Coursera Machine Learning Week5

Coursera Machine Learning Lunar's note

`MachineLearning` `Coursera`

神经网络代价函数 cost function

1. 二元分类

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

2. 反向传播算法 Backpropagation

我们利用反向传播算法来计算代价函数的偏导数$\frac{\partial}{\partial\Theta_{ji}^l}J(\Theta)$
对于第l层的第j个节点，我们可以用如下方以计算出它的偏差(error)
+ 对于输出层 $\delta^{(l)} = a_j^{(l)} - y_j$
+ 对于隐藏层 $\delta^{(l)} = (\Theta^{(l)})^T\delta^{(l+1)}.*g'(z^{(l)})$
.*表示矩阵点乘，g'表示g的导数
+ 对于输入层，因为输入不存在误差，因此没有$\delta^{(1)}$
反向传播的意思是，我们最先计算出的是最后一层的误差，然后反向向前传播。
具体算法：

> Traning set {(x1,y1)...(xm,ym)}
> Set $\Delta_{ij}^l = 0(for \quad all\ l,i,j)$
> for i=1 to m
> 、、Set $a^1 = x^i$
> 、、顺序计算出 $a^2 \ldots a^l$
> 、、利用y^i,反向传播计算出各个δ
> 、、$\Delta_{ij}^l := \frac{1}{m}\Delta_{ij}^l + a_j^l\delta_i^{l+1}$
> $D_{ij}^l := \frac{1}{m}\Delta_i^lj + \lambda\Theta_i^jj \qquad if\ j \neq 0$
> $D_{ij}^l := \frac{1}{m}\Delta_i^lj \qquad\qquad if\ j = 0$

那么我们可以得到

$$\frac{\partial}{\partial \Theta_{ji}^{l}} J(\Theta) = D_{ij}^{l}$$

反向传播算法实现中的细节

- 展开参数 Unrolling parameter
- 梯度检验 Gradient Checking
  在用反向传播实现梯度下降时可能会出现一些不易察觉的错误，即使代价函数每次都在下降也不代表答案的正确，我们可以用梯度检验来检查是否正确。
  我们知道
  $\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon)-J(\theta-\varepsilon)}{2\varepsilon}$
  我们可以用这种近似的导数解来检验反向传播得到的DVec是否可以判定为正确。
- 初始化θ
  神经网络中如果采用全零初始会导致每层的神经元雷同(对称性 symmetry)，所以应该采用随机初始化。

3. 神经网络架构选择

- 对于输入层，神经元个数等于特征向量长度。
- 对于输出层，神经元个数等于分类数量。
- 对于中间层，可以选择只有一层中间层，当选择多层中间层时，通常来说每层的神经单元个数一致。理论上，每层中间层神经元个数越多越好，但是太多的神经元会导致算法变慢。