神经网络反向传播通用算法快速入门指南

刘佳 编写

Email： 281033201@qq.com

2018.9.16

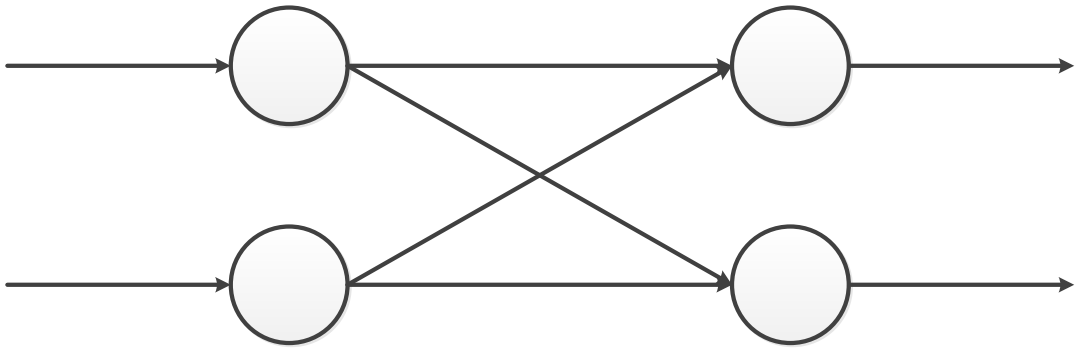
https://liujia1.github.io

前言

写这个文章的目的是为了帮助想利用反向传播自己计算神经网络梯度的人，文章里不对算法做过多解释，而是直观的帮助读者知道如何使用该算法，希望读过这篇文章后，读者能自行计算任何形状的神经网络梯度。 由于本人能力与时间有限，行文难免仓促，若有不妥与错误之处请读者指出，帮助完善这篇文章，使更多的人能从中受益。

规范

神经网络由于结构复杂，如果不规范命名，往往搞不清楚公式中的符号到底指代网络中的哪部分。 一般我们将神经网络画成



其中，把神经元按层分组，上图从左至右依次是第一层，第二层神经元组。而信号线也是从左至右称作第一组参数，第二组参数，以此类推。 第 n 层的神经元组的输入就是 n 组参数，输出就是 n+1 组参数。 本文将给出反向梯度通用算法的说明。

正文

大部分的书籍会把单个神经元描述成如下形式

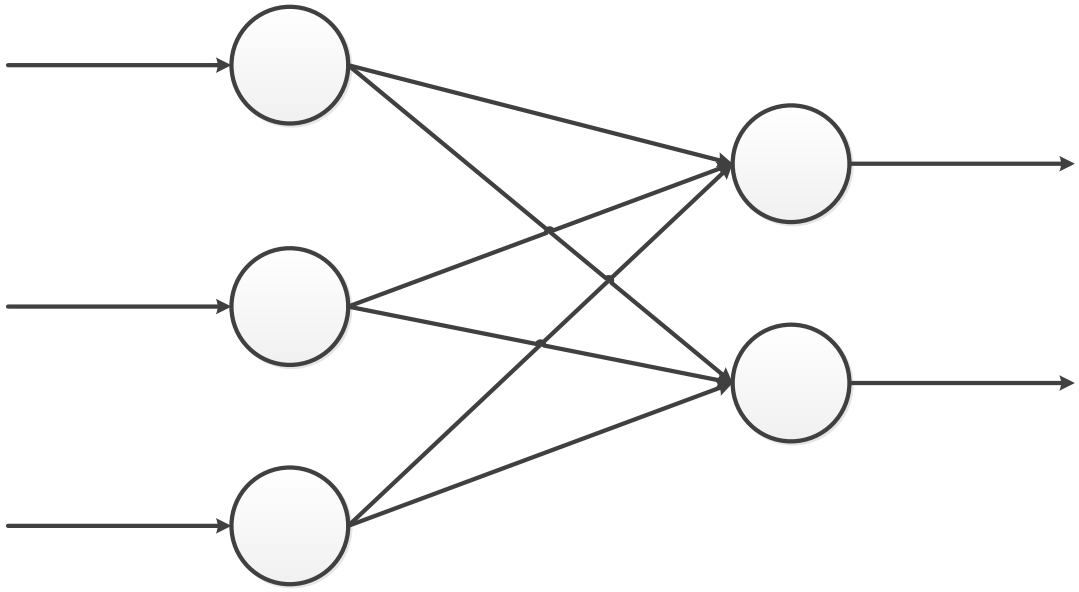
$$Y_i=\sigma(\sum a_i X_i+b_i)$$

用矩阵来描述由多个这样的神经元组成的一层神经元组是这样

子的（一层指竖着的一排）

$$Y=\sigma(A*X+B)$$

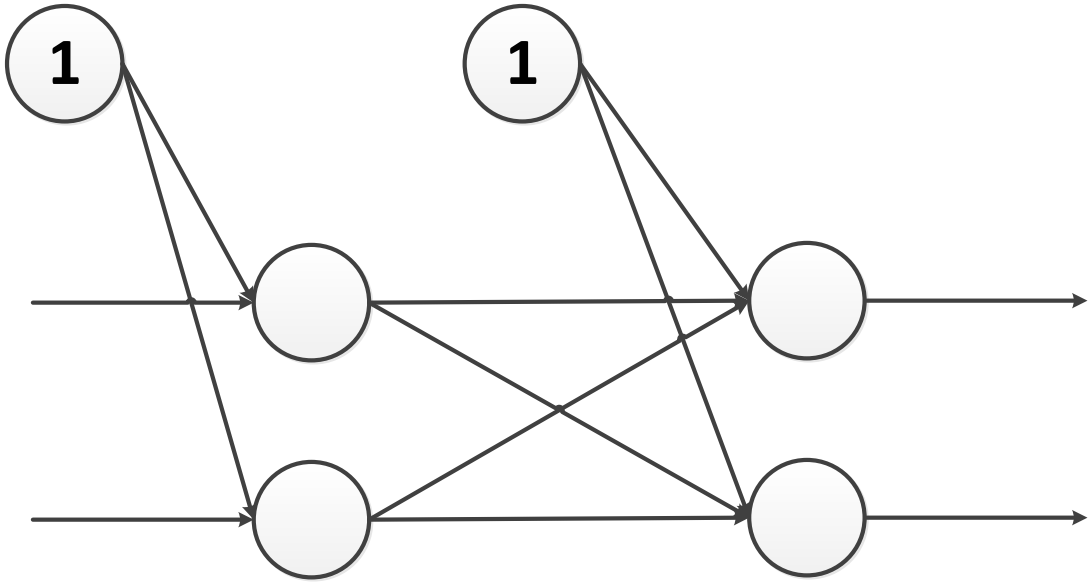
神经网络的形状如下图



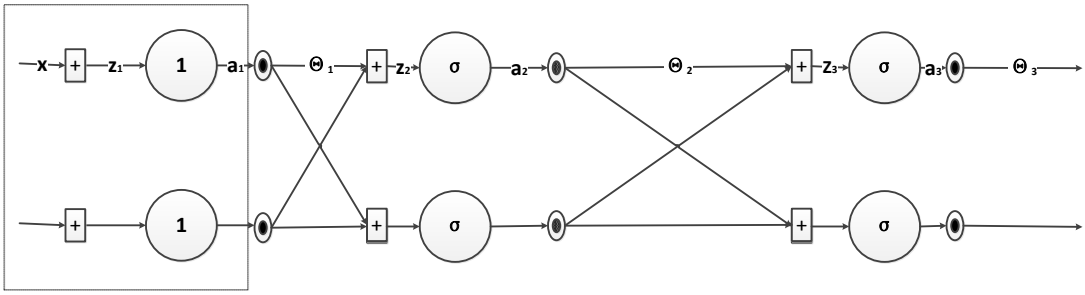
我们最右边的输出层为例解释上面那个公式。其中 Y 是 2*1 矩阵，对应神经元组输出， A 是 3*2 矩阵，对应输入权重， X 是 3*1 矩阵，表示输入， B 是 2*1 矩阵，表示神经元的偏置。我们如果使用这个的数学模型来考虑问题，面对反向传播时，我们需要把 A*X 和 B 割裂开来处理，因为他们数学形式是不同的。为了思维的一致性，我们可以调整一下神经网络的结构，使得我们可以使用如下的式子来表示前面的式子

$$Y=\sigma(A*X)$$

如此，涉及到对每一层需要额外放置一个偏置输入，网络图如下

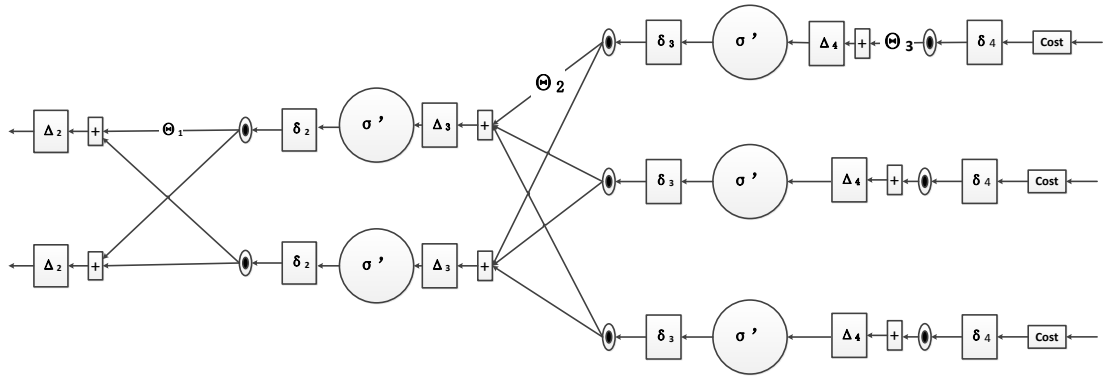


为了方便理解反向传播， 我们还需要对上面的网络画法再做进一步改进，为了方便，我去掉了偏置，只考虑输入相关，每层包含 2 个神经元：



上图把原来直线相连的信号传递线，分步进行了拆分，并做了标记命名。需要注意，我为了节省画图时间，省去了偏置量，但是读者需要心理明白，他们和神经网络的输入是一样的，需要被平等无差别对待。这个神经网络， 有一个输入层，1 个隐藏层和 1 个输出层。其中 记号 a 表示神经元的计算输出，特别的与传统记法不同，我把输入输入层的输入 X 看作第一个输出 a1（虚线框框起的输入层），即把神经网络的输入层也看作是一个神经网络层，它的神经元是常数变换（*1），神经导线的权重是 1。Θ 表

示神经导线的权重值。我们反向传播运算的目的就是找到一组 Θ，使得神经网络的损失（误差）函数最小，而且反向传播的目的，就是为了求 Θ 的梯度。z 是每个神经元的输入，它是前一层 a 与神经导线权重 Θ 的乘积的和。每层输入乘以权重的值，即神经网络的基本运算 A*X 的值。 σ 是单个神经元的运算函数，可以是 sigmoid 或者别的什么，我们不关心。由于不讨论正向传播，文章只关心如何计算反向传播，下图是根据反向传播对上图的正向运算方向进行了反转。



为了图示方便，上图省略了一些正向传播时需要计算的量。反向传播是为了求梯度，梯度即导数，因此神经元的计算变成了σ'(z)求导。顾名思义，反向传播，我们就要烦着来进行计算。一切都要从图示最右边的cost误差函数开始计算。需要特别留意，δ表示误差，是反向运算的核心参数，δ4数值上等于cost对a3求偏导，而后下一层（从右往左看）的δ就是上一层的Δ经过σ'(z)运算后

$$\delta 3 = \Delta 4 .* \sigma'(z)$$ （这里.*表示按元素乘法）

得到的。而Δ则是δ验证神经导线的发散汇总值。我以隐藏层与输入层之间的网络来解释何谓发散汇总：δ3 在◉处沿着神经导

线做Θ计算后发散到不同的神经元上，然后在⊞处进行汇总，这个过程正好和矩阵的乘法运算匹配，可以用

$$\Delta^{3}_{21}=(\Theta^{2}_{32})^{T} * \delta^{3}_{31}$$

来表示，上标δ^{3}表示δ3，下标表示n*m的矩阵，其中n表示右边的神经元个数，m表示左边的神经元个数，因为反向传播，这么记更自然。要计算出每个Θ的梯度（导数），最终的运算是

$$\Theta'_{b}=\delta_{b+1}*a_{b}$$

根据这个式子，上图的 $\Theta'_{2}=\delta_{3}*a_{2}$。说一句题外话，由于神经网络采用批量训练，我们需要一次性求出一个批中的所有 Θ'后，再求他们的平均值作为最后返回的梯度值。下面我将模拟演算一次上图的求梯度过程，假设我有 100 个样本，每个样本输入层有 2 个输入，中间层有 2 个，输出有 3 个神经元。 正向传播后我们能得到：

样本数 m=100,

初始化参数 $\Theta^{1}_{22}, \Theta^{2}_{32}$

$a^{1}_{21},z^{2}_{21},a^{2}_{21},z^{3}_{31},a^{3}_{31}$

据此计算一遍反向传播：

$\delta^{4}_{31}=\partial Cost/\partial a3$

$\Delta^{4}_{31}=\delta^{4}_{31}$

$\delta^{3}_{31}=\Delta^{4}_{31}.*\sigma'(z^{3}_{31})$

$\Theta'^{2}_{32}=\delta^{3}_{31}*(a^{2}_{21})^{T}$

$\Delta^{3}_{21}=(\Theta^{2}_{32})^{T} * \delta^{3}_{31}$

$\delta^2_{21} = \Delta^3_{21} .* \sigma'(z^2_{21})$

$\Theta'^1_{22} = \delta^2_{21} * (a^1_{21})^T$

matlab 伪代码：

```
For i=1:100
δ4=∂Cost/∂a3
Δ4=δ4
δ3=Δ4 .* σ'(z3)
Θ'2=Θ'2+δ3* (a2)^T
Δ3=(Θ2)T    * δ3
δ2=Δ3 .* σ'(z2)
Θ'1=Θ'1+δ2* (a1)^T
End
Θ'1=(1/m).* Θ'1
Θ'2=(1/m).* Θ'2
```

伪代码需要注意，求 Θ 的导数时，由于是对一个批次样本进行计算，所以 Θ'是取累加值，一个批量都算完后再求 Θ'的均值。另外地整个步骤中，δ4=∂Cost/∂a3 得根据损失（代价）函数来手工导出式子，一些神经网络框架会提供自动求导，但是知道怎么算的还是有益的，如果损失函数是交叉熵，那么 δ3 正好等于 y-a3（样本期望-实际网络输出）。

进一步讨论

把神经网络简化的看作这么一个函数

$$A(X_1,X_2)=F(G(X_1,T(X_2))$$

分别对 X_1 和 X_2 求偏导

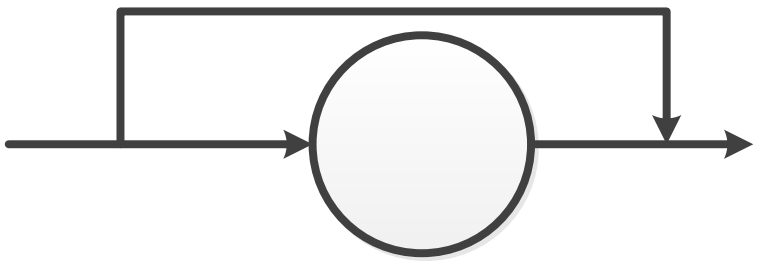
$$A'(X_1)=F'*G'(X_1)$$

$$A'(X_2)= F'*G'*T'(X_2)$$

我们在求 A 的时候，需要先计算 T，再计算 G，最后计算 F。在求偏导时，我们先求 F'，再求 G',最后再是 T'。这个和反向传播的理念是一样的。 另外可以发现，要求 $A'X_2$，需要求 F'*G'，这个结果再计算 $A'X_1$ 时已经计算过了，因此结合之前的说明，我们可以发现这个 F'*G'其实就是对应算法中的 δ。而上一层的输出 a 则对应 T'作用于 X_2 上的系数。

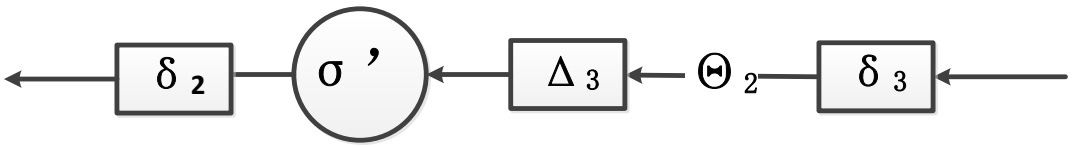
拓展

在使用梯度下降法时，针对神经网络，存在梯度消失的情况（=0）。为了克服这个问题，有人引入了残差网络，一个残差的神经元如图：



Y=X+σ(X)

利用之前提到的算法，我们只需要把 X+σ(X)看作一个新的 Y=σ(X)即可，这样网络就又变成了



再根据之前的算法，逐步运算即可。