Neural Network Back Propagation 公式推导

来源：http://blog.csdn.net/firethelife/article/details/51326931

2017年3月19日

1 Notions

n_l: neural network layer number
s_l: 第l层神经元个数
$z_i^{(l)}$: 第l层第i个神经元的输入
$a_i^{(l)}$: 第l层第i个神经元的输出
$W_{ij}^{(l)}$: 第l层第i个神经元到第$l+1$层第j个神经元的权重，维度：$s_{l+1}\times s_l$

2 feedforward

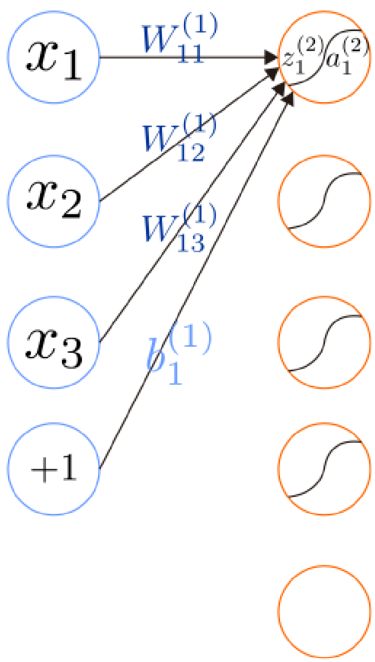

图 1: 神经元图

如图 1 所示，第二层各神经元的计算方法如下：

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
a_4^{(2)} &= f(W_{41}^{(1)}x_1 + W_{42}^{(1)}x_2 + W_{43}^{(1)}x_3 + b_4^{(1)})
\end{aligned}
$$

其中，$f()$为activation function. 向量表示为：

$$
\begin{aligned}
z^{(2)} &= W^{(1)}x + b \\
a^{(2)} &= f(z^{(2)})
\end{aligned}
$$

矩阵W为：

$$
W_{4\times 3} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix}
$$

3 损失函数 cost function

3.1 cost function

1个样本的损失函数为：

$J(W,b;x,y) = \frac{1}{2}||h_{W,b}(x) - y||^2$

K个样本的损失函数为(添加了正则项)：

$$
\begin{aligned}
J(W,b) &= \left[\frac{1}{K}\sum_{k=1}^{K} J(W,b;x^{(k)},y^{(k)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_{l+1}}\sum_{j=1}^{s_l}(W_{ij}^{(l)})^2 \\
&= \left[\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{2}||h_{W,b}(x^{(k)}) - y^{(k)}||^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_{l+1}}\sum_{j=1}^{s_l}(W_{ij}^{(l)})^2
\end{aligned}
$$

3.2 update function

使用批量梯度下降算法寻求神经网络的最优参数 $W^{(l)}, b^{(l)}$ 我们先来看对于第 l+1 层第 i 个神经元来说，第 l 层第 j 个神经元的权值可按如下方

式迭代更新：

$$\begin{aligned}
W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) \\
&= W_{ij}^{(l)} - \alpha \left[\left(\frac{1}{K} \sum_{k=1}^{K} \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(k)},y^{(k)}) \right) + \lambda W_{ij}^{(l)} \right]
\end{aligned}$$

类似的，对于第 l+1 层第 i 个神经元来说，第 l 层的偏置单元的权值可按如下方式迭代更新：

$$\begin{aligned}
b_{i}^{(l)} &= b_{i}^{(l)} - \alpha \frac{\partial}{\partial b_{i}^{(l)}} J(W,b) \\
&= b_{i}^{(l)} - \alpha \left[\frac{1}{K} \sum_{k=1}^{K} \frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x^{(k)},y^{(k)}) \right]
\end{aligned}$$

我们现在的目的是求出以下两个式子就可以对参数进行迭代了：

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(k)},y^{(k)})$$
$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x^{(k)},y^{(k)})$$

又我们知道第 $l+1$ 层第 i 个神经元的输入 $z_i^{(l+1)}$ 可以由以下式子计算：

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$$

再进一步的对上面的式子进行变形：

$$\begin{aligned}
&\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(k)},y^{(k)}) \\
&= \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}} \\
&= \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}} a_j^{(l)}
\end{aligned}$$

同样的，对于 $b_i^{(l)}$的偏导数：

$$\begin{aligned}&\frac{\partial}{\partial b_i^{(l)}}J(W,b;x^{(k)},y^{(k)})\\&=\frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}}\frac{\partial z_i^{(l+1)}}{\partial b_i^{(l)}}\\&=\frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}}\end{aligned}$$

4 残差的定义

接下来我们定义：

$$\delta_i^{(l)}=\frac{\partial}{\partial z_i^{(l)}}J(W,b;x^{(k)},y^{(k)})$$

对于$W_{ij}^{(l)}$我们有：

$$\begin{aligned}W_{ij}^{(l)}&=W_{ij}^{(l)}-\alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b)\\&=W_{ij}^{(l)}-\alpha\left[\left(\frac{1}{K}\sum_{k=1}^{K}\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b;x^{(k)},y^{(k)})\right)+\lambda W_{ij}^{(l)}\right]\\&=W_{ij}^{(l)}-\alpha\left[\left(\frac{1}{K}\sum_{k=1}^{K}\frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}}a_j^{(l)}\right)+\lambda W_{ij}^{(l)}\right]\\&=W_{ij}^{(l)}-\alpha\left[\left(\frac{1}{K}\sum_{k=1}^{K}\delta_i^{(l+1)}a_j^{(l)}\right)+\lambda W_{ij}^{(l)}\right]\end{aligned}$$

类似的，对于 $b_i^{(l)}$ 我们有：

$$\begin{aligned}b_i^{(l)}&=b_i^{(l)}-\alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b)\\&=b_i^{(l)}-\alpha\left[\frac{1}{K}\sum_{k=1}^{K}\frac{\partial}{\partial b_i^{(l)}}J(W,b;x^{(k)},y^{(k)})\right]\\&=b_i^{(l)}-\alpha\frac{1}{K}\sum_{k=1}^{K}\frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}}\\&=b_i^{(l)}-\alpha\frac{1}{K}\sum_{k=1}^{K}\delta_i^{(l+1)}\end{aligned}$$

现在的核心问题只剩下一个了，这个残差该如何求？
我们先计算最后一层第 i 个神经元上的残差，这里为了简单起见，不再指定为第 k 个样本。

$$\begin{aligned}
\delta_i^{(n_l)} &= \frac{\partial}{\partial z_i^{(n_l)}} J(W,b;x,y) \\
&= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} ||h_{W,b}(x) - y||^2 \\
&= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{s_{n_l}} (y_i - a_j^{(n_l)})^2 \\
&= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{s_{n_l}} (y_i - f(z_j^{(n_l)}))^2 \\
&= -(y_i - f(z_i^{(n_l)})) f'(z_i^{(n_l)})
\end{aligned}$$

然后计算倒数第二层即第 $n_l - 1$ 层第 i 个神经元的残差：

$$\begin{aligned}
\delta_i^{(n_l-1)} &= \frac{\partial}{\partial z_i^{(n_l-1)}} J(W,b;x,y) \\
&= \frac{\partial}{\partial z_i^{(n_l-1)}} \frac{1}{2} \sum_{j=1}^{s_{n_l}} (y_i - a_j^{(n_l)})^2 \\
&= \frac{1}{2} \sum_{j=1}^{s_{n_l}} \frac{\partial}{\partial z_i^{(n_l-1)}} (y_i - f(z_j^{(n_l)}))^2 \\
&= \sum_{j=1}^{s_{n_l}} -(y_j - f(z_j^{(n_l)})) \frac{\partial}{\partial z_i^{(n_l-1)}} f(z_j^{(n_l)}) \\
&= \sum_{j=1}^{s_{n_l}} -(y_j - f(z_j^{(n_l)})) f'(z_j^{n_l}) \frac{\partial z_j^{(n_l)}}{\partial z_i^{(n_l-1)}} \\
&= \sum_{j=1}^{s_{n_l}} \delta_j^{(n_l)} \frac{\partial}{\partial z_i^{(n_l-1)}} \sum_{q=1}^{s_{(n_l-1)}} W_{jq}^{(n_l-1)} f(z_q^{(n_l-1)}) \\
&= \sum_{j=1}^{s_{n_l}} W_{ji}^{(n_l-1)} \delta_j^{(n_l)} f'(z_i^{(n_l-1)})
\end{aligned}$$

上式中，

$$
\begin{aligned}
z_j^{(n_l)} &= \sum_{q=1}^{s_{(n_l-1)}} W_{jq}^{(n_l-1)} a_q^{n_l-1} + b_j^{(n_l-1)} \\
&= \sum_{q=1}^{s_{(n_l-1)}} W_{jq}^{(n_l-1)} f(z_q^{(n_l-1)}) + b_j^{(n_l-1)}
\end{aligned}
$$

所以

$$
\begin{aligned}
\frac{\partial z_j^{(n_l)}}{\partial z_i^{(n_l-1)}} &= \frac{\partial}{\partial z_i^{(n_l-1)}} \left[\sum_{q=1}^{s_{(n_l-1)}} W_{jq}^{(n_l-1)} f(z_q^{(n_l-1)}) + b_j^{(n_l-1)} \right] \\
&= \frac{\partial}{\partial z_i^{(n_l-1)}} \left[W_{ji}^{(n_l-1)} f(z_i^{(n_l-1)}) + b_j^{(n_l-1)} \right] \\
&= W_{ji}^{(n_l-1)} f'(z_i^{(n_l-1)})
\end{aligned}
$$

更一般的，可以将上述关系表述为：

$$
\delta_i^{(l)} = \sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} f'(z_i^{(l)})
$$

再次回顾我们的参数更新公式：

$$
\begin{aligned}
W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha \left[\left(\frac{1}{K} \sum_{k=1}^{K} \delta_i^{(l+1)} a_j^{(l)} \right) + \lambda W_{ij}^{(l)} \right] \\
b_i^{(l)} &= b_i^{(l)} - \alpha \frac{1}{K} \sum_{k=1}^{K} \delta_i^{(l+1)}
\end{aligned}
$$

我们需要先计算输出层神经元的残差，然后一级一级的计算前一层的神经元的残差，利用这些残差就可以更新神经网络参数了。

5 向量化表示

这里我们尝试将上述结果表示成向量或矩阵的形式，比如我们希望能一次性更新某一层神经元的权值和偏置，而不是一个一个的更新。

输出层的残差： $\delta^{n_l} = (a^{n_l} - y) \bigodot f'(z^{(n_l)})$

$\delta_i^{(l+1)}$表示的是第 $l+1$ 层第 i 个神经元的残差，那么整个第 $l+1$ 层神经元的偏差是多少呢？

$$
\delta_i^{(l+1)} = \sum_{j=1}^{s_{l+2}} W_{ji}^{(l+1)} \delta_j^{(l+2)} f'(z_i^{(l+1)})
$$

从而得到：

$$\delta^{l+1} = (W^{(l+1)})^T \delta^{(l+2)} \bigodot f'(z_i^{(l+1)})$$

注：这里的 • 是指点乘，即对应元素相乘，$\delta^{(l+1)}$ 是一个 $s_{l+1} \times 1$ 维的列向量。 $a_j^{(l)}$ 表示第 l 层第 j 个神经元的输出，因此整个第 l 层的神经元的输出可用 $a^{(l)}$ 表示，是一个 $s_l \times 1$ 维的列向量。

6 梯度下降法

因此对于矩阵 $W^{(l)}$ 来说，我们记：

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T$$

我们将 $\triangle W^{(l)}$ 初始化为 0，然后对所有 K 个样本将它们的 $\nabla_{W^{(l)}} J(W, b; x, y)$ 累加到 $\triangle W^{(l)}$ 中去：

$$\triangle W^{(l)} := \triangle W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$$

然后更新一次 $W^{(l)}$：

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{K} \triangle W^{(l)} \right) + \lambda W^{(l)} \right]$$

这里再强调一下：上式中的 $\triangle W^{(l)}$ 是所有 K 个样本的 $\delta^{(l+1)} (a^{(l)})^T$ 累加和，如果希望做随机梯度下降或者是mini-batch，这里就不用把所有样本的残差加起来了。类似的，令：

$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)}$$

我们将 $\triangle b^{(l)}$ 初始化为 0，然后对所有 K 个样本将它们的 $\nabla_{b^{(l)}} J(W, b; x, y)$ 累加到 $\triangle b^{(l)}$ 中去

$$\triangle b^{(l)} := \triangle b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$$

于是有：

$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{K} \triangle b^{(l)} \right]$$

同样的，上式中的 $\triangle b^{(l)}$ 是所有 K 个样本的 $\delta^{(l+1)}$ 累加和。

7 随机梯度下降法

当采用随机梯度下降法时，就不需要将所有样本累计相加了。此时样本数K = 1

计算方法：

- 每次使用一个样本时，使用feedforward计算结果，再根据此计算输出层的$\delta^{(n_l)}$
- 由$\delta^{(n_l)}$计算对应的 $W^{(n_l-1)}, b^{(n_l-1)}$
- 由$\delta^{(n_l)}$计算上一层的 $\delta^{(n_l-1)}$
- 同理继续往前推
- 推导第一层，计算得到 $W^{(1)}, b^{(1)}$，此样本就结束了，然后使用下一个样本，从头来。

8 梯度下降法，随机梯度下降法比较

- 梯度下降：在梯度下降中，对于θ的更新，所有的样本都参与调整θ.其计算得到的是一个标准梯度。因而理论上来说一次更新的幅度是比较大的。如果样本不多的情况下，当然是这样收敛的速度会更快。如果样本很多，速度就会比较慢。
- 随机梯度下降:随机也就是说我用样本中的一个例子来近似我所有的样本，来调整θ，因而随机梯度下降是会带来一定的问题，因为计算得到的并不是准确的一个梯度，容易陷入到局部最优解中。样本很多时，速度会比标准梯度下降法快。
- 批量梯度下降：其实批量的梯度下降就是一种折中的方法，他用了一些小样本来近似全部的，其本质就是我1个指不定不太准，那我用个30个50个样本那比随机的要准不少了吧，而且批量的话还是非常可以反映样本的一个分布情况的。