Machine Learning - Week 5

赵燕

目录

1 Cost Function and Backpropagation

1.1 Cost Function

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m个，每个包含一组输入x和输出信号y，L表示神经网络层数，s_l表示每层的neuron的个数，s_L表示最后一层中处理单元的个数。

Let's first define a few variables that we will need to use:

- L = total number of layers in the network
- s_l = number of units (not counting bias unit) in layer l
- K = number of output units/classes

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（Binary classification）:

- $s_L = 1$，y= 0 or 1 ,表示哪一类;
- 1 output unit

K类分类（Multi-class classification（K classes））:

- $s_L = K$,$y_i = 1$，表示分到第i类（$K \geqslant 3$）;
- K output units

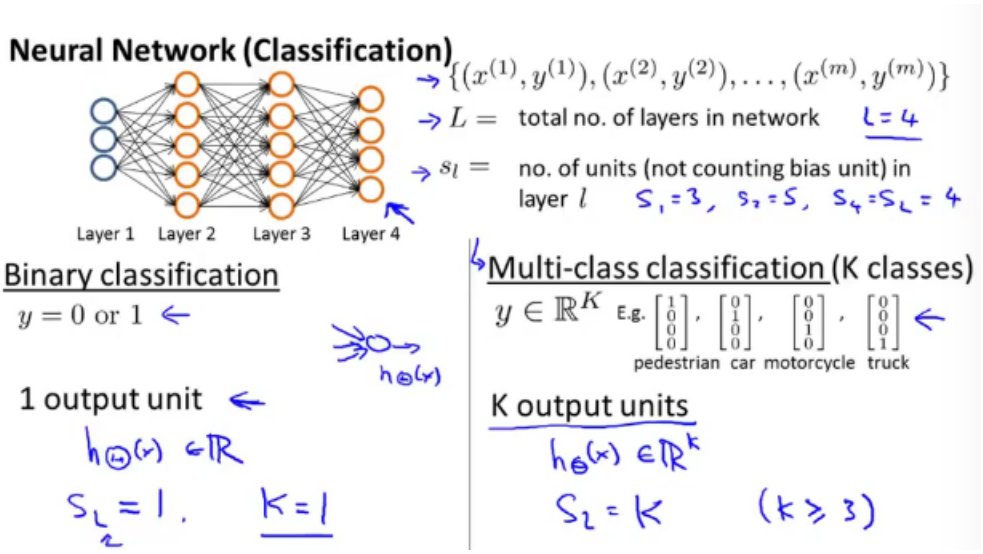

图 1: 二类分类和K类分类

Recall that in neural networks, we may have many output nodes. We denote $h_\Theta(x)_k$ as being a hypothesis that results in the kth output. Our cost function for neural networks is going to be a generalization of the one we used for logistic regression.

Recall that the cost function for regularized logistic regression was:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})) + (1-y^{(i)})log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{1}$$

在逻辑回归中，只有一个输出变量，又称标量（scalar），也只有一个因变量y，但是在神经网络中，可以有很多输出变量，$h_\Theta(x)$是一个K维度的向量，并且训练集中的因变量也是同样维度的一个向量，因此神经网络的代价函数会比逻辑回归更加复杂一些。

For Neural Networks:

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}[y_k^{(i)}log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})log(1-(h_\Theta(x^{(i)}))_k)] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(\Theta_{j,i}^{(l)})^2 \tag{2}$$

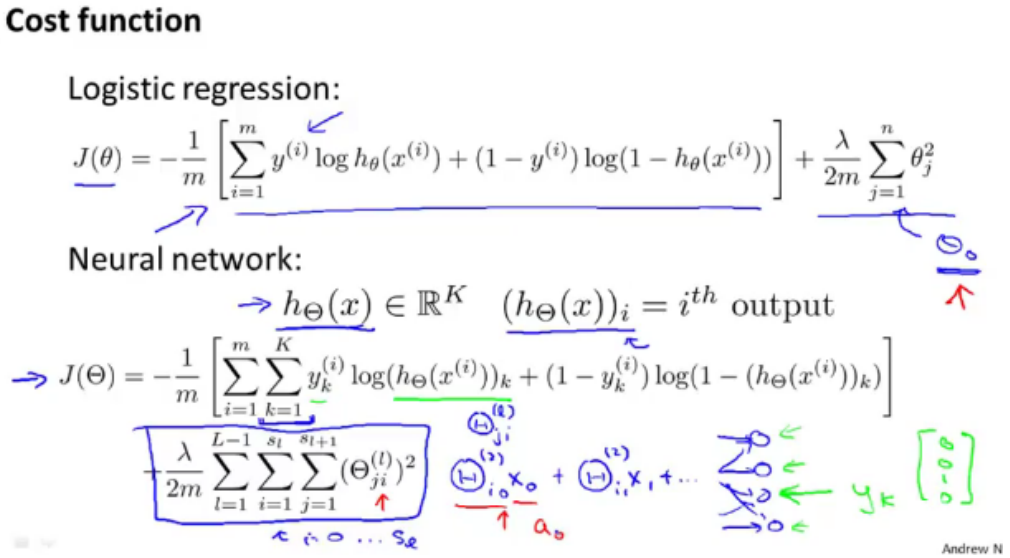

图 2: 逻辑回归和神经网络的代价函数

这个看起来复杂很多的代价函数背后的思想还是一样的，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K个预测，基本上可以利用循环，对每一行特征都预测K个不同的结果，然后利用循环在K个预测中选择可能性最高的一个，将其与y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的Θ_0后，每一层的Θ矩阵的和，最里层的循环j循环所有的行（由s_l+1层的激活单元数决定），循环i则循环所有的列，由该层（s_l层）的激活单元数所决定。即：$h_\Theta(x)$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。

We have added a few nested summations to account for our multiple output nodes. In the first part of the equation, before the square brackets, we have an additional nested summation that loops through the number of output nodes.

In the regularization part, after the square brackets, we must account for multiple theta matrices. The number of columns in our current theta matrix is equal to the number of nodes in our current layer (including the bias unit). The number of rows in our current theta matrix is equal to the number of nodes in the next layer (excluding the bias unit). As before with logistic regression, we square every term.

Note:

- the double sum simply adds up the logistic regression costs calculated for each cell in the output layer
- the triple sum simply adds up the squares of all the individual Θs in the entire network.
- the i in the triple sum does not refer to training example i

1.2 Backpropagation Algorithm

1.3 Backpropagation Intuition

2 Backpropagation in Practice

2.1 Implementation Note:Unrolling Parameters

2.2 Gradient Checking

2.3 Random Initialization

2.4 Putting it Together

3 Application of Neural Networks

3.1 Autonomous Driving