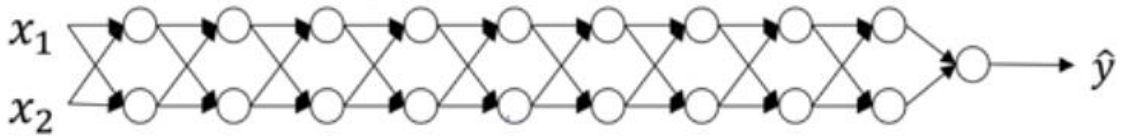
1、梯度消失/爆炸

先从一个直观的例子看起，下图所示是一幅简单的神经网络，假设每个参数至矩阵



都比单位矩阵大一点，那么最后的输出值将与参数呈指数级增长，反之，若比单位矩阵小一点，那么输出值将与参数呈指数级下降。

所以在最后求偏导的时候，对于浅层参数来说，梯度爆炸将使得其更新步伐变大，可以采用梯度修剪避免此类问题，以及在初始化矩阵的时候不要超过 1，一般要乘以 0.01 的系数减小参数值。

而梯度消失将使得其更新步伐极缓，可以这样理解，如果梯度更新缓慢，那么深层计算得到的损失值对浅层参数的优化是没有效果的，在 RNN 里，长文本的序列生成模型就存在这样的问题，如果第 n+t 个词汇的生成不准确，要想更新参数，那么就很难改变浅层的参数，（其实本质上 CNN 和 RNN 的梯度消失和爆炸问题属于同一类问题，只是解决方案有所不同）。

2、权值优化

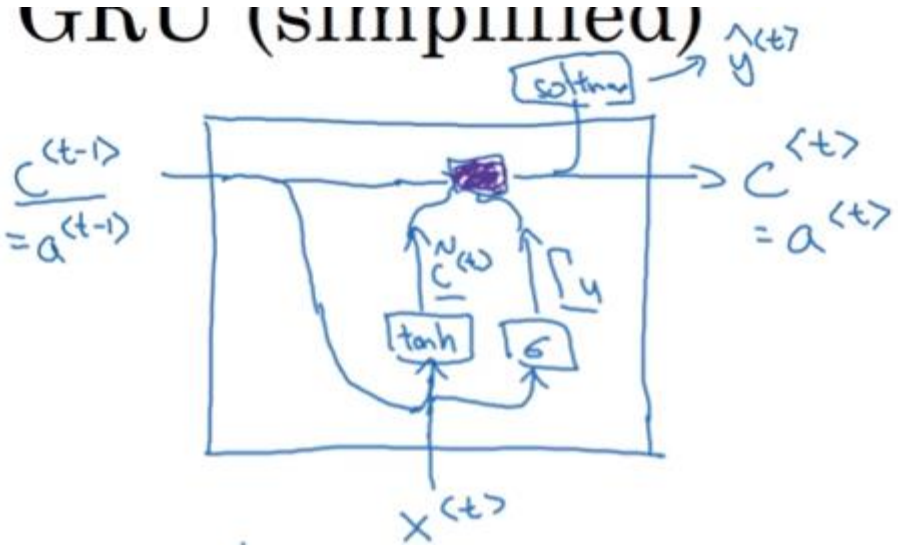
有个遗留问题是，为什么浅层参数值优化不好会使得深层网络预测效率低下，这个问题归根到底就是权值优化问题，这样做是有原因的，首先，深层网络由许多非线性层堆叠而来，每一层非线性层都可以视为是一个非线性函数 f(x)f(x)f(x)(非线性来自于非线性激活函数），因此整个深度网络可以视为是一个复合的非线性多元函数：

$$F(x)=f_n(...f_3(f_2(f_1(x)*\theta_1+b)*\theta_2+b)...)$$

那么如果浅层参数优化不好，就会导致基础数值传递失策（没有考虑这样的权重传递对于深层次网络的预测效果），梯度爆炸如此（使得浅层参数变化过快失去精准度），梯度消失亦是如此（浅层参数优化过慢导致效率低下），梯度消失导致的后果即：不是浅层特征没有传递下去，而是浅层特征传递下去几乎都是偏离正确预测方向的。所以回到问题根本上来，解决梯度消失/爆炸问题的最终目的就是要使浅层特征计算能够准确无误的传递到深层网络中去，解决方案：一是梯度修剪，使其变化幅度降低到合理范围内，二是 GRU/LSTM

3、GRU

既然梯度消失导致特征不能正确地传递下去，那么何不直接将参数“拷贝”一份传递下去，而不再该层进行特征计算。基于此，GRU 算法得以实现，简单来说，就是上一层的特征输出不在该层进行特征计算，直接小部分的改动（更新门）并传递到下一层，这样就保留了原来的特征但不会破坏该特征（实属无奈之举），类比于残差网络的恒等连接，如果网络在该层学到了有用的东西，非但不会产生梯度消失现象，还学到了额外有用的东西，这就是更新门的作用。（需要注意的是，这不是从本质上解决梯度消失问题）



上图即为 GRU 门控单元的示例图，根据图中所示，我们可以得到如下公式：相比

$$\tilde{c}^{<t>} = \tanh(W_c[\, c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

于之前的传递公式，a^t 是在该层进行了类似于公式 1 的计算后就直接传递到下一层去了，而没有进行更新门更新操作，那么这样一来由于梯度消失导致权值计算失策却传递到了深层网络中。在 GRU 单元中，由于更新门（向量）大部分都为 0（公式①和公式②使得该层计算特征和更新门向量长度一致），那么就很容易将浅层特征传递下去。

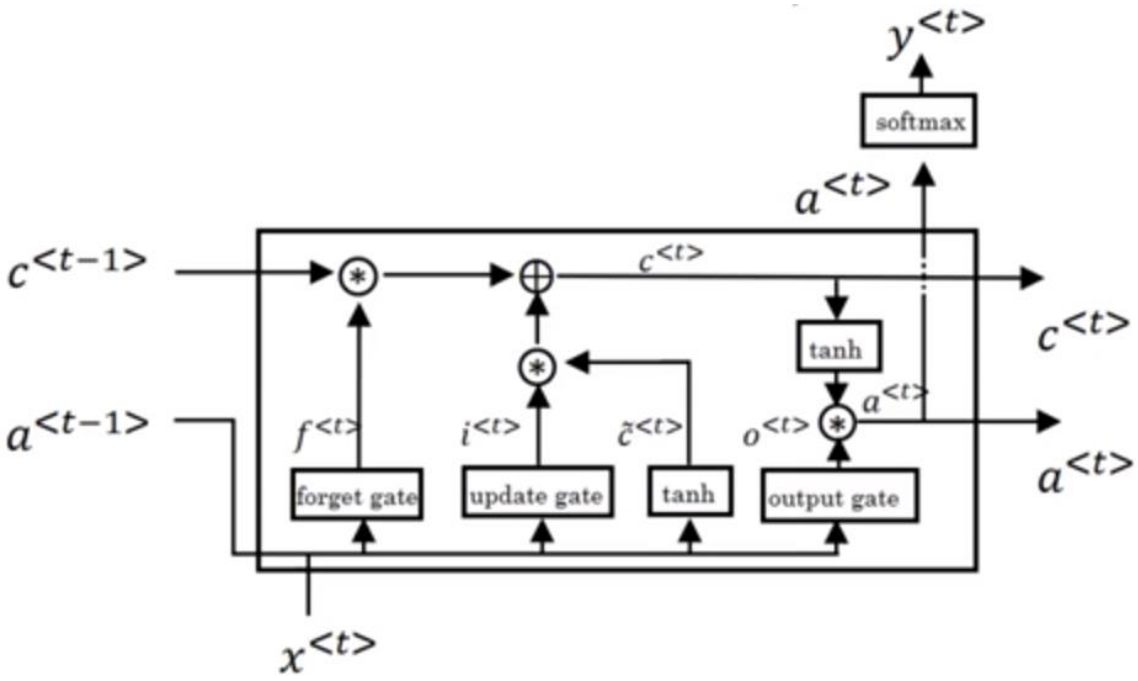
以上只是简化过的 GRU 单元，实际上，历史上存在过很多个 GRU 版本，常见的 GRU 单元还会改变公式①

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

即在计算本层特征的时候，添加一个相关性门，该门的计算过程与更新门计算类似，同样保持维度一致。

4、LSTM 长短期记忆



先来看下 LSTM 单元门的示例图和公式：

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

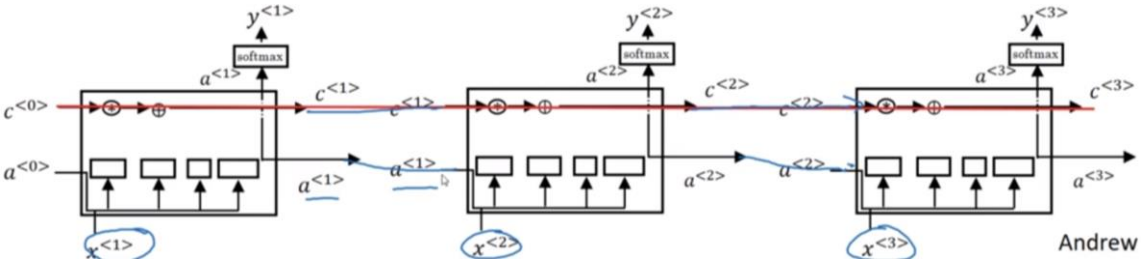
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

相比 GRU 单元，它多了一个门—输出门，在计算特征传递的时候，它与 GRU 单元类似，使用更新门和遗忘门（而不是只使用更新门和 1-更新门），多出来的一个输出门是为了更好的方便本层计算新特征的，所以可以这样理解，LSTM 不仅也能保持浅层特征传递下去，同时能够更加高效的计算每层新的特征值。



总结来说，GRU 单元更加简单，计算快速，而 LSTM 单元灵活强大，准确度更高，但使用起来稍显繁琐。