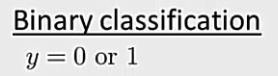
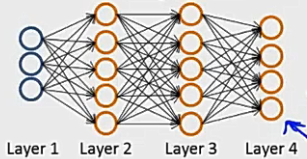
**1、基本知识：**

（1）神经网络的分类：二元分类、多元分类

 Eg：二元分类：

 多元分类： ，每次输出多个值

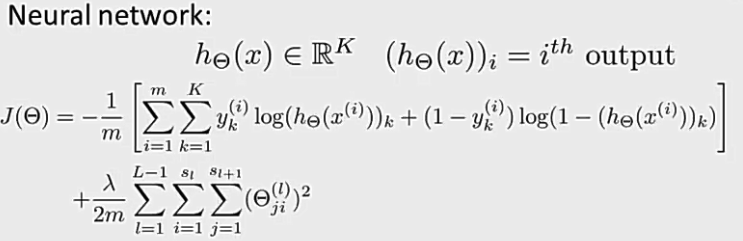
（2）神经网络拓扑图：



L：表示神经网络的层数；

 表示所在层数的神经元数目（不包括：外加的偏置单元（ ））；

eg：，表示第二层有5个神经元。

（3）神经网络的代价函数：

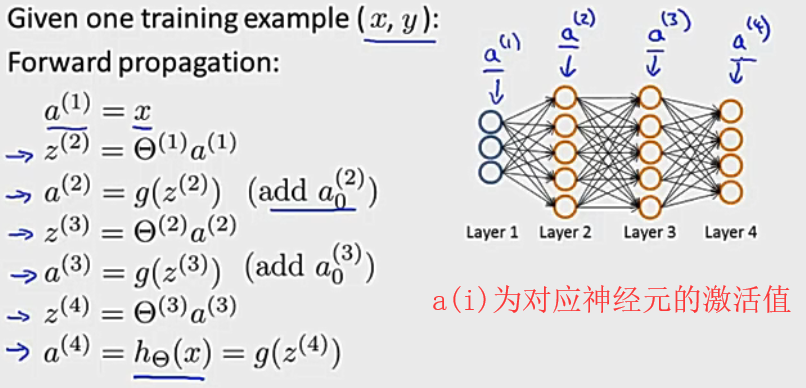
 ：为模型输出，是一个K\*1维的向量（当K=1时，也就是二元分类）；

：模型输出值的第i个元素；

**注意**：此处的正则化，也不计算 。

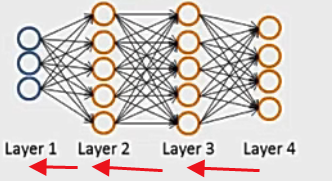
**（4）求解梯度：**

1）正向传播（未添加偏置单元）



正向计算出对应的对应的激活值，得到最终的模型输出值 。

**2）反向传播**



例：以第4层为例子：

定义：第4层的偏差为 ，即实际输出-预测输出

第3-2层都是一样的：





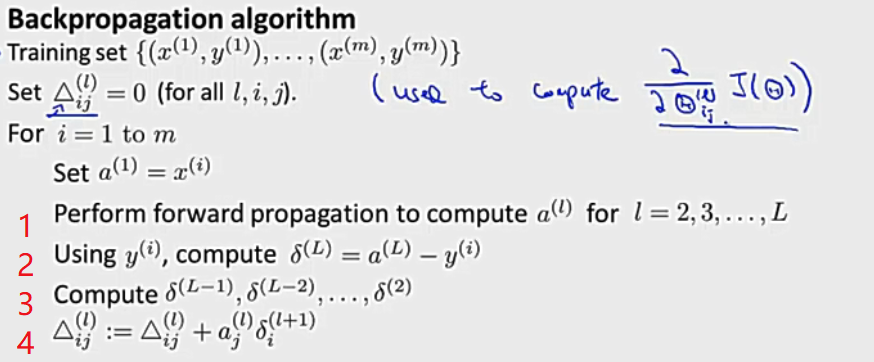
此处，没有 ，由于第一层为输入，不存在偏差。

得到每组输入数据的偏差叠加：

最终：梯度：1、未进行正则化时：

2、正则化后： 

**（整理）反向传播，整体执行步骤：**



（正则化后）梯度： 

此后，求解对应的theta参数，就可以使用梯度下降等方法进行求解。

**（详解）对于上述公式的解释：**

以下的公式，**每输入一组数据计算一次**（使用for实现），最终得到在所有训练数据下的梯度函数。

**必须先计算L-1层与L层的偏差，再从后往前推算各层偏差。**

1.  ：：（L-1层）与（L层：输出层）的偏差

：最后一层隐藏层的输入数据

：模型的输出数据（元素）

示例代码：带有一个隐藏层的神经网络，实现0-9数字分类

% y：模型输出的数组

% err\_3：隐藏层与输出层的偏差（数组）

err\_3 = zeros(10,1);

for k = 1:10

err\_3(k) = a\_3(k) - (y(t) == k);

end

1. ：

计算L-1层的偏差；

=(L-1层的权重)\*(前一层偏差).\*(L-1层激活函数的导数)

：前一层L的偏差（实际编程中，应该去除第一个偏差值，此处不计算偏置单元的偏差）

：L-1层的输入数据

：为L-1层的输入逻辑函数（使用sigmoid）；

（L-1层→2层的偏差 都可以这样求解）

1.  ：

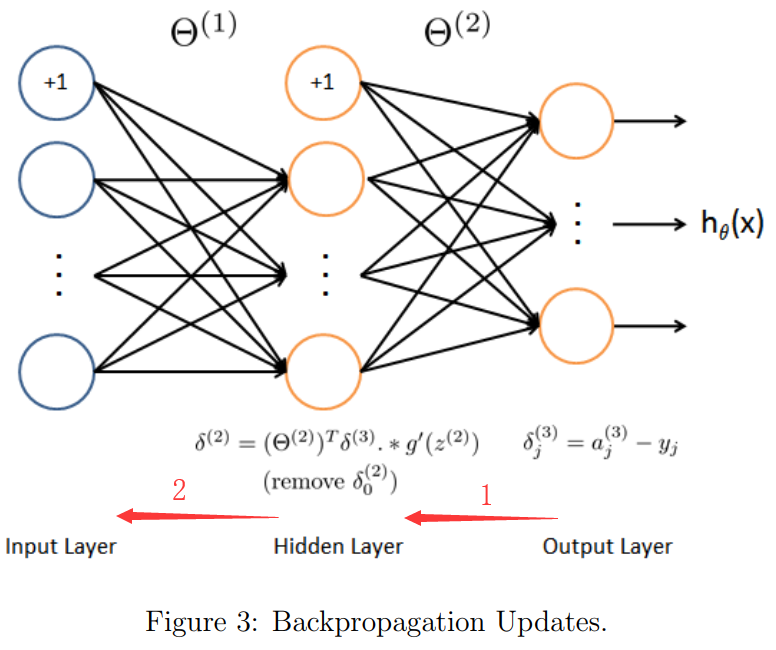
：记录，每组输入数据，对应的L层的偏差（共有L-1个 ）

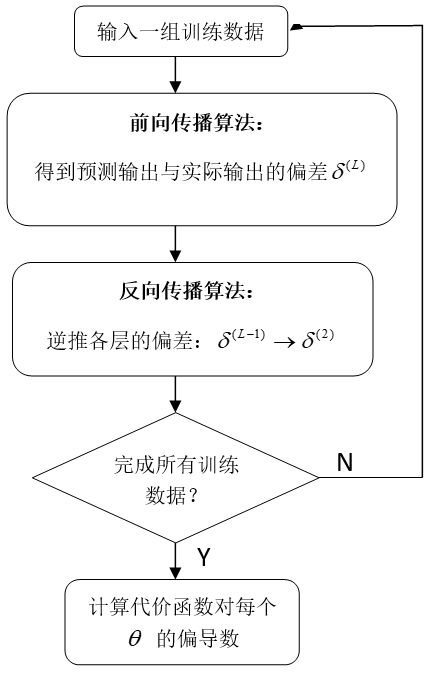
=（L层的输入数据）\*（L+1层的偏差）的叠加

1. 最终计算代价函数对于每个theta的偏导数，使用如下公式：

**（PS：注意，添加正则化的位置）**

附：**添加偏置单元**的反向传播网络



**具体思路如下：**

（5）梯度检测：

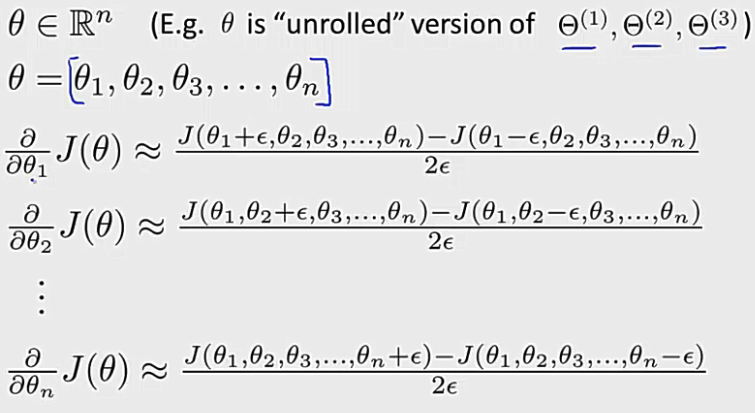
1）单侧差分：

检测： ，一般来说 。

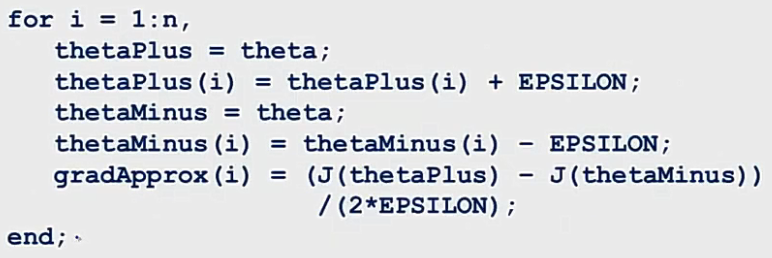
2）双侧差分：（更准确）

检测：，一般来说。

**判断依据：**如果式子左右两侧接近相等，即可说明，梯度计算正确。在之后计算theta时可以放心地使用。

3）实现的方法：

具体代码实现方式如下：



4）使用梯度检测注意事项：

1、确保计算的梯度 双边检测值；

2、在编写代码时，当使用梯度检测确保了程序可以正确计算出梯度时，必须将梯度检测关闭（由于梯度检测需要耗费极大的计算时间）。

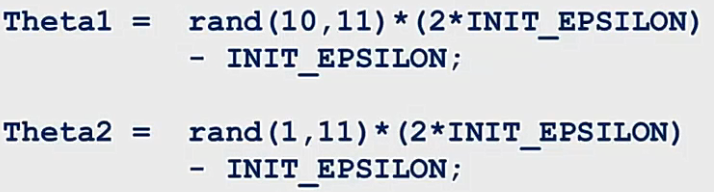
（6）神经网络的theta参数初始化（不能都为0：）

1）若初始化theta为零向量，会导致所有计算的梯度都一样，进而使得反向传播计算得到的结果都一样（theta所有元素相等，但），这就使得所有神经元的连接权重theta都一样，最终导致神经网络无法实现学习功能。

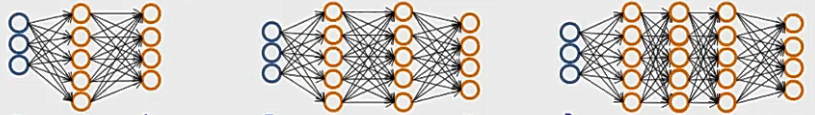
2）因此，在初始化theta时，必须对theta进行随机初始化（ ）。在matlab中可是使用如下代码：

Eg：生成 theta1 = 10\*11矩阵，theta2 = 1\*11矩阵。

INIT\_EPSILON：为自定义的参数，一般为0.10-0.30之间



（7）神经网络结构选择：

1）示例神经网络结构图：

一般，常用的为左边的结构图（1个输入层+1个隐藏层+一个输出层）

2）**要点：**

1、隐藏层的神经元个数应该和特征数量相关（多为倍数关系），隐藏层的神经元个数越多效果越好（但会增加计算量）。

2、隐藏层之间：神经元个数一般是相等的。

（8）神经网络训练步骤：

1）随机初始化theta参数（注意：有添加偏置变量）；

2）正向传播：计算出每一个输入 对应的 ，即：对应的输出值；

3）计算代价函数J ；

4）反向传播：计算对应的梯度

在程序中，可以使用for循环，来计算出对应的梯度；

5）使用梯度检测来判断所求的梯度是否正确

即：是否有；（检测完后，要关闭梯度检测。）

6）使用梯度下降（或更高级的算法）求解theta，得到 。在matlab中可以使用fminunc( )函数求解theta参数。