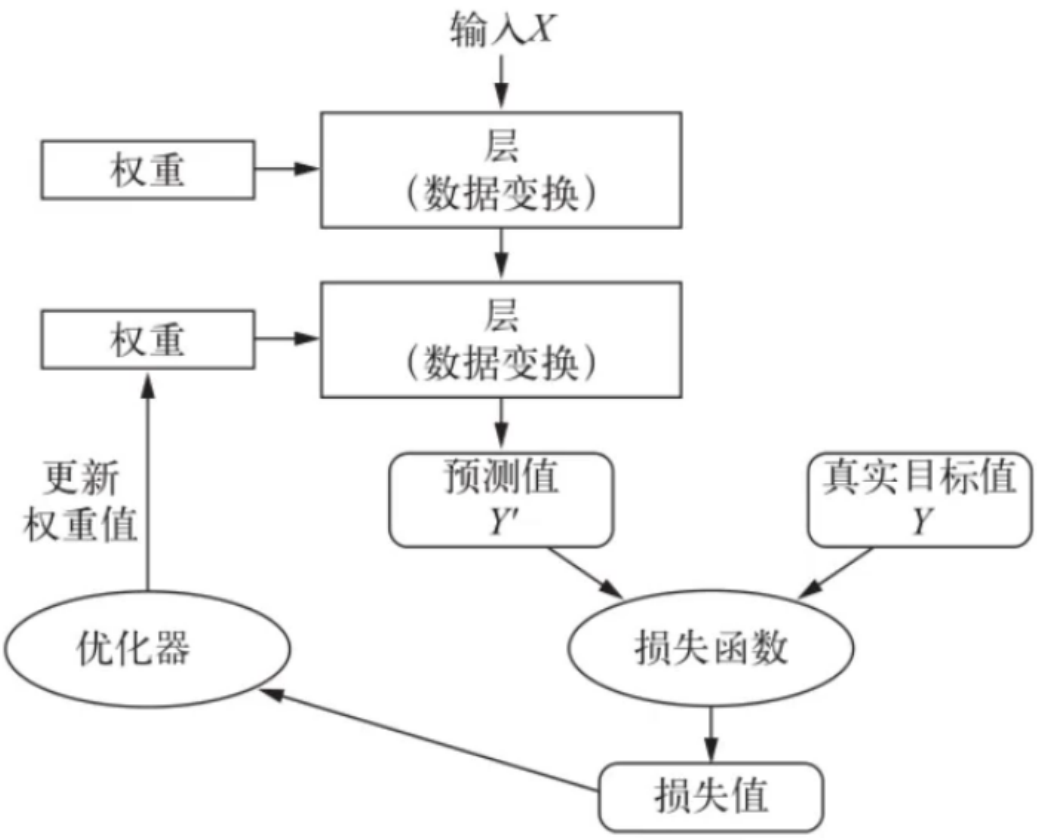
感知机：多个输入，每个输入有权，权与输入值的积求和再加一个偏置，再经过一个非线性函数，就能输出。例子，参加展览考虑（天气、同伴、价格），可以从现实反推感知机的各项参数。因此，可以训练出权重和偏置，得到真实模型。

反向传播

信息的正向传播和误差的反向传播

输入数据，数据变换中赋权重，得到预测值，对比真实目标，得到损失函数

分析损失值，优化器，梯度下降法，对损失函数求偏导，更新权重值



前向传播

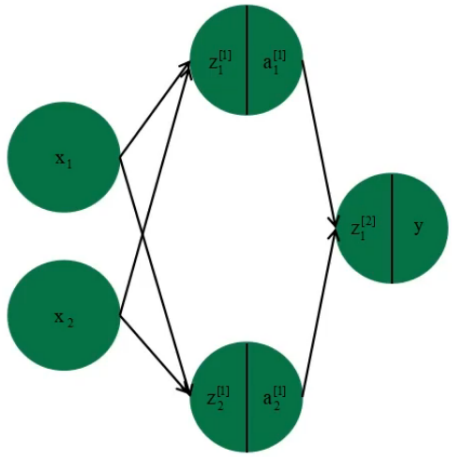
假设f(猫的图片)=猫，计算机只认识数字

把图像数字化，结果分开自定义

将图片拉直成向量，10*10的拉成100列的向量，即输入每个像素点

第一个样本放在第一行，第二个样本放在第二行，以此类推，每一行代表一个样本

样本有多少特征列，就在输入层有多少神经元（比如100列向量对应100个神经元）

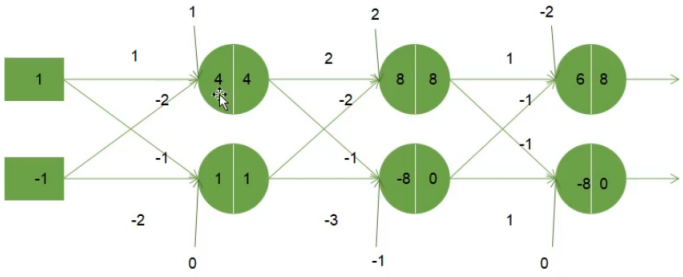


$$x_1 w_{11}^{[1]} + x_2 w_{12}^{[1]} + b_1^{[1]} = z_1^{[1]}$$
$$x_1 w_{21}^{[1]} + x_2 w_{22}^{[1]} + b_2^{[1]} = z_2^{[1]}$$
$$a_1^{[1]} = g(z_1^{[1]})$$
$$a_2^{[1]} = g(z_2^{[1]})$$
$$z_1^{[2]} = a_1^{[1]} w_{11}^{[2]} + a_2^{[1]} w_{12}^{[2]} + b_1^{[2]}$$
$$g(z_1^{[2]}) = y$$

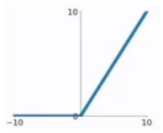
z1是输入*权+偏置和值，经过非线性g得到两个输入层的值

两个输入层的值，再利用同样的方法，在输出层得到最终的y

比如激活函数Relu(x)=max(0,x)



$$\mathrm{Re}lu(x) = \max(0, x)$$



为了简单，需要向量化算式

$$x_1 w_{11}^{[1]} + x_2 w_{12}^{[1]} + b_1^{[1]} = z_1^{[1]}$$
$$x_1 w_{21}^{[1]} + x_2 w_{22}^{[1]} + b_2^{[1]} = z_2^{[1]}$$

$$\downarrow$$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} & b_2^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} & z_2^{[1]} \end{bmatrix}$$

$$\downarrow$$

$$XW^{[1]} + B^{[1]} = Z^{[1]}$$

$$g(z_1^{[1]}) = a_1^{[1]}$$
$$g(z_2^{[1]}) = a_2^{[1]}$$

$$\downarrow$$

$$g(\begin{bmatrix} z_1^{[1]} & z_2^{[1]} \end{bmatrix}) = \begin{bmatrix} a_1^{[1]} & a_2^{[1]} \end{bmatrix}$$

$$\downarrow$$

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

W作为权重矩阵

$$X=\begin{bmatrix} x_1^{(1)}x_2^{(1)}x_3^{(1)} \\ x_1^{(2)}x_2^{(2)}x_3^{(2)} \\ \cdots \\ x_1^{(m)}x_2^{(m)}x_3^{(m)} \end{bmatrix}$$

(m)表示第m个样本
[L]表示第L层神经元

$$W^{[1]}=\begin{bmatrix} w_{11}^{[1]}w_{21}^{[1]}w_{31}^{[1]}w_{41}^{[1]} \\ w_{12}^{[1]}w_{22}^{[1]}w_{32}^{[1]}w_{42}^{[1]} \\ w_{13}^{[1]}w_{22}^{[1]}w_{33}^{[1]}w_{43}^{[1]} \end{bmatrix}$$

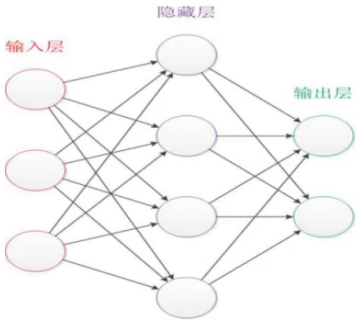
$$B^{[1]}=\begin{bmatrix} b_1^{[1]}b_2^{[1]}b_3^{[1]}b_4^{[1]} \end{bmatrix}$$

$$w_{in}^{[L]}$$

表示第L层第i个神经元与第L-1层第n个神经元的权重

$$w$$

维度
行：上一层神经元个数
列：当层神经元个数



$$Z^{[1]}=XW^{[1]}+B^{[1]}$$
$$A^{[1]}=f(Z^{[1]})$$
$$Z^{[2]}=A^{[1]}W^{[2]}+B^{[2]}$$
$$A^{[2]}=f(Z^{[2]})$$

比如右图中输入层到隐藏层，W是3*4

层数从隐藏层算起，输入层是第0层

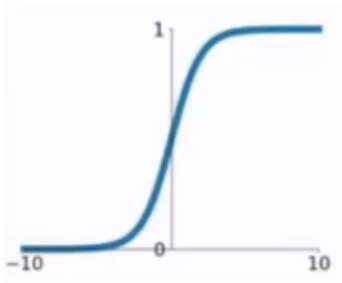
L是从1开始的，因为涉及L-1

激活函数

如果没有**激活函数**，永远都是线性的，学习能力没有加强

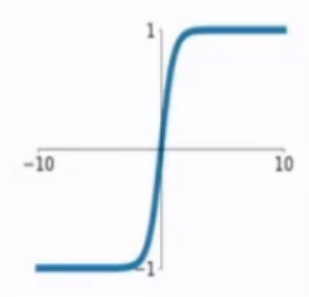
sigmoid函数

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



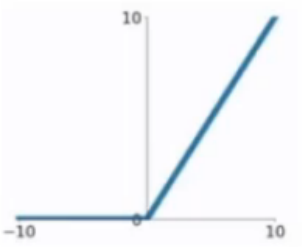
tanh函数

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



Relu函数

$$Relu(x)=\max(0,x)$$



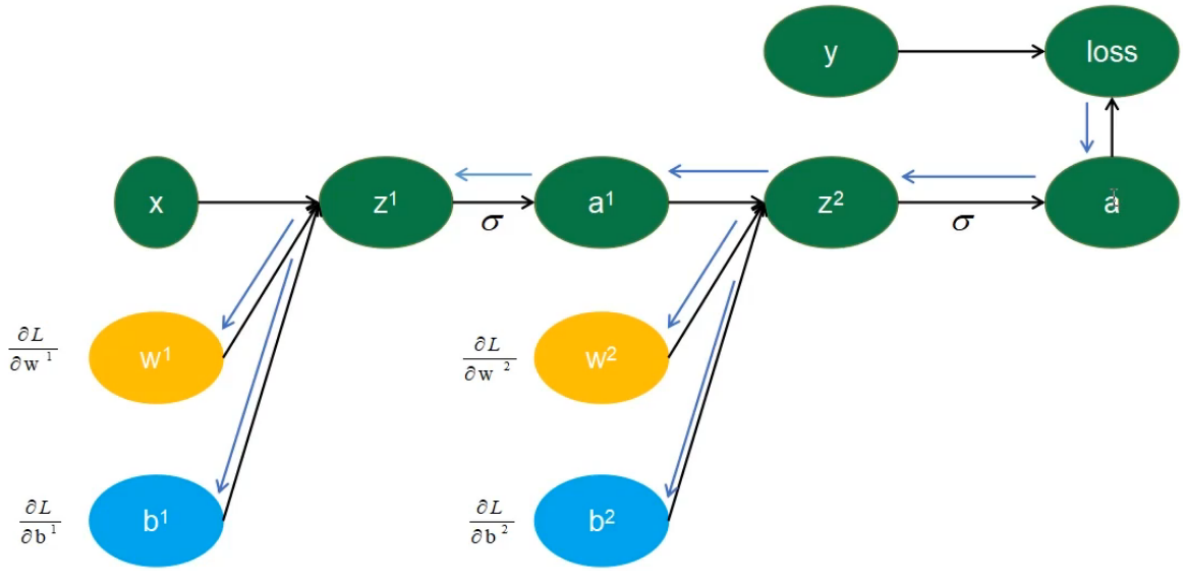
sigmoid：分类，类似概率。输出层二分类

tanh：循环神经网络

以上两种的劣势：趋向无限大时，导数约等于0

Relu：输入层最常用

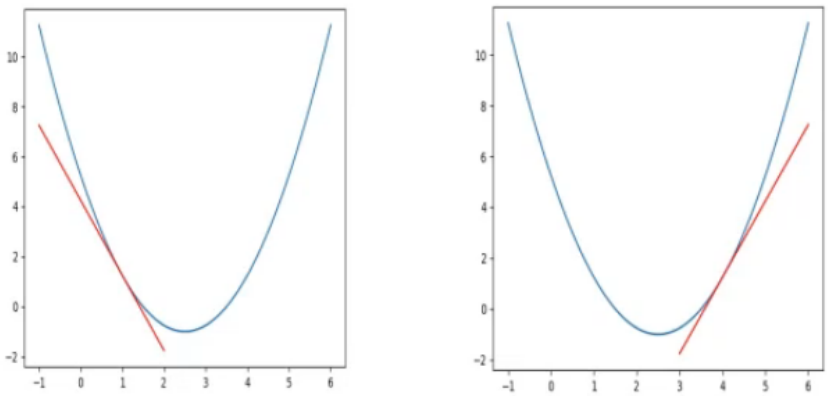
反向传播



求偏导（梯度），利用梯度下降的方法进行梯度的更新

$$\theta_j := \theta_j - \eta \frac{\partial J(\theta_j)}{\partial \theta_j}$$

更新后的权重 = 老权重 - 学习率*权重对应的梯度



θ每一次更新，都朝着J最小的方向

让损失函数不断变小

链式法则

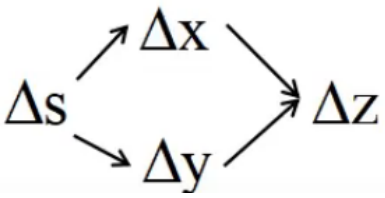
case1

$$y = g(x) \qquad z = h(y)$$

$$\Delta x \to \Delta y \to \Delta z \qquad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

case2

$$x = g(s) \qquad y = h(s) \qquad z = k(x, y)$$



$$\frac{\partial z}{\partial s} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial s}$$

利用链式法则，求偏导

反向传播其实就是利用前向传播得到的预测值，与真实值比较，得到一个损失函数J。求出损失函数对各个参数的梯度，再利用梯度下降法，对所有的参数进行优化更新。

$$dw^{[2]} = \frac{\partial J}{\partial w^{[2]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w^{[2]}}$$

$$dw^{[1]} = \frac{\partial J}{\partial w^{[1]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial w^{[1]}}$$

$$db^{[2]} = \frac{\partial J}{\partial w^{[2]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial b^{[2]}}$$

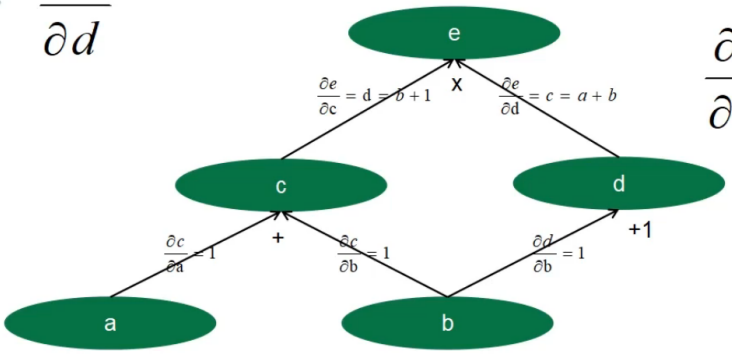
$$db^{[1]} = \frac{\partial J}{\partial w^{[1]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial b^{[1]}}$$

求偏导的例子

已知 $e = (a + b)(b + 1)$

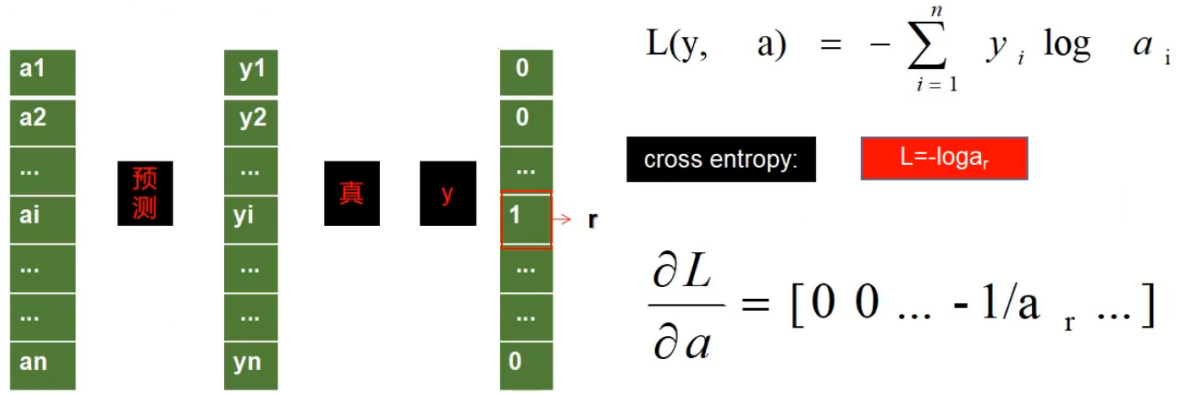
$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c}\frac{\partial c}{\partial b} + \frac{\partial e}{\partial d}\frac{\partial d}{\partial b}$$

求 $\frac{\partial e}{\partial d}$



$$\frac{\partial e}{\partial b} = a + 2b + 1$$

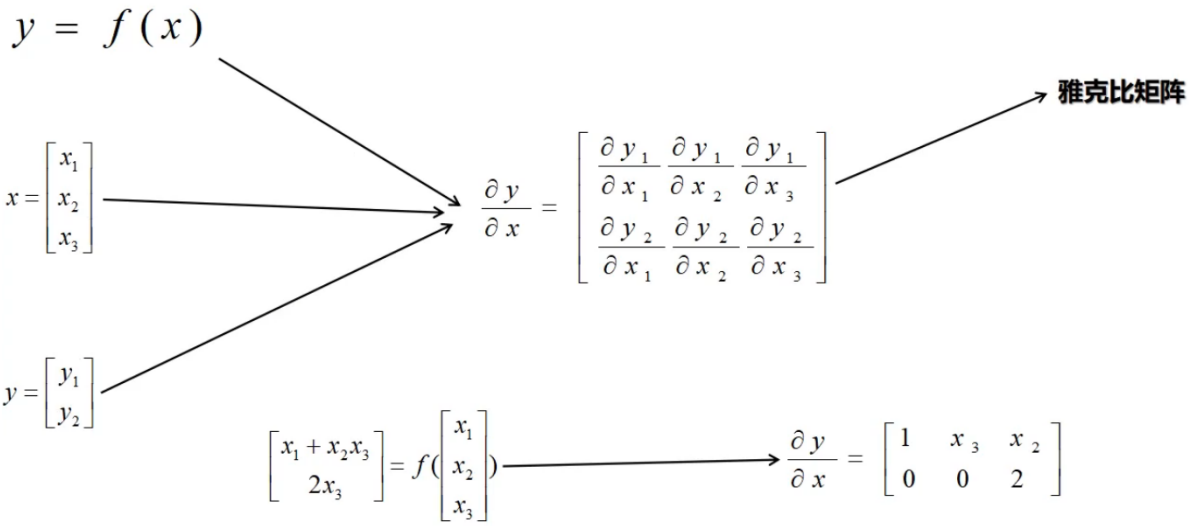
损失函数



交叉熵的损失函数，编程中log底数默认为e

如识别手写数字，每个数字有概率，将最高的非线性变换成0.1

向量求导的方式：



矩阵的求导