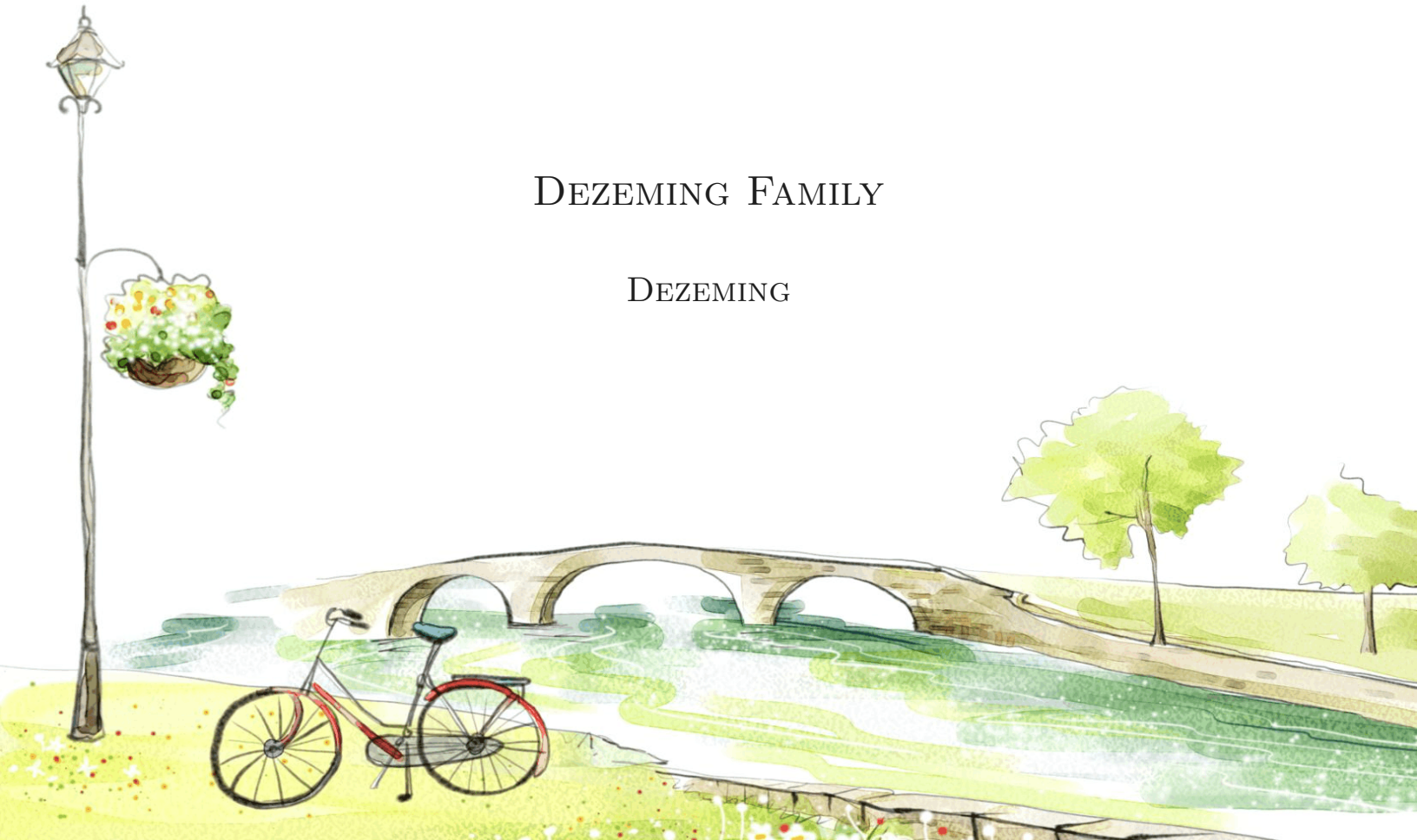
反向传播 BackPropagation 详细推导教程

Dezeming Family

Dezeming



Art. No 0
ISBN 000–00–0000–00–0
Edition 0.0

Cover design by Dezeming Family

Published by Dezeming
Printed in China

目录

0.1 本文前言 5

1 构建前向网络 6

1.1 神经网络结构 6

1.2 代价函数 7

1.3 训练任务 8

2 偏导求解 9

2.1 预备工具 9

2.2 偏导分解 9

2.3 第二步的求解思路 11

2.4 最后一层的偏导求解 11

2.5 其余层的偏导求解 12

2.6 正则化项偏导 13

3 反向转播整体过程 14

3.1 反向传播算法的过程 14

3.2 算法扩展说明 15

Literature 15

前言及简介

DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 *DezemingFamily* 的系列书，可以从我们的网站 *[https://dezeming.top/]* 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

0.1 本文前言

考虑到反向传播是比较重要的算法，并且推导还是有一定的难度，因此我打算好好写一下该算法的推导过程。

如何尽可能写得足够简洁和通俗是一个不太容易的事情，考虑到很多人都会使用吴恩达的《机器学习》课程 [1]，因此我的符号设置也会遵循该课程的符号。该课程并没有推导反向传播算法，但我认为在自学和巩固时一定要学会它的推导，否则就不能算是真正学会了 BP 的原理。在写这本书的时候，我避开了所有参考资料，而是完全自己进行的推导，因此加深了里面的印象，现在我打算把这个推导过程严谨而又不失通俗地呈现出来。

1. 构建前向网络

1.1 神经网络结构 6

1.2 代价函数 7

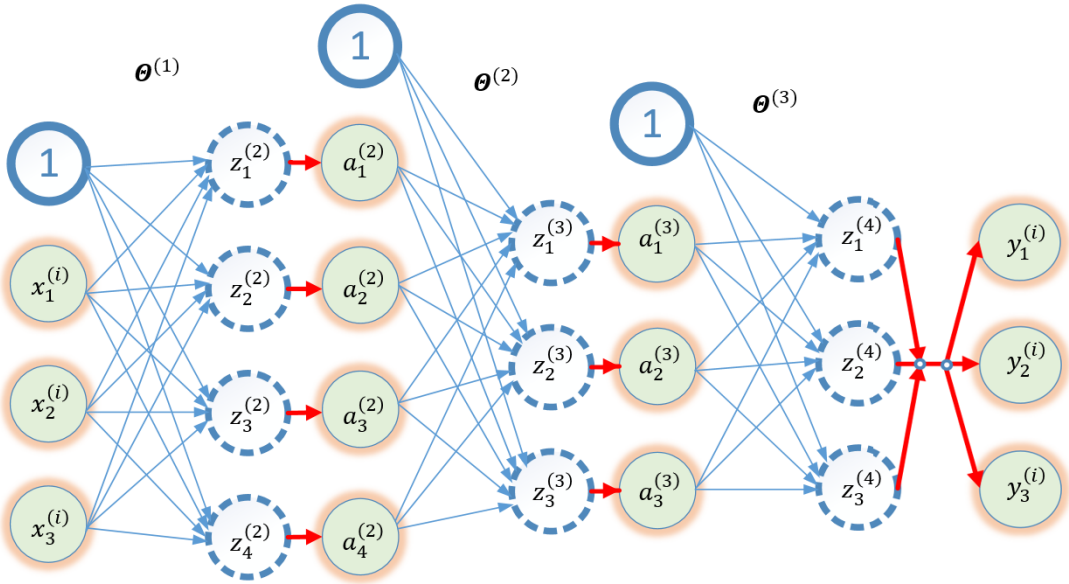
1.3 训练任务 8

本章的内容就是构建前向网络，定义所有的符号标志，为下一章的推导奠定基础。

1.1 神经网络结构

对于多分类神经网络而言，我们设第 b 个样本的特征向量（feature vector）为 $x^{(b)}$，输出向量值为 $y^{(b)}$。对于 n 分类（$n>2$）而言，$y^{(b)}$ 是一个 n 维向量，如果是第 k 个类别，就让该向量第 k 个元素 $y_k^{(b)}$ 为 1，其余值为 0。

我们假设 x 的特征向量一共有 3 个特征，输出为 3 分类，即 $y^{(b)}$ 为三维向量。



如上图，中间红色的箭头表示激活层，最后的数值经过 softmax 函数得到输出，中间连接第 l 层和 $l+1$ 层的权重为 $\Theta^{(l)}$。

但我们这里并不直接推导 softmax 作为最后激活函数的网络，我们假设所有激活层都是 sigmoid 函数，即使最后输出也是 sigmoid 函数激活的。我们设激活函数表示为 $g(\cdot)$。

$$a^{(1)} = x^{(i)} \tag{1.1.1}$$

$$z^{(2)} = \Theta^{(1)} a^{(1)} \tag{1.1.2}$$

$$a^{(2)} = g(z^{(2)}) \quad a^{(2)}\ add\ bias\ a_0^{(2)} \tag{1.1.3}$$

$$z^{(3)} = \Theta^{(2)} a^{(2)} \tag{1.1.4}$$

$$a^{(3)} = g(z^{(3)}) \quad a^{(3)}\ add\ bias\ a_0^{(3)} \tag{1.1.5}$$

$$z^{(4)} = \Theta^{(3)} a^{(3)} \tag{1.1.6}$$

$$a^{(4)} = g(z^{(4)}) = h_\Theta(x) \tag{1.1.7}$$

我们设最后输出值为 $h_\Theta(x)$。注意第 l 层加入的偏置 $bias\ a_0^{(l)}$ 的值都是 1。

1.2 代价函数

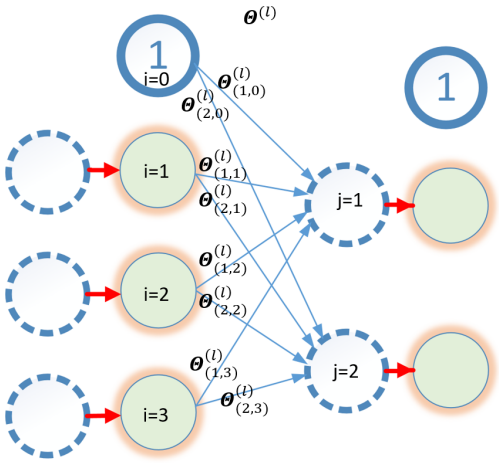
我们需要定义一个代价函数（cost function），这个代价函数一般定义为交叉熵。我们先定义针对一个样本 $x^{(b)}$ 的代价：

$$J(\Theta)^{(b)} = -\sum_{k=1}^{K}\left(y_k^{(b)} \log\left((h_\Theta(x^{(b)}))_k\right) + (1 - y_k^{(b)}) \log\left(1 - (h_\Theta(x^{(b)}))_k\right)\right) \tag{1.2.1}$$

所有的 m 个样本的代价函数就是：

$$J(\Theta) = \frac{1}{m}\sum_{b=1}^{m} J(\Theta)^{(b)} \tag{1.2.2}$$

但我们还有正则化项，我们需要把参数权重写入代价函数里。我们设第 l 层连接层的参数 $\Theta^{(l)}$ 中，前一层的输出为 i，后一层的输入为 j，每个连接线表示为 $\Theta_{ji}^{(l)}$：



通常情况下我们并不惩罚 $bias$ 项。我们设第 l 层输入神经元个数为 S_l，输出神经元个数为

S_{l+1} ，因此，第 l 层的参数代价项为：

$$\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2 \tag{1.2.3}$$

一共网络有 L 层，因此一共有 $L-1$ 个连接层，因此所有的参数代价项为：

$$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2 \tag{1.2.4}$$

前面 $2m$ 中的 2 是为了求导以后计算方便。

因此总的代价函数就定义为：

$$J(\Theta)=-\frac{1}{m}\left[\sum_{b=1}^{m}\sum_{k=1}^{K}\left(y_k^{(b)}\log\left((h_\Theta(x^{(b)}))_k\right)+(1-y_k^{(b)})\log\left(1-(h_\Theta(x^{(b)}))_k\right)\right)\right] \tag{1.2.5}$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2 \tag{1.2.6}$$

1.3 训练任务

我们的任务是尽可能让代价函数更小。根据梯度下降的方法，需要对每个参数求偏导：

$$\frac{\partial J(\Theta)}{\partial \Theta_{ji}^{l}} \tag{1.3.1}$$

利用它来更新参数值：

$$\Theta_{ji}^{l}\leftarrow\Theta_{ji}^{l}-\lambda\frac{\partial J(\Theta)}{\partial\Theta_{ji}^{l}} \tag{1.3.2}$$

偏导相当于函数在该点的变化趋势，当偏导为正时，说明要沿着反方向走代价函数才会减小；偏导为负时，说明就要沿着该方向走代价函数才会减小。上面的 λ 是移动步长，也叫学习率。

而我们发现所有参数在函数中，求导非常困难，因此我们需要分步骤求导，然后分析如何更新参数。我们下一章就是解决求神经网络偏导这个困难的问题。

大家在看后面推导的时候一定要把神经网络图中定义的符号记牢，这样推导才会事半功倍，否则会浪费大量时间。

2. 偏导求解

2.1 预备工具 9

2.2 偏导分解 9

2.3 第二步的求解思路 11

2.4 最后一层的偏导求解 11

2.5 其余层的偏导求解 12

2.6 正则化项偏导 13

本章先介绍反向传播中的偏导求解问题，主要是证明。然后过渡到反向传播算法中。

2.1 预备工具

我们首先求解激活函数 $g(\cdot)$ 的导数：

$$g(x) = \frac{1}{1+e^{-x}} \tag{2.1.1}$$

$$g'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = g(x)(1-g(x)) \tag{2.1.2}$$

链式法则：设 $y = f(x)$ 和 $z = g(y)$ 两个函数，则：

$$\frac{dg(f(x))}{dx} = \frac{dg(f(x))}{df(x)}\frac{df(x)}{dx} \tag{2.1.3}$$

$$that\ is\ :\quad \frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{2.1.4}$$

多元函数：

$$y_i = f_i(x_1, x_2, \ldots, x_n) \tag{2.1.5}$$

$$z = g(y_1, y_2, \ldots, y_m) \tag{2.1.6}$$

则函数对于任意某个变量 x_i 的偏导数为：

$$\frac{\partial z}{\partial x_j} = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}\frac{\partial y_i}{\partial x_j} \tag{2.1.7}$$

2.2 偏导分解

我们先考虑去掉正则化项，代价函数表示如下：

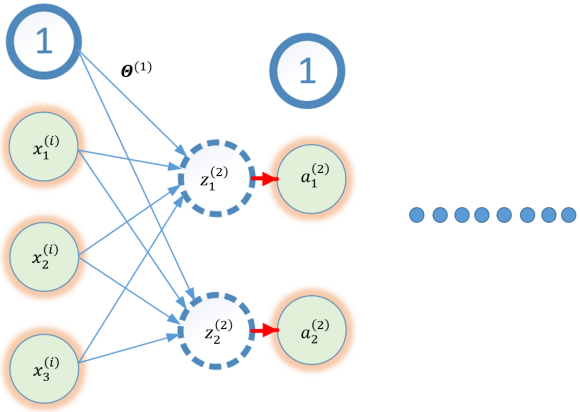
$$J(\Theta) = -\frac{1}{m}\left[\sum_{b=1}^{m}\sum_{k=1}^{K}\left(y_k^{(b)}\log\left((h_\Theta(x^{(b)}))_k\right) + (1-y_k^{(b)})\log\left(1-(h_\Theta(x^{(b)}))_k\right)\right)\right] \tag{2.2.1}$$

好像很复杂的样子，我们只考虑一个样本，毕竟利用多个样本只需要多次计算累加就好了。

$$J(\Theta)^{(b)} = -\sum_{k=1}^{K}\left(y_k^{(b)}\log\left((h_\Theta(x^{(b)}))_k\right) + (1-y_k^{(b)})\log\left(1-(h_\Theta(x^{(b)}))_k\right)\right) \tag{2.2.2}$$

困难的地方在于里面有 $h_\Theta(x^{(b)})$ 。

我们重新回顾一下前向网络这个过程：



$$z_1^{(2)} = \Theta_{1,0}^{(2)} + x_1\Theta_{1,1}^{(2)} + x_2\Theta_{1,2}^{(2)} + x_3\Theta_{1,3}^{(2)} \tag{2.2.3}$$

$$a_1^{(2)} = g(z_1^{(2)}) \tag{2.2.4}$$

为了简洁，我们不再显示样本号 (b)。我们可以发现：

$$\frac{\partial J(\Theta)}{\partial \Theta_{(1,2)}^{1}} = \frac{\partial J(\Theta)}{\partial z_2^2}\frac{\partial z_2^2}{\partial \Theta_{(1,2)}^{1}} \tag{2.2.5}$$

$$that\ is\ : \tag{2.2.6}$$

$$\frac{\partial J(\Theta)}{\partial \Theta_{ji}^{l}} = \frac{\partial J(\Theta)}{\partial z_i^{l+1}}\frac{\partial z_i^{l+1}}{\partial \Theta_{ji}^{l}} \tag{2.2.7}$$

于是我们现在就把求偏导分成了两步，第一步是求 $\frac{\partial z_i^{l+1}}{\partial \Theta_{ji}^{l}}$，第二步是求 $\frac{\partial J(\Theta)}{\partial z_i^{l+1}}$。

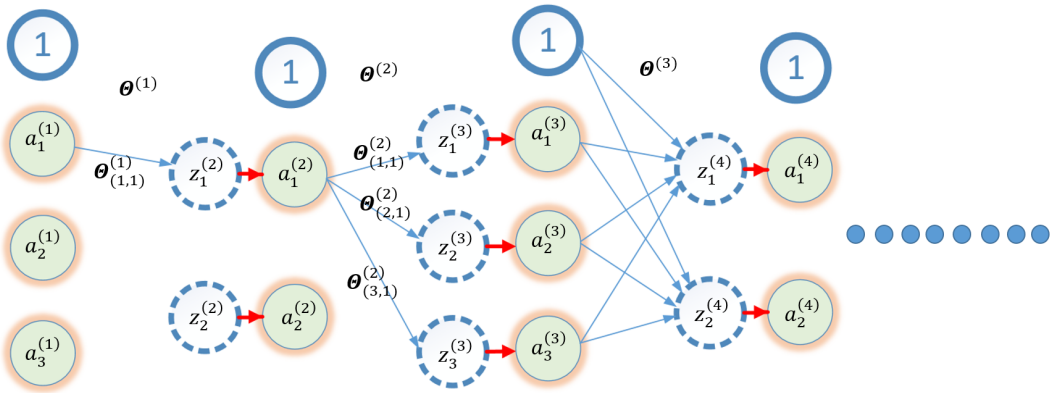
在接下来的证明中，我们需要明确一下，根据神经网络图的标注，我们可以让输入向量 x 表示为 $a^{(1)}$，即 $x_i \equiv a_i^{(1)}$。

第一步 $\frac{\partial z_i^{l+1}}{\partial \Theta_{ji}^{l}}$ 特别好求，我们可以直接得到：

$$\frac{\partial z_i^{l+1}}{\partial \Theta_{ji}^{l}} = a_j^{(l)} \tag{2.2.8}$$

2.3 第二步的求解思路

第二步就比较难了，我们先看看这个网络的整体框架：



根据链式法则，$\frac{\partial J(\Theta)}{\partial z_i^{l+1}}$ 可以写为：

$$\frac{\partial J(\Theta)}{\partial z_i^{l+1}} = \frac{\partial J(\Theta)}{\partial a_i^{l+1}} \frac{\partial a_i^{l+1}}{\partial z_i^{l+1}} \tag{2.3.1}$$

$$a_i^{l+1} = g(z_i^{l+1}) \tag{2.3.2}$$

我们从上面可以看出 $\frac{\partial a_i^{l+1}}{\partial z_i^{l+1}}$ 很好求，但是 $\frac{\partial J(\Theta)}{\partial a_i^{l+1}}$ 就没那么直观了，因为 a_i^{l+1} 还会参与后面的网络。我们对该部分继续使用链式法则。我们以上图为例：

$$\frac{\partial J(\Theta)}{\partial a_1^{(2)}} = \frac{\partial J(\Theta)}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} + \frac{\partial J(\Theta)}{\partial z_2^{(3)}} \frac{\partial z_2^{(3)}}{\partial a_1^{(2)}} + \frac{\partial J(\Theta)}{\partial z_3^{(3)}} \frac{\partial z_3^{(3)}}{\partial a_1^{(2)}} \tag{2.3.3}$$

我们可以发现 $\frac{\partial z_i^{l+1}}{\partial a_i^l}$ 还是比较好算的，但是里面又出现了 $\frac{\partial J(\Theta)}{\partial z_i^{l+1}}$ 这种项，我们又得对该部分根据链式法则求偏导。

也就是说，前一层的神经元计算偏导时，需要借助后面的内容。如果我们从前面算起，那么算第一层连接权重的偏导需要把第一层到最后一层都计算一遍，算第二层时需要把第二层到最后一层都算一遍，以此类推。这样不就有大量重复工作了吗？因此我们可以直接从后面算起，也就是说，先算最后面的偏导，然后逐步往前计算。

我们现在已经明确了步骤，下一节我们就开始真正求出里面每个偏导。

2.4 最后一层的偏导求解

对于 L 层的网络（加上输入层），设最后一层 $h_\Theta(x)$ 表示为 $a^{(L)}$。代价函数表示为：

$$J(\Theta) = -\sum_{k=1}^{K}\left(y_k \log\left(a_k^{(L)}\right) + (1-y_k)\log\left(1-a_k^{(L)}\right)\right) \tag{2.4.1}$$

本节求最后一层的偏导，也就是说我们要求：

$$\frac{\partial J(\Theta)}{\partial z_i^{(L)}} = \frac{\partial J(\Theta)}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \tag{2.4.2}$$

其中链式法则的第一项如下：

$$\frac{\partial J(\Theta)}{\partial a_i^{(L)}} = \left(\frac{-y_i}{a_i^{(L)}} - \frac{-(1-y_i)}{1-a_i^{(L)}}\right) \tag{2.4.3}$$

第二项我们已知：

$$a_i^{(L)} = g(z_i^{(L)}) \tag{2.4.4}$$

我们第一节已经求过 sigmoid 的导数了，因此偏导可以写为：

$$\frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} = a_i^{(L)} \cdot (1 - a_i^{(L)}) \tag{2.4.5}$$

最终我们得到最后一层的偏导：

$$\frac{\partial J(\Theta)}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} = \left(\frac{-y_i}{a_i^{(L)}} - \frac{-(1-y_i)}{1-a_i^{(L)}}\right)\left(a_i^{(L)} \cdot (1 - a_i^{(L)})\right) = a_i^{(L)} - y_i \tag{2.4.6}$$

2.5 其余层的偏导求解

除去第一层不需要求偏导以外，其余所有层 l 都需要求：

$$\frac{\partial J(\Theta)}{\partial z_i^{(l)}} \tag{2.5.1}$$

因为 $z_i^{(l)}$ 会直接作用于后面连接它的网络，因此链式法则得到：

$$\frac{\partial J(\Theta)}{\partial z_i^{(l)}} = \sum_{d=1}^{S_{l+1}} \frac{\partial J(\Theta)}{\partial z_d^{(l+1)}} \frac{\partial z_d^{(l+1)}}{\partial z_i^{(l)}} \tag{2.5.2}$$

其中 S_{l+1} 表示第 $l+1$ 层一共有多少个神经元。
我们观察一下该项：

$$\frac{\partial z_d^{(l+1)}}{\partial z_i^{(l)}} = \sum_{c=1}^{S_{l+1}} \frac{\partial z_d^{(l+1)}}{\partial a_c^{(l)}} \frac{\partial a_c^{(l)}}{\partial z_i^{(l)}} \tag{2.5.3}$$

当 $c \neq i$ 时，$a_c^{(l)}$ 与 $z_i^{(l)}$ 无关，所以上面除了 $c = i$ 其他项都是 0。我们得到：

$$\frac{\partial z_d^{(l+1)}}{\partial z_i^{(l)}} = \frac{\partial z_d^{(l+1)}}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} \tag{2.5.4}$$

其中第一项偏导为：

$$\frac{\partial z_d^{(l+1)}}{\partial a_i^{(l)}} = \Theta_{(d,i)}^{(l)} \tag{2.5.5}$$

第二项偏导为：

$$\frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} = a_i^{(l)} \cdot (1 - a_i^{(l)}) \tag{2.5.6}$$

因此我们可以得到：

$$\frac{\partial J(\Theta)}{\partial z_i^{(l)}} = \sum_{d=1}^{S_{l+1}} \left(\frac{\partial J(\Theta)}{\partial z_d^{(l+1)}} \cdot \Theta_{(d,i)}^{(l)} \cdot a_i^{(l)} \cdot (1 - a_i^{(l)}) \right) \tag{2.5.7}$$

$$= \left(a_i^{(l)} \cdot (1 - a_i^{(l)}) \right) \sum_{d=1}^{S_{l+1}} \left(\frac{\partial J(\Theta)}{\partial z_d^{(l+1)}} \cdot \Theta_{(d,i)}^{(l)} \right) \tag{2.5.8}$$

后面的 $\sum$ 项可以写为矩阵相乘的形式：

$$\begin{bmatrix} \Theta_{(1,i)}^{(l)} & \Theta_{(2,i)}^{(l)} & \dots & \Theta_{(S_{l+1},i)}^{(l)} \end{bmatrix} \begin{bmatrix} \frac{\partial J(\Theta)}{\partial z_1^{(l+1)}} \\ \frac{\partial J(\Theta)}{\partial z_2^{(l+1)}} \\ \dots \\ \frac{\partial J(\Theta)}{\partial z_{S_{l+1}}^{(l+1)}} \end{bmatrix} \tag{2.5.9}$$

2.6 正则化项偏导

现在除了正则化项以外，所有需要的偏导都已经求完了，我们本节求正则化项的偏导。

因为 *bias* 不被惩罚，所以正则化项中，我们得到：

$$\frac{\partial \left(\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(\Theta_{ji}^{(l)} \right)^2 \right)}{\partial \Theta_{(j,i)}^{(l)}} = \begin{cases} \frac{\lambda}{m} \Theta_{(j,i)}^{(l)} & i \neq 0 \\ 0 & i \neq 0 \end{cases} \tag{2.6.1}$$

其实到现在为止，我们已经将反向传播算法搞定了。但是我们是将整体拆散为了多个部分，我们下一章再从整体上介绍一下 BP 算法的实现过程。

3. 反向转播整体过程

3.1 反向传播算法的过程 14

3.2 算法扩展说明 15

本章从整体上介绍反向传播算法的步骤。

3.1 反向传播算法的过程

我们设样本序号为 b，一共 m 个样本。对一个样本求得的代价函数偏导为 Δ（除去正则化项）：

$$\Delta_{(j,i)}^{(l)}(x^{(b)}) = \frac{\partial J(\Theta)}{\partial \Theta_{(j,i)}^{l}}(x^{(b)}) \equiv \frac{\partial J(\Theta)}{\partial \Theta_{(j,i)}^{l}}^{(b)} \tag{3.1.1}$$

我们设全部样本的：

$$D_{(j,i)}^{(l)} = \frac{1}{m}\sum_{b=1}^{m}\Delta_{(j,i)}^{(l)}(x^{(b)}) \tag{3.1.2}$$

总的代价函数为：

$$\frac{\partial J(\Theta)}{\partial \Theta_{(j,i)}^{l}} = \begin{cases} D_{(j,i)}^{(l)} + \frac{\lambda}{m}\Theta_{(j,i)}^{(l)} & i \neq 0 \\ D_{(j,i)}^{(l)} & i = 0 \end{cases} \tag{3.1.3}$$

而反向传播就是从最底层开始，逐步计算里面的偏导：

$$\frac{\partial J(\Theta)}{\partial z_i^{(l)}} \tag{3.1.4}$$

该偏导是求得如下代价函数对于某网络权重偏导的前提条件：

$$\frac{\partial J(\Theta)}{\partial \Theta_{(j,i)}^{l}} \tag{3.1.5}$$

这就是反向传播算法。

3.2 算法扩展说明

我们上面只是为了简单，把所有网络激活层都设置为了 $sigmoid$ 函数，但多分类任务中，通常最后一层会使用 $softmax$ ，里面的激活函数也各种各样，例如不会造成梯度消失的 $Relu$ 函数。但是相信大家在熟练推导反向传播算法以后（我本人对照着书本看了两遍，才最终能自己亲手推导出来），就能有能力去扩展到各种情况了。

有些神经网络的内部会使用大量 if-else 结构，用来做各种判断，例如循环神经网络等，这里面的推导也并不是那么复杂，只要我们能够清晰构建出前向网络，那么就可以从后往前把分解出的一些偏导进行求解了。

Bibliography

[1] https://www.coursera.org/learn/machine-learning