


17.4.25(神经网络的使用)

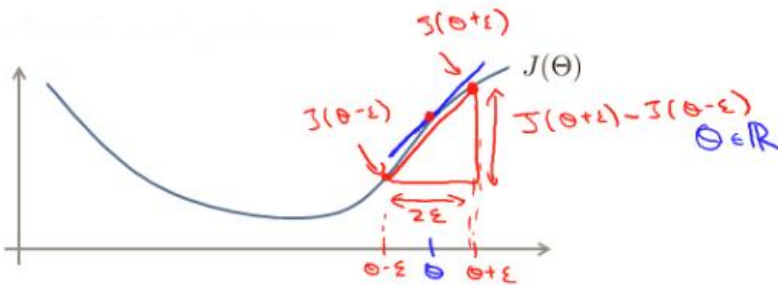
神经网络在使用最优化高级算法进行优化时,要把所有的theta矩阵转化成一列的向量。在使用完高级算法以后,还要把theta转化回来。具体方法如下:

Example

$s_1 = 10, s_2 = 10, s_3 = 1$
 $\rightarrow \Theta^{(1)} \in \mathbb{R}^{10 \times 11}, \Theta^{(2)} \in \mathbb{R}^{10 \times 11}, \Theta^{(3)} \in \mathbb{R}^{1 \times 11}$
 $\rightarrow D^{(1)} \in \mathbb{R}^{10 \times 11}, D^{(2)} \in \mathbb{R}^{10 \times 11}, D^{(3)} \in \mathbb{R}^{1 \times 11}$
 $\rightarrow \text{thetaVec} = [\text{Theta1}(:); \text{Theta2}(:); \text{Theta3}(:)];$
 $\rightarrow \text{DVec} = [D1(:); D2(:); D3(:)];$
 $\text{Theta1} = \text{reshape}(\text{thetaVec}(1:110), 10, 11);$
 $\text{Theta2} = \text{reshape}(\text{thetaVec}(111:220), 10, 11);$
 $\text{Theta3} = \text{reshape}(\text{thetaVec}(221:231), 1, 11);$



当我们训练出来一个复杂的神经网络模型以后,可能会存在一些无法察觉的错误,这时候我们就可以用数值计算来进行检测。这种方法是我们在代价函数上找几个点,求出他们的代价,然后与算法求出来的进行比较。具体方法是:对于某个特定的 θ ,我们计算出在 $\theta - \epsilon$ 处和 $\theta + \epsilon$ 的代价值(ϵ 是一个非常小的值,通常选取0.001),然后求两个代价的平均,用以估计在 θ 处的代价值。



Suppose you have a function $f_i(\theta)$ that purportedly computes $\frac{\partial}{\partial \theta_i} J(\theta)$; you'd like to check if f_i is outputting correct derivative values.

$$\text{Let } \theta^{(i+)} = \theta + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \epsilon \\ \vdots \\ 0 \end{bmatrix} \quad \text{and} \quad \theta^{(i-)} = \theta - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \epsilon \\ \vdots \\ 0 \end{bmatrix}$$

So, $\theta^{(i+)}$ is the same as θ , except its i -th element has been incremented by ϵ . Similarly, $\theta^{(i-)}$ is the corresponding vector with the i -th element decreased by ϵ . You can now numerically verify $f_i(\theta)$'s correctness by checking, for each i , that:

$$f_i(\theta) \approx \frac{J(\theta^{(i+)}) - J(\theta^{(i-)})}{2\epsilon}.$$

参数初始化, 任何一个优化算法都需要初始化参数,我们之前的初始参数都是0,在线性,逻辑,回归中是可行的,但在神经网络中,初始化参数都为零是不也可行的。这会让你第二层求出来的激活值都一样,这种情况是错误的,同理任何一个相同的参数都不行。所以我们要初始化随机参数。

综合神经网络的用法, 在使用神经网络的时候,我们首先要做的是决定我们的神经网络算法的网格结构,就是决定我们要有多少层,每层有多少个神经元。首先输入层的神经元个数就是我们训练集样本的特征数,输出层的神经元个数就是我们要分的类数。所以说我们要决定的就是隐藏层的层数,和每层的神经元数,理论上来说:每层的神经元个数多点好,也就是每层的激活函数多,可能越复杂的隐藏层预测的结果就越好。还有一点是:隐藏层数大于1时,每层的神经元个数要保持一致。下面的几步分别是: 1.初始化参数

2.利用正向传播算法,算出所有的假设函数 $h(\theta)$

3.利用公式求cost function

4.用反向传播算法求每个代价函数 $J(\theta)$ 的偏导数

5.用数值计算求偏导数和反向传播求出来的进行比较

6.把求出来的cost function 和 grad 代入高级优化算法，来最小化代价cost.

.