Machine Learning Week-5

ramay7

March 1, 2017

Contents

1 What is BP(Back Propagation) Algorithm ?

Back Propagation is used to compute the partial derivation of $J(\theta)$:$\frac{\partial}{\partial\theta_{i,j}^{l}}J(\theta)$.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\theta_{j,i}^{l})^2$$

The main reason that we use the back propagation algorithm rather than the numerical gradient computation method during learning is that the latter is very slow.

The gradient for the sigmoid function can be computed as:

$$g'(z) = \frac{d}{dz}g(z) = g(z)(1-g(z))$$

where sigmoid(z) = g(z) = $\frac{1}{1+e^{-z}}$.

2 Random Initialization: Symmetry breaking

Initialize each $\theta_{i,j}^{(l)}$ to a random value in $[-\varepsilon, \varepsilon]$ (i.e. $-\varepsilon \leq \theta_{i,j}^{(l)} \leq \varepsilon$).

E.g. Theta1 = rand(10, 11) * (2 * INIT_EPSILON) - INIT_EPSILON.

When training neural networks, it is important to randomly initialize the parameters for symmetry breaking. One effective strategy for random initialization is to randomly select values for $\theta^{(l)}$ uniformly in the range $[-\varepsilon_{init}, \varepsilon_{init}]$. One effective strategy for choosing ε_{init} is to base it on the number of units in the network. A good choice of ε_{init} is $\varepsilon_{init} = \frac{\sqrt{6}}{\sqrt{L_{in}+L_{out}}}$, where $L_{in} = s_l$ and $L_{out} = s_{l+1}$ are the number of units in the layer adjacent to θ^{l}.

3 Concrete Steps

- set $\Delta_{i,j}^{(l)} := 0$ for all (l, i, j)(hence you end up having a matrix full of zero)
- for i = 1 to m
  1. Set $a^{(1)} = x^{(i)}$
  2. Perfom forward propagation to compute $a^{(l)}$ for $l = 2, 3, ..., L$
  $$a^{(l)} = g(z^{(l)}) = g(\theta^{(l-1)} a^{(l-1)}))$$
  3. Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
  4. Computing $\delta^{(L-1)}, \delta^{(L-2)}, ..., \delta^{(2)}$ using $\delta^{(l)} = (\theta^{(l)})^T \delta^{l+1} . * a^{(l)} . * (1 - a^{(l)})$
  5. $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_j^{(l+1)}$ or with vectorization, $\Delta^l := \Delta^l + \delta^{l+1} (a^l)^T$
- Hence we update our new Δ matrix:

$$D_{i,j}^{(i)} := \begin{cases} \frac{1}{m} \Delta_{i,j}^{(l)} & \text{j} = 0 \\ \frac{1}{m} (\Delta_{i,j}^{(l)} + \lambda \theta_{i,j}^{(l)}) & \text{j} \neq 0 \end{cases}$$

The captial-delta matrix D is used as an "accumulator" to add up our values as we go along and eventually compute our partial derivative. Thus, we get:

$$\frac{\partial}{\partial \theta_{i,j}^{(l)}} J(\theta) = D_{i,j}^{(l)}$$

4 The End

To be honest, BP algorithm is a little bit hard to understand, and a little difference may cause a compelety wrong answer while doing exercise. In addition, I have written some proof about BP Algorithm. This may be helpful to understand the algorithm.