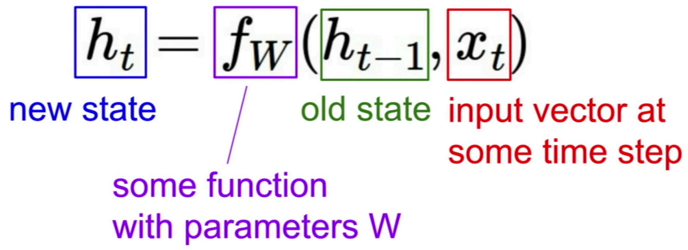
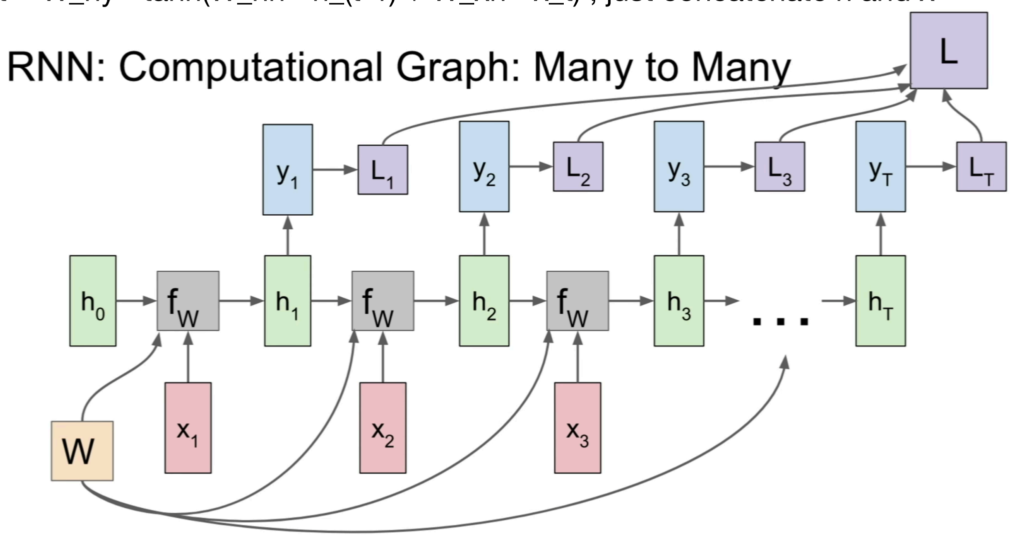
cs231n-3: Reccurent Neuron Network

1. RNN综述
    1. Get some input
    2. Update its hidden state
    3. produce an output
    4. 

2. Functional form
    1. yt = W_hy * tanh(W_hh * h_(t-1) + W_xh * x_t) , just concatenate h and x
    2. 

    3. 可以看到，t个x输入，W是同一个W，h_1...t随着x的输入更新，注意W在这一轮是共享的，而且函数的应用是连续的，所以W的梯度就是小梯度的sum，在这样的情况下，y可以在每次运算中被求得，那么就得到了一个连续的输出，L可以分别计算然后求和得到。
    4. **下面解释RNN如何操作**
        1. **Training Time**
            1. **输入是x1, x2, x3是<start>, word1, word2, …, wordn, 无结束符号**
                1. **也就是说，根据前一个单词得到下一个单词**
            2. **输出是y1, y2, y3，也就是最后的caption，可以和word1, word2, …, wordn, <END>对比**
                1. **也就是说，对比得到loss**
            3. **那么我们的图片在哪里呢？图片先由CNN训练，然后用一个affine函数转换成h0**
                1. **图片决定的hidden state能够告诉我们该输出什么y**

    2. **Testing Time**
        1. **从图片的到h0，然后输入<start>**
        2. **然后就可以依次的到yi，直到yi=<END>**
        3. **注意yi是每个单词的概率，我们要在这个概率上做一次sample，提高准确率（但是我们实现的版本貌似只取了最大值）**
3. RNN通常用在Language Modeling Problem
    1. Such as how to produce natural language
    2. Example: Character-level Language Model
        1. 比如training阶段，我们学习的到下一个字母的概率矩阵，然后取得hot vector，重新放到input
        2. testing阶段，我们用这个下一个字母的概率矩阵求得一个字母串，其中每个字母是在求出softmax loss以后sample得到的
        3. sample保证了多样性，可以得到不同的结果
4. Truncated back-propagation through time
5. Vanilla RNN Gradient Flow
    1. compute gradient of h0 involves many factors of W
        1. 最大奇异值>1，梯度爆炸
        2. 最大奇异值<1，梯度消失
        3. 启发式方法
            1. gradient clipping for exploding
            2. change architecture for vanishing -> LSTM
6. **LSTM-Long Short Term Memory, raised in 1997, but used anywhere now**
    1. it can avoid vanishing or exploding gradient
    2. cell state/ hidden state
        1. four functions, “**ifog**”, the input of these four gates is W(h_t-1, x_t)
            1. **f**, forget gate, whether to erase the cell (sigmoid)
            2. **i**, input gate, whether to write this cell (sigmoid)
            3. **g**, gate gate, how much to write this cell (tanh)
            4. **o**, output gate, how much to reveal this cell (sigmoid)

$$\boxed{\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

        2. $h_t = o \odot \tanh(c_t)$ where it’s element wise multiplication, not matrix multiplication
        3. W’s size is 2h x 4h
    3. benefit
        1. back propagation from c_t to c_t-1 only element wise multiplication by f, no matrix multiply by W
        2. from hidden state to initial cell state c_0, it only back propagate through a single tanh gate
        3. give a highway for gradient back propagation, similar to RNN
        4. it’s element wise multiplication that avoid the vanishing

4. LSTM的变种
    1. GRU等
    2. these variance of LSTM is not significantly better then vanilla LSTM
    3. 最重要的事是如何管理gradient flow