神经网络的学习(Neural Networks: Learning)

### 9.1 代价函数

参考视频: 9 - 1 - Cost Function (7 min).mkv

首先引入一些便于稍后讨论的新标记方法：

假设神经网络的训练样本有个，每个包含一组输入和一组输出信号，表示神经网络层数，表示每层的**neuron**个数(表示输出层神经元个数)，代表最后一层中处理单元的个数。

将神经网络的分类定义为两种情况：二类分类和多类分类，

二类分类：表示哪一类；

类分类：表示分到第i类；

图片包含 文字

描述已自动生成

我们回顾逻辑回归问题中我们的代价函数为：



在逻辑回归中，我们只有一个输出变量，又称标量（**scalar**），也只有一个因变量，但是在神经网络中，我们可以有很多输出变量，我们的是一个维度为的向量，并且我们训练集中的因变量也是同样维度的一个向量，因此我们的代价函数会比逻辑回归更加复杂一些，为： ，



这个看起来复杂很多的代价函数背后的思想还是一样的，我们希望通过代价函数来观察算法预测的结果与真实情况的误差有多大，唯一不同的是，对于每一行特征，我们都会给出个预测，基本上我们可以利用循环，对每一行特征都预测个不同结果，然后在利用循环在个预测中选择可能性最高的一个，将其与中的实际数据进行比较。

正则化的那一项只是排除了每一层后，每一层的 矩阵的和。最里层的循环循环所有的行（由 +1 层的激活单元数决定），循环则循环所有的列，由该层（层）的激活单元数所决定。即：与真实值之间的距离为每个样本-每个类输出的加和，对参数进行**regularization**的**bias**项处理所有参数的平方和。

图片包含 文字

描述已自动生成

### 9.2 反向传播算法

参考视频: 9 - 2 - Backpropagation Algorithm (12 min).mkv

**计算神经网络预测结果**的时候我们采用了一种**正向传播方法**，我们从第一层开始正向一层一层进行计算，直到最后一层的。

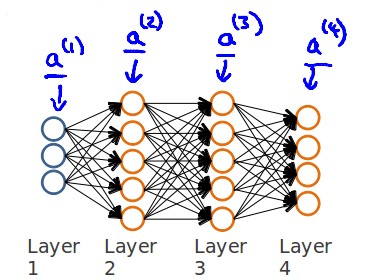
现在，为了计算代价函数的偏导数，我们需要采用一种反向传播算法，也就是首先计算最后一层的误差，然后再一层一层反向求出各层的误差，直到倒数第二层。 以一个例子来说明反向传播算法。

假设我们的训练集只有一个实例，我们的神经网络是一个四层的神经网络，其中：

前向传播算法：

图片包含 文字

描述已自动生成



**我们从最后一层的误差开始计算**，误差是激活单元的预测（）与实际值（）之间的误差，（）。

我们用来表示误差，则：

我们利用这个误差值来计算前一层的误差： 其中 是 形函数的导数，。而则是权重导致的误差的和。下一步是继续计算第二层的误差：

因为第一层是输入变量，不存在误差。我们有了所有的误差的表达式后，便可以计算代价函数的偏导数了，假设，即我们不做任何正则化处理时有：

重要的是清楚地知道上面式子中上下标的含义：

代表目前所计算的是第几层。

代表目前计算层中的激活单元的下标，也将是下一层的第个输入变量的下标。

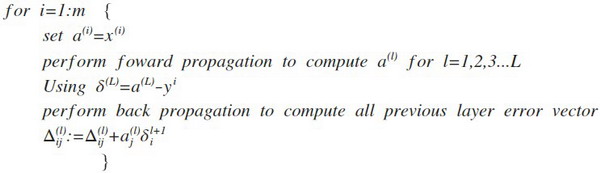
代表下一层中误差单元的下标，是受到权重矩阵中第行影响的下一层中的误差单元的下标。

图片包含 文字

描述已自动生成

如果我们考虑正则化处理，并且我们的训练集是一个特征矩阵而非向量。在上面的特殊情况中，我们需要计算每一层的误差单元来计算代价函数的偏导数。在更为一般的情况中，我们同样需要计算每一层的误差单元，但是我们需要为整个训练集计算误差单元，此时的误差单元也是一个矩阵，我们用来表示这个误差矩阵。第 层的第 个激活单元受到第 个参数影响而导致的误差。

我们的算法表示为：



即首先用正向传播方法计算出每一层的激活单元，利用训练集的结果与神经网络预测的结果求出最后一层的误差，然后利用该误差运用反向传播法计算出直至第二层的所有误差。

在求出了之后，我们便可以计算代价函数的偏导数了，计算方法如下：

在**Octave** 中，如果我们要使用 fminuc这样的优化算法来求解求出权重矩阵，我们需要将矩阵首先展开成为向量，在利用算法求出最优解后再重新转换回矩阵。

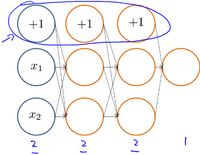
假设我们有三个权重矩阵，Theta1，Theta2 和 Theta3，尺寸分别为 10\*11*，*10\*11 和1\*11， 下面的代码可以实现这样的转换：

thetaVec = [Theta1(:) ; Theta2(:) ; Theta3(:)]  
...optimization using functions like fminuc...  
Theta1 = reshape(thetaVec(1:110, 10, 11);  
Theta2 = reshape(thetaVec(111:220, 10, 11);  
Theta1 = reshape(thetaVec(221:231, 1, 11);

### 9.3 反向传播算法的直观理解

参考视频: 9 - 3 - Backpropagation Intuition (13 min).mkv

前向传播算法：



图片包含 文字

描述已自动生成

反向传播算法做的是：

图片包含 文字

描述已自动生成

图片包含 文字

描述已自动生成