

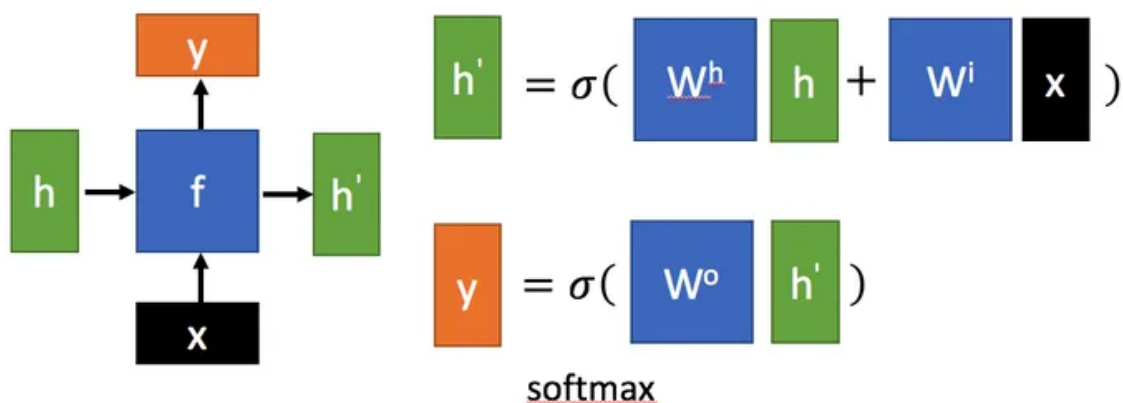
chpt_6 - RNN - 报告

1、RNN 模型

RNN 的全称是循环神经网络 (Round Neural Network) 。RNN 的神经元结构如下图：

Naïve RNN

- Given function $f: h', y = f(h, x)$



Ignore bias here

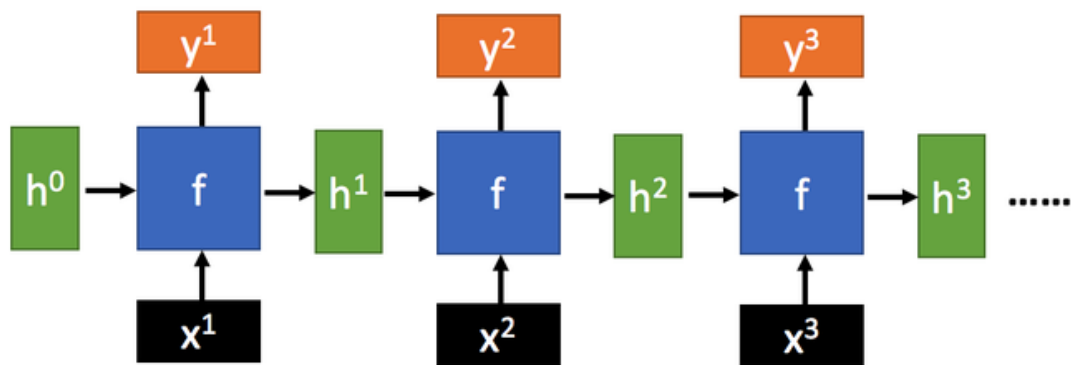
其中， x 表示当前状态下的输入数据， h 表示接收到的上一个状态的输入， h' 表示节点传递给下一个状态的输出， y 表示当前状态下的输出。

各个量之间的数学关系可以表示成上图中右侧的两个公式，其中： W^h 表示隐藏层的权重矩阵， W^i 表示输入层的权重矩阵， W^o 表示输出层的权重矩阵。如果将 RNN 的循环过程展开，可以表示成如下图的形式：

Recurrent Neural Network

- Given function $f: h', y = f(h, x)$

h and h' are vectors with the same dimension



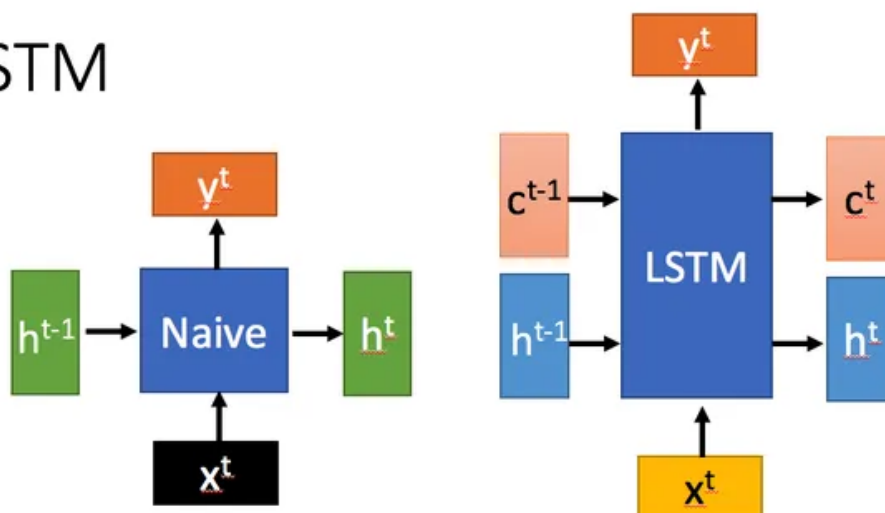
No matter how long the input/output sequence is,
we only need one function f

2、LSTM 模型

LSTM 的全称是长短时记忆神经网络 (Long Short - Term Memory) , 是一种特殊的 RNN, 用来解决长序列训练时产生的梯度爆炸和梯度消失的问题。

LSTM 的神经元相较于普通 RNN 的不同可以用下图来表示:

LSTM



c change slowly $\Rightarrow c^t$ is c^{t-1} added by something

h change faster $\Rightarrow h^t$ and h^{t-1} can be very different

可以看到，LSTM 神经元有两个传递的状态 c 和 h ， c 的变化慢， h 的变化快。

LSTM 神经元首先利用上个状态的 h^{t-1} 和当前状态的输入 X^t 得到下面四个值：

$$Z = \tanh(W \cdot [X^t, h^{t-1}])$$

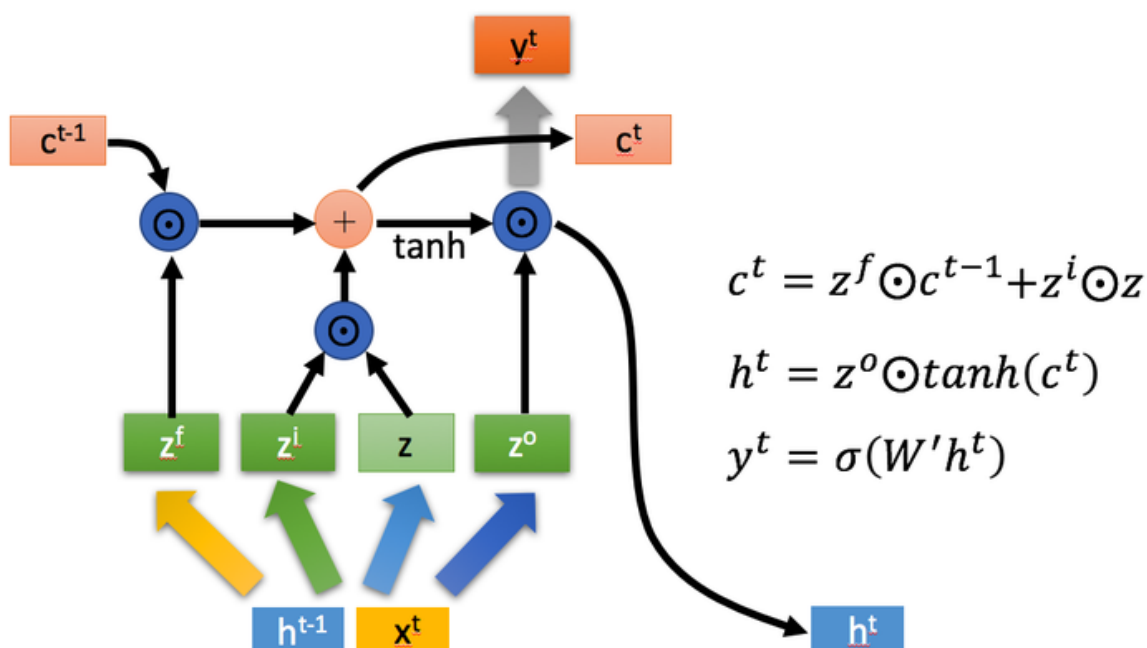
$$Z^i = \sigma(W^i \cdot [X^t, h^{t-1}])$$

$$Z^f = \sigma(W^f \cdot [X^t, h^{t-1}])$$

$$Z^o = \sigma(W^o \cdot [X^t, h^{t-1}])$$

其中，四个 W 是需要通过训练得到的权重矩阵， \tanh 和 σ (sigmoid) 是两种激活函数。

利用这四个值，进一步计算当前状态的输出值，过程可用下图表示。



其中， \odot 表示两矩阵的对应元素相乘，即两矩阵的形状需要相同。

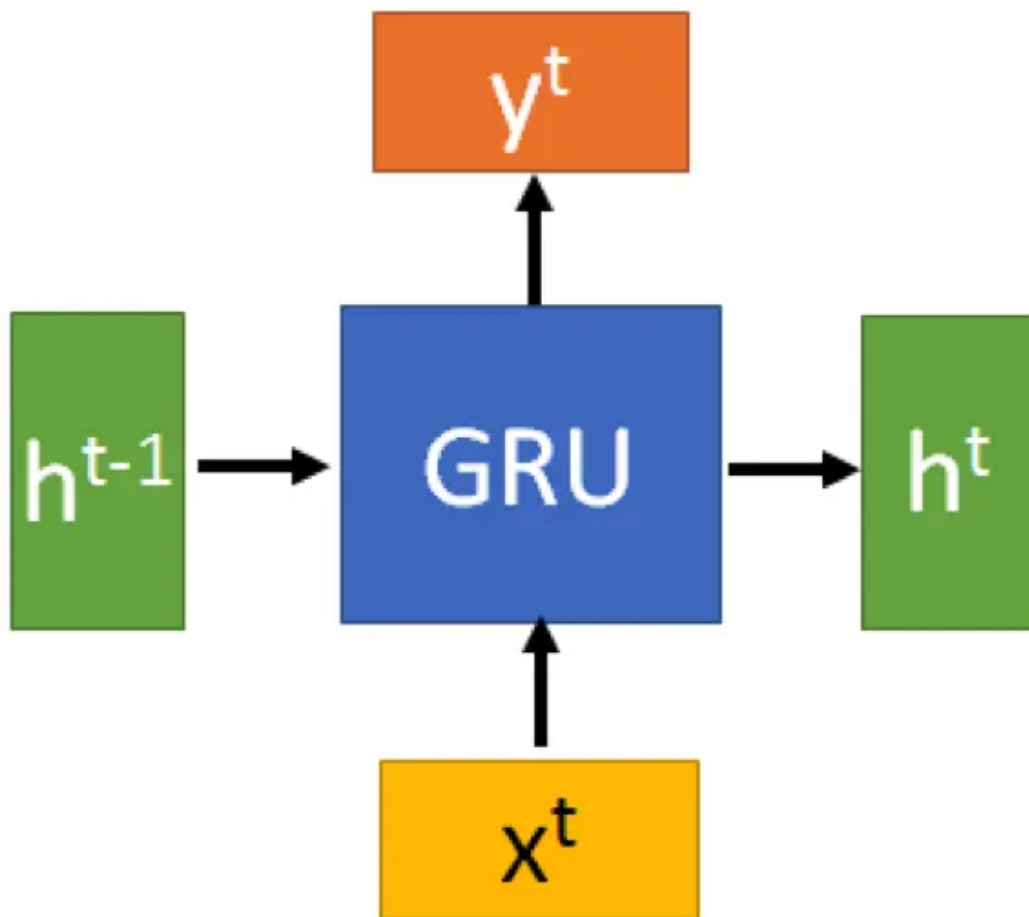
通过上述的计算过程，来解释在 LSTM 神经元内部进行的三个阶段。

1. 遗忘阶段。LSTM 会选择性地遗忘一些输入，而不像 RNN 那样处理所有输入。
 z^f 作为忘记门控 (f 表示 forget)，它是通过 sigmoid 激活函数得到的，那它为 0 的位置表示遗忘，为 1 表示记忆，来控制上一个状态的 c^{t-1} 中的哪些信息需要忘记。
2. 记忆阶段： z^i 作为记忆门控，来对输入 z 进行选择记忆。遗忘部分和记忆部分相加，作为传递给下一个状态的输出 c^t 。
3. 输出阶段：利用 z^o 来决定将哪些数据作为当前状态的输出。

3、GRU 模型

GRU 的全称是门控循环单元神经网络 (Gate Recurrent Unit)，同样用来解决长期记忆中的梯度问题，相较于 LSTM，它的训练更容易。

GRU 的输入输出结构和普通 RNN 相同，如下图：



GRU 神经元首先利用输入得到如下两个门控状态：

$$r = \sigma(W^r \cdot [X^t, h^{t-1}])$$

$$z = \sigma(W^z \cdot [X^t, h^{t-1}])$$

然后利用门控 r “重置”输入的数据，得到如下两个数据：

$$h^{t-1'} = h^{t-1} \odot r$$

$$h' = \tanh([X^t, h^{t-1'}])$$

然后再“更新”输出，得到 h^t ：

$$h^t = (1 - z) \odot h^{t-1} + z \odot h'$$

这里 GRU 只用了一个门控就同时完成了遗忘和选择记忆。

4、叙述诗歌生成过程

tf 版本的诗歌生成代码如下：

```

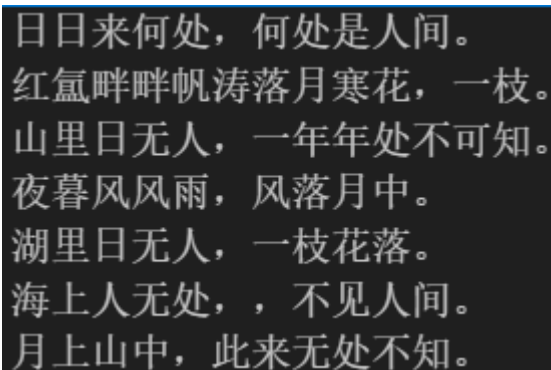
1 def gen_poem(begin_word):
2     # state = [tf.random.normal(shape=(1, 128), stddev=0.5),
3     tf.random.normal(shape=(1, 128), stddev=0.5)]
4     state = tf.zeros((1, 128))
5     cur_token = tf.constant([word2id[begin_word]],
6     dtype=tf.int32)
7     collect = []
8     collect.append(cur_token.numpy()[0])
9     while cur_token != word2id[end_token]:
10         cur_token, state = model.get_next_token(cur_token,
11         state)
12         collect.append(cur_token.numpy()[0])
13     if len(collect) > 30:
14         break
15     return [id2word[t] for t in collect]

```

首先需要指定一个 shape 为 1*128 的初始状态 state，可以随机生成，也可以设全零。然后将指定的 begin_word 通过 word2id 字典得到向量形式 cur_token，利用 collect 记录已经生成的 token，开始循环，在循环中根据 cur_token 和 state 两个输入，通过 RNN model 得到预测的下一组 token 和 state，如此循环直到生成了 end_token，这样就得到了一句话，返回时再利用 id2word 字典得到字符串的形式返回。

5、诗歌生成结果

初始 state 置全 0 得到的结果如下。



日日来何处，何处是人间。
 红氍毹畔帆涛落月寒花，一枝。
 山里日无人，一年年处不可知。
 夜幕风风雨，风落月中。
 湖里日无人，一枝花落。
 海上人无处，，不见人间。
 月上山中，此来无处不知。

初始 state 随机生成得到的较好的一次结果如下。

日暮风风不见，人间不可知。
红蓉畔迴堪怜柳树寒。
山上水开。日来无处，不知不得。
夜暮风如此，何处不知。
湖里，不知人事不知人。
海上山风不见，山上月中春。
月闻不见，风吹落云声。